

PROCEEDINGS OF THE

12th INTERNATIONAL **MODELICA** CONFERENCE

May 15–17, 2017

Clarion Congress Hotel Prague

Czech Republic

www.modelica.org



ČSKI



POLITECNICO
MILANO 1863

MODELICA

The conference is organized by The Czech Society for Cybernetics and Informatics (ČSKI) and Politecnico di Milano in cooperation with the Modelica Association.

Proceedings of the 12th International Modelica Conference
Prague, Czech Republic, May 15-17, 2017

Editors:

Doc. MUDr. Jiří Kofránek, CSc. and Prof. Francesco Casella

Published by:

Modelica Association and Linköping University Electronic Press

ISBN: 978-91-7685-575-1

Series: Linköping Electronic Conference Proceedings, No 132

ISSN: 1650-3686

eISSN: 1650-3740

DOI: <http://dx.doi.org/10.3384/ecp17132>

Organized by:

ČSKI

(Czech Society for Cybernetics and Informatics)

Pod Vodárenskou věží 2

182 07 Praha 8 - Libeň

Czech Republic

Politecnico di Milano

Dipartimento di Elettronica, Informazione e Bioingegneria

Piazza Leonardo da Vinci, 32

20133 Milano

Italy

in co-operation with:

Modelica Association

c/o PELAB, Linköpings Univ.

SE-581 83 Linköping

Sweden

Conference location:

Clarion Congress Hotel Prague

Freyova 33

190 00 Praha 9 - Vysočany

Czech Republic

Copyright © Modelica Association, 2017

WELCOME



Jiří Kofránek
Conference Chair

We would like to welcome you to Prague for the 12th international Modelica Conference. The conference was organized by the Modelica Association in cooperation with the Czech Society for Informatics and Cybernetics and Politecnico di Milano.

Modelica is not only a unique modeling language, which is widely used in numerous branches of industry and also in research and science, but most of all it is an immensely effective tool for complex simulations in the automotive industry, building energy management, aerospace and many other fields of engineering.

The program of the conference is interesting not only for the participants, who already use Modelica, but also for those who would like to be introduced to the possibilities of this new modern modeling language by our numerous tutorials. The usage of the language is facilitated by Modelica libraries focused on diverse fields. Consequently, an important part of the conference is the traditional Library Award Announcement.

We welcome you to Prague, the city of many historic sites, culture and also the music festival Prague Spring, taking place in Prague this week.



Francesco Casella
Program Chair

The International Modelica Conference is the most important place for the Modelica and FMI communities to meet, exchange ideas and advance the state of the art in object-oriented modelling.

This year we received 129 paper submissions for the scientific program. After a thorough peer review process by the International Program Committee, 83 were accepted for full oral presentation and 19 for poster presentation, with authors coming from 18 different countries in Europe, Asia, America, and Oceania. The scientific program is completed by two distinguished keynote talks, one from industry and one from academia.

The conference also hosts nine tutorials, the FMI User Meeting, as well as vendor presentations and a commercial exhibition.

I warmly welcome you to the 12th International Modelica Conference and I wish you a successful, pleasant, and rewarding stay in Prague!

KEYNOTE SPEAKERS



Challenges of Future Robotics

Presenter:

Bernd Liepert
President of the euRobotics AISB
Chief Innovation Officer at KUKA AG

Abstract: Robotics will change the world! It will unleash the same if not an even more disruptive and transformational power within the next 50 years as mainstream IT-technology and the Internet have over the last half a century. Nurtured by technological breakthroughs in industrial automation, robotics will exhaustively permeate all domains of the human living realm. Hence, our grandchildren will grow up in a society that is enriched and enhanced by assistive technologies in every imaginable way. Robotics and automation will be tailored into many everyday objects, becoming an integral part of all kinds of appliances. This Generation „R“ will be without fear of these technologies perceiving their beneficial nature - they will grow up as Robotic Natives. This implies, that today's people are already born to become the first society of Robotic Immigrants. Although it is not possible to precisely predict the world of tomorrow, the presented model of the 4 Robotic Revolutions provides a compelling, holistic approach to describe the future phases of robotic evolution, characterizing them according to their technological enablers and underlying interaction paradigms.

Bio: Dr. Bernd Liepert is the Chief Innovation Officer of KUKA AG, a world leading manufacturer of industrial robots. Dr. Liepert earned his diploma in mathematics in 1990 from the University of Augsburg and his honorary doctor degree from University of Magdeburg in 2011. Since 1990 Dr. Liepert has worked in changing positions for KUKA. From 1990 to 1996 he worked as mathematician and developer at KUKA Schweissanlagen + Roboter GmbH before he took charge as head of development of the newly founded company KUKA Roboter GmbH until 1997. From 1998-1999 he was a member of KUKA Roboter GmbH Board of Management, responsible for development and design. From 2000-2009 Dr. Liepert was the CEO of KUKA Roboter GmbH. From 2010 to January 2015 he was the CTO of KUKA AG, responsible for technology and development of the whole KUKA group. As Chief Innovation Officer of KUKA AG, Dr. Liepert is now responsible for expanding innovations at KUKA where he can apply his vast robotics experience at the interface between technology and the market. From 2008-2012 Dr. Liepert was President of EUROP, the European Robotics Technology Platform, and subsequently President of euRobotics AISBL – the European Robotics Association. euRobotics was founded in September 2012 and has become the private side of SPARC, the European Public-Private Partnership in Robotics in 2013. As president of these associations Dr. Liepert has been leading the European robotics community and representing it at high political levels.



Synchronous Programming and its fit with Modeling

Presenter:

Gérard Berry
Paris, France

Abstract: The family of Synchronous programming languages was born in the 1980's in three different French labs that gathered researchers in Computer Science and Control Theory. The three first languages were Esterel, dedicated to control-dominated problems in embedded systems, telecom protocols and later digital circuit design, Lustre, dedicated to continuous control, and Signal, oriented towards signal processing. They share a common perfect synchrony principle that expresses that the reaction to an input should be viewed as conceptually instantaneous. This simple principle is well-adapted to the targeted applications and greatly simplifies programming by reconciling parallelism and determinism. It also leads to well-defined mathematical semantics that directly ground their formal compiling, simulation and verification environments. Synchronous programming rapidly became used in Industry for safety-critical production systems in avionics (Dassault Aviation, Airbus, etc.), railways, etc., as well as in robotics and circuit design. In the 2000's, Esterel and Lustre have been unified in two new languages industrialized by Esterel Technologies (now part of Ansys): SCADE 6 for safety critical software and Esterel v7 for hardware design, both also incorporating ideas from Harel's reactive graphical formalism Statecharts.

The talk will explain the practical and mathematical concepts of synchronous programming and stress its advantages over asynchronous concurrent programming for the considered applications. It will also explore the links between synchronous programming and modeling / simulation. In one direction, synchronous languages are ideal targets to generate embedded code from executable parts of simulation models. In the other direction, embedding synchrony into conventional modelers may be necessary to solve the current tricky issues due to the coupling of discrete and continuous computations in modelers, in particular for the currently mishandled case where external or internal events provoke cascades of discrete reactions. Pouzet and Bourkes's new Zelus language is a step in this direction.

Bio: Former student of the Ecole polytechnique, Member of the Academy of sciences, of the Academy of technology and the Academia Europaea, CNRS Gold medal 2014, Gérard Berry was a researcher at the Ecole des mines of Paris and INRIA from 1973 to 2000, Chief Scientist of the company Esterel Technologies from 2001 to 2009, then Research Director at INRIA and President of the Evaluation Committee of this Institute from 2009 to 2012. He holds the Chair Algorithms, Machines and Languages at the Collège de France from 2012, after having held two annual chairs in 2007-2008 and 2009-2010.

His scientific contribution concerns four main topics: the formal treatment of programming languages and their relations with mathematical logic, reactive and real-time programming for embedded systems, integrated circuit computer-aided design, and formal verification of programs and circuits. He is the creator of the Esterel programming language.

Program Committee

Conference Chair

Doc. MUDr. Jiří Kofránek, CSc., Charles University in Prague, Czech Republic

Program Chair

Prof. Francesco Casella, Politecnico di Milano, Italy

Conference Board

Dr. Hilding Elmqvist, Mogram AB, Lund, Sweden

Prof. Peter Fritzson, Linköping University, Sweden

Prof. Martin Otter, DLR, Germany

Dr. Michael Tiller, Xogeny, Michigan, USA

Program Committee

Dr. Johan Åkesson, Modelon AB, Lund, Sweden

Prof. Bernhard Bachmann, Univ. Applied Sciences Bielefeld, Bielefeld, Germany

Prof. John Baras, University of Maryland, Maryland, USA

Dr. John Batteh, Modelon Inc., Ann Arbor, USA

Dr. Albert Benveniste, INRIA, Rennes, France

Christian Bertsch, Robert Bosch GmbH, Stuttgart, Germany

Volker Beuter, VI-grade GmbH, Marburg, Germany

Torsten Blochwitz, ITI GmbH, Dresden, Germany

Dr. Scott Bortoff, MERL Cambridge, USA

Dr. Timothy Bourke, INRIA, France

Dr. Marco Bonvini, Whisker Labs, USA

Daniel Bouskela, EDF R&D, Paris, France

Prof. David Broman, KTH Royal Institute of Technology, Stockholm, Sweden

Dr. Dan Burns, MERL, Cambridge, USA

Prof. Francesco Casella, Politecnico di Milano, Milano, Italy

Prof. Massimo Ceraolo, University of Pisa, Italy

Prof. François E. Cellier, ETH Zürich (retired), Zürich, Switzerland

Dr. Christoph Clauß, Fraunhofer IIS EAS (retired), Dresden, Germany

Dr. Johan de Kleer, PARC, Palo Alto, USA

Mike Dempsey, Claytex Services Ltd, UK

Dr. Hilding Elmqvist, Mogram AB, Lund, Sweden

Dr. Olaf Enge-Rosenblatt, Fraunhofer IIS Dresden, Dresden, Germany

Prof. Gianni Ferretti, Politecnico di Milano, Italy

Dr. Rüdiger Franke, ABB AG, Mannheim, Germany

Dr. Jens Frenkel, ESI ITI GmbH, Dresden, Germany

Prof. Peter Fritzson, Linköping University, Sweden

Leo Gall, LTX Simulation GmbH, Munich, Germany

Peter Harman, CAE Tech Limited, UK

Prof. Anton Haumer, OTH Regensburg, Regensburg, Germany

Dr. Dan Henriksson, Dassault Systèmes, Lund, Sweden

Dr. Yutaka Hirano, Toyota, Japan

Christoph Höger, TU Berlin, Germany

Prof. Bengt Jacobson, Chalmers Technical University, Gothenburg, Sweden

Prof. Tommi Karhela, VTT / Aalto University, Espoo, Finland

Åke Kinnander, Siemens Turbo, Sweden

Jochen Köhler, ZF AG, Friedrichshafen, Germany

Dr. Christian Kral, TGM, Vienna, Austria

Dr. Chris Laughman, MERL, Cambridge, USA

Prof. Alberto Leva, Politecnico di Milano, Italy

Kilian Link, Siemens AG, Erlangen, Germany

Prof. Edward Lee, UC Berkeley, USA

Prof. Marco Lovera, Politecnico di Milano, Italy

Kristin Majetta, Fraunhofer IIS, Dresden, Germany

Dr. Jakob Mauss, QTronic GmbH, Berlin, Germany
 Dr. Alexandra Mehlhase, Arizona State University, USA
 Dr. Lars Mikelsons, Bosch-Rexroth GmbH, Lohr am Main, Germany
 Ramine Nikoukhah, Altair Engineering, Paris, France
 Prof. Henrik Nilsson, University of Nottingham, Nottingham, Great Britain
 Prof. Mattias Nyberg, Scania AB, Södertälje, Sweden
 Prof. Akira Ohata, Toyota Motor Corporation, Tokyo, Japan
 Dr. Hans Olsson, Dassault Systèmes, Lund, Sweden
 Prof. Martin Otter, DLR, Oberpfaffenhofen, Germany
 Dr. Andreas Pillekeit, dSPACE, Germany
 Dr. Adrian Pop, Linköping University, Sweden
 Johan Rhodin, 84 Codes, Missouri, USA
 Dr. Adrijan Ribaric, Sentient Science, Idaho Falls, USA
 Dr. Clemens Schlegel, Schlegel Simulation, Munich, Germany
 Prof. Gerhard Schmitz, Technical University Hamburg-Harburg, Germany
 Dr. Peter Schneider, Fraunhofer IIS EAS, Dresden, Germany
 Prof. Stefan-Alexander Schneider, Kempten University of Applied Sciences, Germany
 Dr. Martin Sjölund, Linköping University, Sweden
 Prof. Thierry Soriano, Supmeca, France
 Dr. Rita Streblow, RWTH Aachen, Aachen, Germany
 Dr. Ed Tate, Exa, Livonia, USA
 Dr. Wilhelm Tegethoff, TLK-Thermo GmbH, Germany
 Bernhard Thiele, Linköping University, Sweden
 Dr. Michael Tiller, Xogeny, Michigan, USA
 Dr. Jakub Tobolar, DLR Oberpfaffenhofen, Munich, Germany
 Dr. Hubertus Tummescheit, Modelon Inc., West Hartford, USA
 Prof. Alfonso Urquía, UNED, Madrid, Spain
 Prof. Luigi Vanfretti, KTH Royal Institute of Technology, Stockholm, Sweden
 Prof. Hans Vangheluwe, McGill University, Canada and University of Antwerp, Belgium
 Dr. Subbarao Varigonda, Cummins, Columbus, USA
 Dr. Stéphane Velut, Lund, Sweden
 Dr. Michael Wetter, Lawrence Berkeley National Laboratory, Berkeley, USA
 Prof. Dietmar Winkler, University College of Southeast Norway, Norway
 Dr. Dirk Zimmer, DLR Oberpfaffenhofen, Germany

Conference Organization Team:

Prof. Francesco Casella, Politecnico di Milano, Italy
 Filip Ježek, Czech Technical University in Prague, Czech Republic
 Doc. MUDr. Jiří Kofránek, CSc., Charles University in Prague, Czech Republic
 Dr. Marek Matejak, Charles University in Prague, Czech Republic
 Veronika Sýkorová, Creative Connections s.r.o., Prague, Czech Republic
 Milena Zeithamlová, Action M Agency, Prague, Czech Republic

Contents

Session 1: Keynote 1	17
Session 4A: Automotive I	17
Development of an Integrated Control of Front Steering and Torque Vectoring Differential Gear System Using Modelica	17
Virtual Occupant Model for Riding Comfort Simulation	27
A Simulation-Based Digital Twin for Model-Driven Health Monitoring and Predictive Maintenance of an Automotive Braking System	35
Improved Aerodynamic Prediction Through Coupled System and CFD Models	47
Session 4B: Buildings I	55
Coupled Simulation between CFD and Multizone Models Based on Modelica Buildings Library to Study Indoor Environment Control	55
Co-Simulation between detailed building energy performance simulation and Modelica HVAC component models	63
Aspects of FMI in Building Simulation	73
Application of Richardson Extrapolation to the Co-Simulation of FMUs from Building Simulation . .	79
Session 4C: Process & Chemical Engineering	89
Development of a Thermodynamic Engine in OpenModelica	89
Integrated Process and Molecular Design with Modelica Using Continuous-Molecular Targeting	101
Dynamic Simulations of the Post-combustion CO ₂ Capture System of a Combined Cycle Power Plant	111
Optimizing the start-up process of post-combustion capture plants by varying the solvent flow rate . .	121
Session 4D: Control Systems I	131
Framework for dynamic optimization of district heating systems using Optimica Compiler Toolkit . .	131
Optimal Control of District Heating Systems using Dynamic Simulation and Mixed Integer Linear Programming	141
Rapid development of an aircraft cabin temperature regulation concept	151
Investigation of the Influence of Controller Approaches on Room Thermal Behaviour A Simulation Study	161
Session 5A: Automotive II	171
Powertrain and Thermal System Simulation Models of a High Performance Electric Road Vehicle . . .	171
Investigating the Effect of a Sonic Restrictor in the Intake of an Engine	181
Engine thermal shock testing prediction through coolant and lubricant cycling in Dymola	189
Session 5B: Buildings II	199
Template based code generation of Modelica building energy simulation models	199
Modelling and Simulation of Standardised Control Functions from Building Automation	209
Modelling of Heat Pumps with Calibrated Parameters Based on Manufacturer Data	219
Session 5C: Electrical & Power Systems I	227
Simulation of Large Grids in OpenModelica: reflections and perspectives	227
A Modelica-based Tool for Power System Dynamic Simulations	235
A Modelica VSC-HVDC Average Value Model Implementation and its Software-to-Software Validation using an EMT Power System Domain Specific Simulator	241
Session 5D: Control Systems II	249
From system model to optimal control - A tool chain for the efficient solution of optimal control problems	249
Nonlinear Model Predictive Control of a Thermal Management System for Electrified Vehicles using FMI	255
Defining and Solving Hybrid Optimal Control Problems with Higher Index DAEs	265

Session 6: Poster Session	275
Large Scale Training through Spoken Tutorials to Promote and use OpenModelica	275
EMOTH The EMobility Library of OTH Regensburg	285
Simulating a Variable-structure Model of an Electric Vehicle for Battery Life Estimation Using Modelica/Dymola and Python	291
Model Reduction Techniques Applied to a Physical Vehicle Model for HiL Testing	299
Towards Virtual Validation of ECU Software using FMI	307
Parameter Estimation based on FMI	313
Generic FMI-compliant Simulation Tool Coupling	321
FMI and IP protection of models: A survey of use cases and support in the standard	329
Model-based virtual sensors by means of Modelica and FMI	337
Dymola-JADE Co-Simulation for Agent-Based Control in Office Spaces	345
Failure Modes of Tearing and a Novel Robust Approach	353
Towards Adjoint and Directional Derivatives in FMI utilizing ADOL-C within OpenModelica	363
PDEModelica and Breathing in an Avalanche	367
Multirotor Aerial Vehicle modeling in Modelica	373
Rotating Machinery Library for Diagnosis	381
Modelling and Simulation of the passive Structure of a 5-Axis-Milling Machine with rigid and flexible bodies for evaluating the static and dynamic behaviour	389
Modeling and Simulation on Environmental and Thermal Control System of Manned Spacecraft	397
Modeling and simulation of complex ThermoSysPro model with OpenModelica - Dynamic Modeling of a combined cycle power plant	407
A Power-Based Model of a Heating Station for District Heating (DH) System Applications	415
Session 7A: Automotive III	425
Model Based Design of a Split Carrier Wheel Suspension for Light-weight Vehicles	425
Development of hierarchal commercial vehicle model for target cascading suspension design process	433
Model Based Analysis of Shimmy in a Racing Bicycle	441
Session 7B: Thermodynamic Systems	449
Optimization-friendly thermodynamic properties of water and steam	449
Modeling of a Thermosiphon to Recharge Phase Change Material Based Thermal Battery for a Portable Air Conditioning Device	459
Extended Modelica Model for Heat Transfer of Two-Phase Flows in Pipes Considering Various Flow Patterns	467
Session 7C: Electrical & Power Systems II	477
Improved Model of Photovoltaic Systems	477
Modelling of a Hydro Power Station in an Island Operation	483
Periodic Steady State Identification of electrical circuits	493
Session 7D: Control Systems III	507
Discrete-time models for control applications with FMI	507
Model-based Embedded Control using Rosenbrock Integration Methods	517
Integration of complex Modelica-based physics models and discrete-time control systems: Approaches and observations of numerical performance	527
Session 8: Keynote 2	533
Session 9A: FMI I	533
Improving Interoperability of FMI-supporting Tools with Reference FMUs	533
The Embedded Simulation via FMI and its Application to the Simulation of Lifetime Tests Including Wear	541
Integration Modelica with Digital Mockup Tool using the FMI	547
Session 9B: Numerical & Symbolic Methods	557
Solving large-scale Modelica models: new approaches and experimental results using OpenModelica	557
Transformation of Differential Algebraic Array Equations to Index One Form	565
Smart Processing of Function Calls to Achieve Efficient Simulation Code	581

Session 9C: Acoustic & Medical Systems	589
Integrative physiology in Modelica	589
Sound Source Extension Library for Modelica	605
Towards Medical Cyber-Physical Systems: Modelica and FMI based Online Parameter Identification of the Cardiovascular System	613
Session 9D: Wind & Naval Engineering	623
The DLR RailwayDynamics Library: the Crosswind Stability Problem	623
The OneWind Modelica Library for Floating Offshore Wind Turbine Simulations with Flexible Structures	633
Modelica Based Naval Architecture Library for Small Autonomous Boat Design	643
Session 10A: FMI II	653
FMI Go! A simulation runtime environment with a client server architecture over multiple protocols .	653
Building Parallel FMUs (or Matryoshka Co-Simulations)	663
Scaling FMI-CS Based Multi-Simulation Beyond Thousand FMUs on Infiniband Cluster	673
Development of an open source multi-platform software tool for parameter estimation studies in FMI models	683
Session 10B: Modelica Language & Tools	693
Innovations for Future Modelica	693
Hierarchical Semantics of Modelica	703
Towards a Standard-Conform, Platform-Generic and Feature-Rich Modelica Device Drivers Library .	713
modelica.university: A Platform for Interactive Modelica Content	725
Session 10C: Mechanical Systems Modelling	735
Object-oriented modelling of a flexible beam including geometric nonlinearities	735
Musculoskeletal Modeling of the Hand and Contact Object in Modelica	745
Modelica Spur Gears with Hertzian Contact Forces	755
Modeling of Roller Bearings	765
Session 10D: HVAC Systems	771
Cabin Thermal Needs: Modeling and Assumption Analysis	771
Simulative Comparison of Mobile Air-Conditioning Concepts for Mechanical and Electrical Driven Systems	783
Duty Cycle for Low Energy Operation of a Personal Conditioning Device	791
A Platform for the Agent-based Control of HVAC Systems	799
Session 11A: Modelica Tools & GUIs	809
MoVE A Standalone Modelica Vector Graphics Editor	809
Mo E A Communication Service Between Modelica Compilers and Text Editors	815
Traceability Support in OpenModelica Using Open Services for Lifecycle Collaboration (OSLC) . . .	823
A Simulation Environment for Efficiently Mixing Signal Blocks and Modelica Components	831
Session 11B: Power Plants & Energy Systems	839
Component Development for Nuclear Hybrid Energy Systems	839
Modeling and simulation of fixed bed regenerators for a multi-tower decoupled advanced solar combined cycle	847
Annual Performance of a Solar-Thermochemical Hydrogen Production Plant Based on CeO2 Redox Cycle	857
Applying the Power Plant Library ClaRa for Control Optimisation	867
Session 11C: Mechanical Systems, Robotics & VR	879
Interactive FMU-Based Visualization for an Early Design Experience	879
Using Modelica for advanced Multi-Body modelling in 3D graphical robotic simulators	887
A New Object-Oriented Approach for Integrating Discrete Element Method into Modelica	895
Modeling and Simulation of Wheel Driving Systems based on Terramechanics for Planetary Exploration Rover using Modelica	901

Session 11D: Aerospace	909
The Jet Propulsion Library: Modeling and simulation of aircraft engines	909
Virtual flight testing of a controller for gust load alleviation using FMI for cosimulation	921
The DLR Environment Library for Multi-Disciplinary Aerospace Applications	929

Author Index

Abel, Dirk	613	Dempsey, Mike	181, 299
Åberg, Marcus	449	Desmet, Wim	337
Adib Murad, Mohammed Ahsan	241	Dhumane, Rohit	459, 791
Albertelli, Paolo	735	Dominik, Andreas	809, 815
Albin, Thivaharan	613	Duncan, Brad	47
Altshuller, Dmitry	477	Durling, Erik	329
Andreasson, Johan	745	El Hefni, Baligh	407
Aoyama, Kazuhiro	643	Elci, Mehmet	415
Asghar, Adeel	823	Elmqvist, Hilding	565, 693
Aute, Vikrant	459, 791	Emhofer, Johann	605
Bachmann, Bernhard	363, 557, 581	Évora Gómez, José	663
Baharev, Ali	353	Fanli, Zhou	397
Banakar, Shivakumar	783	Febres, Jesús	847
Bardaro, Gianluca	887	Ferretti, Gianni	441, 735
Bardow, André	101	Fischer, Torben	255
Bartolini, Andrea	227	Fleps-Dezasse, Michael	425
Bascetta, Luca	887	Franke, Rüdiger	363, 507
Batteh, John	35, 47, 171, 527	Fraulob, Sebastian	541
Bau, Uwe	101	Fritzsche, Jörg	249
Baumgartner, Daniel	425	Fritzson, Peter	89, 275, 823
Baviere, Roland	141	Fütterer, Johannes	799
Bayon, Alicia	857	Galindo, Eduardo	189
Beaude, Francois P.	235	Gallagher, Stephen	299
Bellmann, Tobias	713	Gallarotti, Maura	181
Bender, Daniel	151	Galtier, Virginie	663
Berenguel, Manuel	683	Gargoloff, Joaquin	47
Bergianti, Luca	171	Gauterin, Frank	255
Bertsch, Christian	533	Gertig, Christoph Udo	101
Beutlich, Thomas	713, 895	Gesenhues, Jonas	613
Blochitz, Torsten	507	Gillot, Romain	299
Bonilla, Javier	683	Giraud, Loïc	141
Bouskela, Daniel	407	Gonzalez Cocho, Mikel	337
Braun, Willi	363, 557	Gottelt, Friedrich	467, 867
Brembeck, Jonathan	425	Gräber, Manuel	249
Breque, Florent	771	Greenwood, Scott	839
Briese, Lâle Evrim	929	Greiner, Christopher	527
Bünning, Felix	799	Grether, Gustav	623
Bünthe, Tilman	425	Griffin, John	47
Bürger, Christoff	517	Grimm, Alexander	285
Carballo, Jose Antonio	683	Gross, Joachim	101
Casella, Francesco	227, 557, 887	Gundermann, Julia	541
Caujolle, Mathieu	663, 673	Hagemann, Jan	581
Choi, Hyung Yun	27	Han, Bing	381
Cimmino, Massimo	219	Han, Manyong	27
Clauss, Christoph	79, 161	Härdin, Tomas	653
Constantin, Ana	345	Haumer, Anton	285, 389
Corniglion, Remi	673	Heckmann, Andreas	623
Corniglion, Rémi	663	Hein, Marc	613
Corves, Burkhard	765	Henningsson, Maria	329
Croes, Jan	337	Henningsson, Toivo	693
Dad, Cherifa	673	Henriksson, Dan	517
Dahash, Abdulrahman	415	Heo, Seungjin	433
Dahl, Markus	755	Hernández Cabrera, José Juan	663
Datta, Kaushik	275	Heyberger, Jean-Baptiste	235
Davoudabadi, Peyman	35	Hirano, Yutaka	17, 425
de La Calle, Alberto	857	Hirao, Akinari	27

Höger, Christoph	703	Matsuda, Shinji	547
Hoppe, Timm	467, 867	Matsuoka, Hisayoshi	27
Huchtemann, Kristian	345	Matteucci, Matteo	887
Hüsson, Peter	477	Mattsson, Sven Erik	507, 517
Hyun, Minsu	433	Menager, Nils	313
Ianotto, Michel	663	Mengist, Alachew	823
Inderfurth, Alexander	199	Merabet, Massinissa	141
Invernizzi, Davide	735	Mesonero, Iván	847
Ishibashi, Tatsuro	381	Meyer, Richard	79
Jain, Rahul	89, 275	Mikelsons, Lars	307
Janczyk, Leonard	477	Mocholí Montañés, Rubén	111
Ježek, Filip	367, 589	Mösch, Danny	313
Jian, Jin	397	Moudgalya, Kannan	89, 275
Jo, Jaehun	433	Mucha, Katharina	199
Johnson, Lee	35	Mukbil, Awad	533
Jones, Christopher	477	Müller, Dirk	345, 799
Junghanns, Andreas	533	Müller, Reiko	921
Justus, Nicola	809, 815	Müller, Wolfgang	321
Kampfmann, Rüdiger	313	Najafi, Masoud	831
Kang, Daeoh	433	Nassif, Fady	831
Kaul, Werner	199	Nayak, Priyam	275
Kawai, Tadao	381	Nemer, Maroun	771
Kerling, Ines	151	Nemmaru, Bhargava	275
Kesarkar, Omkar	35	Neumaier, Arnold	353
Ketelhut, Maike	613	Nicolai, Andreas	63
Kirches, Christian	255	Nielsen, Lasse	867
Klöckner, Andreas	929	Nikoukhah, Ramin	831
Ko, Kwangchan	433	Nonaka, Kenichiro	901
Köckeis, Rupert	389	Nord, Lars Olof	111
Kofránek, Jiří	367, 589	Nytsch-Geusen, Christoph	161, 199
Kraus, Tom	255	Ochel, Lennart	507, 581
Kremers, Enrique	663	Oda, Takatsugu	901
Krishnaswamy, Sivasubramani	35	Ohser, Florian	895
Kuhn, Martin	493	Ohtomi, Koichi	547
Kulhanek, Tomas	589	Oizumi, Kazuya	643
Kulshreshtha, Kshitij	363	Olsson, Hans	507, 517
Kuric, Muhamed	373	Osmic, Nedim	373
Lacoursière, Claude	653	Otter, Martin	507, 517, 565, 693
Lanzerath, Franz	101	Paepcke, Anne	63
Larsson, Per-Ola	131	Palmkvist, Elias	329
Lefeng, Sun	397	Park, Jongchan	433
Leimeister, Mareike	633	Peßler, Georg Ambrosius	209
Leon, Gladys E.	235	Pfeiffer, Andreas	517
Letschert, Thomas	815	Picarelli, Alessandro	181, 189, 299
Leva, Alberto	227	Pipiorke, Jörg	73
Limperich, Dirk	783	Pitchaikani, Anand	35, 909
Ling, Jiazhen	459, 791	Pluymers, Bert	337
Liping, Chen	397	Pollok, Alexander	151
López Pérez, Susana	847	Ponci, Ferdinanda	345
Löwen, Artur	345	Pop, Adrian	89, 275, 823
Magargle, Ryan	35	Pytlak, Radoslaw	265
Magnani, Gianantonio	441	Qi, Liu	397
Magnúsdóttir, Arnór	483	Radermacher, Reinhard	459, 791
Magnusson, Fredrik	131, 449	Rädler, Jörg	199
Majetta, Kristin	79, 161	Reichl, Christoph	605
Mandloi, Padmesh	35	Reinbold, Vincent	663, 673
Marx-Schubach, Thomas	121	Reiner, Matthias	929
Matejak, Marek	589	Richter, Christian	895

Ritter, Markus	921	Vialle, Stephane	673
Roca, Lidia	683	von Manstein, Arnim	783
Runvik, Håkan	131, 449	Walther, Andrea	363
Salgado, Oscar	337	Walther, Susanne	541
Samlaus, Roland	307	Wang, Kai	527
Sammak, Majed	909	Waurich, Volker	713, 879
Sangi, Roozbeh	799	Weber, Jürgen	879, 895
Scaglioni, Bruno	441, 735	Wei, Liu	397
Schichl, Hermann	353	Weiser, Tobias	765
Schilling, Johannes	101	Wernersson, Karl	507
Schmitz, Gerhard	121	Wetter, Michael	219
Schnabel, Uwe	541	Wettergren, Håkan	755
Schneider, Georg Ferdinand	209	Widl, Edmund	321
Schneider, Michael	389	Windahl, Johan	449
Schölzel, Christopher	809, 815	Winkler, Dietmar	483, 725
Schwan, Torsten	73	Wischhusen, Stefan	467
Schweiger, Gerald	131	Yoshikawa, Hiroki	901
Sekiguchi, Kazuma	901	Zawadzki, Tomasz	265
Selvan, Nithish	909	Zimmer, Dirk	151
Sevilla, Thomas	55	Zitzenbacher, Raimund	605
Sielemann, Michael	909	Zuo, Wangda	55
Šilar, Jan	367, 589		
Sjölund, Martin	713		
Sohn, Michael	55		
Soler, Rodolfo	189		
Steiger, Simone	209		
Steingrube, Annette	415		
Stellato, Massimo	171		
Stüber, Moritz	291		
Suski, Damian	265		
Sutherland, Joshua	643		
Suzuki, Hiromasa	547		
Swaminathan, Shashank	745		
Tahirovic, Adnan	373		
Tarnawski, Tomasz	265		
Tate, Ed	47		
Täuber, Patrick	581		
Tavella, Jean-Philippe	663, 673		
Tegethoff, Wilhelm	249		
Thiele, Bernhard	713		
Thiele, Matthias	541		
Thomas, Philipp	633		
Thorade, Matthis	199		
Tian, Wei	55		
Tidefelt, Henrik	755		
Tiller, Michael	725		
Tillmanns, Dominik	101		
Tobolar, Jakub	425		
Todtermuschke, Karsten	541		
Tomati, Nicolò	441		
Toriya, Hiroshi	547		
Trentelman, Thom	643		
Tugores, Carles Ribas	199		
Tummescheit, Hubertus	47		
Unger, René	73		
Vallée, Mathieu	141		
Vanfretti, Luigi	241		
Velut, Stéphane	131		

Development of an Integrated Control of Front Steering and Torque Vectoring Differential Gear System Using Modelica

Yutaka Hirano¹

¹ Toyota Motor Corporation, Japan, yutaka_hirano@mail.toyota.co.jp

Abstract

To achieve future low carbon mobility society, many new-type electric vehicles (EVs) are developed actively in recent period. Those EVs have integrated power unit which take place of conventional engine, transmission and differential gear components. Additionally it is rather easy to integrate torque vectoring function to those power units using gear sets to control torque distribution between left wheel and right wheel. In this paper, model-based development of an integrated control of the front steering angle and torque vectoring differential (TVD) gear system is described. New integrated control logic was developed using model matching control to let the vehicle yaw rate and vehicle slip angle follow the desired dynamics. Simulation results using an extended single track model of vehicle dynamics are shown to prove the efficacy of the proposed control. Though, full vehicle model considering all of vehicle dynamics and drive train motion using Modelica clarified the problem of this method in actual cases. Difference between the extended single track model and full vehicle model was compared to estimate the reason of the problem.

Keywords: *Model Based Development, Vehicle Dynamics, Torque Vectoring, Model Matching Control*

1 Introduction

To satisfy needs for future low-carbon mobility society, development of many new EVs is increasingly active in recent years. Additionally many new proposals about integrated electric power train which also has torque vectoring capability are presented (Höhn *et al.*, 2013). (Burgess, 2009) showed a model-based control design of TVD using an inverse model for feed-forward control. (Efsthathios *et al.* 2015) introduced a model predictive control of TVD considering non-linear tire characteristics. On the other hand, authors have researched a new control of TVD by using traditional PI feedback control (Hirano *et al.*, 2013)(Hirano *et al.*, 2014). The author also utilized a model matching control theory to develop a new control of TVD (Hirano, 2016a). Additionally the author expand the control to the integrated control of TVD and active front steering (AFS) by model matching control (Hirano, 2016b). The purpose of

using both TVD and AFS is to control both vehicle yaw rate and slip angle independently. In the last paper, the derived control was based on simple LQR (Linear Quadratic Regulator) and there was no measure to cope with steady state deviation. In this paper, the LQR design was modified by augmenting the plant model to include integral of the state variables. As same as the last research, an extended single track model of vehicle dynamics was used to derive and verify the new control. Finally the developed control was verified by using the full vehicle model using Modelica. Some measures about solving problems when applying Modelica to this kind of problem are also mentioned.

2 Experimental EV

Table 1 Specifications of new experimental EV

	New EV	Conventional car
Vehicle Mass	750 kg	1240 kg
Yaw Moment Inertia	869 kgm ²	2104 kgm ²
Wheelbase	2.6 m	2.6 m
Front : Rear Weight Distribution	0.48 : 0.52	0.62 : 0.38
Height of CG	0.38 m	0.55 m
Tire RRC	5×10^{-3}	8.8×10^{-3}
Tire Normalized CP	16.1	20.4

The proposed experimental EV has specifications as shown in Table 1. Compared with a conventional small-class passenger car, the new EV has better characteristics of lighter vehicle weight, smaller yaw moment of inertia, lower height of the center of gravity (CG) and lower rolling resistance coefficients (RRC) value of tires. Because of these characteristics, this new EV is expected to have better handling and lower energy consumption than conventional vehicles. On the other hand, because of lighter weight and lower value of tire normalized CP (Cornering Power), this new EV seems more sensitive against external disturbances such as crosswind and road irregularity than the conventional cars. To cope with this problem, direct yaw moment control (DYC) was applied by using a

new integrated transaxle unit for rear axle which has a main electric motor and also TVD gear unit with a control motor. Additionally, to control both yaw rate and slip angle of the vehicle independently, another control input of active front steering (AFS) was introduced.

3 Vehicle Model

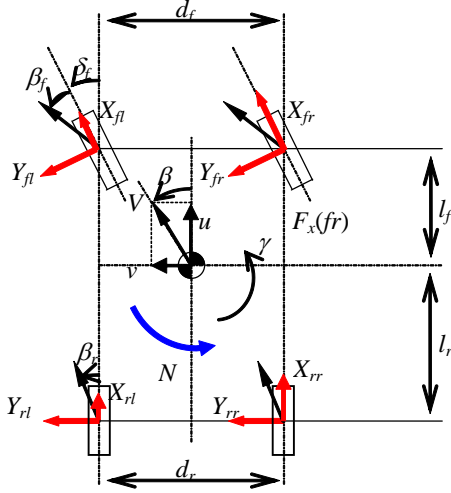


Figure 1. Extended single track vehicle model

Figure 1 shows an extended single track vehicle model to derive the control logic. Usually the single track model calculates front and rear tire side forces by adding both tire forces of right tire and left tire respectively. But in this paper, the model was extended to separate the tire longitudinal forces of right tire and left tire to consider direct yaw moment generated by the difference of the longitudinal forces of right tire and left tire. The coordinate system of this model follows FLU (x: forward, y: leftward, z; upward) convention. The simplified equations of motion by this extended single track model become as follows.

$$M \frac{dV}{dt} = F = (X_{rr} + X_{rl}) \quad (1)$$

$$MV \left(\frac{d\beta}{dt} + \gamma \right) = 2Y_f + 2Y_r \quad (2)$$

$$I_z \frac{d\gamma}{dt} = 2l_f Y_f - 2l_r Y_r + N \quad (3)$$

where

$$Y_f = -K_f \beta_f = -K_f \left(\beta + \frac{l_f}{V} \gamma - \delta_f \right) \quad (4)$$

$$Y_r = -K_r \beta_r = -K_r \left(\beta - \frac{l_r}{V} \gamma \right) \quad (5)$$

$$N = d_r (X_{rr} - X_{rl}) \quad (6)$$

Here,

- β : Vehicle slip angle,
- γ : Vehicle yaw rate,
- M : Vehicle mass,
- V : Vehicle velocity,
- I_z : Vehicle yaw moment of inertia,
- $l_f (l_r)$: Distance from the CG to front (rear) axle, (CG: Center of Gravity)
- $d_f (d_r)$: Tread of front (rear) axle,
- X_{**} : Longitudinal force of each tire,
- $Y_f (Y_r)$: Lateral force of front (rear) tires,
- δ_f : Steering angle of front tire,
- F : Vehicle driving force,
- N : Direct yaw control moment by TVD.

K_f and K_r are the equivalent cornering power of front and rear tire respectively.

If driving force F and DYC moment N can be calculated by some control logic, then the target longitudinal forces of left and right rear wheels to be realized by TVD power unit become as follows from Equation (1) and (6).

$$X_{rr} = \frac{1}{2} \left(F + \frac{N}{d_r} \right) \quad (7)$$

$$X_{rl} = \frac{1}{2} \left(F - \frac{N}{d_r} \right) \quad (8)$$

4 Control Design

4.1 Longitudinal Driving Force Control

Let us suppose that the desired value of vehicle speed, vehicle yaw rate and vehicle slip angle are defined as V_{ref} , γ_{ref} and β_{ref} respectively.

The desired vehicle driving force F can be calculated as below by PI feedback control and Equation (1)..

$$F = M \frac{dV_{ref}}{dt} + K_{PF} (V_{ref} - V) + K_{IF} \int (V_{ref} - V) dt \quad (9)$$

Here K_{PF} is a proportional feedback gain and K_{IF} is an integral feedback gain.

4.2 Model Matching Control of Lateral Dynamics

For the lateral dynamics, the state space form of the vehicle dynamics with TVD and AFS control becomes as follow from Equations (2) and (3).

$$\dot{x} = Ax + Bu \quad (10)$$

Here,

$$x = \begin{bmatrix} \beta \\ \gamma \end{bmatrix} : \text{State variables} \quad (11)$$

$$u = \begin{bmatrix} \delta_f \\ N \end{bmatrix} : \text{Control inputs} \quad (12)$$

$$A = \begin{bmatrix} -\frac{2(K_f + K_r)}{MV} & -1 - \frac{2(l_f K_f - l_r K_r)}{MV^2} \\ \frac{2(l_f K_f - l_r K_r)}{I_z} & -\frac{2(l_f^2 K_f + l_r^2 K_r)}{I_z V} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad (13)$$

$$B = \begin{bmatrix} \frac{2K_f}{MV} & 0 \\ \frac{2l_f K_f}{I_z} & \frac{1}{I_z} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \quad (14)$$

Please note that the elements of the matrix A of the Equation (10) are dependent on the vehicle velocity V as shown in the Equation (13). So the vehicle dynamics system described by the Equation (10) is a time-variant system.

It is well known that the response of both yaw rate and slip angle become to the second order lag function of the steering input when no control is applied. This fact results in that the ordinary drivers tend to respond to steer with time lag against the vehicle motion and tend to result in vehicle spin when the vehicle motion becomes unstable such as on the slippery road. On the other hand, it becomes easier for drivers to stabilize the vehicle if the response of the vehicle motion will become to the first order lag function, i.e. there is no resonance characteristics about the vehicle dynamics.

Thus, the desired dynamics of vehicle yaw rate and vehicle slip angle are assumed as the first order lag function of the driver's steering wheel input, as shown by the Equation (15).

$$x_d = \begin{bmatrix} \beta_{ref} \\ \gamma_{ref} \end{bmatrix} = \begin{bmatrix} \frac{k_\beta}{1 + s\tau_\beta} G_{\beta 0} \\ \frac{k_\gamma}{1 + s\tau_\gamma} G_{\gamma 0} \end{bmatrix} \delta_s \quad (15)$$

Here, s is Laplace operator. k_β and k_γ are gain of desired slip angle and desired yaw rate from the steady state gain of each state variables, while $G_{\beta 0}$ and $G_{\gamma 0}$ are steady state gain of the slip angle and the yaw rate respectively from the steering wheel input angle δ_s . Also τ_β and τ_γ are time constant of the desired slip angle and the desired yaw rate as the first order lag function.

$G_{\beta 0}$ and $G_{\gamma 0}$ are calculated as follows. Considering the case of steady state as $x = x_0$ and without any active control, the Equation (10) becomes as bellow.

$$\dot{x}_0 = 0 = Ax_0 + E\delta_s \quad (16)$$

where

$$E = \begin{bmatrix} \frac{2K_f}{G_s MV} \\ \frac{2l_f K_f}{G_s I_z} \end{bmatrix} \quad (17)$$

Here, G_s : steering gear ratio. From the Equation (16), x_0 is obtained as follow.

$$\begin{aligned} x_0 &= -A^{-1}E\delta_s \\ &= -\frac{MI_z V^2}{4K_f K_r (l_f + l_r)^2 - 2MV^2 (l_f K_f - l_r K_r)} \\ &\quad \times \begin{bmatrix} -\frac{4K_f K_r l_r (l_f + l_r)}{MI_z V^2} + \frac{2l_f K_f}{I_z} \\ \frac{-4K_f K_r (l_f + l_r)}{MI_z V} \end{bmatrix} \frac{1}{G_s} \delta_s \end{aligned} \quad (18)$$

Thus, $G_{\beta 0}$ and $G_{\gamma 0}$ can be calculated as the following equation.

$$\begin{aligned} \begin{bmatrix} G_{\beta 0} \\ G_{\gamma 0} \end{bmatrix} &= -\frac{MI_z V^2}{4K_f K_r (l_f + l_r)^2 - 2MV^2 (l_f K_f - l_r K_r)} \\ &\quad \times \begin{bmatrix} -\frac{4K_f K_r l_r (l_f + l_r)}{MI_z V^2} + \frac{2l_f K_f}{I_z} \\ \frac{-4K_f K_r (l_f + l_r)}{MI_z V} \end{bmatrix} \frac{1}{G_s} \end{aligned} \quad (19)$$

The model matching control can be derived as below. The state space form of the desired dynamics can be written as below from the Equation (15).

$$\dot{x}_d = A_d x_d + E_d \delta_s \quad (20)$$

Here,

$$A_d = \begin{bmatrix} -\frac{1}{\tau_\beta} & 0 \\ 0 & -\frac{1}{\tau_\gamma} \end{bmatrix} \text{ and } E_d = \begin{bmatrix} \frac{k_\beta}{\tau_\beta} G_{\beta 0} \\ \frac{k_\gamma}{\tau_\gamma} G_{\gamma 0} \end{bmatrix}.$$

Assume the error between actual state variables and desired state variables as $e = x - x_d$. A dynamic state equation of this error variable can be obtained as below by subtracting Equation (20) from Equation (10).

$$\dot{e} = Ae + Bu + (A - A_d)x_d - E_d \delta_s \quad (21)$$

In the previous research (Hirano, 2016b), the simulation results of full vehicle model showed that there were some steady state deviation remained after stabilizing the vehicle motion. Thus, it is suggested that augmenting the state space equation of the Equation (21) to include the integral of the state variables is necessary. By assuming a new state vector of the error vector as

$$\hat{e} = \begin{bmatrix} \beta - \beta_{ref} \\ \gamma - \gamma_{ref} \\ \int (\beta - \beta_{ref}) dt \\ \int (\gamma - \gamma_{ref}) dt \end{bmatrix} \quad (22)$$

the Equation (21) is augmented as below.

$$\begin{aligned} \dot{\hat{e}} &= \begin{bmatrix} A & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \hat{e} + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} u \\ &+ \begin{bmatrix} (A - A_d) \\ 0 & 0 \\ 0 & 0 \end{bmatrix} x_d - \begin{bmatrix} E_d \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \delta_s \\ &\stackrel{\text{def}}{=} \hat{A}\hat{e} + \hat{B}u + \hat{A}_{de}x_d - \hat{E}_{de}\delta_s \end{aligned} \quad (23)$$

Here,

$$\begin{aligned} \hat{A} &\stackrel{\text{def}}{=} \begin{bmatrix} A & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} A & O_2 \\ I_2 & O_2 \end{bmatrix} \\ \hat{B} &\stackrel{\text{def}}{=} \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} B \\ O_2 \end{bmatrix} \\ \hat{A}_{de} &\stackrel{\text{def}}{=} \begin{bmatrix} (A - A_d) \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} (A - A_d) \\ O_2 \end{bmatrix} \\ \hat{E}_{de} &\stackrel{\text{def}}{=} \begin{bmatrix} E_d \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} E_d \\ O_2 \end{bmatrix} \end{aligned}$$

where O_2 is the zero matrix of order 2 and I_2 is the unit matrix of order 2.

Let's assume a virtual control input of the augmented state space equation of the error vector as follow.

$$\hat{U} \stackrel{\text{def}}{=} \hat{B}u + \hat{A}_{de}x_d - \hat{E}_{de}\delta_s \quad (24)$$

Then the Equation (23) becomes as follow.

$$\dot{\hat{e}} = \hat{A}\hat{e} + I_4\hat{U} \quad (25)$$

Here I_4 is the unit matrix of order 4. For the linear system of the Equation (25), we can design a feedback control

$$\hat{U} = \hat{K}\hat{e} \quad (26)$$

by linear control theory. In this paper, the feedback gain \hat{K} is calculated by using LQR (Linear Quadratic Regulator) so that the following criteria function is minimized.

$$J = \int_0^\infty (\hat{e}^T Q \hat{e} + \hat{U}^T R \hat{U}) dt$$

Here, Q and R are weight matrixes of order 4. Please note that \hat{K} is dependent on vehicle velocity because the matrix A included in \hat{A} is velocity dependent as shown in Equation (13). **Figure 2** shows some

elements of feedback gain \hat{K} as a function of the vehicle velocity V .

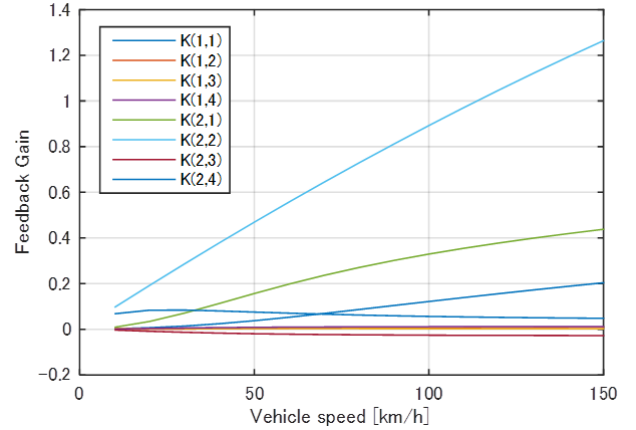


Figure 2. Plot of LQR gain according to vehicle speed

Finally the actual control input u is calculated by the Equations (24) and (26) as below. At first the Equation (24) is rewritten as below.

$$\begin{bmatrix} \hat{K}_1 \\ \hat{K}_2 \end{bmatrix} \hat{e} = \begin{bmatrix} Bu + (A - A_d)x_d - E_d\delta \\ S \end{bmatrix} \quad (27)$$

Here, \hat{K}_1 is the upper part of size (2×4) of the gain matrix \hat{K} and \hat{K}_2 is the lower part of size (2×4) of the gain matrix \hat{K} . S is an unknown variable.

From the upper part of the Equation (27) the actual control input u can be calculated as below.

$$u = B^{-1} \{-\hat{K}_1\hat{e} - (A - A_d)x_d + E_d\delta_s\} \quad (28)$$

Here, B^{-1} is the inverse matrix of B . ($B^{-1}B = I_2$.)

From the Equation (28) it is understood that the control input consists of a feedback term of the augmented state error and two feedforward terms of desired state variables and also of driver's steering input.

5 Simulation Models and Results

5.1 Single Track Vehicle Model

To confirm the validity of above mentioned model matching control, simulation test based on the single track vehicle model was performed by using Modelica. First of all, we should handle time-varying linear state space system expressed by Equations (10) and (13). It was easy to describe time-varying state space system as Equations (10) and (13) by Modelica as mentioned below.

A new class of time-varying linear state space system was defined by using Modelica. For this purpose, the existing class of the linear state space system of Modelica Standard Library (MSL) was modified to release the constraint of variability of variables (i.e. by eliminating 'parameter' qualifier). The definition of the new class becomes as follow.

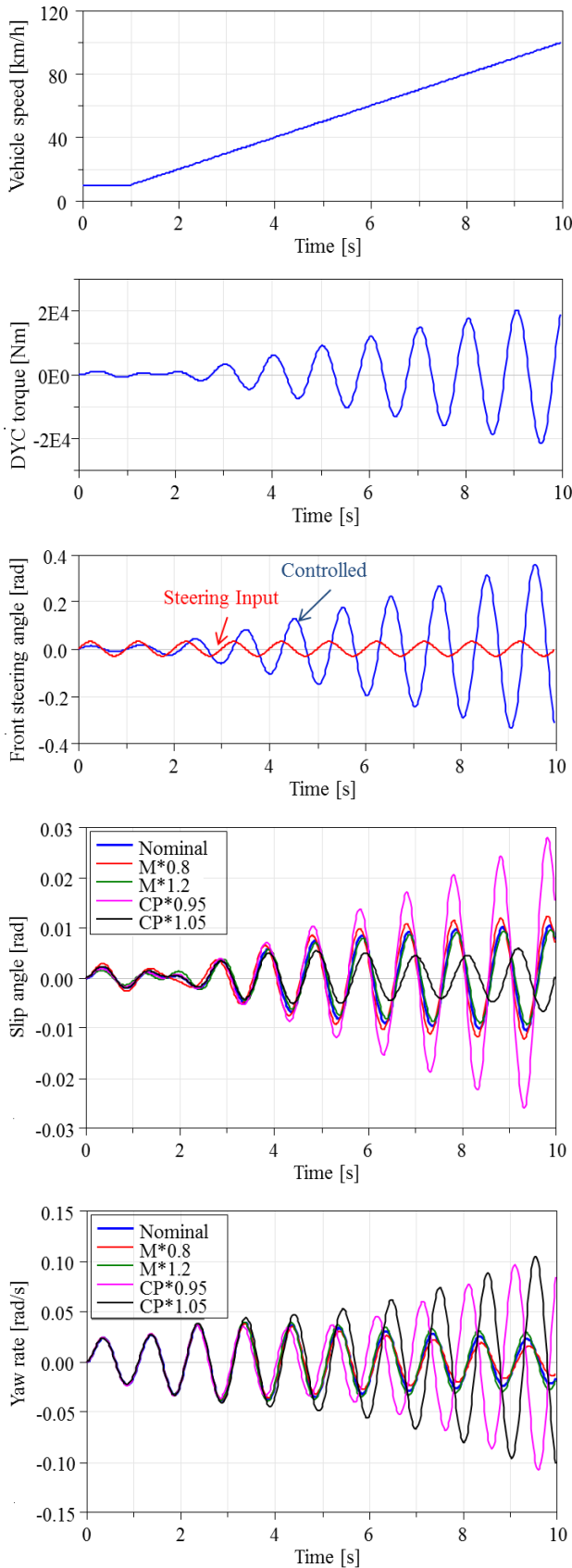


Figure 4. Plot of vehicle velocity and steering input angle

5.2 Full vehicle model

The full vehicle model as previous research (Hirano *et al.*, 2013)(Hirano *et al.*, 2014) was used for full-vehicle simulation. The model was developed based on Vehicle Dynamics Library of Modelica (Modelon, 2014) and was built as a full 3-dimensional (3D) multi-body-dynamic system (MBS) model. Component models of control systems such as TVD gearbox, electric motor and inverter were added with the full vehicle model.

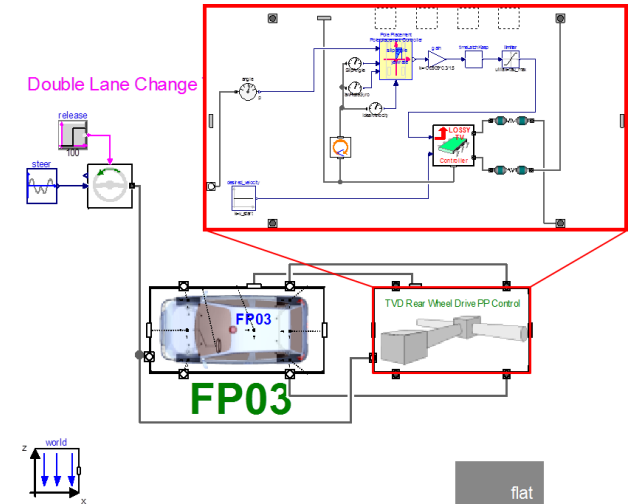


Figure 5. Structure of full vehicle test model

Figure 5 shows the top level of the model hierarchy of the full vehicle test model and also the power train model with the controller.

For the TVD gear train, a driveline structure referencing the MUTE project of the Technische Universität München (Höhn *et al.*, 2013) was selected and the TVD model was constructed using Modelica Power Train Library (DLR, 2013). **Figure 6** shows the configuration of the gear trains. Torque from the main motor is distributed equally to the left wheel and the right wheel through the differential gear. The torque distribution between the left wheel and the right wheel can be controlled by changing the torque input of the control motor.

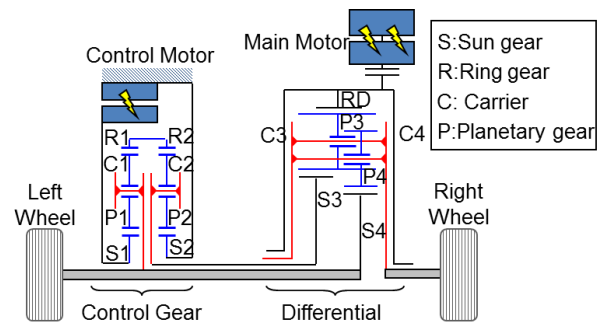


Figure 6. Torque vectoring differential (TVD) driveline

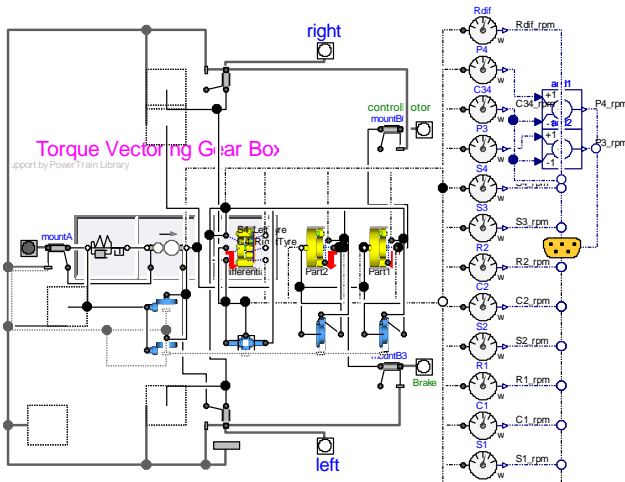


Figure 7. Modelica model of TVD gear train

Figure 7 shows a diagram of Modelica model of the torque vectoring gear train. The model is provided with elements that define the relational expression between the torque and speed of each gear engagement portion.

3D MBS model of suspension, steering and body were installed to calculate vehicle dynamics characteristics. Suspension model was constructed as an assembled model of each suspension linkage, joints and force elements such as spring, damper and bushing. Non-linear tire model based on 'Magic Formula' model (Pacejka02) was used to calculate combined lateral force and longitudinal force of each tire. Steering model considered the characteristics of viscous friction of steering gear box and steering shaft as well as steering shaft stiffness.

5.3 Results of full vehicle simulation

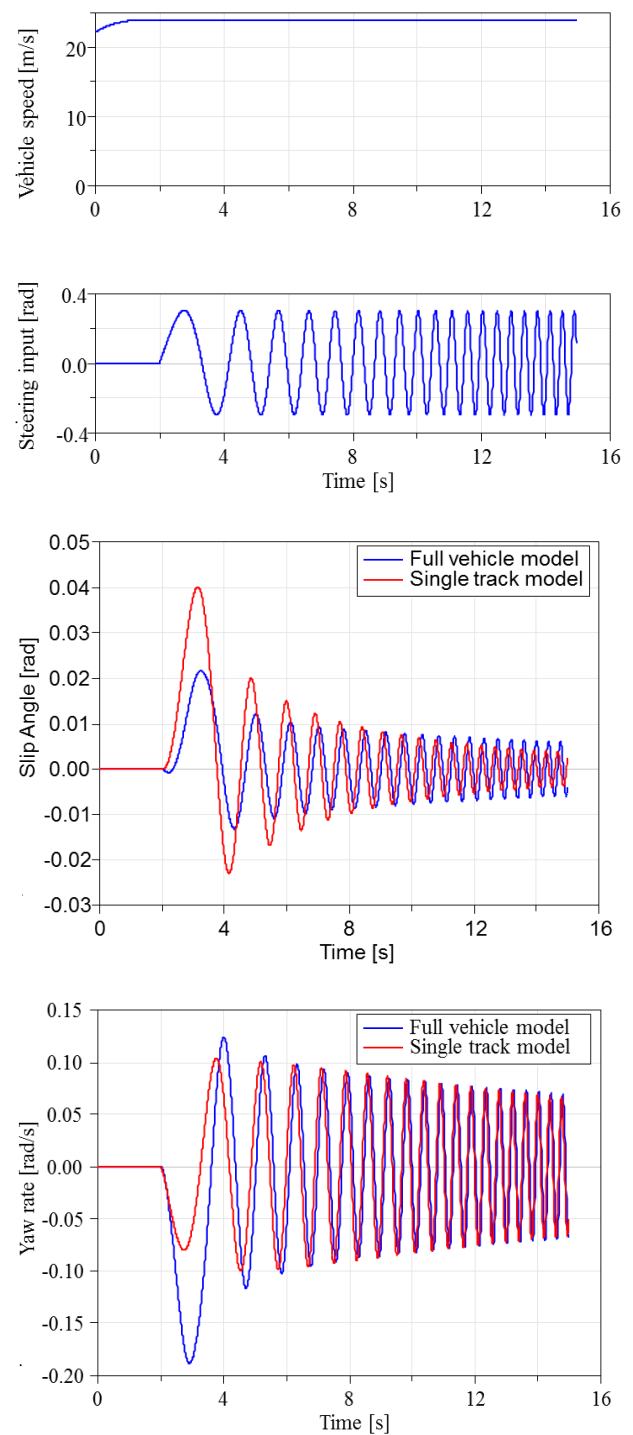


Figure 8. Comparison between full vehicle model and single track model

At first, the result of the full vehicle model and the single track model was compared in a case that no control was applied. Steering input angle was given as a sinusoidal sweep signal from 0.1 Hz to 5Hz at constant vehicle speed $V=80[\text{km/h}]$. Figure 8 shows the results of vehicle slip angle and yaw rate response. It is shown that some difference exists between the single

track model and the full vehicle model especially in the low frequency response. The reason of this result is assumed that the approximation when used to derive the equation of the single track model was too big. Actually the Equation (2) and (3) about the Figure 1 should be

$$M \frac{d}{dt} (u \tan^{-1} \beta + u\gamma) \cong 2Y_f \cos \delta_f + 2Y_r \quad (29)$$

$$I_z \frac{d\gamma}{dt} \cong 2l_f Y_f \cos \delta_f - 2l_r Y_r + N \quad (30)$$

in precise. Also the non-linearity of the tire characteristics and effects of many losses and stiffness of mechanical parts are not considered in the single track model. This result indicates that we should be careful when designing controllers based on the single track model.

Next, a simulation emulating double lane change maneuver was performed. Though, in this case, a problem that the vehicle motion of the full vehicle model became unstable when applying the control law shown in Eq. (28). The reason was that by the default gain of the feed forward control parts, the controlled steering angle exceeded the actual physical limit and turned more than 6[rad], that is, about 360[deg]. So the compensation for the feed forward parts was applied so that the controlled front steering angle will not be so different from the steering input angle. (Actually the feed forward parts were gained by 0.1.) After this modification, the vehicle response became stable in the actual case using the full vehicle model. Also for the feedback part, we should be careful to select the value of weight matrix element when designing LQR controller. Also the weight for the steering angle control was lowered than that for the DYC torque because of the physical limit of the steering angle. These problems may be solved by modifying the controller design from LQR to MPC (Model Predictive Control) which can consider the limitation of the actuators, but there would be a conflict of calculation time of the controller in such a case.

Figure 9 shows the results of the full vehicle simulation imitating the double lane change test by open-loop driver model. Though there seems necessity of further gain tuning, the modified model matching control seems to work to let the actual state variables trace the desired variables.

Also side wind test was simulated using the full vehicle model. Figure 10 shows the results. There is a side wind of 20[m/s] while time = 2 sec to 3.5 sec when the vehicle is running at 120[km/h] with fixed steering input angle of 0[rad]. The effect of the proposed control to stabilize both slip angle and yaw rate response against the side wind was shown.

For comparison, the result of the previous research in which design of the model matching control was done without considering the integral of the error (Hirano, 2016b) is shown in Figure 11. The new

control (Figure 10) showed less steady state deviation and also better regulation of the state variables against the external disturbance as the side wind, though it is not perfect yet.

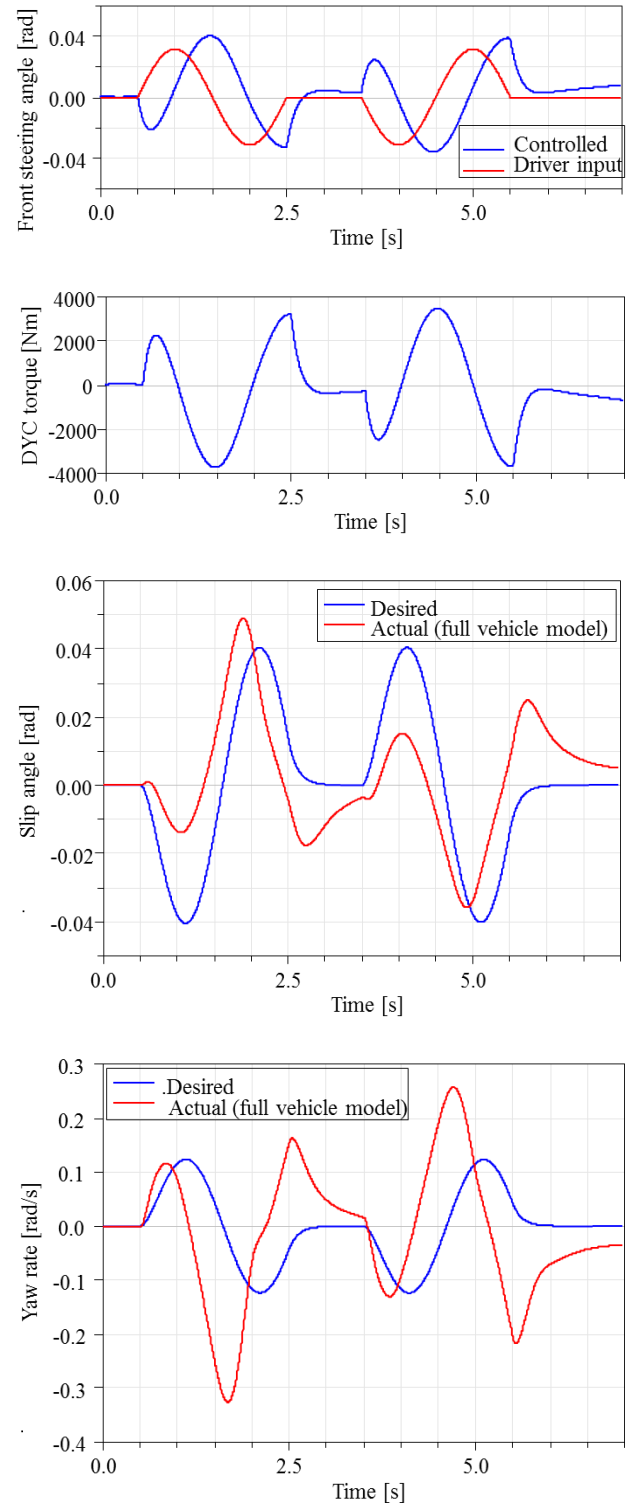


Figure 9. Full vehicle simulation result for double lane change test

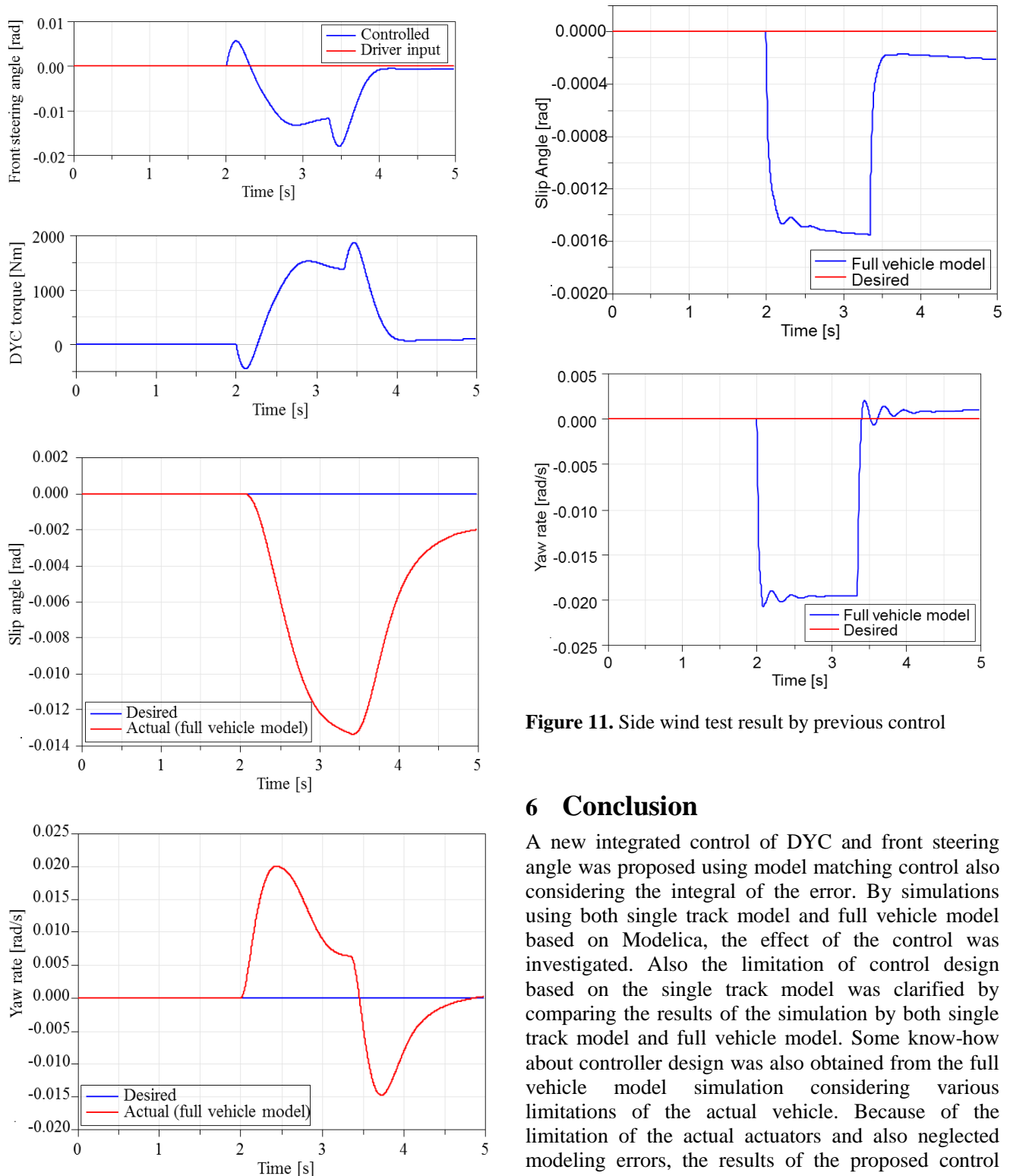


Figure 10. Full vehicle simulation result of side wind test

Figure 11. Side wind test result by previous control

6 Conclusion

A new integrated control of DYC and front steering angle was proposed using model matching control also considering the integral of the error. By simulations using both single track model and full vehicle model based on Modelica, the effect of the control was investigated. Also the limitation of control design based on the single track model was clarified by comparing the results of the simulation by both single track model and full vehicle model. Some know-how about controller design was also obtained from the full vehicle model simulation considering various limitations of the actual vehicle. Because of the limitation of the actual actuators and also neglected modeling errors, the results of the proposed control was not satisfactory.

On the other hand, Modelica was always powerful to express any kind of controllers as well as multi-physics full vehicle model. A new technique to expand Modelica model to write time-variant models was also introduced.

References

- M. Burgess, Torque vectoring. *Lotus Engineering*, 2009.
- DLR(German Aerospace Center), Power Train Library Users Guide (Version 2.1.0), 2013
- S. Efstathios, E. Velenis, and S. Longo, Model predictive torque vectoring control for electric vehicles near the limits of handling. *European .Control Conference (ECC) 2015 IEEE*, 2015.
- Y. Hirano, S. Inoue and J. Ota, Model-based Development of Future Small EVs using Modelica, *Proceedings of Modelica Conference 2014*, 2014.
- Y. Hirano, S. Inoue and J. Ota, Model Based Performance Development of a Future Small Electric Vehicle by Modelica, *Proceedings of Modelica Conference 2015*, 2015.
- Y. Hirano, Research of Model Matching Control of Torque Vectoring Differential Gear System, *Proceedings of Japanese Modelica Conference 2016*, 2016a
- Y. Hirano, Model Based Development of an Integrated Control of Front Steering and Torque Vectoring Differential Gear System, *Proceedings of SICE 2016*, 2016b
- B. Höhn et al., Torque Vectoring Driveline for Electric Vehicle, *Proceedings of the FISITA 2012 World Automotive Congress*, Vol. 191, pp. 585-593, 2013.
- Modelon, A.B., Vehicle Dynamics library Users Guide (Version 1.8), 2014

Virtual Occupant Model for Riding Comfort Simulation

Hyung Yun Choi¹ Manyong Han¹ Akinari Hirao² Hisayoshi Matsuoka²

¹Hongik University, Korea, hychoi@hongik.ac.kr, myhan@mail.hongik.ac.kr

²Nissan Motor Co. Ltd, Japan, a-hirao, hisayoshi_m@mail.nissan.co.jp

Abstract

A digital human body model as a virtual occupant surrogate for the riding comfort simulation is developed for both 1D lumped network (Modelica) and 3D mesh based (Finite Element) solutions. Since the composition of 1D and 3D versions of the human body model has a similar multibody system architecture, the kinematic responses from both solutions are almost equivalent. The models are therefore complementary, since the economic 1D models can serve effectively in design exploration and optimization, while their sophisticated 3D counterparts can serve in final design validation. The detailed modeling process and validation results against standard seat vibration excitation test are introduced in this paper, preparing the models for use in seat design.

Keywords: digital human body model, riding comfort simulation, 1D lumped network Modelica model, 3D mesh based Finite Element model, vibration excitation

1 Introduction

A virtual human body model (VHBM) is developed for quantitative and objective assessments of the riding comfort design of vehicles. The VHBM has biofidelic dynamic characteristics of human occupants during the vehicle ride. The anthropometry of the finite element human body model is based on a previous study [Kim 2007] and represents a standard North American 50th %tile male from the SizeUSA population survey 2000-03.

There have been many CAE studies to virtually simulate static and dynamic interactions between the human occupant and the vehicle seat. Montmayeur et al [Montmayeur 2004] used a human body model to predict the sitting pressure distribution and head-to-seat vertical transmissibility. There was a good correlation against the experiment but the position of the human body model was limited to upright erect sitting without a back support. Choi et al [Choi 2008] used a human body model to evaluate lumbar support design. They investigated postural changes of sitting occupants such as seat back pressure distribution and lordotic curvatures of the lumbar spine with the different configurations of lumbar support, and the prominence and height of the support. Yamada et al [Yamada 2016] used the THUMS model (Total Human Model for Safety, version 5) to investigate the influence of muscular strength and seat reaction force on occupant kinematics in single lane

change maneuvers. It was found that some skeletal muscles in the THUMS model were needed to activate, e.g., 350N by abdominal oblique muscles to resist against 1.0G lateral vehicle motion. Han et al [Han 2016] presented an efficient way to model muscle forces of vehicle occupants as they maintain the postural stability during the ride. The active joint torque controlled by a proportional integral derivative (PID) closed loop was introduced at the elbow joint to simulate voluntary and reflexive response of the human subjects.

The main focus of the VHBM in this paper is showing its capability of not sophisticated but quite effective representations of active skeletal muscle forces by developing PID-controlled active torques at articulated joints. It is hypothesized that vehicle occupants brace their limbs and trunk to maintain the initial upright (comfortable) sitting posture. Accordingly, the VHBM model autonomously develops the skeletal muscle forces, i.e. active torques, at articulated joints against the external perturbations.

The VHBM was built for two kinds of solution, 1D lumped network (Modelica) and 3D mesh based (Finite Element) solutions. The 1D lumped network solution is very effective for the multi physical system with many controllers. It is also suitable for the calculation of large numbers of variants. On the other hand, the 3D mesh based solution with its fine geometry and material properties can provide detailed interactions with the neighboring structure, the vehicle seat in our case. However the 3D solution requires a great deal of computing power due to its high level of modeling complexity. The topological composition of the 1D version of the human body model is the same as that of the 3D version since they are both based on a multibody system with PID controllers. The outcomes of two different solutions, e.g., dynamic response of human body model to external loadings, are thus almost identical. Therefore, the use of the 1D model to calibrate intrinsic and extrinsic parameters of the human body model, such as joint properties and weighting factors (gains) in the PID controllers, is quite beneficial. An optimization process is normally adopted for determining those modeling parameters, which becomes an extremely lengthy task when a 3D model is used. In case with the Genetic Algorithm at the optimization process, a number of around several hundred model runs (15 generations X 50 populations) are often necessary for the convergent result. However the 3D human body model is also necessary at the practical application stage

because it produces many informative outcomes at the riding comfort simulation, e.g., dynamic sitting pressure distribution.

2 Human Body Modeling

2.1 Multibody Modeling

The whole human body are segmented into 15 body regions (see Fig. 1), i.e., head, neck, upper/center/lower trunks, left and right upper/lower arms, left and right upper/lower legs, and left and right foot. And they are articulated by 14 joints as listed in Table 1. The dynamic properties of the 15 body segments modeled as rigid bodies: mass, center of gravity, and 2nd mass moment of inertia of each body region, are calculated by GEBOD program [Cheng 1996]. The averaged values of 32 body dimensions measured from 10 test subjects of this study are used as input parameters at the GEBOD calculation. Kinematic joint elements are used for the articulation of the 15 body segments of which the main biomechanical characteristics are defined by stiffness and damping coefficients. The kinematic joint element describes the passive characteristics of the human joint, together with, the active torques. Assuming that a co-contraction of agonist and antagonist muscles stiffens the joint

articulation, the damping coefficients of the passive kinematic joint element are adjusted for the different levels of pre-tensions, which is considered as a major mechanism of voluntary muscle activation. Meanwhile, the spring constant of the kinematic joint element represents the inter-subject variation of the muscular structure, e.g., male versus female, younger versus older.

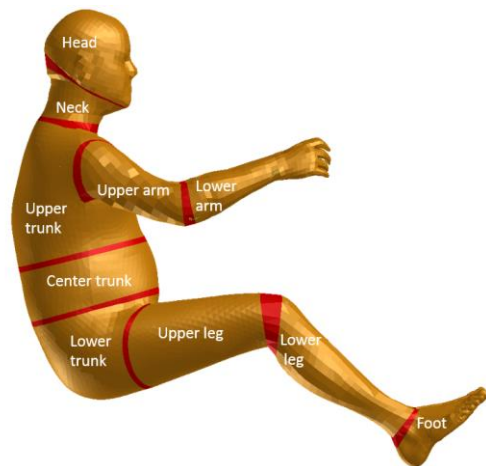


Figure 1. Whole body model segmented by 15 rigid bodies and 14 articulated joints (in 3D FE model view)

Table 1. Fourteen articulated joints with their anatomical positions

#	Articulated joint	DOF	Anatomical position
1	Head-neck	3	OC joint
2	Neck-Upper trunk	3	sC7/T1
3	Upper-Center trunk	3	T12/L1
4	Center-Lower trunk	3	L5/S1
5, 6	Upper trunk-arm, R, L	3	Right, Left shoulders
7, 8	Upper-Lower arm, R, L	1	Right, Left elbows
9, 10	Lower trunk-leg, R, L	3	Right, Left hip joints
11, 12	Upper-Lower leg, R, L	1	Right, Left Knees
13, 14	Lower leg-foot, R, L	3	Right, Left ankle

2.2 Wobbling Masses

The internal organs in the ventral body cavity, such as lungs, heart, stomach, intestines, liver, spleen, kidneys, and bladder, are classified by their anatomical locations, either above or below the diaphragm, i.e., in thoracic and abdominal/pelvic cavities, respectively. The organs in a same or adjacent cavity are grouped together as a single lumped mass in the virtual occupant model, Fig. 2. The wobbling behavior of the internal organs at whole body vibration is thus characterized by two lumped masses, one for the thoracic cavity and the other one for the abdominal and pelvic cavities. Each wobbling mass was estimated respectively as 5kg and 10kg for the thoracic and abdominal/pelvic masses.

All sides of the two wobbling masses are tied by elastic spring elements to the inner surfaces of the

thoracic and abdominal cavities. There are also elastic spring elements between two wobbling masses connecting the bottom side of thoracic mass and the top side of abdominal/pelvic mass. The mechanical characteristics of spring elements such as stiffness and damping coefficients were assigned to reproduce the biofidelic dynamic behavior of the two wobbling masses.

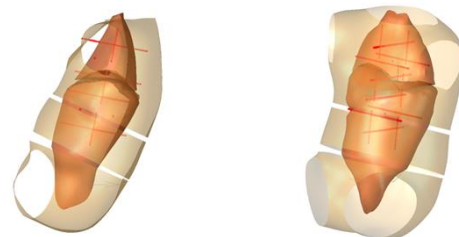


Figure 2. Wobbling masses in the trunk (in 1D SimulationX model view)

2.3 Active Joint Torque with PID Close Loop Control

Voluntary and reflexive muscle activation of a vehicle occupant is modeled by active joint elements at each anatomical joint position (e.g., shoulder, knee, spine, etc.). There are two basic elements at each joint, i.e., the passive kinematic joint element and the torque actuator. Contrastively to voluntary activation of individual muscles, i.e., the pre-tension and consequent stiffening of the articulated joint represented by the passive kinematic joint element, a vestibular reflexive muscle activation for the posture stabilization is modeled by the introduction of active torques with PID closed loop

control. As an example, the modeling of the head-neck joint (C0-C1) is shown in Fig. 3. The active torque, the control signal, is a sum of proportional, integral, and derivative terms between the current and the reference (initial) joint angles. The gain values at the PID control determine the rates of torque generation. Faster torque generation with larger gain values stands for the pre-recognition of the upcoming external perturbation. Each term at PID can be adjusted to calibrate the rate of muscle recruitment for fine control of the reflexive response of the human occupant. Authors of this paper showed a successful application of the proposing active joint modeling with the elbow reacting to the jerk loading [Choi 2016, Han 2016].

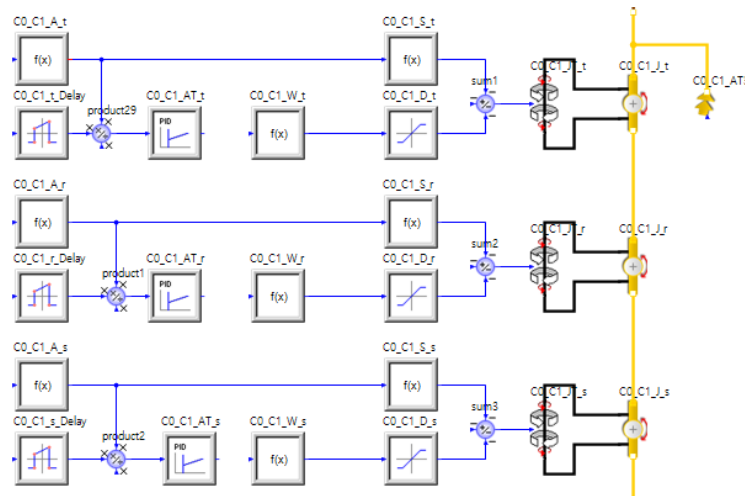


Figure 3. Block diagram of head-neck joint (C0-C1) with active torque using PID close loop control (in 1D model view)

2.4 Finite element model vs. Modelica model

The composition of 1D lumped network (Modelica) version and 3D mesh based (Finite Element) version of the whole body model in Fig. 4 has similar multibody system architecture. The same segmental dynamic properties, joint characteristics, and PID control gains, are assigned to both 1D and 3D models. Consequently, the outcomes such as dynamic responses from both models to external loadings are almost identical. So, utilizing the computational efficiency of 1D Modelica model and solution instead of 3D FE model, the calibration process of the intrinsic and extrinsic modeling parameters become much faster.

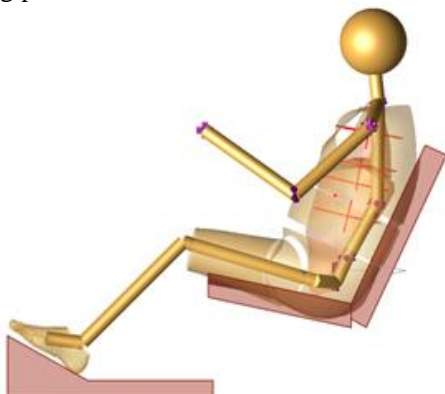


Figure 4. 1D lumped network (top) and 3D finite element (bottom) whole body human models.

The one of lacking feature at the 1D lumped network solution compared to the 3D mesh based solution is the pragmatic sliding contact algorithm to handle the nonlinear boundary conditions. In the context of riding comfort simulation, the main application of the occupant model, a dynamic interaction between the occupant and the vehicle seat, is the most relevant case. In general, a riding comfort design of the vehicle seat is responsible for the quality of static support at a sitting posture and the dynamic isolation against floor level

vibration. The measurement of the sitting body pressure distribution and its pattern analysis provide the design assessment of the static support, while the body regional transmissibility characterizes the dynamic isolation of the excitation vibration. There are model libraries available at SimulationX to handle the sliding contact between objects, such as polygon-polygon contact. The polygon-polygon contact library computes the penalty contact force based on the amount of overlapping depth between two-dimensional cross sectional outline polygons of objects. This is a suboptimal choice at 1D solutions but an appropriate selection of cross section and contact parameters is always required to reproduce the same outcome from the sliding contact in 3D mesh based solutions. Fig. 5 shows a polygon-polygon contact definition between buttock and seat at the 1D model.

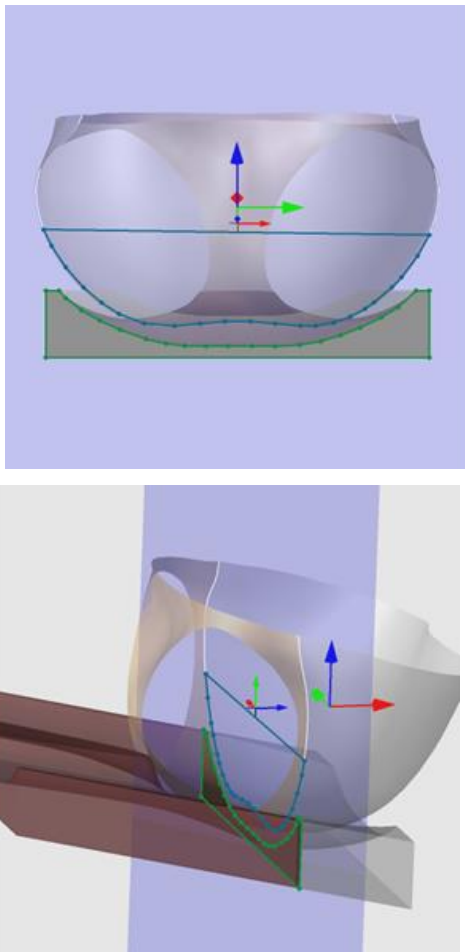


Figure 5. 2D polygon-polygon contact at 1D lumped network models.

In addition, the 3D FE whole body model has deformable flesh layers modeled by a visco-elastic Ogden rubber material of solid element at those body regions, i.e., dorsal back, buttock and thigh, which are normally in touch with vehicle seat. (Fig. 6) This deformable flesh layer can simulate the precise distribution of dynamic sitting pressure, which is hardly obtainable from the 1D lumped network solution.

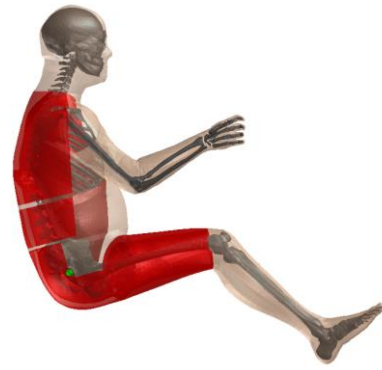


Figure 6. Deformable flesh layer in 3D FE model

The comparison of computation times between three models, two 3D finite element models with and without flesh layer and 1D Modelica model is listed in Table 2.

Table 2. Simulation time of a loading case (X direction excitation, 5Hz, 0.2g relaxed condition for 4sec)

Model	1 core CPU* time (sec)	8 core CPU time (sec)
3D FE with flesh	108,900 sec (30.3 hours)	14,770 sec (4.1 hours)
3D FE w/o flesh	13,880 sec (3.9 hours)	1,980 sec (0.6 hours)
1D Modelica	1,851 sec (0.5 hours)	NA

* CPU processor: I7-4770K 3.5GHz,

3 Validation of Human Body Model against Vibration Excitation Test

3.1 Excitation Vibration Test

A total of ten male subjects with standard North American 50th %tile anatomies between 35 and 45 years old were recruited. The same selection process of test subjects is adopted from the previous study [Kim 2007]. From the statistical factor analysis of the study, six primary dimensions listed in Table 3 were chosen to represent the overall body shape and size of the target population. Based on the SizeUSA survey (2000-03), the specific ranges (average $\pm\sigma/4$) in Table 3 for 50th %tile male were assigned as the selection criteria of test subjects. The mean and standard deviation for the six primary dimensions of 10 test subjects in this study are also listed in Table 3.

Table 3. Ranges for selection and mean of primary dimensions for test subjects

Body dimension	Selection range	Mean(SD) of test subjects
Weight (kg)	81.5 - 89.9	85.9 (2.43)
Height (m)	1.759 - 1.799	1.780(0.013)
Hip Height (m)	0.870 - 0.925	0.898(0.018)

Back Waist Length(m)	0.476 - 0.548	0.527(0.013)
Bust Girth (m)	1.052 - 1.170	1.067(0.015)
Hip Girth (m)	1.000 - 1.080	1.034(0.026)

Both standing and sitting postures of all subjects are scanned in three dimensions, and 32 body dimensions of each subject are digitally measured for the estimation of dynamic properties of 15 body segments at the GEBOD calculation [Cheng 97]. Approval to conduct testing in this study with human subjects was granted by the Pusan National University Institutional Review Board (IRB, PNU IRB/2015_30_HR).

All test subjects hold a driving posture as sitting on the wood seat engraved with the skin shape of HPM-II machine (SAE J4002) which is designed to minimize the slip on the seat during the excitation. The three translational degree of freedom exciter machine (IMV i-220) was used for the test. The typical sitting posture on the exciter and the wood seat are shown in Fig. 7. The test subjects were exposed to the discrete sinusoidal vibrations in uncoupled three translational directions. Three frequencies, 3Hz, 5Hz, and 10Hz, at two amplitudes, 0.2g and 0.4g (c.f., 0.1g and 0.2g for 3Hz excitation), were respectively applied to each of the three directions, fore/aft(X), lateral(Y), and vertical(Z). The test subjects were exposed to the excitations in two

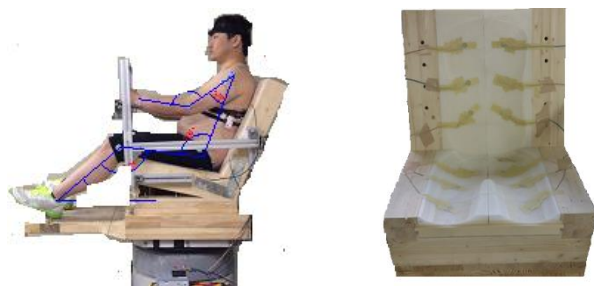


Figure 7. Typical sitting posture of test subject (left) and wood seat engraved with skin shape of HPM-II machine (SAE J4002) (right)

conscious muscle conditions of being relaxed and tensed. In the relaxed muscle condition, the test subjects were requested to strain against the gravity and the excitation just enough to sustain the initial sitting posture. In the tensed muscle condition, the test subjects were instead asked to fully brace their limbs to maximize the resistance against the excitation. There were a total of 36 cases in this excitation test, 3 excitation directions X 3 frequencies X 2 amplitudes X 2 muscle condition. The body segmental accelerations were measured in three directions at the forehead, chest and two thighs, specifically on the anterior side of the mid femur. The vibration was monitored by the accelerometer (Kistler 8310B) placed on the seat buck platform for a feedback control of the input signal by using NI-PXI8187 controller and Labview software to maintain the

frequency and amplitude of the target excitation. The excitation of each vibration case was applied for 20 seconds with a random order. The data of 16 seconds record were just used in the analysis by excluding the first and the last transient 2 seconds.

The frequency analysis of measured body regional acceleration signals was performed by taking Fast Fourier Transformation (FFT) with 99% overlap and 2-second unit time. The 1st peak head acceleration of 6 representative subset cases are listed in Table 4.

Table 4. 1st peak head acceleration of 6 representative subset cases.

Excitation cases*		Head acc. (SD), (m/s ²)		
		X	Y	Z
#1	X_5Hz_0.2g_R	0.985 (±24%)	0.270 (±59%)	2.100 (±48%)
#2	Y_5Hz_0.2g_R	0.090 (±73%)	0.254 (±63%)	0.153 (±81%)
#3	Z_Hz_0.2g_R	1.338 (±24%)	0.311 (±63%)	2.625 (±53%)
#4	Z_5Hz_0.4g_R	2.335 (±26%)	0.582 (±58%)	6.259 (±37%)
#5	Z_10Hz_0.2g_R	1.041 (±32%)	0.199 (±51%)	1.384 (±74%)
#6	X_5Hz_0.2g_T	0.934 (±44%)	0.379 (±74%)	3.761 (±39%)

*: excitation direction_frequency_amplitude_muscle condition

3.2 Exciting vibration simulation with virtual human body model

Using the 3D FE version of the virtual human body model described in Section 2, the vibration response to excitation was simulated in the following two steps:

- Step #1: Quasi-static sitting phase by gravity loading;
- Step #2: Dynamic excitation phase by discrete sinusoidal loadings.

The gravity driven sitting phase at step #1 simulates the equilibrium state of the virtual human whole body model in a driving posture. The driver at tensed muscle condition braces articulated joints at upper and lower limbs [Choi 2005]. This bracing behavior at the tensed muscle condition is reproduced by increasing the level of active joint torques. The change of initial sitting posture at tensed muscle condition from the relaxed, especially the slightly more extended elbow joint and tucked-in chin is noticeably shown in Fig. 8. The effect of bracing in the sitting posture on pressure distribution is shown in Fig. 9. The contact area to the seat back at the tensed posture shifts to the upper dorsal back while the contact area in buttock to seat cushion remains

similar to the relaxed muscle condition. The simulation time for relaxed and tensed initial postures are respectively 1,500 milliseconds and 2,000 millisecond.

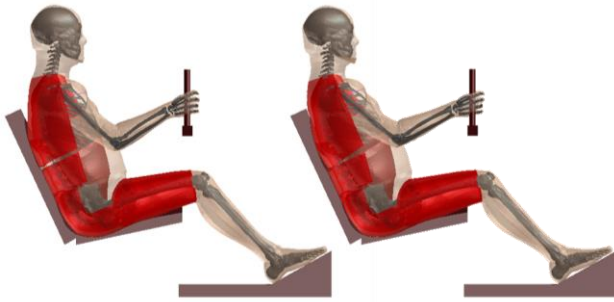


Figure 8. Comparison of simulated initial sitting posture between relaxed (left) and tensed (right) muscle conditions (3D FE model).

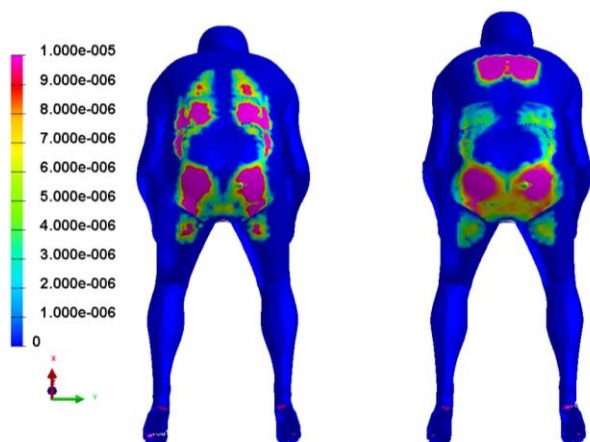


Figure 9. Comparison of simulated sitting pressure distribution between relaxed (left) and tensed (right) muscle conditions (3D FE model).

In Step #2, a discrete sinusoidal excitation loading is applied for additional 4,000 milliseconds to the equilibrated sitting virtual driver model. The same 3D FE model used for Step #1 is also utilized to calculate 6dof kinematic outcomes at the COG point of the lower trunk, i.e., time profiles of 3 translational and 3 rotational displacements. This vibration response at the lower trunk body segment is further used as an input signal of the 1D lumped network model (and the 3D FE model without deformable flesh layer) for the calibration process of intrinsic and extrinsic modeling parameters, which is to be described in detail at Section 3.3. Assuming the negligible effect of intrinsic and extrinsic parameters on the kinematics of the lower trunk body segment which is right top of the seat cushion but more to the upper body and the head, the use of the 1D model for the calibration process is far more efficient than the equivalent 3D FE model in terms of the computation time.

3.3 Calibration of modeling parameters

Two kinds of modeling parameters, intrinsic and extrinsic variables which are, respectively, independent

and dependent on external loadings, are calibrated as in the process shown in Fig. 10. The most important steps in the calibration process are preliminary and decisive optimizations. Both intrinsic and extrinsic parameters are design variables in the preliminary optimization but only extrinsic parameters in the decisive optimization. At the decisive optimization process, the intrinsic parameters adopted from case #4, the best matching case at the preliminary optimization among 6 loading cases, are used for all cases since they are supposedly independent on external loadings, the excitation direction, frequency, and amplitude. As described in Section 2.2, the mechanical characteristics of tied springs for wobbling masses belong to the intrinsic parameters. The discrete damping values in the kinematic joint element are separately assigned to relaxed (cases #1-5 in Table 4) and tensed (case#6) muscle conditions, which represent the level of bracing (co-contraction). The three gain terms at PID controllers for the active joint torque in Section 2.3 fall into the list of extrinsic parameters.

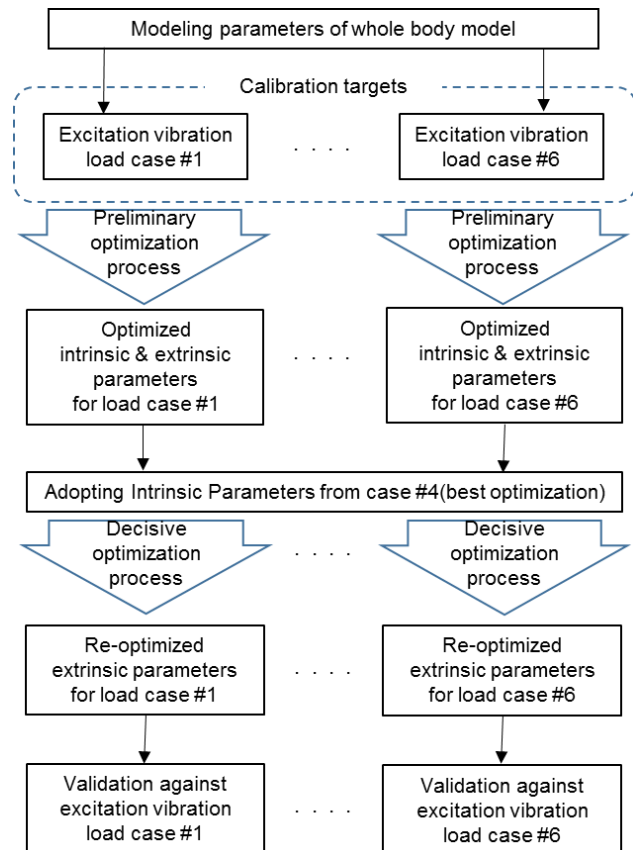


Figure 10. Calibration process for model parameters.

3.4 Optimization process

The results from two optimizations in the calibration process in Fig. 10 is listed in Tables 5 and 6. The Genetic Algorithm (GA) at PAM-OPT (www.esi-group.com) is adopted to optimize design variables, given by the intrinsic and extrinsic modeling parameters. The objective function is defined as the following equation;

$$\begin{aligned} \text{Obj. function} = & ((a_{x_{sim}} - a_{x_{test}})/a_{x_{test}})^2 + \\ & ((a_{y_{sim}} - a_{y_{test}})/a_{y_{test}})^2 + \\ & ((a_{z_{sim}} - a_{z_{test}})/a_{z_{test}})^2 + \\ & 0.1 * (\max(\theta_{c0-c1\ t})/1.5)^2 \end{aligned}$$

Where,

$a_{i_{sim}}$: 1st peak FFT acceleration in i direction from simulation.

$a_{i_{test}}$: 1st peak FFT acceleration in i direction from test

$\theta_{c0-c1\ t}$: Rotation angle of C0_C1 (head-neck) joint in t -direction (yawing).

Each iteration (generation) in the GA optimization has a number of 30 points (populations). As an exception, case #2 Y_5Hz_0.2g_R has 90 populations, three times more than other cases just for the decisive optimization process. The termination criteria is satisfying one the following two conditions;

#1 Objective function value becomes less than 0.1

#2 No change in objective function values for last 5 iterations

The objective function values in Table 5 for the preliminary optimization result is smaller than those from the decisive optimization result in Table 6 for all 6 loading cases. This becomes obvious that only extrinsic parameters are optimized as design variables while the uniform intrinsic parameters are assigned as fixed modeling variables in the decisive process.

It is also noted that relatively high objective function values associated with the lateral Y direction excitation

case (#2 in Table 5 and 6) is mainly due to the small baseline effect, i.e., the measured head acceleration is quite smaller than the other excitation directions (see Table 4).

4. Conclusion

A virtual human body model is developed to predict the riding comfort design of vehicles. The active response of the human occupant to maintain the upright sitting posture is virtually reproduced by using active joint torques with PID closed loop control. Both 1D lumped network (Modelica) and 3D mesh based (Finite Element) solutions are adopted to model the multibody system architecture of human body. The characteristics of virtual human body model is verified and validated against the excitation test with human subjects.

5. Future Study

The virtual human body model will be further validated against the subject test of angular excitations such as rolling and pitching which was performed with 6dof exciter [Choi 2017(2, 3)]. Also a riding comfort index based on ergonomic criteria is under development. Assuming the occupant is trying to develop active joint torques to maintain the upright sitting posture against external perturbations, the total amount of skeletal muscle energy together with body regional transfer function could be a quantitative and objective tool for the assessment of seat, suspension system, and chassis designs in the dynamic performance of vehicle.

Table 5. Result from the preliminary optimization process in Fig. 10

Excitation cases*		Preliminary Optimization 1 st Peak Head acceleration (g)									Obj. func. Value	Iter. No.	Iter. Term.
		X			Y			Z					
		Sim	Test (SD)	% err.	Sim	Test (SD)	% err.	Sim	Test (SD)	% err.			
#1	X_5Hz_0.2g_R	0.101	0.100 (±24%)	1%	0.026	0.028 (±59%)	-6%	0.259	0.214 (±48%)	21%	0.099	12	#1
#2	Y_5Hz_0.2g_R	0.006	0.009 (±73%)	-29%	0.044	0.026 (±63%)	68%	0.008	0.016 (±81%)	-51%	0.896	15	#2
#3	Z_5Hz_0.2g_R	0.112	0.136 (±24%)	-18%	0.034	0.032 (±63%)	7%	0.272	0.268 (±53%)	2%	0.059	7	#1
#4	Z_5Hz_0.4g_R	0.225	0.238 (±26%)	-5%	0.063	0.059 (±58%)	6%	0.564	0.638 (±37%)	-12%	0.040	11	#1
#5	Z_10Hz_0.2g_R	0.119	0.106 (±32%)	12%	0.013	0.020 (±51%)	-36%	0.212	0.141 (±74%)	51%	0.414	25	#2
#6	X_5Hz_0.2g_T	0.115	0.095 (±44%)	21%	0.040	0.039 (±74%)	3%	0.233	0.383 (±39%)	-39%	0.201	20	#2

*: excitation direction_frequency_amplitude_muscle condition

Table 6. Result from the decisive optimization process in Fig. 10

Excitation cases*		Decisive Optimization 1 st Peak Head acceleration (g)									Obj. func. Value	Iter. No.	Iter. stop
		X			Y			Z					
		Sim	Test (SD)	% err.	Sim	Test (SD)	% err.	Sim	Test (SD)	% err.			
#1	X_5Hz_0.2g_R	0.199	0.100 (±24%)	99%	0.033	0.028 (±59%)	17%	0.049	0.214 (±48%)	-77%	1.610	15	#2
#2	Y_5Hz_0.2g_R	0.025	0.009 (±73%)	177%	0.118	0.026 (±63%)	354%	0.037	0.016 (±81%)	130%	17.58	29	#2
#3	Z_5Hz_0.2g_R	0.104	0.136 (±24%)	-24%	0.034	0.032 (±63%)	6%	0.258	0.268 (±53%)	-4%	0.126	12	#2
#4	Z_5Hz_0.4g_R	0.225	0.238 (±26%)	-5%	0.063	0.059 (±58%)	6%	0.564	0.638 (±37%)	-12%	0.040	7	#1
#5	Z_10Hz_0.2g_R	0.153	0.106 (±32%)	44%	0.025	0.020 (±51%)	25%	0.345	0.141 (±74%)	144%	2.349	18	#2
#6	X_5Hz_0.2g_T	0.104	0.095 (±44%)	9%	0.039	0.039 (±74%)	-1%	0.237	0.383 (±39%)	-38%	0.167	14	#2

*: excitation direction_frequency_amplitude_muscle condition

References

- H. Cheng, L. Obergefell and A. Rizer, The development of the GEBOD program, Biomedical Engineering Conference, Proceedings of the 1996 Fifteenth Southern, 1996.
- H. Y. Choi, S.J. Sah, B. Lee, H.S. Cho, S.J. Kang, M.S. Mun, I. Lee, J. Lee, Experimental and numerical studies of muscular activations of bracing occupant. *Proc. of Enhanced Safety of Vehicle*, Washington D.C. USA, 2005
- H.Y. Choi, K. Kim, C. Kim, S. Sah, S. Kim, S. Hwang, K. Lee, J. Pyun, N. Montmayeur, I. Lee, Challenge of Lumbar Support Design Using Human Body Models. *SAE Int. J. Passeng. Cars - Mech. Syst.* 1(1), 1078-1084, 2009
- H.Y. Choi, M. Han, W. Lee, Active Elbow Joint Model. *The First Japanese Modelica Conference*, 2016
- H.Y. Choi, M. Han, J. Park, K. Yang, Air ride seat for Heavy Duty Vehicle. *12th International Modelica Conference*, submitted, 2017
- H.Y. Choi, M. Han, A. Hirao, H. Matsuoka, Occupant kinematics at vibration excitations: Part I Pure rolling and pitching vibrations. In preparation, 2017
- H.Y. Choi, M. Han, A. Hirao, H. Matsuoka, Occupant kinematics at vibration excitations: Part II Real road vibrations. In preparation, 2017
- M. Han, H.Y. Choi, Elbow joint model with active muscle force, *Journal of Mechanical Science and Technology* 30/12 5847~5853, 2016
- S. Kim, S. Hwang, K. Lee, J. Pyun, H.Y. Choi, K. Kim, S. Sah, N. Montmayeur, New Anthropometry of Human Body Models for Riding Comfort Simulation. *SAE Technical Paper* 2007-01-2457, 2007
- N. Montmayeur, C. Marca, E. Haug, H.Y. Choi, S. Sah, Experimental and Numerical Analyses of Seating Pressure Distribution Patterns. *SAE Technical Paper* 2005-01-2703, 2005
- K. Yamada, H. Motojima, Y. Kitagawa, T. Yasuki, Investigation of relations between occupant kinematics and supporting by the seat in lane change maneuvers. (In Japanese) *Proceedings of JSAE spring conference*, No.38-16, pp.941-946, 2016

A Simulation-Based Digital Twin for Model-Driven Health Monitoring and Predictive Maintenance of an Automotive Braking System

Ryan Magargle¹ Lee Johnson¹ Padmesh Mandloi¹ Peyman Davoudabadi¹ Omkar Kesarkar¹
Sivasubramani Krishnaswamy¹ John Batteh² Anand Pitchaikani²

¹ANSYS Inc., USA, {ryan.magargle, lee.johnson, padmesh.mandloi, mohammad.davoudabadi, omkar.kesarkar, sivasubramani.krishnaswamy}@ansys.com

²Modelon Inc., USA, {john.batteh, anand.pitchaikani}@modelon.com

Abstract

This paper describes a model-driven approach to support heat monitoring and predictive maintenance of an automotive braking system. This approach includes the creation of a simulation-based digital twin, or numerical model, that combines different modeling formalisms into an integrated model of the braking system that can be used for monitoring, diagnostics, and prognostics. The paper provides an overview of the basic models including Modelica models, reduced order models for various key components of the system model, and controls and sensor models. The Modelica models are implemented in the ANSYS Simplorer simulation to leverage existing modeling work and connections with other ANSYS finite element software to utilize reduced order models. The simulation results include both baseline results for the system and the results of injecting failures into the system for monitoring and predictive maintenance.

Keywords: *digital twin; electronics; electromagnetics; hydraulics; pneumatics; braking system; automotive; FEA;*

1 Introduction

Beyond the challenges of developing complex products, companies are increasingly seeking to monitor and manage the performance of those products in operation to improve safety, performance, and customer satisfaction (GE, 2016; Siemens, 2016). Model-based approaches and physics simulation are powerful components of creating a digital twin of a physical asset in operation-- a simulated replica of the asset that is used to diagnose anomalies in the performance of the asset and for predicting the state of health and remaining useful life of that asset. These insights can subsequently be used with machine learning algorithms to optimize operational downtime, trigger pre-emptive maintenance, and mitigate costly failures (Prytz, 2014). The work shows multi-domain system simulation modeling that can be used with a

variety of machine learning analytics engines, such as PTC Thingworx (PTC, 2016), which are not discussed in detail here.

The automotive industry produces millions of vehicles operating in diverse conditions and require periodic maintenance to replace worn components and address faulty conditions. Current vehicle health management practices rely heavily on data science, which has become quite powerful (Holland, 2010); however, the role of engineering and physics in these practices is limited and are included only in the form of simplified relations, material data, etc. This approach therefore has limited the applicability of health management systems to diagnostics and managed maintenance for a few automotive components and systems. The need for higher value capabilities like prognostics for critical components and systems, e.g. engine, exhaust aftertreatment, and safety, are driving the evolution of vehicle health management.

Digital twins offer automotive manufacturers an enhanced ability to diagnose anomalous conditions and predict remaining useful life of degradable components, thereby improving owner safety and satisfaction.

An approach of combining physics-based modeling techniques (0-D, 1-D, 3-D) at the system level is applied to create a digital twin for predicting brake pad wear in a conventional passenger vehicle. Versus relying solely on physical measurements, a simulation-based approach produces a high-fidelity model that can be used to predict brake pad wear, given a set of operating conditions. Further, the physics-based models are subjected to failure modes that can produce abnormal brake pad wear and unsafe conditions, and simulated to observe the sensor signatures that will subsequently aid in improving the diagnosis and mitigation of unsafe or undesirable conditions in the vehicle.

2 Modeling Overview

This section of the paper provides an overview of the integrated braking system model and the individual model components.

2.1 Integrated Model

The integrated braking system model in Simplorer (ANSYS, 2016) with all subcomponents is shown in **Figure 1**. Simplorer offers a system modeling and integration platform that supports multiple modeling formalisms, including Modelica based on the OPTIMICA Compiler (Modelon AB, 2016) from Modelon, system models using other formalisms like VHDL-AMS, reduced order models, and controls. The power converter is on the left, followed by the electromagnetic solenoid actuator, pneumatic and hydraulic braking system, and vehicle dynamics (including the speed sensor), and brake wear model. The controller is on the top, providing feedback from the measured speed and slip output to the command signal input to the power converter.

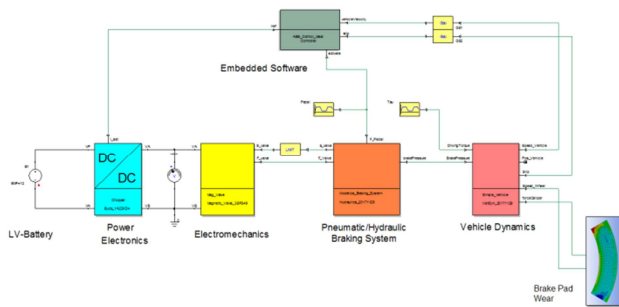


Figure 1. The full system schematic including electrical, pneumatic, hydraulic, mechanical and control logic components.

2.2 Reduced Order Modeling (ROM)

Several reduced order models generated from detailed 3-D simulations are used in this system simulation to capture component effects that might be difficult to describe with closed form solutions. The electromagnetics models of the ABS valve solenoid actuator, the magnetic wheel speed sensor, and the mechanical brake wear are all based on finite element numerical models to capture the relevant nonlinear physics.

The basis of all three reduced order models is a lookup table based on the numerical model outputs versus specified input variables. The electromagnetic solenoid actuator uses the lookup table to capture the dependence of the magnetic flux linkage vs current and magnetic force versus displacement (Woodson, 1968). The magnetic wheel speed sensor has a lookup table that represents the angular displacement of the magnetic fields on the sensor surface, in degrees, versus the position of the wheel sensor. The brake pad wear model lookup table represents the wear rate, in

mm, of the pad surface vs pad normal pressure and wheel speed.

The following sections will describe all of the major sub-component models and reduced order modeling implementation from each of the finite element numerical models.

2.3 Model Components

This section of the paper details the physical, control system, and sensor components that comprise the braking system model.

2.3.1 Electronics

To drive the electromagnetic solenoid actuator, a DC/DC buck converter, **Figure 2**, is used to drop the 12V DC supply to a level that will allow a current to flow as determined by the controller.

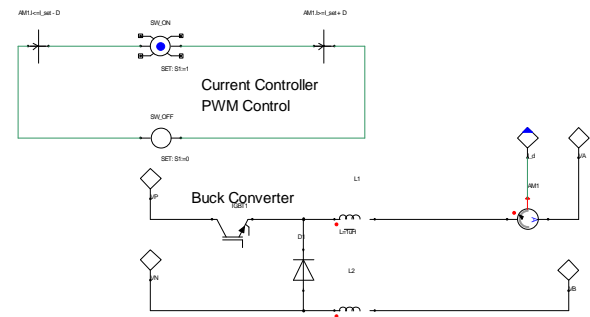


Figure 2. Electrical circuit representation of buck converter with setpoint hysteresis controller state-diagram.

The buck converter has a hysteresis current controller built-in using the state transition elements that control the MOSFET switch. The current controller opens the switch when the current exceeds the set point of the ABS controller by a user-defined threshold, 0.2A in this case, and closes the switch when the current falls below the set point.

In the system schematic, the converter is placed in a sub-circuit, as shown in **Figure 3**. The graph in **Figure 3** shows the output of the buck converter alone driving an inductive 5-ohm load with a 1A set point and 0.2A threshold without any filtering capacitance.

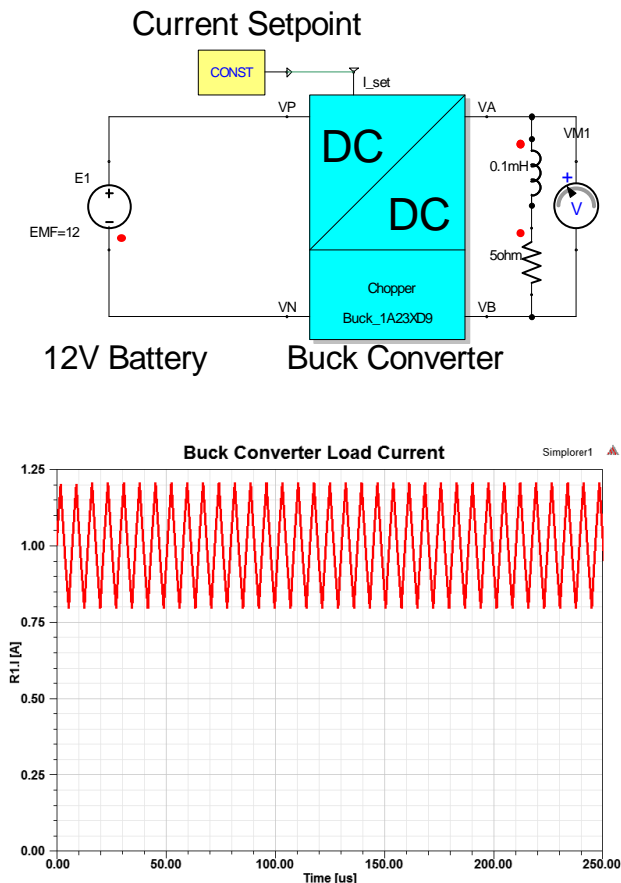


Figure 3. Power converter subcircuit output current results for 5 ohm, 0.1 mH load.

2.3.2 Electromagnetics

The DC/DC power converter is used to drive the electromagnetic solenoid actuator that the ABS controller uses to bypass brake actuator and vary braking pressure. To calculate the force generated by the current flowing through the solenoid coil, including any nonlinear effects from the steel and airgap displacement, not amenable to closed form solution, a 2-D axi-symmetric magnetostatic model is created in Maxwell2D (ANSYS, 2016), as shown in **Figure 4**. Maxwell2D uses the finite element method to calculate the magnetic field (Chari, 1977), as seen in **Figure 5**, and uses the virtual work method (Fu, 2004) to calculate the force on the moving armature. The winding is shown as a solid object, but it represents 400 turns of copper wire in this model.

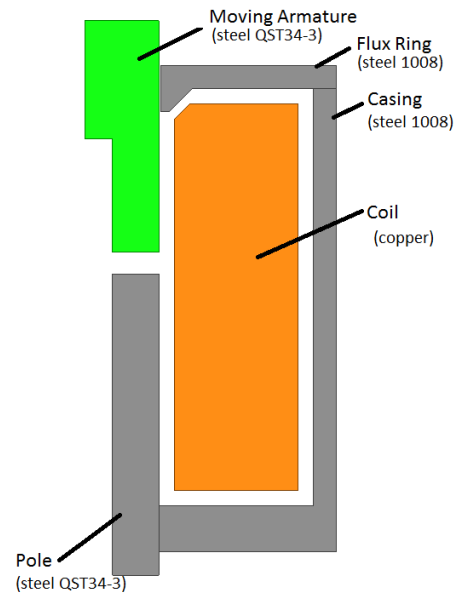


Figure 4. 2-D model of electromagnetic solenoid actuator

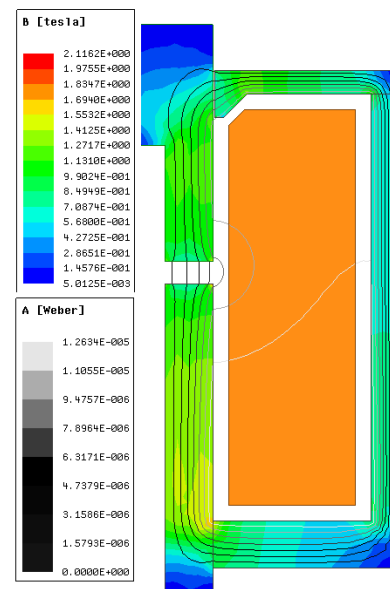


Figure 5. Magnetic flux density and field lines within the solenoid for a DC current excitation of 1.8 Amps through 400 turns.

The airgap and current are parametrically varied and the force and inductance are calculated to create a table of results, as seen in **Figure 6**.

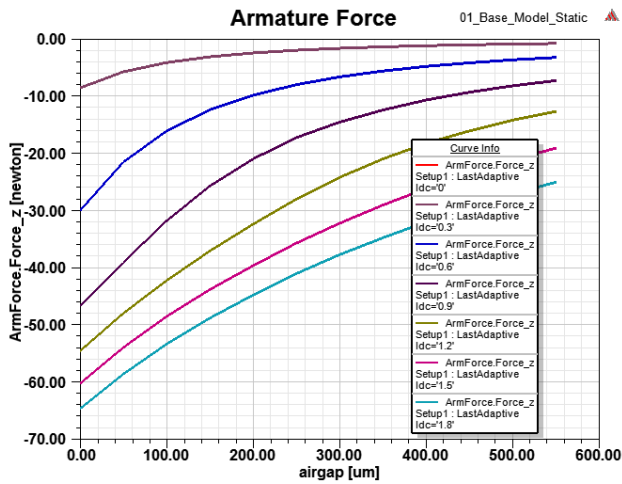


Figure 6. Force vs airgap curves for different current excitations.

The result of the parametric sweep creates a multidimensional lookup table of current, flux linkage, displacement, and force. Maxwell automatically creates a circuit element that uses dependent sources and a lookup table to relate the electrical input energy and mechanical output energy, where losses can be lumped and made external to the component, as in **Figure 7** (Woodson, 1968).

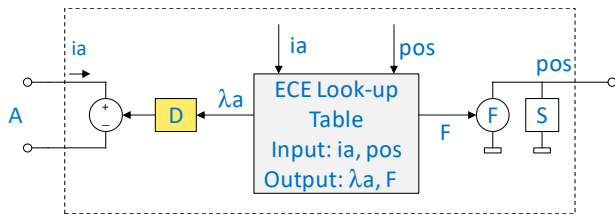


Figure 7. Equivalent circuit of electrical solenoid actuator using the lookup table results from the finite element simulation.

Together with the rest of the system circuit, the electrical actuator is implemented with an icon, containing all of the complexity of **Figure 7**. The circuit model is connected with mechanical elements, such as the restoration spring and mechanical damping, as seen in **Figure 8**.

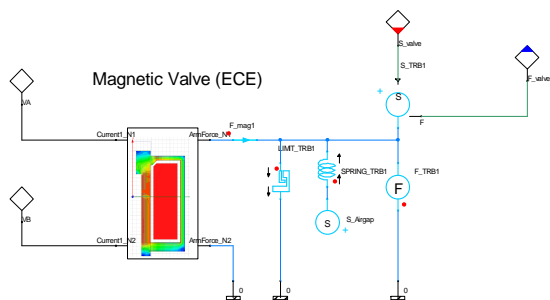


Figure 8. System model implementation of actuator with mechanical elements such as spring and mechanical damping connected externally.

2.3.3 Brake pad

A detailed 3-D model consisting of brake rotor, hub, and pads was constructed to predict brake pad wear as a function of rotor velocity and pressure applied to the brake pads, shown in **Figure 9** as a meshed geometry simulated using nonlinear structural finite element analysis (FEA) within ANSYS Mechanical (ANSYS, 2016).

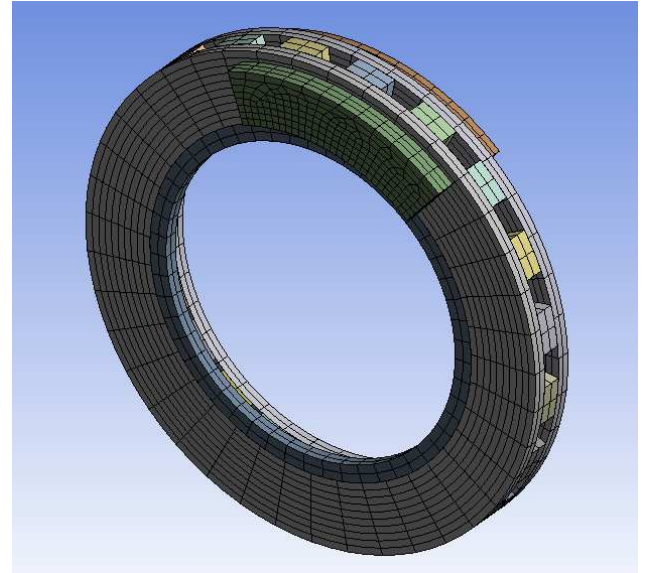


Figure 9. Meshed 3-D geometry of the rotor discs, rotor vents, hub, and pad assembly.

Brake pad wear is calculated as part of the FEA solution using the Archard Wear Model (Archard, 1980), shown in generalized form in **Figure 10**.

$$\text{Rate of Wear} \quad \frac{dW}{dt} = \frac{k p^n v_s^m}{H} \mathbf{n}$$

Contact pressure Material Hardness Sliding velocity

Surface inward normal

Figure 10. Generalized Archard Wear Model used by ANSYS Mechanical.

During the FEA simulation, constant pressure is applied on the outer faces of the two brake pads to interact with the rotor, spinning at a constant velocity. Boundary conditions constrain brake pad movement in the direction normal to the rotor, whereas in more detailed studies the motion would be determined by the brake calipers and guide pins. The simulation produces a wear rate of the brake pads as a function of the applied pressure and rotor velocity, shown in **Figure 11**.

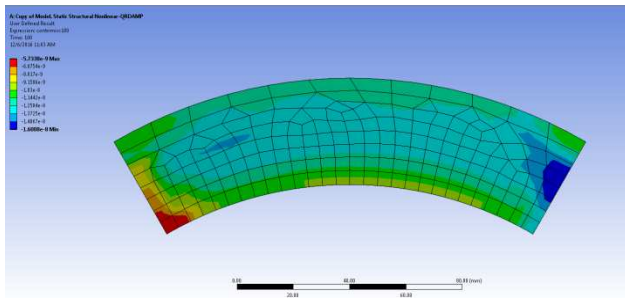


Figure 11. Wear rate on the brake pad for a constant applied pressure and rotor velocity.

Using distributed computing, the FEA simulation was performed for a number of combinations of pressure applied on the brake pads and rotor velocity to produce a response surface of wear rate at a selected location on the surface of the brake pad, illustrated in **Figure 12**. This response surface model was then encapsulated as a Functional Mock-up Unit (FMU) and connected to the speed and force inputs produced by the vehicle dynamics and ABS subsystems in the Simplorer system model. The output of this model can be integrated, using a standard integration block to measure accumulated wear.

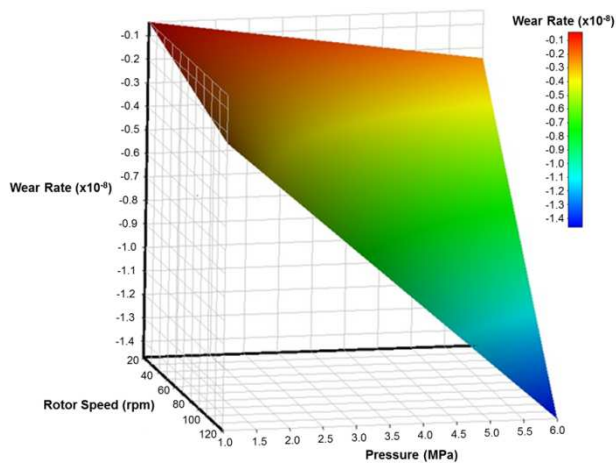


Figure 12. Response surface model of brake pad wear versus brake pressure and rotor velocity.

2.3.4 Braking System

Leveraging the latest capability in Simplorer for modeling with Modelica, the braking system is modeled natively in Modelica as a pneumatic and hydraulic system using the Pneumatics Library and Hydraulics Library (Modelon AB, 2016). The model shown in **Figure 13** consists of a pneumatic system for the brake booster (a) and the hydraulic system (b) for the brake actuation.

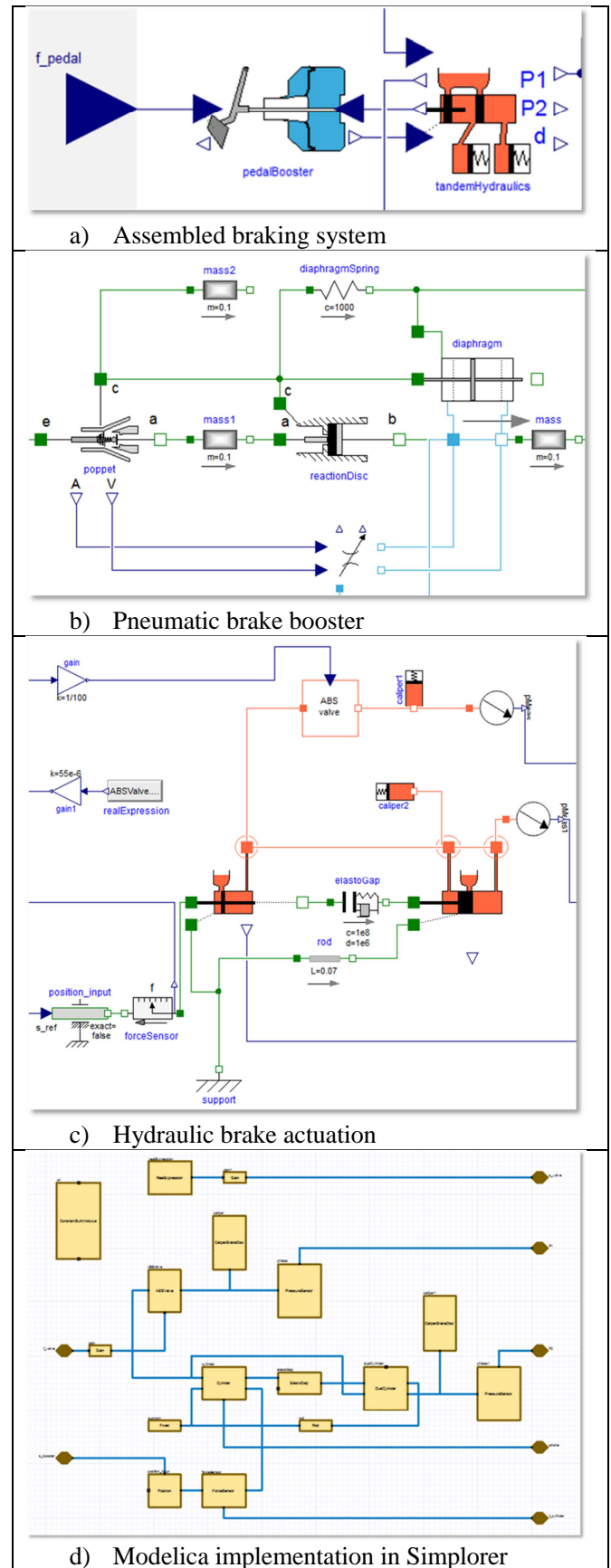


Figure 13. Pneumatic and hydraulic braking system model

Based on the brake pedal input from the driver, the models calculate the resulting caliper pressure that is provided to the vehicle model for use in calculation of the brake torque. The ABS valve that modulates the brake pressure based on the electromagnetic force actuation is shown in **Figure 13b**.

2.3.5 Vehicle Dynamics

The vehicle is modeled in Modelica with basic longitudinal dynamics considering a lumped vehicle mass and a simple single wheel approach. The Modelica model in Simplorer is shown in **Figure 14**. The tire dynamics include the effects of slip via the Pacejka magic tire formula (Pacejka, 1993). The vehicle model takes the drive torque and the brake caliper pressure and calculates the resulting vehicle response and wheel conditions, including wheel speed and slip. The wheel speed is passed to the model of the wheel speed sensor for slip estimation, and the vehicle speed is provided to the controller.

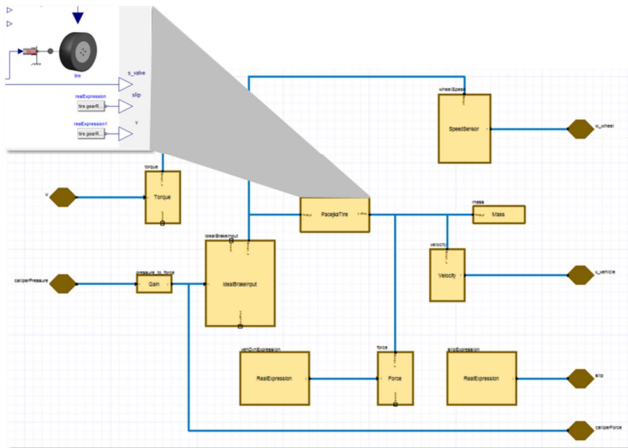


Figure 14. Vehicle dynamics model implemented in Modelica in Simplorer

While the vehicle dynamics considered are fairly simple, the native Modelica capability in Simplorer allows for more complex models to be included to capture higher frequency dynamics. For example, models from the Vehicle Dynamics Library (Modelon AB, 2016) can be integrated to capture higher fidelity chassis dynamics and also more detailed tire dynamics and tire/ground interactions.

2.3.6 Controls

Shown in **Figure 15**, control of the ABS valve is implemented as a state machine in ANSYS SCADE Suite, a model-based environment for developing embedded software, often for applications where safety is critical (ANSYS, 2016).

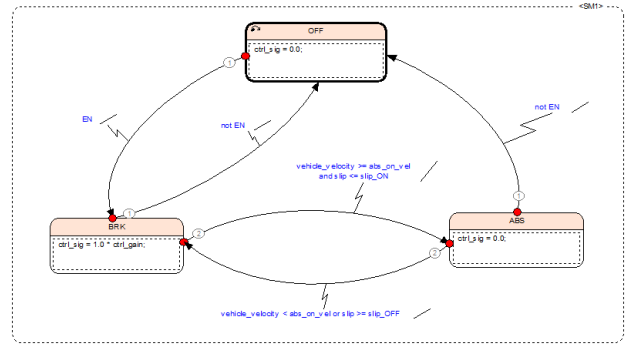


Figure 15. ABS control as a state diagram which modulates the activation of the ABS valves.

During a braking event, the control algorithm uses the measured vehicle speed and calculation of wheel to determine which braking mode to activate. If vehicle speed is below a low-speed threshold or if the calculated wheel slip is less than a minimum value, an unmodulated braking command is sent to the ABS actuator. If vehicle speed is above the low-speed threshold, the controller sends a modulated command signal to the ABS actuator when wheel slip exceeds the established threshold.

Using SCADE code generation, the model-based representation of the controller was transformed into C code and compiled into an FMU, shown in **Figure 16**. The FMU was directly integrated into the braking system model within ANSYS Simplorer. Simulated braking tests were applied to the system model to validate the operation of the control algorithm under various conditions. **Figure 17** shows the modulation of the ABS during the application of full braking on dry pavement at a speed of 20 m/s.

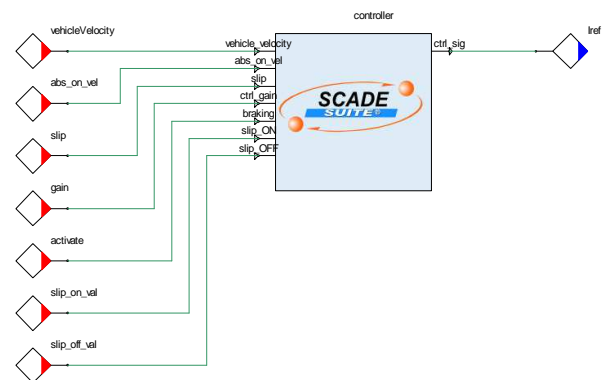


Figure 16. Generated C code for the ABS controller, encapsulated as a Functional Mock-up Unit.

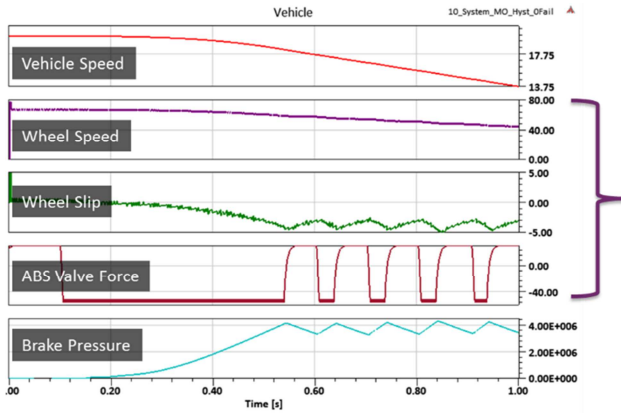


Figure 17. Simulated results of maximum braking applied on dry pavement at 20 m/s.

2.3.7 Speed sensor

The sensor for measuring the wheel speed is an anisotropic magnetoresistive sensor (Lenz, 1990). As the teeth on a magnetic tone ring pass by the sensor, the changes in the direction of the magnetic field relative to the current flow in the sensor cause changes in the resistance of the modules, as seen in **Figure 18** and **Figure 19**.

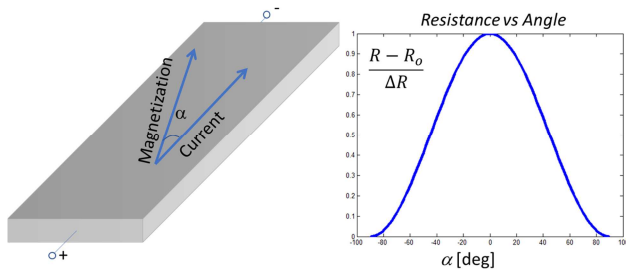


Figure 18. Magnetoresistive material, showing a varying resistance as an external magnetic field angle of incidence changes vs the direction of current flowing through the material.

The variation in resistance follows a squared sinusoidal dependence on angle from -90deg to 90deg shown in **Figure 18**, according to the relation (McGuire, 1975):

$$R(t) = R_o + \Delta R \cos^2 \alpha(t) \quad (1)$$

Taking advantage of this variation, the modules are arranged in Wheatstone bridge configuration so measurable voltage changes result on their output. The change in direction of the magnetic field due to the variable reluctance of the teeth as they pass is calculated by the Maxwell3D finite element program (ANSYS, 2016), as seen in **Figure 19** using a 2-D view to visualize the magnetic flux lines. The flux lines can be seen originating from a permanent magnet and then linking with the nearest tooth creating an angle relative to the face of the magnet and sensor.

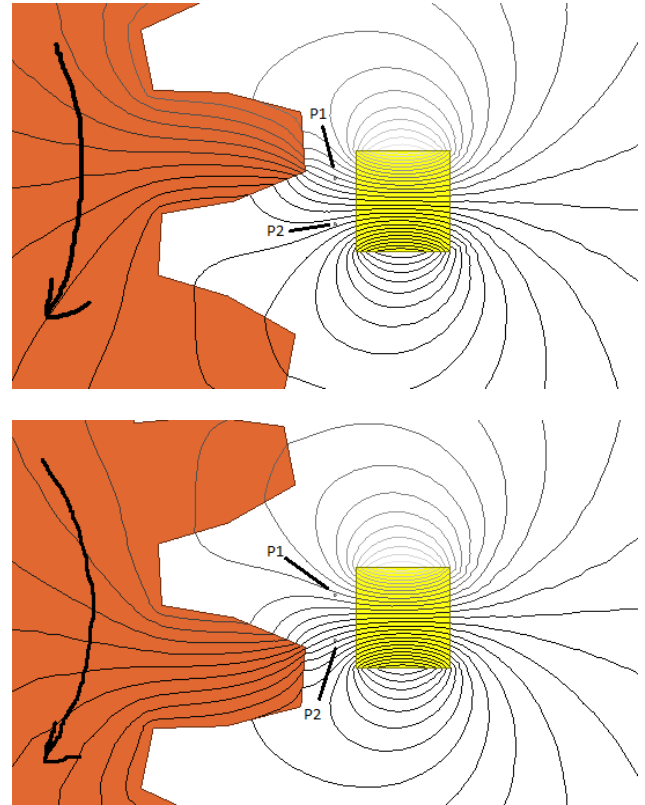


Figure 19. Magnetic flux bending as the tone ring teeth move past the stationary sensor, measuring flux direction as points, *P1* and *P2*.

The two sets of resistive sensor modules are at each point, *P1* and *P2*, as shown in **Figure 19**. The resistive Wheatstone bridge shown in **Figure 20** has equal resistors in opposing positions within the bridge.

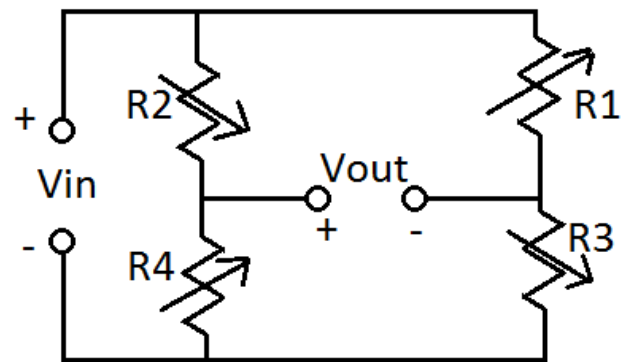


Figure 20. Magnetoresistor bridge with matching resistors placed in opposite positions.

With this arrangement, the output voltage, V_{out} , can be seen to change with a change in direction leading to incremental changes in resistance, dR , as the following:

$$V_{out} = V_{out+} - V_{out-} = V_{in} \left(\frac{R_4}{R_2 + R_4} - \frac{R_3}{R_1 + R_3} \right) \quad (2)$$

$$R_1 = R_4 = R_o + dR(t) \quad (3)$$

$$R_2 = R_3 = R_o - dR(t) \quad (4)$$

where from (1),

$$dR(t) = \Delta R \cos^2 \alpha(t), \quad (5)$$

$$\therefore V_{out} = V_{in} \left(\frac{dR(t)}{R_o} \right) = \left(\frac{V_{in}}{R_o} \right) dR(t) = I_o \Delta R \cos^2 \alpha(t) \quad (6)$$

So it can be seen that the output voltage would be equal to the current, I_o , through the nominal resistance, R_o , times the change in the resistance. ΔR is the maximum possible change in resistance, R_o is the nominal resistance, and $\alpha(t)$ is the field angle relative to the current flow. If there is no change in resistance, the output voltage is zero, as expected from a balanced bridge configuration.

To model the resistor bridge in a circuit simulation using Simplorer (ANSYS, 2016), the equation for the resistance in **Figure 18** and (1) is used. R_o is given as 1200 ohm and ΔR is given as 0.02 ohm. α is measured for all positions of the sensor in the FEA simulation at both $P1$ and $P2$, which are used for R_1 , R_4 and R_2 , R_3 respectively. The average magnetic fields in the volumes of $P1$ and $P2$ are used according to the following equation for α :

$$\alpha = \tan^{-1} \left(\frac{\int H_y dV}{\int H_x dV} \right) \quad (7)$$

Therefore, for any given velocity of the tone ring, the position can be instantaneously measured and evaluated against a precomputed table of values of α , for $P1$ and $P2$, for different tone ring positions. This method excludes any dynamic eddy currents.

In the circuit, this table is represented by a lookup table model with the angle position as an input and the corresponding value of α at $P1$ and $P2$, **Figure 21**.

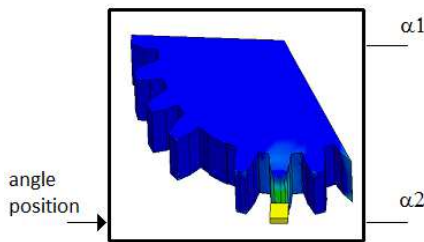


Figure 21. Circuit model of finite element based lookup table, with tone ring position as an input and magnetic field angle measured at $P1$ and $P2$.

The values of α are then passed to an equation block where the resistance in (1) is calculated for $R1$, $R2$, $R3$, and $R4$, as shown in **Figure 22**.

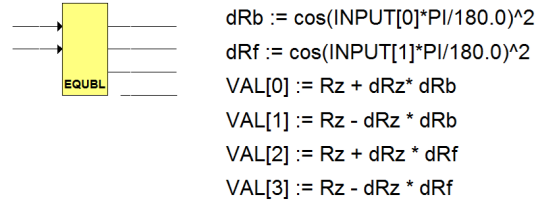


Figure 22. Circuit model equations for the value of the magnetoresistors as the field angles change.

These computed values of resistance are then linked to a resistor bridge, with the value of each resistance being fed from each corresponding output of the equation block, **Figure 23** and **Figure 24**.

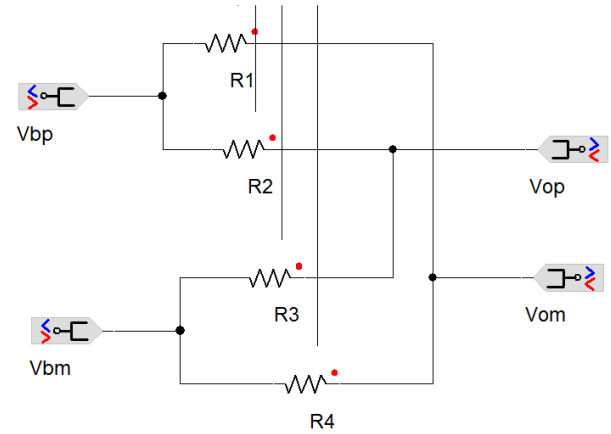


Figure 23. Circuit magnetoresistor bridge model, graphically using resistance values calculated from the equation block in **Figure 22** with wire connections.

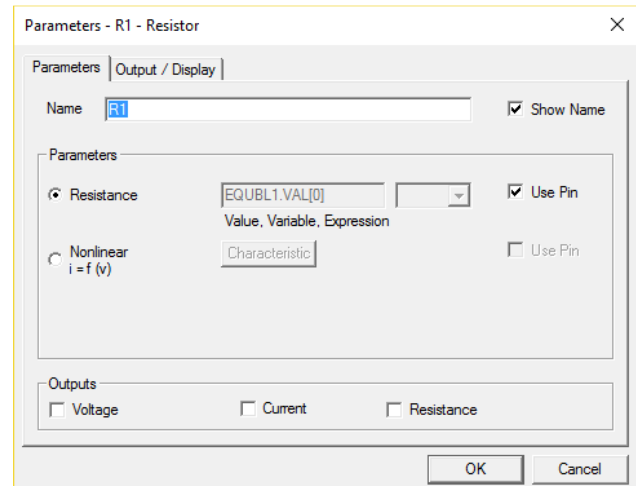
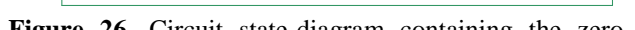
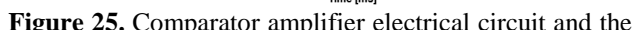


Figure 24. Resistor model properties obtaining resistance from the equation block.

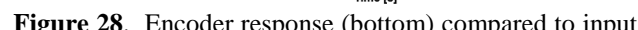
The output of this resistor bridge generates a voltage on the order of 30μV for a wheel rotation of 600rpm, so this measurement voltage is passed onto a



All of these sensor components are placed into sub-



The nominal operation of the speed sensor is shown



1 **1** **1** \ **1** /

[illegible]

3.1 Baseline System Performance

The baseline performance is shown in **Figure 29** for a 3 second braking event from 20 m/s to a complete stop including normal ABS activation.

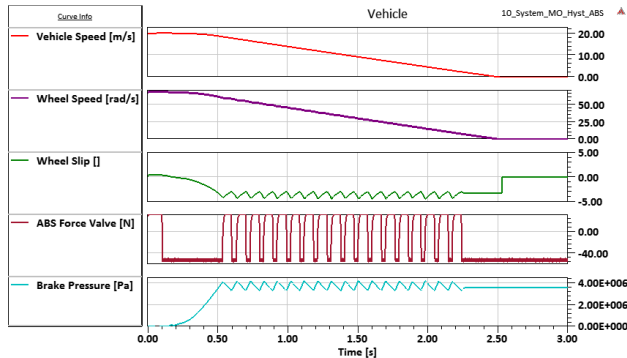


Figure 29. Normal vehicle telemetry results for a full stop from 20 m/s using normal ABS activation.

The amount of brake wear that results, measured in mm, is shown in **Figure 30**.

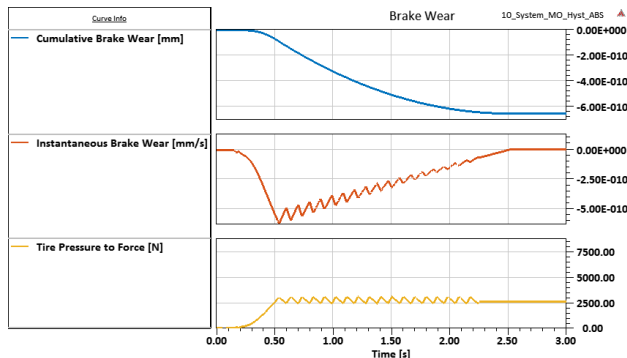


Figure 30. Brake wear for normal stopping condition with ABS activation.

3.2 Fault Injection: Disconnected ABS Controller

In the first fault scenario, the controller fails and is disconnected from the circuit. In this case, the ABS activation does not occur. The vehicle telemetry results in this scenario can be seen in **Figure 31** for the same braking scenario from 20 m/s to a complete stop.

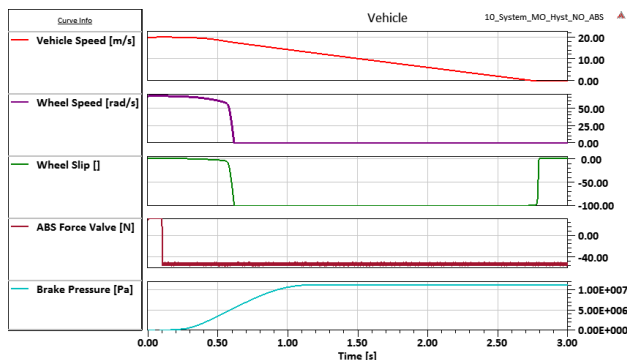


Figure 31. Vehicle telemetry results for a full stop from 20 m/s with abnormal ABS failure.

The brake wear that results in the absence of the ABS activation can be seen in **Figure 32**.

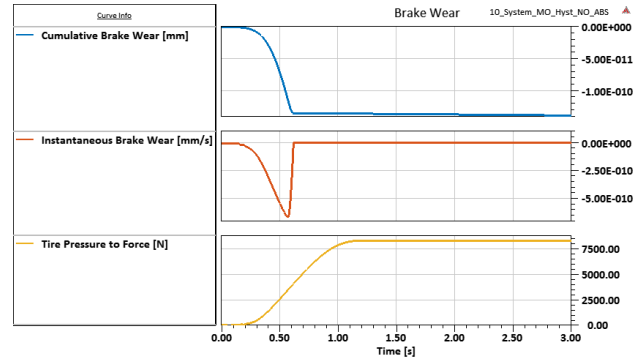


Figure 32. Brake wear for abnormal stopping condition without ABS activation.

It can be seen that with ABS activation there is a cumulative brake wear for this single braking event of 6.25×10^{-10} mm. Without ABS, the brake wear is 1.38×10^{-10} mm. There is less wear in the absence of ABS since the wheel spends more time locked to the brake pads, which also causes the vehicle to spend more time coming to a complete stop. The cumulative wear numbers are very low, since only a single braking event is being investigated.

3.3 Fault Injection: Broken Sensor

In the second scenario, the speed sensor tone ring is modeled with missing teeth (Obrochta, 2015), as seen in **Figure 33**, and causes the sensor to misread the wheel speed.

For this simulation, the first second of braking behavior is investigated for a sensor signature and the effect on ABS behavior. **Figure 34a** shows the normal sensor behavior. **Figure 34b** shows the effect of missing one tooth, and **Figure 34c** shows the effect of missing two non-adjacent teeth.



Figure 33. Tone ring mechanical failure with missing teeth.

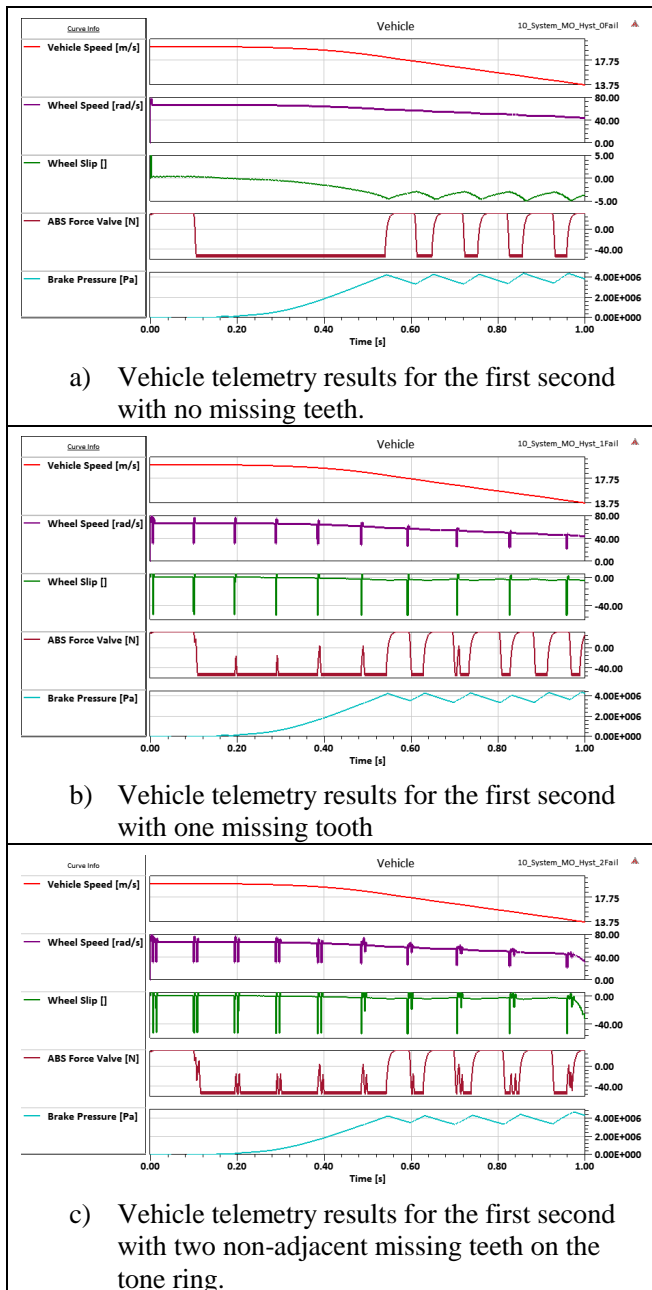


Figure 34. Vehicle telemetry results showing the effect of missing teeth

As teeth go missing from the tone ring, the wheel speed and slip measurements show intermittent dips. The dips represent a perceived reduction in speed since the time between zero crossings increases in the gap, resulting in an interpreted slowdown in speed. This perceived reduction in wheel rotation results in inadvertent ABS activation since it is interpreted as wheel slip since the vehicle speed does not change with it. The simulation can be run for any combination of missing teeth to create signatures leading to more accurate diagnosis of the ABS sensor and specific tone ring failure.

4 Summary and Future Work

A model-driven simulation approach combining 0-D and 3-D physics-based models with controls to support heat monitoring and predictive maintenance for automotive braking stems was shown. These simulation-based digital twins are useful for providing inputs into analytics systems that support predictive maintenance for critical sub-systems, especially under abnormal operating conditions. As an example, the difference in wear rate was identified for normal conditions and abnormal conditions where the ABS controller becomes disconnected, to predict when maintenance is required on the braking system.

Simulation-based digital twin models are also useful for obtaining sensor signatures of the fault conditions needed to train machine learning algorithms that support advanced system diagnostics. These models are particularly useful for observing abnormalities and failures which cannot be observed easily in physical tests or in sufficient quantities to reliably train learning algorithms. In this paper, the unique ABS activation and speed-slip signals were recorded using simulation for several cases of mechanical failure of the teeth on the wheel sensor.

In order to realize these simulation-based digital twins, several methods of system, circuit, and reduced order modeling were shown using 3-D finite element analysis and 0-D multi-domain circuit simulation.

Future work will include more detailed physical models to support more vehicle operating scenarios and adding fault tree analysis with rigorous and automated scenario analysis for detailed root-cause and diagnostics analysis of brake wear. Further work can also be done to connect the simulation-based digital twin with real-time data from the vehicle and adding HMI (Human-Machine Interface) components to depict vehicle health management parameters (diagnostics, prognostics) to the driver.

Acknowledgements

The authors gratefully acknowledge Leon Voss and Michael Sielemann for their work on the original braking system simulation model.

References

- ANSYS, Inc, Canonsburg, PA. (2016). *Maxwell2D*. <http://www.ansys.com>.
- ANSYS, Inc, Canonsburg, PA. (2016). *Maxwell3D*. <http://www.ansys.com>.
- ANSYS, Inc, Canonsburg, PA. (2016). *Mechanical*. <http://www.ansys.com>
- ANSYS, Inc, Canonsburg, PA. (2016). *SCADE Suite*. <http://www.ansys.com>.
- ANSYS, Inc, Canonsburg, PA. (2016). *Simplorer*. <http://www.ansys.com>.

- GE, Boston, MA. (2016). How a 'Digital Twin' for physical assets can help achieve no unplanned downtime [Online]. Available: <http://www.geglobalresearch.com/impact/how-a-digital-twin-for-physical-assets-can-help-achieve-no-unplanned-downtime>
- J.F. Archard. Wear theory and mechanisms. *Wear control handbook*. Peterson MB, Winer WO, editors. New York ASME, 1980.
- Donald C. Augustin, Mark S. Fineberg, Bruce B. Johnson, Robert N. Linebarger, F. John Sansom, and Jon C. Strauss. The SCi Continuous System Simulation Language (CSSL). *Simulation*, No 9, pp. 281–303, 1967.
- M. V. K. Chari, Z. J. Csendes. Finite Element Analysis of the Skin Effect in Current Carrying Conductors. *IEEE Transactions on Magnetics*, 13(5): 1125–1127, September 1977.
- Iain S. Duff and John K. Reid. An Implementation of Tarjan's Algorithm for the Block Triangularization of a Matrix. *ACM Transactions on Mathematical Software*, 4(2):137–147, 1978. doi:
- W.N Fu, P. Zhou, D. Lin, S. Stanton, Z.J. Cendes. Magnetic force computation in permanent magnets using a local energy coordinate derivative method. *IEEE Trans. on Magnetics*, 40(2): 683–686, 2004. doi: 10.1109/TMAG.2004.824774
- S. Holland. Integrated Vehicle Health Management in the Automotive Industry. *Health Management, Krzysztof Smigorski (Ed.)*. InTech, 2010. doi:10.5772/9889.J.E. Lenz. A review of magnetic sensors. *Proc. of the IEEE*, 78(6): 973–989, 1990. doi: 10.1109/5.56910
- T. R. McGuire. Anisotropic magnetoresistance in ferromagnetic 3d alloys. *IEEE Trans. Magn.*, 11(4), 1018–1038, 1975.
- Modelon AB, Lund, Sweden. (2016). *OPTIMICA Compiler Toolkit*. <http://www.modelon.com/products/optimica-compiler-toolkit/>
- Modelon AB, Lund, Sweden. (2016). *Hydraulics Library*. <http://www.modelon.com/products/modelica-libraries/hydraulics-library/>
- Modelon AB, Lund, Sweden. (2016). *Pneumatics Library*. <http://www.modelon.com/products/modelica-libraries/pneumatics-library/>
- Modelon AB, Lund, Sweden. (2016). *Vehicle Dynamics Library*. <http://www.modelon.com/products/modelica-libraries/vehicle-dynamics-library/>
- Eric Obrochta. (2015, Dec. 5). Saturn S series - unwanted ABS activation at all speeds. [YouTube video]. Available: <https://www.youtube.com/watch?v=oGwyrLxtaNY&t=368s>. Accessed Dec. 15, 2016.
- Pacejka, H.B., and Bakker, E. (1993): The Magic Formula tyre model. Proceedings of 1st Colloquium on Tyre Models for Vehicle Analysis, Delft 1991, ed. H.B. Pacejka, Suppl. Vehicle System Dynamics, 21, 1993.
- PTC, Needham, MA. (2016). *Thingworx Analytics*. <http://www.ptc.com/internet-of-things/analytics>.
- R. Prytz. Machine Learning Methods for Vehicle Predictive Maintenance using Off-Board and On-Board Data. *Halmstad University Dissertations*, No. 9, 2014. Siemens, Munich, GmbH (2016). The Digital Twin [Online]. Available: <https://www.siemens.com/customer-magazine/en/home/industry/digitalization-in-machine-building/the-digital-twin.html>
- H. H. Woodson and J. R. Melcher. *Electromechanical Dynamics: Part I: Discrete Systems*. New York, NY: John Wiley & Sons, 1968.

Improved Aerodynamic Prediction Through Coupled System and CFD Models

Ed Tate¹ Joaquin Gargoloff¹ Brad Duncan¹
Hubertus Tummescheit² John Griffin² John Batteh²

¹Exa, USA, {edtate, joaquin, brad}@exa.com

²Modelon, USA, {hubertus.tummescheit, john.griffin, john.batteh}@modelon.com

Abstract

Accurate predictions of aerodynamic forces using computational fluid dynamics require accurate geometry. The aerodynamic forces on the vehicle body affect the vehicle posture or the vehicle position with respect to the ground. When a vehicle is cruising on the road, the change in vehicle posture is usually relatively small with respect to the size of a vehicle. However, these small changes in geometry can lead to significant differences in aerodynamic drag and airflow structures. To address this issue, a coupled simulation approach was developed to predict vehicle posture in typical cruise and wind tunnel test conditions. This coupled approach was tested using Exa's PowerFLOW and Modelon's Vehicle Dynamics Library (VDL). In this approach, the aerodynamic forces on the body are used to calculate the movement of the body and the suspension geometry. This modified geometry is then used to recalculate the operating aerodynamic forces. The modified geometry shows changes in total force, the distribution of forces, and the structure of the airflow over the vehicle. The results provided by correct geometry under loaded conditions offer better correlation to test and provide car makers with the increased accuracy to confidently improve real world fuel economy.

Keywords: aerodynamics, suspension, co-simulation

1 Introduction

One of the most important aspects of a vehicle for fuel economy is the aerodynamic drag. Reducing drag improves fuel economy in conventional vehicles and range in electric vehicles. When a new vehicle is designed, a car maker must decide where to invest resources in meeting mandated and customer expected efficiency requirements. Meeting efficiency targets usually involves improving drag, reducing powertrain losses, and reducing vehicle mass. Improvements in each of these areas represent significant investments on any new program. Accurately predicting the drag is critical to predicting the performance that a production vehicle will achieve. If this value is accurately predicted, an OEM can confidently direct the large

investments associated with improving fuel economy and range. If this value is incorrectly predicted, then late design changes that carry a large risk and expense are needed to meet the original vehicle targets. Predicting vehicle efficiency involves many tools that are used for simulating the different aspects of a vehicle. The vehicle drag prediction requires 3D CFD simulation. The efficiency is usually predicted in system simulations that consider drag, body, and powertrain behavior.

Two common assumptions are used when determining drag for fuel economy, range, and vehicle dynamics simulations. The first is that vehicle aerodynamic forces are accurately represented by a load curve that is a function of vehicle speed. The second is that vehicle geometry is fixed for characterizing aerodynamic forces. Both assumptions are valid, but only for limited conditions. In both the wind tunnel and the real world, these assumptions reduce the accuracy of the resulting predictions.

In a system simulation, the effect of aerodynamic forces on a vehicle is usually calculated using the coefficient of drag. This coefficient is determined from a CFD simulation, measured in a wind tunnel, or derived from a coast down test. For fixed geometry in still air, the drag force, F_D , is a function of the square of the vehicle speed, v , the air density, ρ , the coefficient of drag, C_D , and the frontal area of the vehicle.

$$F_D = \frac{1}{2} \cdot \rho \cdot v^2 \cdot C_D \cdot A \quad (1)$$

The drag force works against the direction of travel of a vehicle. However, in addition to the drag force, the airflow over the vehicle also generates lift forces. These lift forces cause the posture of the vehicle to change, with a 2 millimeter front ride height increase and a 3 millimeter rear ride height for our example, as illustrated in Figure 1. The lift force is calculated similar to the drag force using a lift coefficient. To calculate the vehicle posture, lift forces are calculated over the front and rear axles using an equation similar to the drag equation.

$$F_{L,Front} = \frac{1}{2} \cdot \rho \cdot v^2 \cdot C_{L,Front} \cdot A \quad (2)$$

$$F_{L,Rear} = \frac{1}{2} \cdot \rho \cdot v^2 \cdot C_{L,Rear} \cdot A \quad (3)$$

The lift is determined in this way so that the effect of different lift forces on the front and rear of the vehicle are considered. The lift driven changes in posture mean that the assumption of fixed geometry doesn't hold. Therefore, for accurate prediction of forces on a vehicle, this interaction between vehicle posture and aerodynamic forces should be considered. CFD accuracy is improved by considering the impact of aerodynamic forces on vehicle posture.

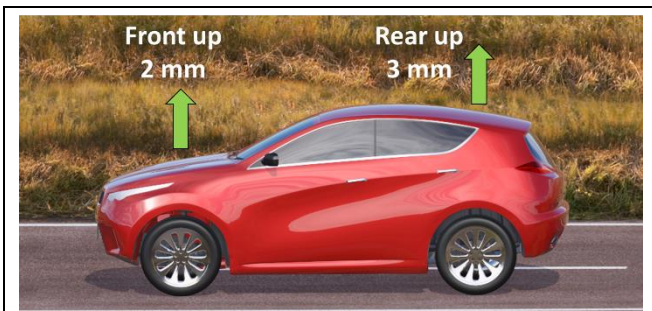


Figure 1. Lift forces and displacement.

2 Determining Posture Change

Changes in vehicle posture affect the position and orientation of suspension parts and wheels. These small changes in geometry affect the airflow over the entire vehicle. The changes in airflow change the pressure on the vehicle surfaces. This change in the pressure distribution and magnitude cause a change in the lift and drag forces. In some cases, this change in posture has an easily observable effect on the airflow. For example, a part of the underbody which was shielded from high speed airflow might be exposed and act like an aeronautic air brake. In other cases, the effect may be subtle, causing changes in the distribution of the flow over the vehicle body and relative change in the flow under and over the vehicle. This effect is similar to how changing an airfoil's angle of attack changes its lift and drag. A key difference is that a vehicle's geometry is much more complex than an airfoil. It has complex surfaces, heat exchangers, fans, airflow through the engine bay, rotating tires, and airflow around the vehicle body.

To accurately determine the effect of these geometry changes a full 3D flow simulation is required. This simulation is done using the Lattice-Boltzmann (LB) solver in PowerFLOW (Exa Corporation, 2017). This solver offers several advantages over traditional Navier-Stokes (NS) based solvers. The LB solver is inherently a transient solver, and the PowerFLOW

implementation is able to handle fully detailed automotive geometry without simplification. This ability to handle geometry changes without special consideration simplifies implementation of geometry movement. This solver is used by OEM's globally for aerodynamic, thermal, and acoustic simulation. Its accuracy and robustness are well documented (Kotopati, 2009; Duncan, 2010; Duncan, 2012).

Modelica was used in this application because it is capable of describing problems in many engineering domains. Most importantly, it can elegantly describe multi-body problems such as suspension simulations. The features inherent in the language make it easy to present the model in a form that can be used by someone who is not an expert in a particular engineering domain, such as suspension simulation.

Furthermore, since Modelica is able to address multiple engineering domains, it provides a solution to describing different functional behavior in the vehicle using a single language. Vehicle Dynamics Library (VDL) (Modelon AB, 2016) has been used extensively in the automotive domain and proven for simulation of complex vehicle behavior (Andreasson, 2011; Andreasson, 2016; Griffin, 2012; Klomp, 2016) in Dymola (Dassault Systemes, 2017). VDL is a commercial Modelica library with a wide range of full fidelity, multibody suspension configurations. VDL can solve for the effect of aerodynamic load, like in a wind tunnel or the open road, and inertial loads, like on the track. In conjunction with OPTIMICA Compiler Toolkit (OCT) (Modelon AB, 2016), Functional Mockup Units (FMUs) (MODELISAR, 2010) from VDL can be created to simplify the task of interfacing between multiple solvers.

A model of a proprietary vehicle from Exa known as the EV12 was implemented using the Vehicle Dynamics Library (VDL) from Modelon. The EV12 vehicle has a McPherson strut front suspension and a twist beam rear suspension. As suspension topologies are available in VDL, modelling the EV12 was simply a matter of modifying the suspension geometry parameters to match those of the EV12. The resulting vehicle model in Dymola is shown in Figure 2.

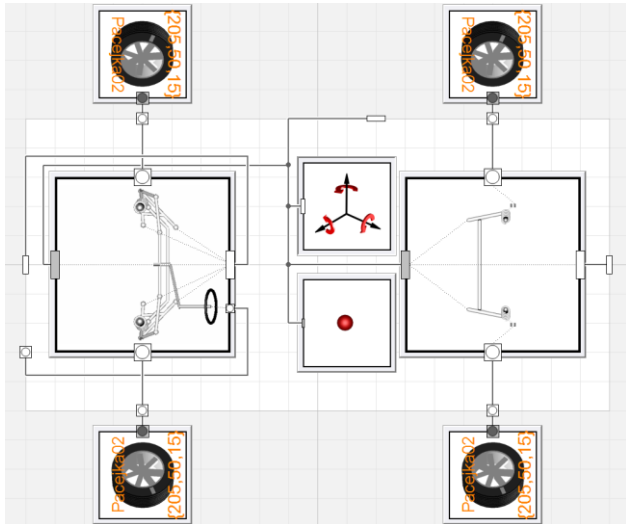


Figure 2. Diagram view of EV12 chassis model

Two different approaches were used during the investigation to quantify the difference in drag forces and vehicle pressure distribution.

2.1 Change in vehicle posture based on downforce

In the first approach, the goal was to determine the effect changes in aerodynamic forces had on the static vehicle posture.

To quantify this effect, the vehicle posture was controlled by aerodynamic downforce. As changes in downforce directly relate to changes in tire vertical forces, tire vertical forces were used to resolve the ride height. This change was implemented as a controller in the system model.

The ride height controller, shown in Figure 3, was implemented by defining the fender height, or vertical height of the chassis at each vehicle corner, versus tire vertical force as tabular data and adjusting the force in the actuator until the desired fender height was achieved. The tire vertical force is a standard output in VDL for vehicle simulations. Therefore, accessing the tire vertical force to use it in the actuator was simply a matter of pulling this signal off the signal Bus. A PID-controller from the Modelica Standard Library was used to control the force.

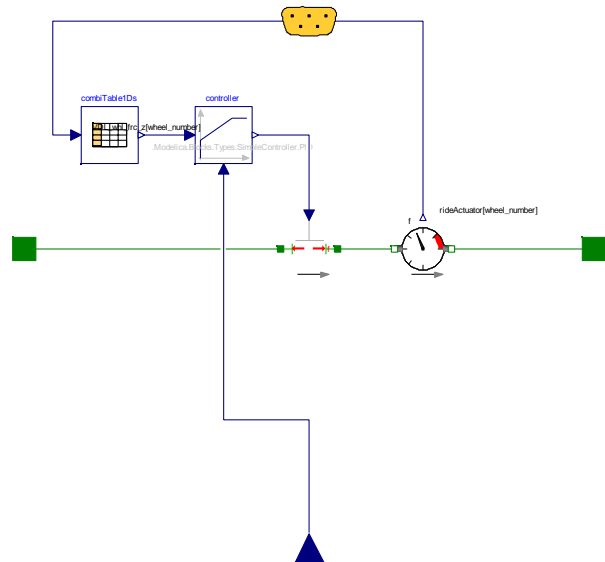


Figure 3. Ride height controller

Control of the vehicle posture was achieved by using ride height actuators as shown in Figure 4. VDL uses both standardized templates and interfaces to describe vehicle components and sub-components. The ride height controller described above used a consistent interface as the standard ride springs. As such changing from the standard ride spring model to ride height actuator was simply a matter of changing classes.

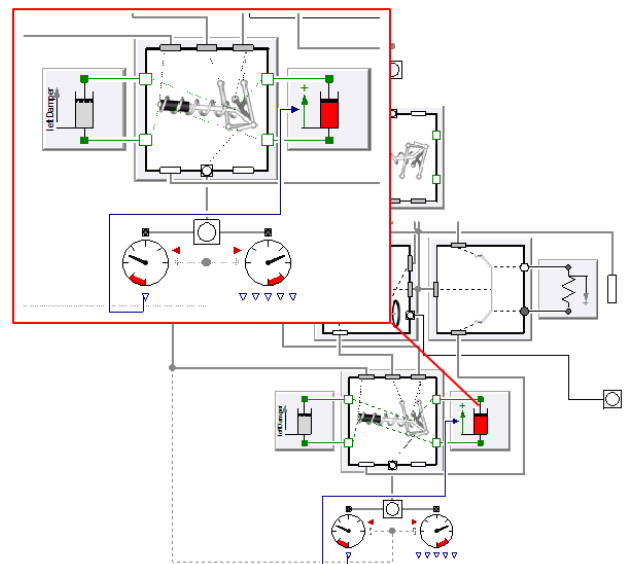


Figure 4. Standard ride springs replaced with actuators

Based on this approach, we concluded that even small changes in aerodynamic downforce affected both vehicle posture and the position and orientation of the suspension components.

2.2 Controlled vehicle posture

In the second approach, the vehicle posture was explicitly controlled.

The vehicle posture was changed using a standard experiment in VDL in which the wheel hubs are held at a fixed vertical position and the chassis is pulled down by two actuators. The attachment points of the actuators on the vehicle were located at positions consistent with the sensors that measure the front and rear ride height in the CFD simulation. The diagram layer of the heave rig experiment is shown in Figure 5.

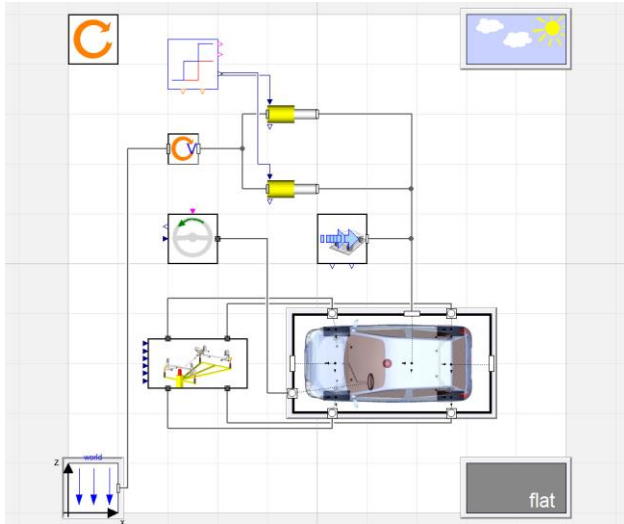


Figure 5. Diagram layer of HeaveRig experiment

The resulting animation of the heave rig experiment is shown in Figure 6.

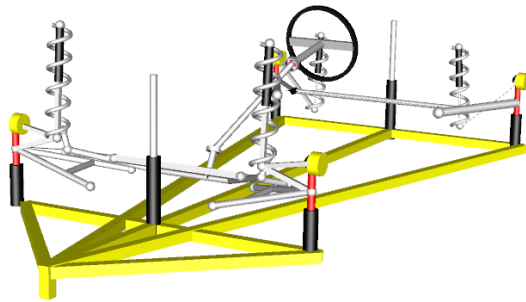


Figure 6. Animation of HeaveRig experiment

The desired results of the heave rig were the time history of all suspension part positions and orientations at all front and rear right heights. To generate this data, a full variable sweep was used in which the front and rear ride heights were varied from -15 to 15 mm of travel at 1 mm intervals. This full sweep resulted in 961 different vehicle postures.

As is evidenced in Figure 7, the suspension components of the vehicle move significantly across

the various vehicle postures. The image below was generated by superimposing all the animation frames.

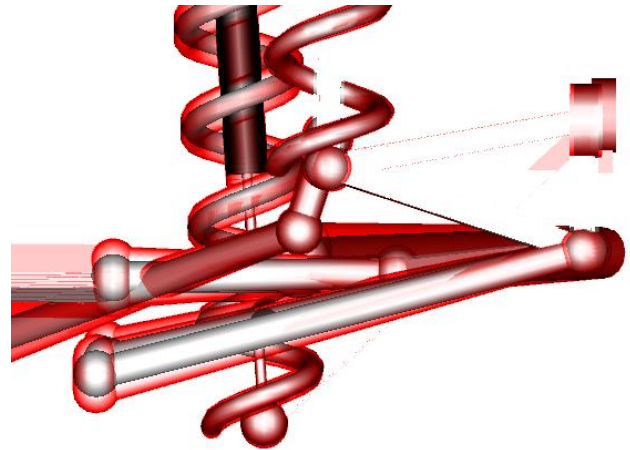


Figure 7. Superimposed frames of HeaveRig animation

The overall magnitude of the change in suspension component position and orientation is shown in Figure 8. This plot shows the change in the height of the outer tierod point vs. front and rear heave changes.

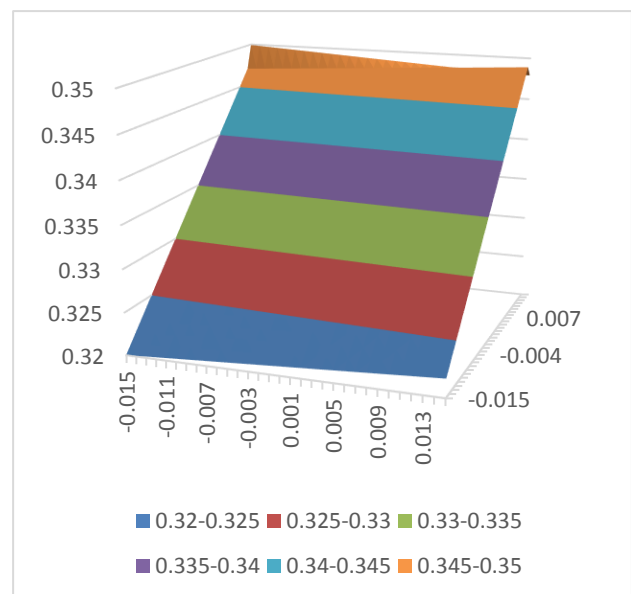


Figure 8. Variation of outer tierod height with changes in vehicle posture

The HeaveRig simulation provided a complete time history of suspension parts position and orientation during the vehicle posture changes. These results were exported and reformatted for use in PowerFLOW.

3 Improved Drag Prediction

The most important design point for a vehicle's aerodynamics is the performance under steady speed conditions on a flat road. This condition is the one

replicated in most wind tunnels. When operating in this manner, the vehicle's change in posture is caused by the drag force and the lift forces. When the vehicle posture changes, many components in the suspension move. This movement is illustrated in Figure 9. Most importantly, the vehicle body position changes.

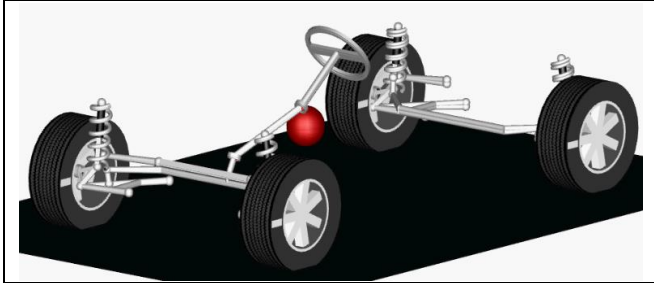


Figure 9. Suspension displacement under aerodynamic load.

While small changes in posture of a few millimeters may appear to be inconsequential, these effects are often a source of error for accurately predicting the drag of a vehicle. Furthermore, these small changes in posture lead to appreciable changes the flow structures on the vehicle. Such an effect is illustrated in Figure 10.

The change in the vehicle posture exposes the front suspension to more incoming flow, which increases the static pressure on the surface of the vehicle, increasing drag. This effect (higher static pressure) is visible in both the lower A-arm attachment to the body as well as the front wheel arch pressure, behind the front suspension. These two areas are marked with white arrows on each image of Figure 10. Both areas show a redder shade of static pressure, contributing to about 1 count of aerodynamic drag (a count is 0.001 or a tenth of a percent).

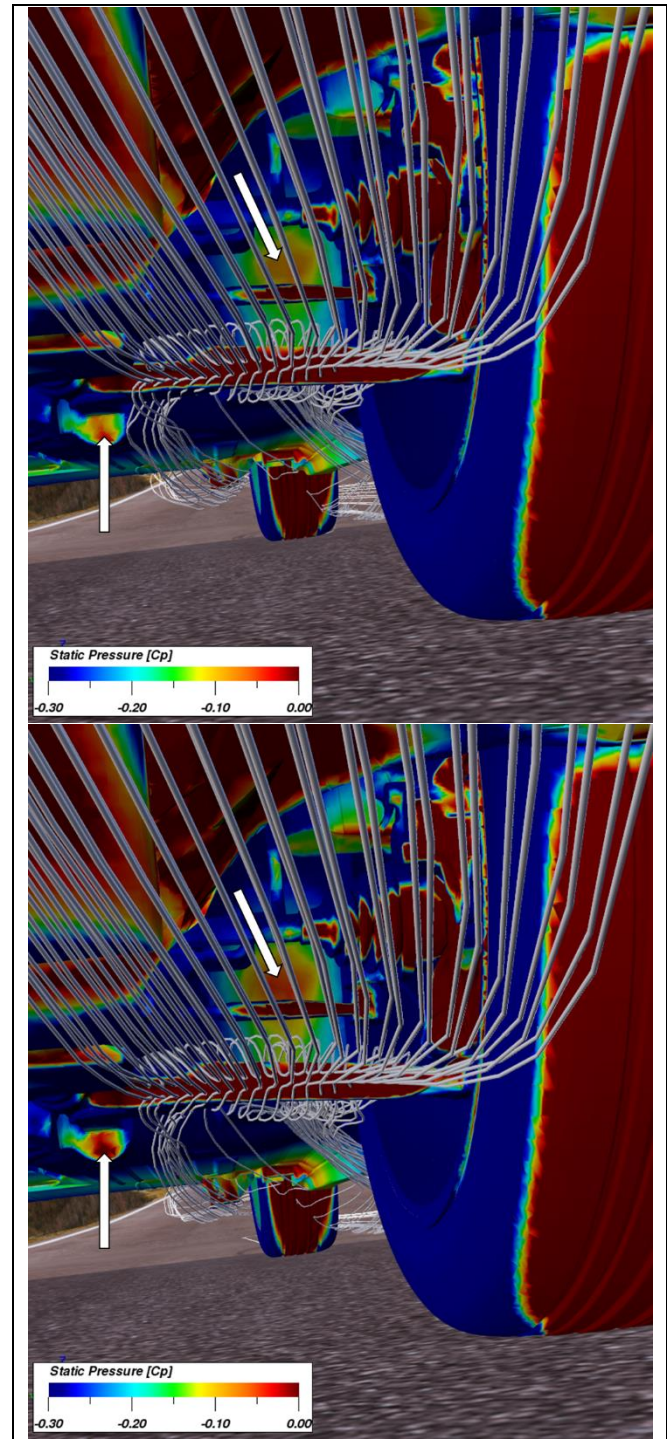


Figure 10. Differences in surface pressure and underbody airflow. Original posture [top] vs. realistic posture [bottom]

Focusing on the rear of the vehicle, Figure 11 shows the surface pressure for both the baseline vehicle (top) as well as the realistic posture (bottom). It can be seen that updating the posture and the suspension yield a lower surface pressure on the back of the vehicle, which contributes to 3 count of drag increase.

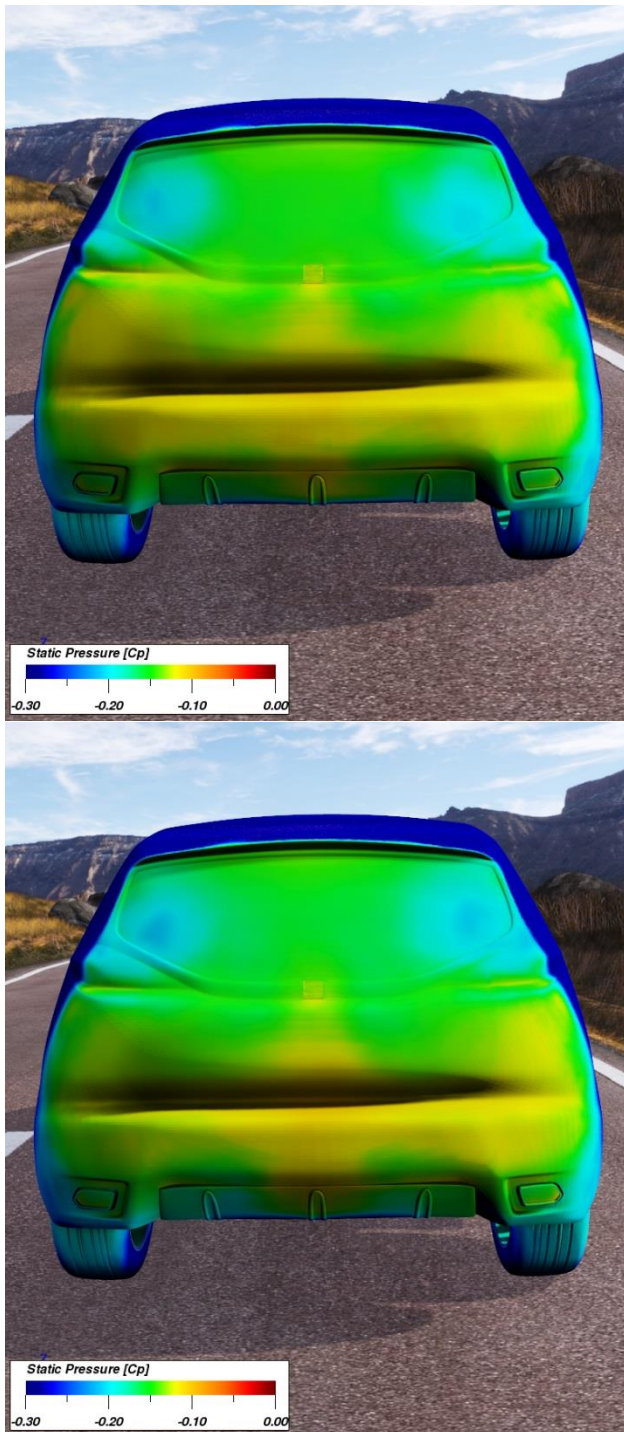


Figure 11. Differences in surface pressure on the back of the vehicle. Original posture [top] vs. realistic posture [bottom]

In this case study, this coupling improved the accuracy of the drag predictions by 4 counts, about a 1% improvement. Simulating a step in between (updating the body posture alone without updating the suspension) enabled us to find that of the 4 total counts of drag increase, 3 were due to the body posture update and 1 count was due to updating the suspension geometry. These effects can be seen in Table 1.

Case:	Cd [counts]	Δ Cd [counts]
Baseline	387	
Posture alone	390	+3
Posture+Suspension	391	+4

Table 1. Difference in vehicle drag due to posture and suspension change.

This improved accuracy was achieved by first finding the drag and lift forces on the vehicle using the at-rest posture. This model included all the vehicle geometry details such as underhood and suspension components. After finding the lift forces, they were applied to the suspension model described in Section 2.1.

The change in posture resulted in changes in the airflow pressure on the vehicle body. Using the corrected geometry, these refined forces are a more accurate representation of the vehicle forces and flows. The local impact of the changes can be seen in the drag development show in Figure 12. This graph shows that the body posture effect of 3 counts is mainly felt at the back of the vehicle, manifesting in a reduction in the base pressure. The suspension effects, on the other hand, have a 1 count of drag impact that is felt mainly on the front axle suspension and the front wheel arches.

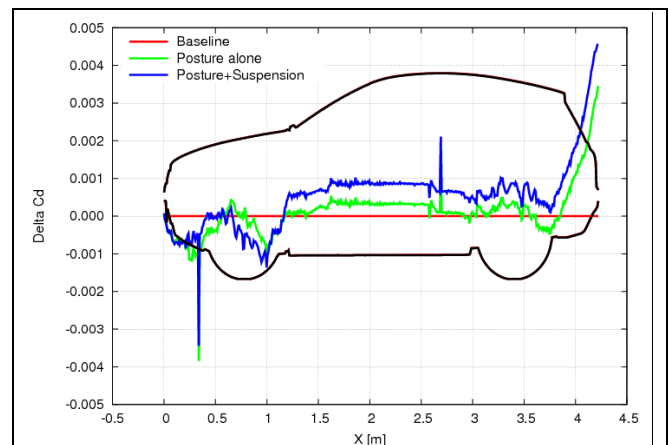


Figure 12. Difference in drag due to posture change.

This iterative coupling solved for the vehicle lift, posture change, and then for the improved drag value. The inputs to the process are the vehicle geometry and the suspension characteristics. The vehicle geometry provides the surface data for CFD simulation and the location of the hard points in the suspension which are used to setup the system simulation. The suspension characteristics allow the system model to properly calculate the changes in geometry due to the lift forces.

4 A Case Study

This process was applied to a proprietary vehicle from Exa known as the EV12. This vehicle has dimensions and details similar to a small SUV. To calculate the posture change, Vehicle Dynamics Library was used to determine the vehicle geometry changes in response to these forces. These changes in geometry were used to modify the vehicle geometry. This modified vehicle geometry was used to find an updated drag value. For this case study, the need to iterate on this process was investigated. It was found that a single iteration was sufficient to account for posture driven changes in drag. This quick convergence allows improved drag prediction at an affordable computational and wall-time cost.

The system model was integrated in a rig which simulates the body motion in response to the drag and lift forces. This rig is shown in Figure 4. The remaining part of this solution is the translation of the suspension movement back to changes in geometry. This translation is accomplished by using consistent frames of reference between the system model and the CFD model. The changes of the frames of reference are determined in response to the aerodynamic forces.

5 Conclusions

A methodology was developed that improves the correlation of vehicle geometry to real world loading conditions and thus improves accuracy in replicating test conditions in CFD. This improvement was accomplished by coupling a Vehicle Dynamics Library model of the vehicle with Exa's PowerFLOW for 3D CFD simulation. The vehicle model was coupled with the CFD simulation via an FMU generated using OPTIMICA Compiler Toolkit.

In the case study considered, changes in drag of about 1% were seen due to the changes in vehicle posture and consequently changes to the suspension geometry. Improving aerodynamic prediction accuracy is critical because of the large impact on certification and real world fuel economy. This case study examined a single aspect of coupling aerodynamic and suspension simulations. This coupling is important enough that it is expected to be part of every vehicle aerodynamic simulation. Future applications will consider the impact of real world conditions, tire tread, and driving cycles to improve designs for efficiency and comfort.

References

- Andreasson, J., "The Vehicle Dynamics Library: New Concepts and New Fields of Application", *Proceedings of 8th International Modelica Conference*, 2011.
- Andreasson, J., Machida, N., Tsushima, M., Griffin, J., Sundström, P.: Deployment of high-fidelity vehicle

models for accurate real-time simulation. *Japanese Modelica Conference 2016*, Tokyo, Japan, May 23-24, 2016.

Dassault Systemes, Velizy, France (2017) *Dymola 2017 FD01*. <https://www.3ds.com/products-services/catia/products/dymola/>

Duncan BD, Fischer A, and Kandasamy S.: Validation of lattice-Boltzmann aerodynamics simulation for vehicle lift prediction. In: ASME 2010 3rd joint US–European fluids engineering summer meeting, 8th international conference on nanochannels, microchannels, and minichannels, Montreal, Quebec, Canada, 1–5 August 2010, ASME paper FEDSM-ICNMM2010-30891, pp. 2705–2716. New York: ASME.

Duncan B, Kandasamy S, Gau H, et al.: Aerodynamic performance assessment of BMW validation models using computational fluid dynamics. SAE paper 2012-01-0297, 2012.

Exa Corporation, Burlington, Mass, USA (2017) *PowerFLOW*. <http://exa.com/en/product/simulation-tools/powerflow-cfd-simulation/>

Griffin, J., Batteh, J., and Andreasson, J., "Modeling Vehicle Drivability with Modelica and the Vehicle Dynamics Library", *Proceedings of 9th International Modelica Conference*, pp. 599-608, 2012.

Klomp, M., Sundström, P., Johnsson, A.: Real-Time Simulation of Elasto-kinematic Multi-body Vehicle Models. *13th International Symposium on Advanced Vehicle Control*, Munich, Germany, pp. 255-260, Sep. 13-16, 2016.

Kotopati R, Keating A, Kandasamy S, et al.: The lattice-Boltzmann-VLES Method for automotive fluid dynamics simulation, a review. SAE paper 2009-26-057, 2009.

MODELISAR, Functional Mock-up Interface for Model Exchange, Version 1.0, 2010.

Modelon AB, Lund, Sweden. (2017). *OPTIMICA Compiler Toolkit*. <http://www.modelon.com/products/optimica-compiler-toolkit/>

Modelon AB, Lund, Sweden. (2017). *Vehicle Dynamics Library*. <http://www.modelon.com/products/modelica-libraries/vehicle-dynamics-library/>

Coupled Simulation between CFD and Multizone Models Based on Modelica Buildings Library to Study Indoor Environment Control

Wei Tian¹ Wangda Zuo^{1,*} Thomas A. Sevilla¹ Michael D. Sohn²

¹Department of Civil, Architectural, and Environmental Engineering, University of Miami, USA,
w.tian@umiami.edu w.zuo@miami.edu, t.sevilla@umiami.edu; *Corresponding Author

²Sustainable Energy Systems Group, Lawrence Berkeley National Laboratory, USA, mdsohn@lbl.gov

Abstract

Multizone models are widely used in building airflow and energy performance simulations because they are often suitable for the analysis needed, and due to their fast computation speed. However, the results provided by the multizone models are sometimes limited due to the underlying well-mixed assumption of the air in a zone (e.g., a room). For zones where this assumption is not suitable, a Computational Fluid Dynamics (CFD) models may be needed. This paper proposes a coupled simulation model between the multizone and CFD model, which in the paper is fast fluid dynamics, a freely available and publicly released program. The model allows the simulation of a dynamic interaction between airflow and Heating, Ventilation and Air-Conditioning (HVAC) systems for buildings with stratified airflow distribution in some of the zones. The approach is implemented using Modelica and its buildings library. In this presentation, we first discuss the design and implementation of a data synchronization strategy between the two models. We then show a possible validation of the implementation by comparing the simulated results with experimental data from previous research. Finally, we perform a case study by linking a Variable Air Volume (VAV) terminal box to space in order to evaluate the capability of the coupled simulation. Finally, further research needs are discussed at the end of the paper.

Keywords: CFD, Multizone, Coupled Simulation

1 Introduction

On average, Americans spend 90% of their time indoors (Kats 2003). Therefore, in order to maintain thermal comfort using HVAC systems, buildings consume about 41% of total energy in the US (Department of Energy 2011). However, the current indoor environment is far from satisfactory. The estimated loss of productivity due to the poor indoor environment is up to 160 billion dollars in the US (Fisk 2000). Thus, it is critical to improve the indoor environment while decreasing the energy consumption.

To improve the design of HVAC system and indoor environment, we can use numerical simulation. On the airflow simulation, there are various models available,

such as multizone models, zonal models, and CFD models (Chen 2009). For the HVAC simulation, there are some conventional building performance simulation programs such as EnergyPlus (Crawley et al. 2001), ESP-r (Strachan et al. 2008), IDA Indoor Climate and Energy (IDA ICE) (Kropf and Zweifel 2001), TRNSYS (Klein et al. 1976), and some advanced techniques such as Modelica-based modeling (Wetter 2009).

Multizone models are widely used in building energy performance simulation programs to save computation time. By asserting that the air is suitably well mixed in a zone, a multizone model solves the mass balance equation and energy balance equation in a significantly faster fashion, compared to the speed of the CFD models (Chen 2009). However, the underlying well-mixed air assumptions for multizone models may be invalid if, for example, the air in the room is stratified. In this case, the multizone models may calculate incorrect results (Wang and Chen 2008).

To model a multiple air distribution type zone building, Wang (Wang 2007) proposed dynamic coupling between CFD and multizone models. As a result, the multizone models are adopted for zones with well-mixed air distribution and CFD model is used for zones with stratified air distribution. At the synchronization time, data is exchanged between the CFD and multizone models. The data exchange is performed iteratively to ensure a fully-converged solution. To achieve convergence and stability, Wang and Chen (2005) recommended transferring pressure data from multizone models to CFD while simultaneously giving airflow rates from CFD to multizone models. While significant, however, their work only focused on the airflow movement and did not demonstrate their approach for buildings that included HVAC systems, and with HVAC controls.

To model the control and distribution of airflow movement in a building with multiple zones, it is necessary to integrate the HVAC system modeling, multizone model, and CFD model. In previous work, multizone models were implemented in Modelica (Wetter 2006a). Similar models are also implemented in the Modelica *Buildings* library (Wetter et al. 2014) which can link to the HVAC system model to study the

control of airflow. Besides the multizone models, there are several CFD model implementations in Modelica such as a sub-zonal CFD model (Bonvini et al. 2014) and VEPZO (Norrefeldt et al. 2012). Moreover, an externally coupled simulation model between CFD model for airflow, HVAC, building envelopes and control was implemented in the Modelica *Buildings* library to enable the study of their dynamic interactions (Zuo et al. 2016). The coupled simulation model was then validated and used to study a case with stratified non-isothermal airflows with an idealized constant air volume system. The results demonstrated that the model is capable of capturing the dynamics of the system.

Based on the previous efforts, this paper implements the coupled simulation of CFD and multizone models in Modelica to study the interaction between airflow movement and HVAC system. This paper first discusses the data synchronization strategy used in the implementation. Then it focuses on the validation by using a case with well-controlled boundary conditions. Finally, a more complex case stemmed from research (Wang 2007) was used to further evaluate the capability of the coupled simulation.

2 Methodologies

A quasi-dynamic data synchronization strategy (Zhai et al. 2002; Tian and Zuo 2013) is used for the coupled simulation. As shown in Figure 1, CFD and multizone models exchange data at a given data synchronization point t_n and then run on their own till the next point t_{n+1} . The exchange of information \mathbf{x} is dependent on different scenarios. Note that CFD models have a constant time step size and multizone models programmed in Modelica uses an adaptive time step size.

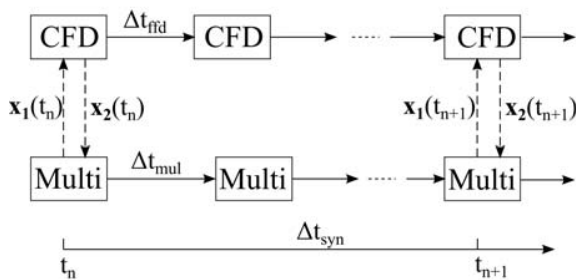


Figure 1. Two-way data synchronization strategy

As shown in Figure 2, we present a simplified physical representation of the data exchange strategy. In this scenario, Zone 1 is simulated by CFD as a non-uniform momentum distribution formed by the inlet directly facing one of the outlets. The mass flow rate and temperature at the inlet of the CFD zone are already known. CFD models feed the mass flow rates and

temperature values at two outlets, which are the averages for time and area, to the multizone modeled zones, namely, Zone 2 and Zone 3.

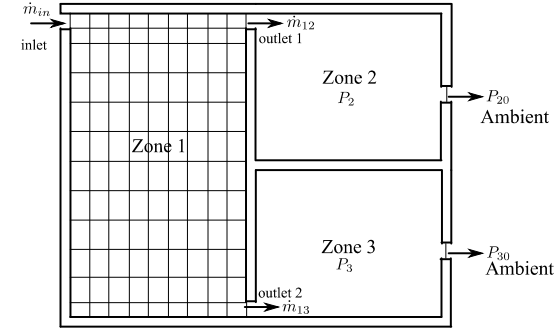


Figure 2. Sketch of the case on which data exchange was implemented

Note that in this simplified data synchronization scheme there is no pressure information exchanged mutually between two programs. After receiving the mass flow rate at openings to from the CFD models, the multizone models can then determine the pressure at zones and mass flow rates at the openings using the equation introduced in the next section.

3 Mathematical Description of Multizone model and FFD

FFD solves the Navier-Stokes equations:

$$\frac{\partial \mathbf{U}_i}{\partial t} = -\mathbf{U}_j \frac{\partial \mathbf{U}_i}{\partial x_j} + \nu \frac{\partial^2 \mathbf{U}_i}{\partial x_j^2} - \frac{1}{\rho} \frac{\partial P}{\partial x_i} + \mathbf{F}_i \quad (1)$$

where \mathbf{U}_i and \mathbf{U}_j are the velocity component in \mathbf{x}_i and \mathbf{x}_j directions, respectively, ν is the kinematic viscosity, ρ is the fluid density, P is the pressure, t is the time, and \mathbf{F}_i is the source term, such as the buoyancy force. FFD splits the Navier-Stokes equation into the following three equations:

$$\frac{\partial \mathbf{U}_i}{\partial t} = -\mathbf{U}_j \frac{\partial \mathbf{U}_i}{\partial x_j} \quad (2)$$

$$\frac{\partial \mathbf{U}_i}{\partial t} = \nu \frac{\partial^2 \mathbf{U}_i}{\partial x_j^2} + \mathbf{F}_i \quad (3)$$

$$\frac{\partial \mathbf{U}_i}{\partial t} = -\frac{1}{\rho} \frac{\partial P}{\partial x_i} \quad (4)$$

FFD first solves the advection equation (2) using a semi-Lagrangian method (Courant et al. 1952). It then solves the diffusion equation (3) with an implicit scheme. Finally, it solves the pressure equation (4) together with the continuity equation

$$\frac{\partial \mathbf{U}_i}{\partial x_i} = 0 \quad (5)$$

using a projection-correction method (Chorin 1967). FFD also applies a similar algorithm to solve the conservation equations of energy and species. The detailed implementation of sequential FFD model can

be found in (Zuo and Chen 2009; Jin et al. 2012). One can also refer to the parallelized FFD model by CUDA and OpenCL in these literature (Zuo and Chen 2010; Yang 2013; Tian, Sevilla, and Zuo 2017).

Typical multizone models, for example, CONTAMW, use the power law to calculate the mass flow rate \dot{Q}_{ij} from zone i to zone j (Dols and Walton 2002). In Modelica Buildings library, the \dot{Q}_{ij} is defined as follows (Wetter 2006b):

$$\dot{Q}_{ij} = C_d A \sqrt{2/\rho} \Delta P^m \quad (6)$$

where C_d is the discharge coefficient normally ranging between 0.6 to 0.75; A is the area size of the opening; ρ is the density of the air; m is constant, which is 0.5 for large openings. ΔP is the pressure difference consisting of total pressure difference $|P_i - P_j|$, pressure difference due to wind ΔP_w , and pressure difference due to density and elevation difference ΔP_t (Wang and Chen 2007).

Since Modelica is an equation-based, object-oriented modeling language (Fritzson 1998), the sign of \dot{Q}_{ij} can be automatically determined based on the pressure in two zones. Thus, we can write the mass conservation for zone i as:

$$\frac{dm_i}{dt} = \sum_{j=1}^n \dot{Q}_{ij} + F_i \quad (7)$$

where m_i is the mass at zone i ; n is number of surrounding neighbours to zone j ; F_i is the air mass source in the zone i . Since the flow in buildings is typically incompressible, we can assume that m_i is not changing with the time. Once the boundary conditions are applied, the pressure at each zone and mass flow rate between neighboring zones can be uniquely determined.

4 Model Implementation

The key obstacle to the implementation is to realize the extraction of the flow rates and the value of the scalar variables at the outlets from CFD and to feed them to the multizone model. To overcome the problem, we put virtual sensors at the outlets to obtain the necessary information. For detailed information of the CFD model in the Modelica *Buildings* library, please refer to previous research (Zuo et al. 2014).

Figure 3 shows the detailed implementation. The CFD zone is modeled using the CFD model in the Modelica *Buildings* library. Three real inputs for radiative heat gain, convective heat gain, and latent heat gain, are connected to the CFD model. At the lower part of the figure, there are fluid and heat ports connected to the CFD model as boundary conditions. Note that the CFD model will calculate the mass flow rates at all ports using the mass balance law and the CFD program will assign the tag of inlet or outlet to the ports based on the

sign of the mass flow rate. On the right side of the figure, the mass flow rates and temperature at the outlets from CFD were given to the prescribed fluid mover through the first order delay model. The delay model is used to mimic reality by making the mass flow rate increase gradually.

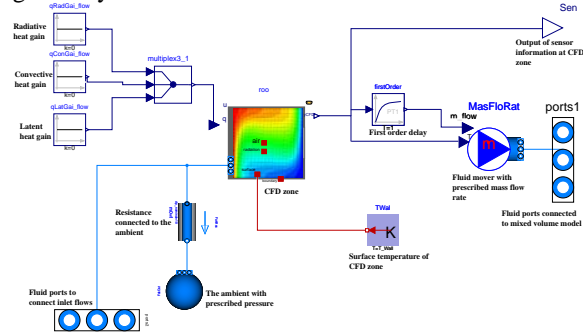


Figure 3. Diagram of Modelica model for coupling

In this paper, we chose Fast Fluid Dynamics (FFD), as an intermediate model between multizone and CFD models, due to its fast computation speed. By sacrificing some accuracy the FFD method is shown to be about 50 times faster than CFD programs if running on the CPU (Zuo and Chen 2009). By taking advantage of the GPU, the FFD program can gain another 30 times computation acceleration, which will be added up to achieve 1500 times faster than CFD program running on CPU (Zuo and Chen 2010).

5 Case Study

5.1 Isothermal with non-uniform momentum distribution

We used one of the three experiments conducted by Wang and Chen (2009) to validate the coupled simulation model. As shown in Figure 4, space consists of four zones. Zone 1, which has one inlet and two outlets, is simulated by FFD, due to the non-uniform momentum distribution as the inlet is directly facing opening 1. Other zones were simulated using multizone models.

Figure 5 shows the Modelica representation of the validation case. A prescribed fluid mover was connected to the CFD zone (Zone 1) to provide the inlet boundary conditions for the FFD program. Other zones were simulated by the multizone models, namely, *MixingVolume*. The openings were simulated by *Orifice*, which nonlinearly correlates the mass flow rates with a pressure difference between zones.

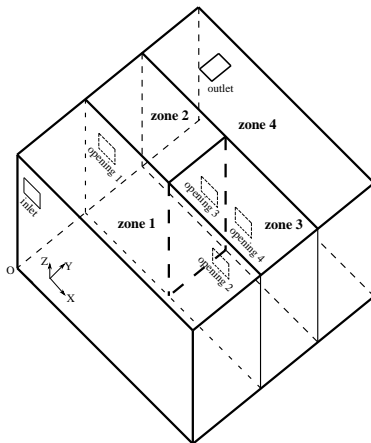


Figure 4. Schematic of a building with two rooms

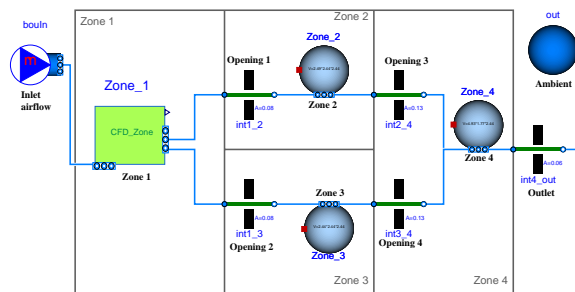


Figure 5. Diagram of Modelica model for a building with two rooms

The radiative heat gain, the convective heat gain, and the latent heat gain inside *CFD_Zone* model are all set to zero. The inlet mass flow rate is changing at 0.033, 0.053, 0.105, 0.14, and 0.215 m³/s. Since this experiment is essentially isothermal, we set the inlet temperature, the temperature at all walls of Zone 1 and initial temperature at fluid cells as 10 °C. The data synchronization time step is set up to 5 s. The simulation span is 100 s and the Radau solver is used. The residual is regulated to be below 1E-6.

FFD uses a mesh of $34 \times 12 \times 18$. The time step size for the former two mass flow rates is 0.1 s and for others is 0.05 s. To simulate the turbulence introduced by the high-velocity jet, we employed the zero equation model proposed by Chen and Xu (1998).

Figure 6 shows the mass flow rates ratio at opening 1 and opening 2 in Zone 1. Our simulated results have good agreement with the experiment when the inlet mass flow rate is generally larger. Due to the fact that there is considerable numerical viscosity (can be acted as turbulence viscosity) in the FFD model as a result of the solution method, we tuned the coefficients of the zero equation turbulence model.

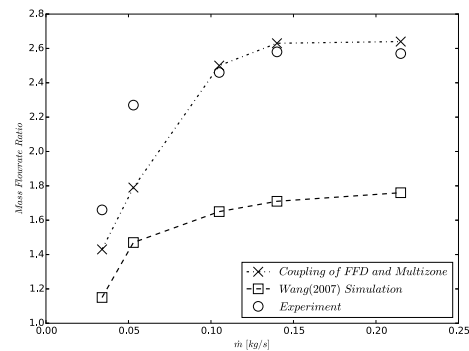


Figure 6. Validation results of mass flow rate ratio at opening 1 and opening 2

5.2 Multizone airflow with a VAV terminal box

In a validation effort, we demonstrate that the coupled simulation model can study the airflow distribution for space with a non-uniform momentum distribution. After adding a VAV terminal box to the validation case, the case study aimed to investigate the control of room temperature for Zone 1, as shown in Figure 7. To increase the efficiency of temperature control, we increased the length of the inlet (in the X direction) by 0.53 m, in order to insert more air from the terminal box in the room.

Here we modeled the heat transfer and radiative heat transfer through and between the envelopes in Zone 1 in Modelica. The exterior surface temperature for floor and other walls are 25 °C and 27 °C, respectively. The initial temperature of the space is 30 °C. The objective is to sustain 25 °C temperature for occupant zone of Zone 1, which is in the lower half part, by adjusting the VAV terminal box.

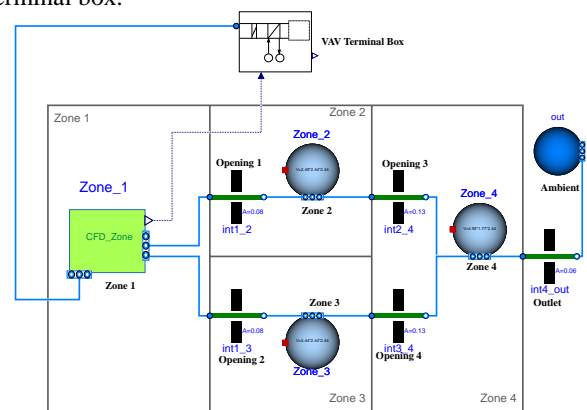


Figure 7. VAV terminal box with validation space

Figure 8 illustrates the detailed model of VAV terminal box. Since we isolated the room from a VAV system which serves multiple rooms, we assume that the pressure difference at terminal box and space outlet as

constant. Thus, we set the pressure of the cold air source as 20 Pa. The temperature of the cold air source is constant as 16 °C. The opening of the valve in the cold air loop is adjustable and reheat coil can be turned on by opening the valve in the hot water loop. A controller is implemented to coordinate the opening position of the valve in cold air and hot water loop.

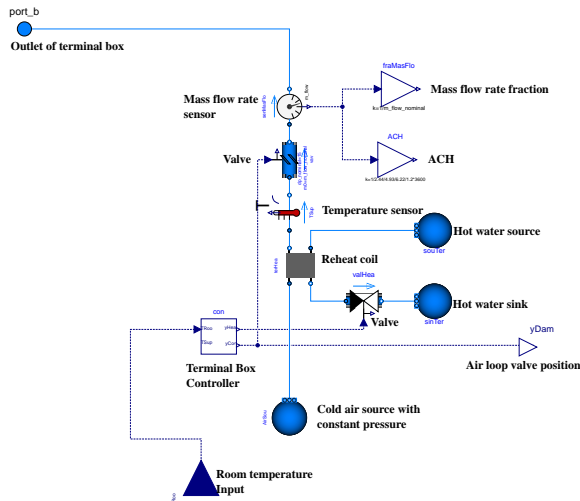


Figure 8. VAV terminal box

As shown in Figure 9, we implemented a pressure-dependent control logic (Liu et al. 2012). The occupant zone temperature signal is first sent to adjust the valve position in the cooling air loop, which is at the lower part of the figure. If the valve opening decreases to 30%, which is deemed as the lower limit, then, the reheat coil will be turned on by feeding the opening position signal to the valve of the reheat coil. The control of the reheat coil is shown in the upper part of the figure. To avoid the short cycling of the reheat coil, we added to the controller a hysteresis, which has lower bound of 0.3 and higher bound of 0.4.

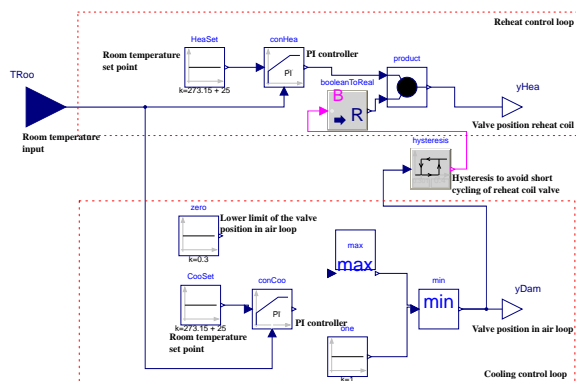


Figure 9. Controller in VAV terminal box

From Figure 10 to Figure 12, the dynamic response of the VAV terminal box and indoor environment is shown. In the beginning, as shown in Figure 10, the room temperature is initially higher than the set point (25 °C), the opening ratio of the valve in the cold air loop is decreasing from 1.0 to 0.3 as shown in Figure 11. The mass flow rate of the supply air as shown in Figure 12 then drops from 0.120 kg/s to 0.044 kg/s. Since the reheat coil does not turn on, the supply air temperature remains constant as 16 °C, as shown in Figure 13.

At around 60 seconds, when the opening ratio of the valve in the cold air loop reaches 30%, and the room temperature is lower than the set point (Figure 10), the reheat coil is turned on. Then, the room temperature is increased and meets the set point at around 160 seconds. Since the room temperature is lower than the set point at this period (60-160 seconds), the opening ratio of the valve in cold air loop remains a minimum of 30% and the opening of the valve in reheat coil first climbs up and then drops, as shown in Figure 11. As a result, the mass flow rate of the supply air remains constant at 0.044 kg/s (Figure 12). Consequently, one can see in Figure 13 that the supply air temperature first increases to a maximum of 25.4 °C and then gradually drops to 23.0 °C, along with the change of opening of the valve in reheat coil.

From 160 to 225 seconds, the room temperature is higher than the set point and their difference is decreasing (Figure 10). As the difference changes, the opening of the valve in the cold air loop increases from 0.3 to 0.4 kg/s. Though the room temperature is higher than set point, due to the hysteresis embedded in the controller, the reheat coil is still on with a small opening (Figure 11). Thus, the supply air temperature is higher than 16 °C and generally decreasing with the valve opening becoming smaller (Figure 13).

After approximately 225 seconds, the room temperature is approaching the set point (Figure 10). At end of the simulation (15 min), the difference between room temperature and the set point is marginal. Since the room temperature is higher than set point and the opening of the valve in cold air loop is larger than 0.4, the reheat coil is turned off (Figure 11) and supply air temperature is 16 °C (Figure 13).

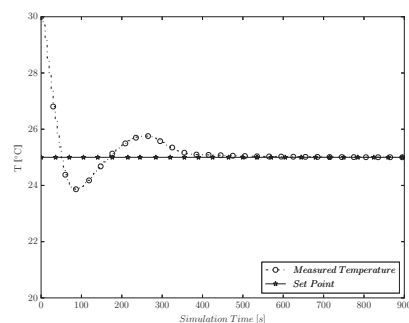


Figure 10. Zone 1 temperature control

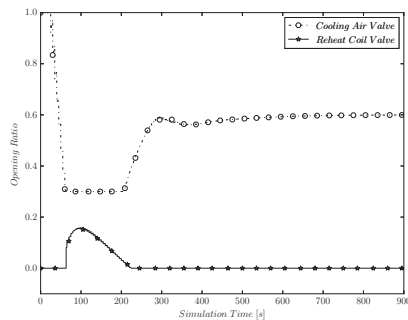


Figure 11. Control outputs from VAV terminal box

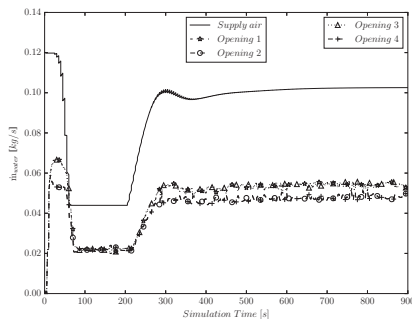


Figure 12. Mass flow rates at different openings

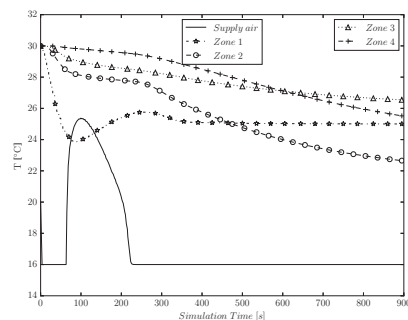


Figure 13. Zone temperature in the space

Note that we presented the mass flow rate of supply air and at different openings in the space in Figure 12. We can clearly identify the mass flow rate difference at Opening 1 and Opening 2, which would be ignored if a multizone model is used. Due to the mass conservation law, the mass flow rate at Opening 1 and Opening 3 are equal, and the same rule applies to Opening 2 and Opening 4.

Figure 13 shows the temperature of supply air and other zones. As the room temperature in Zone 1 approaches set point of 25 °C, the temperature at Zone 3 and Zone 4 gets close to the set point with an error of

1.0 °C. However, in the Zone 2, the temperature is 21.7 °C, which is as expected, because part of the cold supply air in Zone 1 is directly injected into Zone 2 as opening 1 is facing to the inlet of Zone 2.

6 Conclusion and Discussion

The results shown in the validation case prove that the coupled simulation is capable of handling the airflow simulation in a multi-zone space with non-uniform momentum distribution. By further adding a VAV terminal box to the validation case, the coupled simulation model further demonstrates its application potential in indoor climate control and its capability to capture the dynamics of the building system as well as the indoor environment. In the future, more case studies need to be performed to holistically assess the coupled simulation model such as contaminant control and fire or smoke control. Moreover, the FFD simulation can be performed in parallel (Tian, Sevilla, and Zuo 2017) or a reduced order model such as in situ adaptive tabulation (Li et al. 2016; Tian, Sevilla, Li, et al. 2017) can be further used to accelerate the computation speed.

Acknowledgements

This research was supported by the National Science Foundation under Award No. IIS-1633338 and the U.S. Department of Energy under Contract No. DE-EE0007688. This research was also supported in part by the U.S. Defense Threat Reduction Agency. LBNL's research was performed under U.S. Department of Energy Contract No. DE-AC02-05CH11231.

References

- Bonvini, M., M. Popovac, and A. Leva. 2014. Sub-Zonal Computational Fluid Dynamics in an Object-Oriented Modelling Framework. *Proceedings of the Building Simulation*.
- Chen, Q. 2009. Ventilation Performance Prediction for Buildings: A Method Overview and Recent Applications. *Building and Environment*, 44 (4):848-58.
- Chen, Q., and W. Xu. 1998. A Zero-Equation Turbulence Model for Indoor Airflow Simulation. *Energy and Buildings*, 28 (2):137-44.
- Chorin, A. J. 1967. A Numerical Method for Solving Incompressible Viscous Flow Problems. *Journal of Computational Physics*, 2 (1):12-26.
- Courant, R., E. Isaacson, and M. Rees. 1952. On the Solution of Nonlinear Hyperbolic Differential Equations by Finite Differences. *Communications on Pure and Applied Mathematics*, 5 (3):243-55.
- Crawley, D. B., L. K. Lawrie, F. C. Winkelmann, W. F. Buhl, Y. J. Huang, C. O. Pedersen, R. K. Strand, et al. 2001. Energyplus: Creating a New-Generation Building Energy Simulation Program. *Energy and Buildings*, 33 (4):319-31.

- Department of Energy. 2011. "Building Energy Data Book." In.
- Dols, W. S., and G. N. Walton. 2002. *Contamw 2.0 User Manual: Multizone Airflow and Contaminant Transport Analysis Software*: US Department of Commerce, Technology Administration, National Institute of Standards and Technology.
- Fisk, W. J. 2000. Health and Productivity Gains from Better Indoor Environments and Their Relationship with Building Energy Efficiency. *Annual Review of Energy and the Environment*, 25:537-66.
- Fritzon, P. 1998. Modelica - a Language for Equation-Based Physical Modeling and High Performance Simulation. *Applied Parallel Computing*, 1541:149-60.
- Jin, M., W. Zuo, and Q. Chen. 2012. Improvements of Fast Fluid Dynamics for Simulating Air Flow in Buildings. *Numerical Heat Transfer, Part B: Fundamentals*, 62 (6):419-38.
- Kats, G. 2003. *Green Building Costs and Financial Benefits*: Massachusetts Technology Collaborative Boston, MA.
- Klein, S. A., J. A. Duffie, and W. A. Beckman. 1976. Trnsys-a Transient Simulation Program. *Ashrae Transactions*, 82:623.
- Kropf, S., and G. Zweifel. 2001. Validation of the Building Simulation Program Ida-Ice According to Cen 13791 "Thermal Performance of Buildings—Calculation of Internal Temperatures of a Room in Summer without Mechanical Cooling—General Criteria and Validation Procedures". *Hochschule Technik+ Architektur Luzern. HLK Engineering*.
- Li, D., W. Tian, Z. Wetter, Wangda, and Michael. 2016. Simulation Using in Situ Adaptive Tabulation and Fast Fluid Dynamics. *IBPSA-USA Journal*, 6 (1).
- Liu, G., J. Zhang, and A. Dasu. 2012. Review of Literature on Terminal Box Control, Occupancy Sensing Technology and Multi-Zone Demand Control Ventilation (Dcv). *US Department of Energy, Tech. Rep*.
- Norrefeldt, V., G. Grün, and K. Sedlbauer. 2012. Vepzo—Velocity Propagating Zonal Model for the Estimation of the Airflow Pattern and Temperature Distribution in a Confined Space. *Building and Environment*, 48:183-94.
- Strachan, P., G. Kokogiannakis, and I. Macdonald. 2008. History and Development of Validation with the Esp-R Simulation Program. *Building and Environment*, 43 (4):601-9.
- Tian, W., A. T. Sevilla, D. Li, W. Zuo, and M. Wetter. 2017. Fast and Self-Learning Indoor Airflow Simulation Based on in Situ Adaptive Tabulation. *Journal of Building Performance Simulation*.
- Tian, W., T. A. Sevilla, and W. Zuo. 2017. A Systematic Evaluation of Accelerating Indoor Airflow Simulations Using Cross-Platform Parallel Computing. *Journal of Building Performance Simulation*, 10 (3):243-55. doi: 10.1080/19401493.2016.1212933.
- Tian, W., and W. Zuo. 2013. Literature Review and Research Needs to Couple Building Energy and Airflow Simulation. *Proceedings of the Proceedings of the the APEC Conference on Low-carbon Towns and Physical Energy Storage*.
- Wang, L. 2007. *Coupling of Multizone and CFD Programs for Building Airflow and Contaminant Transport Simulations*: ProQuest.
- Wang, L., and Q. Chen. 2005. On Solution Characteristics of Coupling of Multizone and CFD Programs in Building Air Distribution Simulation. *Proceedings of the Proceedings of the 9th International IBPSA Conference (Building Simulation 2005)*, Montreal, Canada.
- Wang, L., and Q. Chen. 2007. Validation of a Coupled Multizone-CFD Program for Building Airflow and Contaminant Transport Simulations. *HVAC&R Research*, 13 (2):267-81.
- Wang, L. L., and Q. Chen. 2008. Evaluation of Some Assumptions Used in Multizone Airflow Network Models. *Building and Environment*, 43 (10):1671-7.
- Wang, M., and Q. Chen. 2009. Assessment of Various Turbulence Models for Transitional Flows in an Enclosed Environment (Rp-1271). *HVAC&R Research*, 15 (6):1099-119.
- Wetter, M. 2006a. Multizone Airflow Model in Modelica. *Proc. of the 5-th International Modelica Conference*, 2:431-40.
- Wetter, M. 2006b. Multizone Airflow Model in Modelica. *Proceedings of the Proc. of the 5-th International Modelica Conference*.
- Wetter, M. 2009. Modelica-Based Modeling and Simulation to Support Research and Development in Building Energy and Control Systems. *Journal of Building Performance Simulation*, 2 (2):143-61.
- Wetter, M., W. Zuo, T. S. Nouidui, and X. Pang. 2014. Modelica Buildings Library. *Journal of Building Performance Simulation*, 7 (4):253-70. doi: 10.1080/19401493.2013.765506.
- Yang, P. 2013. "Real-Time Building Airflow Simulation Aided by GPU and FFD." Concordia University.
- Zhai, Z., Q. Chen, P. Haves, and J. H. Klems. 2002. On Approaches to Couple Energy Simulation and Computational Fluid Dynamics Programs. *Building and Environment*, 37 (8):857-64.
- Zuo, W., and Q. Chen. 2009. Real-Time or Faster-Than-Real-Time Simulation of Airflow in Buildings. *Indoor Air*, 19 (1):33-44.
- Zuo, W., and Q. Chen. 2010. Fast and Informative Flow Simulations in a Building by Using Fast Fluid Dynamics Model on Graphics Processing Unit. *Building and Environment*, 45 (3):747-57.
- Zuo, W., M. Wetter, D. Li, M. Jin, W. Tian, and Q. Chen. 2014. Coupled Simulation of Indoor Environment, HVAC and Control System by Using Fast Fluid Dynamics and Modelica. *Proceedings of the 2014 ASHRAE/IBPSA-USA Building Simulation Conference*, Atlanta, GA, Sep. 10-12.
- Zuo, W., M. Wetter, W. Tian, D. Li, M. Jin, and Q. Chen. 2016. Coupling Indoor Airflow, HVAC, Control and Building Envelope Heat Transfer in the Modelica Buildings Library. *Journal of Building Performance Simulation*, 9 (4):366-81.

Co-Simulation between detailed building energy performance simulation and Modelica HVAC component models

Andreas Nicolai¹ Anne Paepcke¹

¹Institut for Building Climatology, Faculty of Architecture, TU Dresden, Germany,
andreas.nicolai@tu-dresden.de

Abstract

We discuss the application of the FMI Co-Simulation technology to building energy performance simulation, where detailed physical building models are coupled to Modelica-based HVAC component and plant models. First, we describe the generation process of the building FMU from our stand-alone building simulation program NANDRAD and sketch out internal algorithms for FMI version 2 capabilities. Then, coupling scenarios are described and physical interface conventions are presented. Usability is addressed by automatic generation of building-model specific adapters and wrappers. The building FMU and plant FMUs are then simulated together using different Co-Simulation master algorithms. Finally, based on simulation results and performance analysis we conclude with recommendations on suitable master algorithm options and specific features of suitable building FMUs.

Keywords: FMI, Co-Simulation, Energy, Building Simulation, HVAC System, Physical Interface, Master Algorithm

1 Introduction

Building energy performance simulation is a technology used by planners and building designers in the planning process. A typical usage scenario includes evaluation of different options regarding building envelope construction, HVAC systems and control strategies. Currently, available simulation tools, such as EnergyPlus TRNSYS (Klein et al., 1976; Dols et al., 2014), IDA-ICE (Sahlin et al., 2004) and our own development NANDRAD (Nicolai and Paepcke, 2012; Paepcke and Nicolai, 2014) (in C/C++) are conceived as stand-alone tools. Modeling and simulation of integrated modern buildings requires flexible plant and equipment models, which are often case-specific. Extending the source code of existing building simulation models is often only possible by original model developers and also very difficult and time consuming.

Alternatively, Modelica as one example for a modeling language can be used to express such equipment and control systems. There are a number of libraries providing suitable components for modeling building systems, for example the Annex60-based libraries AixLib, BuildingSystems, Buildings and Idias (Wetter et al., 2013; Wetter, 2009; Nytsch-Geusen et al., 2013; Sahlin et al., 2004)

or the GreenBuilding library¹. However, modeling the entire building with sufficient physical detail in Modelica alone is not meaningful for several reasons:

- larger building complexes may involve many zones, constructions, facade elements, thermal storage members resulting in thousands of differential equations,
- Modelica code may become huge and may cause problems with the generic Modelica solvers, even symbolic analysis may be extremely slow,
- modeling the building in Modelica without suitable BIM-style data import or code generation will not be possible for realistic buildings, it is too time-consuming and thus too expensive, and
- manual connection of many building components with corresponding equipment and control models may be extremely time-consuming and error-prone.

For practical purposes, planners and engineers will not accept a procedure that involves creation of such complex models with current Modelica user interfaces, alone. There are, however, tools under development that assist with prototyping Modelica-based building and equipment models, for example TEASER². However, limitations with respect to the detail of the building model and simulation efficiency persist.

1.1 Benefits of Simulation Coupling within the Building Energy Simulation Context

The use of stand-alone simulation tools or Modelica-only based building modeling may not be a satisfying strategy. Instead, a hybrid approach appears meaningful:

- using existing building simulation software tailored to the building engineering user group, preferably Building Information Model (BIM) preprocessing software packages (DesignBuilder³, BIM-HVAC

¹Green City/ SimulationX – Planungstool,
<http://www.ea-energie.de/de/products/↔green-city-simulationsbibliothek-2-2>

²TEASER - Tool for Energy Analysis and Simulation for Efficient Retrofit, <https://github.com/RWTH-EBC/TEASER>

³<https://www.designbuilder.co.uk>

tool⁴, etc.) with database support, graphical representation of the building, and input error control with automatic generation of input data to building simulation engines (e.g. IDF-files for EnergyPlus, or nandrad-files for NANDROID), and

- use of Modelica and suitable libraries by HVAC system planners to model building equipment (heater/chiller/ventilation systems) and required control strategies.

Joining both models in a coupled simulation will combine also the benefits of both modeling approaches. With the FMI standard a unified methodology and technical description for coupled simulation is available. With respect to the two described operation modes ModelExchange and Co-Simulation, we prefer the latter variant that allows individual FMUs to use their own dedicated solver engines. However, it can be expected that the Co-Simulation approach and gained flexibility implies a simulation overhead and performance penalty. In the remainder of the article we always refer to Co-Simulation according to the FMI standard when discussing coupled simulation.

1.2 Envisioned Usage of Co-Simulation

We envision two suitable scenarios of combining a dedicated building energy simulation FMU with one or more HVAC component FMUs created with Modelica. In the first scenario, the user will model the equipment system in Modelica and import a previously generated building FMU into the modeling environment, connect it to the Modelica components and run the simulation within the Environment (Figure 1).

Alternatively, HVAC component or control models may be designed with Modelica and then exported by the modeling tool into FMUs. These are then combined with the building FMU and simulated by an alternative Co-Simulation master. This approach allows prefabrication of HVAC component sub-models.

1.3 Co-Simulation Requirements

A central requirement for the application of Co-Simulation is that obtained results are of a similar accuracy as if the entire model would be calculated stand-alone. Accuracy shall be defined in this respect such that the global error, i.e. the difference between numerical solution and true solution is bounded to a defined limit. In practice, within each integration step the local error is controlled. Every FMU should implement such an error control algorithm to be considered a consistent model.

In the building energy simulation side, this demand restricts the choice of suitable simulation tools, for example, older simulation engines like EnergyPlus and TRNSYS do not implement such an error testing procedure. Our building simulation models THERAKLES and NANDROID (Nicolai, 2013; Nicolai and Paepcke, 2012) belong

to a class of modern solvers that use dynamic time step adjustment schemes based on local error estimates, with the advantage of maintaining required accuracy while improving simulation speed whenever possible (Hindmarsh et al., 2005). This is an important feature, since different building equipment may be active during different annual seasons and may enforce different time integration regimes. For example, heating systems are turned off during summer, and if air conditioning is not used, simulation can speed up since no interaction with actively controlled equipment occurs. Simulation time steps typically vary between 1 second and 30 minutes in annual simulations.

The requirement on error control made for FMUs should also be fulfilled by the Co-Simulation master, which effectively needs to adjust communication interval sizes. When separating control and equipment systems from the building's thermal response in a Co-Simulation scenario, the use of larger communication intervals may cause stability and accuracy problems. Such problems can be avoided by choosing a sufficiently small time step size. In realistic simulation cases it is generally not possible to predict the allowed maximum of the communication step size. Also, using a fixed tiny communication step size leads to unacceptable long simulations and would limit the advantage of performance optimized FMU-internal solvers. Therefore, a master algorithm which supports error/stability control and dynamic adjustment of communication step sizes is desirable. This, in return, requires FMI Version 2.0 capabilities of the slaves, in particular the get and set state functionality (FMI, 2014).

Note, that an error control algorithm within a Co-Simulation master will also detect and compensate, by reducing communication step size, potential numerical instabilities, again leading to excessive and unacceptable simulation times. Phenomena of instability may grow with increased coupling strength of FMUs interface quantities and often arising from the choice of the model interface.

2 Choice of the FMU Interface

The separation of a complex building energy simulation model into subcomponents is not trivial. A natural choice for separation of the entire model into FMUs may be to keep the passive building and its physics regarding interaction with climate and user loads within the building simulation FMU. All active components such as heating, cooling, ventilation and associated equipment and control models will be in one or more HVAC-FMUs. In this article we use a single FMU with all HVAC equipment and control models written in Modelica.

2.1 Building Simulation FMU Input/Output Variables

One option for a flexible interface would be to export all relevant states like temperatures and solar radiation loads from the building simulation FMU, and import calculated

⁴<http://www.building-engineering.de>

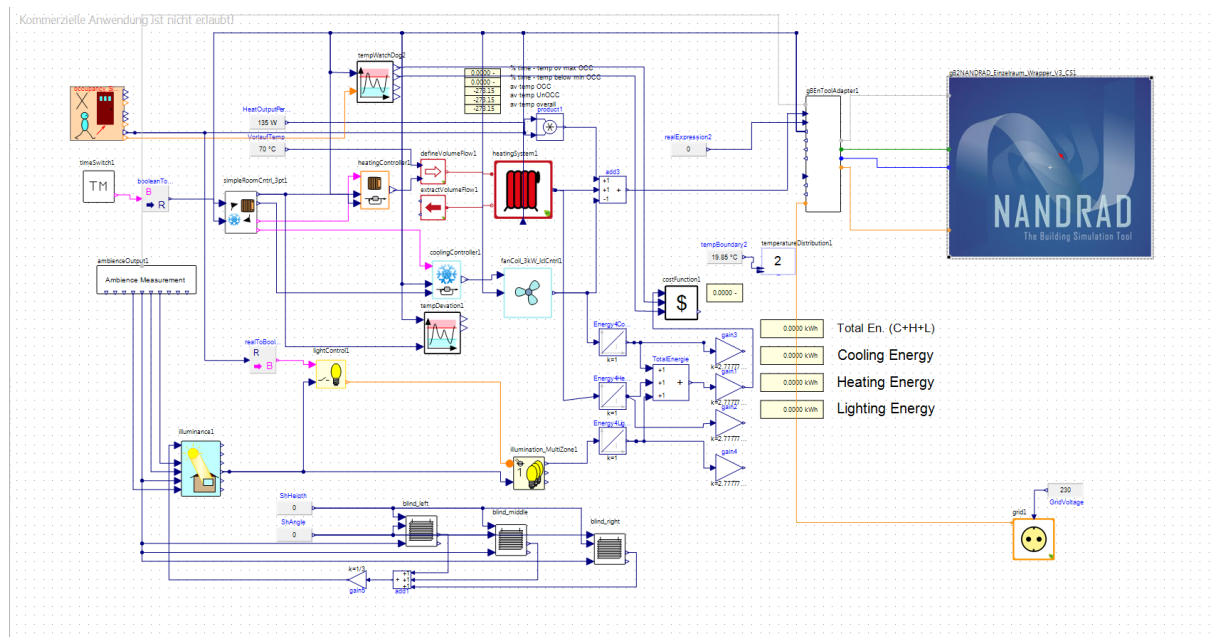


Figure 1. Usage Scenario 1: Building simulation FMU (NANDRAD) imported into Modelica environment (SimulationX)

heating/cooling loads from the HVAC FMUs. This interface can be considered a very universal interface, since any kind of heating/cooling loads can be modeled and imported as energy source to each thermal zone's energy balance. The interface defines for each thermal zone an export of mean air and operative temperature and input of convective and radiative thermal load.

The building simulation FMU includes databases for climatic loads and user behavior and related equipment schedules⁵. Hence, climatic data and schedules are additionally exported via the FMU interface. This allows consistent treatment of climatic input data in building and equipment models. Part of the scheduled user loads are also hot and cold water demand as well as user-related electric power consumption.

2.2 Convenience Adapters and Wrappers

The interface definition allows exporting and importing zonal quantities. Considering typical buildings of more than hundred conditioned zones, a large number of input/output variables need to be connected to the plant FMU. Even if the FMI standard would allow usage of vector variables, the manual connection of exported temperatures to the various input ports on the plant side would not be expedient and may lead to errors that are difficult to identify and track.

Also, when importing a building simulation FMU into a Modelica development environment the graphical representation of the inserted FMU with hundreds of ports is not suitable for practical use. Therefore, we utilize helper components that assist with mapping native FMU inter-

face quantities to Modelica library ports and buses.

Different helper components are used depending on the usage scenario:

- When the building FMU is imported into the Modelica environment, the FMU is encapsulated into a Modelica wrapper model, which internally holds the FMU and connects to the native FMU interface. On the outside it provides port and bus connectors matching the corresponding library interfaces, in our case the GreenBuilding climate, electrical and HVAC buses (see Figure 2, we use the HVAC, HotWater and Electrical port of the GreenBuilding library). This wrapper is therefore specific to each building⁶ and to the interfaced library.
- When the plant model is to be exported from Modelica into a stand-alone Co-Simulation FMU, the adapter (Figure 3) is used instead. It provides the same library-specific connectors as the wrapper, but does not connect to the building FMU. Instead, it exports and imports exactly the counterparts of the building FMU interface variables. When exporting the Modelica model, only these connectors become part of the FMU interface. Also, the connector counterparts are identically named to the building FMU interface quantities, which greatly simplifies automated connection between plant and building FMU connectors⁷. The graphical annotations of zonal con-

⁵Typically, such schedules and databases are part of the building model definition and will be generated/collected within the BIM process

⁶The native interface of the FMU changes with the number of thermal zones, or their IDs, and so does the wrapper component.

⁷Similarly, when importing a building FMU into a Modelica environment an automated matching of connectors between FMU and wrapper/adaptor model would be possible. Unfortunately, none of the currently available modeling environments supports such a procedure.

nectors with same quantities but different zone references are arranged on top of each other, thus keeping the adapter symbol compact.

Figure 3 does not show the actual connector names, but rather a physical description and associated unit (see (Paepcke et al., 2016) for a complete specification).

2.2.1 Adapter/Wrapper Configurations

The use of wrappers/adapters is a compromise between flexibility of the building model interface and easy-of-use within Modelica environments. The current specification of our adapter/wrapper Modelica component is specialized of interfacing all building zones with exactly one HVAC system model in Modelica. For other situations, the adapter/wrapper models may look different. Yet, the principle approach to provide FMU-independent connectors for the remainder of the Modelica model appears to be a promising way to avoid connecting to individual FMU input/output variables directly.

3 Parametrization and Export of NANDRAD FMUs

3.1 Configuration for FMU Export

When NANDRAD is executed as stand-alone building energy simulation model, for example to compute annual energy demand and comfort criteria, it uses a set of input files with the building model (BIM) and database elements (material data, constructions, climatic data, etc.). The input data include definitions of all zones and their heating and cooling requirements, which enables an ideal heating and cooling load calculation.

When NANDRAD is used as building simulation FMU to simulate a realistic heating/cooling system, all conditioned zones need to be connected to the heating cycle or to the electrical grid of the plant model. All zones that are part of the interface and import/export variables are given different usage scenarios, for example, heating scenario or electrical usage scenario. This information is then used during export to generate required import/export quantities and also create the internal data structures

that map FMU input/output variables to existing internal variables. Selecting the usage scenarios and selecting the corresponding zones is part of the FMU preprocessing.

3.2 Export procedure

The export procedure involves several steps:

- NANDRAD is run as stand-alone simulation to generate auxiliary information needed for parametrization of the HVAC/plant model, for example the heating/cooling design day calculation.
- The NANDRAD solver initialization is used to generate the variable dependency information, which is stored in the `modelDescription.xml` file.
- The `modelDescription.xml` is composed (included data for ModelExchange and Co-Simulation and the FMI v2 functionality).
- All referenced databases are collected. All input files, the pre-compiled NANDRAD dynamic library (with implemented FMI functionality), and additional dependent libraries⁸ are copied. Finally, the FMU archive is created.
- Modelica wrapper and adapter models (.mo files) are generated individually for the current building project.
- A report including zone naming, dimensions, unique IDs and heating/cooling design loads is written to be used during configuration of the HVAC component model, and for automatic Modelica model generation scripts.

During export, compilation of source code is not necessary and the model initialization and the design day calculation are usually very fast, except for large buildings with several hundred of zones. The auxiliary files are provided separately from the generated FMU.

⁸Depending on the target platform, different libraries are copied. Currently, one NANDRAD FMU holds only binaries for one platform Win32, Win64, Linux64, Darwin64 at a time.



Figure 2. Modelica wrapper encapsulates NANDRAD FMU and provides collector ports for climate, HVAC and electrical quantities

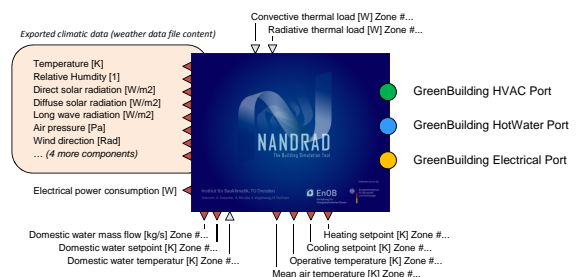


Figure 3. NANDRAD adapter provides Modelica collector ports as well as input and output variables identically named as the building FMU ports

Modeling modern complex integrated buildings will require much more data being commonly used by the building simulation model and HVAC system model. Hereby, the procedure of using BIM-data for consistent parametrization of all model components is desirable and an ongoing research issue.

4 NANDRAD FMU Calculation Functionality

4.1 State-based model evaluation and time integration in NANDRAD

When the building simulation engine NANDRAD was developed at the IBK, its design was heavily influenced by the first version of the FMI for ModelExchange standard. The entire physics evaluation is encapsulated within a state-based model object, whose state changes only by modification of the time point or conservative quantities (solution variables). After spatial discretization of all partial differential equations within the building model, a large sparse system of coupled ordinary differential equations is assembled. The time integration is then performed using our own integration framework, which incorporates the SUNDIALS:CVODE solver (Hindmarsh et al., 2005). Internally, the CVODE integrator is called by the framework for each integration step at a time. It selects/predicts a suitable integration time step, performs a modified Newton iteration⁹ and upon convergence or error test failure reduces integration step until an acceptable solution is found. Note, since integration step sizes are exclusively determined by the integrator engine, synchronization with communication intervals needs to be addressed.

Figure 4 illustrates the architecture of the stand-alone NANDRAD solver. The physical model implementation is encapsulated in a model object which has similar access functions as the ModelExchange specifications require. Therefore, the ModelExchange FMU interface implementation is only a thin layer around our physical model. Our integration framework calls one of the supported time integration methods in a step-wise manner. This core loop, which also signals successful steps (`stepCompleted()`) and tells the model to write interim outputs (`writeOutputs()`), is partially replaced by the Co-Simulation master.

4.2 Implementation of the `doStep` functionality

When NANDRAD runs as a simulation slave, the time integration is now interrupted at the end of communication interval and control is returned to the master. Since internal integration steps may not match interval end, we choose to limit the internal integration step size so that the communication interval is not exceeded. However, this

may lead to situations, wherein the last integration step before end of communication interval is much shorter than previous integration steps¹⁰.

An alternative to limiting the last integration step would be to allow the integrator to take its natural step size. In the case of CVODE, the solution at communication interval end could be easily obtained by backward interpolation. The CVODE integrator could now be re-started with that interpolated solution in the next communication interval. However, such a restart would destroy the history within the multi-step BDF method, effectively forcing CVODE to restart integration from first order with very small time steps. This approach leads to unacceptable simulation times and cannot be recommended.

4.3 Retrieving and restoring the FMU state

The aforementioned functionality is sufficient for executing NANDRAD as FMI for Co-Simulation version 1. However, as soon as the Co-Simulation master is using an iterative or error controlling algorithm, the slaves must be repeatedly set back in time (see, for example (Clauß et al., 2017)). The master needs to retrieve and restore each FMU's state.

Within NANDRAD the internal state is stored in several solver components:

- State of the integrator (time point, state variables and Nordsieck history array, counters, control variables)
- State of linear equation system solver, in case of GMRES only control variables
- state of Jacobian, since with modified Newton algorithm it is only infrequently updated
- state of preconditioner (part of Jacobian matrix and in case of ILU preconditioner also the factorized representation)
- integral model states (integral outputs, state of hysteresis loops etc.)

The data structures are typically very fragmented. The serialization implementation within NANDRAD creates a continuous memory array and then copies all data members into the array, hereby advancing an insertion pointer after each copy operation. With the use of C macro definitions, the entire serialization, deserialization and size computation functionality is only coded once, thus ensuring binary compatibility and improving code maintenance (Nicolai and Paepcke, 2016).

Additionally, the ability to serialize the entire state of model and integrator into a continuous memory block enables implementation of the `fmi2Serialize()` and `fmi2Deserialize()` functions.

⁹Within each Newton iteration the large sparse equation system is solved using a Krylov-subspace method with NANDRAD-specific preconditioner.

¹⁰Drastic changes in time step sizes typically lead to invalidation of Jacobian matrix information, with the related overhead of re-composing and factorizing the Jacobian.

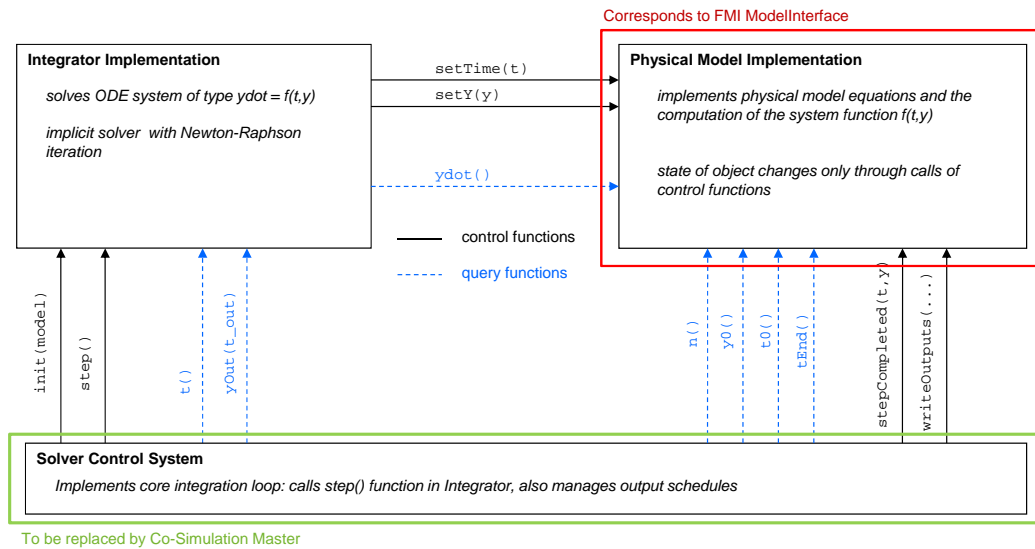


Figure 4. Core components of the NANDRAD stand-alone solver.

4.4 Integration of FMU inputs and outputs in the NANDRAD building model

The physical model of NANDRAD is internally implemented by means of interconnected state-based model objects. We allow a single model object calculation to depend on other model results. For example, room air balance is encapsulated in a single model object that requests heating and cooling load as input quantities. In turn, the power of controlled heating and cooling elements reacts on thermal response of the zone. In complete, NANDRAD owns several model objects with arbitrary interconnections that form an unstructured graph. Indeed, the states of all these models must be updated in the correct order whenever a solver state change is registered. For this purpose we cluster the model graph into nodes with cyclic and sequential connections first and order it afterwards during initialization process. As a result, all model objects appear stacked with respect to their evaluation chronology. This strategy guarantees all internal states to be current whenever an update is necessary because of changes of solver states or solver time.

This modeling concept can be easily extended to FMU inputs and outputs. In detail, we encapsulate all FMU quantities into an FMU import and an FMU export model object. The export model transfers all required output variables from the building model towards the FMI. The import model caches FMI input quantities, such as heating and cooling loads, and provides them just like calculation results to other internal model objects. This structure enables the model initialization to sort FMU inputs and outputs to the correct position inside the model object graph. For evaluation of all models depending on FMU input we store the position of the FMU import model object within the graph. In the case of update due to FMU input changes only the corresponding dependent nodes of the

model graph are taken into account. This allows a model evaluation/update with only small computational effort.

To achieve good simulation performance we follow the concept of lazy evaluation: the call of `fmi2SetReal()` does not enforce an update of dependent building model objects but temporarily fills a data container. Only at the beginning of each communication step the container values are copied and the model evaluation is triggered. So, during each communication interval the model results as well as FMU outputs are consistent to the FMU inputs.

5 Application Cases

The procedure of creating and parametrizing building and equipment models and exporting FMUs has been tested with three application cases of different scales: an office room (1 conditioned zone, 383 ODEs in the building simulation part), a family row-house (2 heated zones, 502 ODEs in the building FMU) and a large apartment complex (178 conditioned zones, 23220 ODEs in the building FMU).

In this article we will only look at the first case and discuss the observed behavior with respect to the different Co-Simulation master algorithms employed. Note, it is generally possible to reduce the number of ODEs, which mostly result from spatial discretization of envelope/interior constructions, by adjusting the grid-generation parameters. As with most spatial discretization techniques, such a variation should be complemented by sensitivity studies which are beyond the scope of the article. In the test case we selected medium-fine discretization settings, leading to the reported number of elements.

5.1 Office Room Model Setup

In this model, the office is represented by four enclosing wall/floor constructions, where internal walls with same behavior are lumped into one. The only external wall

faces west and contains a large window. The old construction is made from massive lime sandstone (simplified as single layer construction) with poor thermal insulation properties. The HVAC system operates with ideal control for a heating and cooling demand calculation, dynamic daily schedules and low setback temperature on weekends. During the week, schedules distinguish between daytime and nighttime use (occupied/not-occupied). Corresponding thermal loads are computed by the plant model and imposed onto the room energy balance. Infiltration is considered with standard settings. Since the heating system is modeled in an idealistic way. The interaction between room response and heating system is very strong, leading to stiffly coupled system.

5.2 Reference Solution and Verification Procedure

Initially, for this problem a Modelica-only solution existed, yet with simplified building representation. The results of this calculation can be used for plausibility tests (Figure 5).

A correct reference solution with detailed building physics can be obtained using the ModelExchange-functionality of the building FMU and the Modelica-based equipment model. Generation of a correct reference solution depends on the following assumptions:

- building FMU correctly implements the ModelExchange interface and internal room physics,
- ModelExchange master correctly implements time integration with error checking,
- Modelica model is correctly solved within the environment.

As Modelica simulation environment and ModelExchange master we use the SimulationX¹¹ software, which has a comprehensive quality testing procedure to ensure correctness of Modelica and FMI master implementation. Our own NANDRAD implementation is tested against standard and customized dynamic test scenarios, and also compared to the thermal room model THERAKLES¹².

Given these testing procedures, we are confident that the results of the ModelExchange calculation will be correct and can be used to evaluate the quality of the Co-Simulation runs.

5.2.1 ModelExchange Reference Simulation

A first step in generating this reference solution was to export the NANDRAD model as FMU for ModelExchange. Since NANDRAD exports a single FMU with both Co-Simulation and ModelExchange specification, the same

¹¹<https://www.simulationx.de>

¹²See <http://bauklimatik-dresden.de/therakles>. THERAKLES was used in the test case as plugin alternative to NANDRAD and gave the same results for this single-zone model problem.

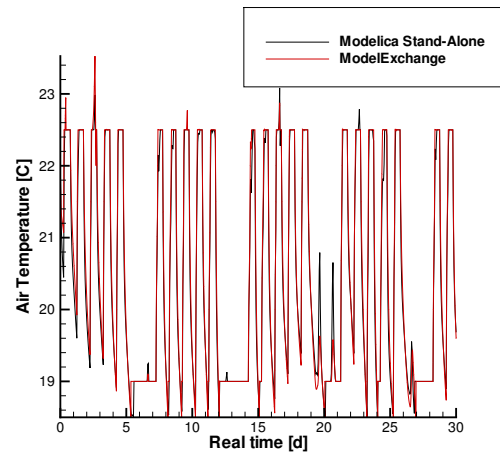


Figure 5. Comparison of reference ModelExchange solution with Modelica-only variant, using SimulationX for both simulations

FMU is used for later Co-Simulation testing. The FMU was imported into SimulationX (version 3.7), connected manually to the plant model and simulated with the SimulationX internal solver (CVODE solver, sparse Jacobian).

The fully coupled ModelExchange simulation is a fairly small problem and was simulated in acceptable time. The results were then compared to the stand-alone simplified Modelica variant and showed good agreement (Figure 5). We also did not expect much difference, since the single-layer constructions with high thermal conductance will be reasonably well approximated by the mean thermal resistance approach used by the Modelica model.

For the office room model, we use the ModelExchange reference solution for the subsequent Co-Simulation tests. However, for larger buildings (more than one hundred zones) the procedure of using ModelExchange for simulation fails, because already the symbolic analysis takes excessive time. For example, in the case of the apartment complex the symbolic analysis was not yet finished after three days.

5.3 Co-Simulation Variants

For the Co-Simulation approach, we exported the Modelica plant model from SimulationX into an FMU for Co-Simulation, version 2, hereby using the CVODE integrator option and numerical Jacobian generation method. Then, we ran the coupled simulation between the plant and building simulation FMUs with MASTERSIM¹³. We developed this open-source Co-Simulation master implementation specifically for testing and evaluation of building simulation applications.

5.3.1 Non-Iterating Gauss-Jacobi with Fixed Step-Size (only FMI v1)

The most trivial approach to Co-Simulation is the use of the Gauss-Jacobi algorithm without iteration and fixed

¹³<http://mastersim.sourceforge.net>

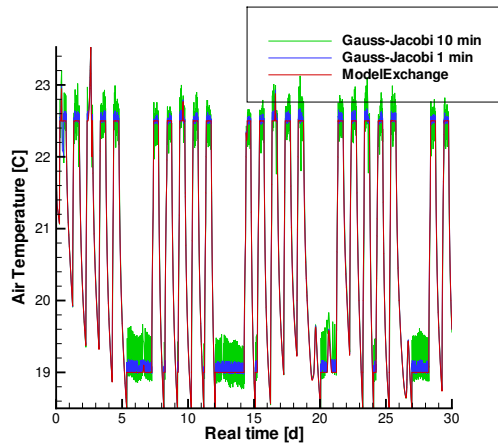


Figure 6. Gauss-Jacobi, non-iterating, fixed communication step sizes (ModelExchange results from SimulationX, Co-Simulation results calculated with MASTERSIM)

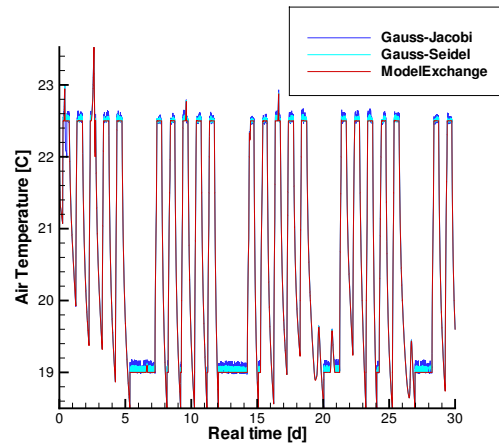


Figure 7. Comparison of non-iterating Gauss-Jacobi and Gauss-Seidel calculation for a fixed communication step of 1 minute (Co-Simulation cases done with MASTERSIM)

time step. This algorithm does not require any FMI v2 features and is thus the most compatible.

For the office room case the simulation was done with a fixed communication step size of 10 and 1 minutes. For both variants, stability problems appear. Figure 6 shows computed room mean air temperatures for the first weeks of the annual simulation.

The two Co-Simulation variants are plotted vs. the reference solution and clearly show unphysical oscillations, even at times when heating setpoints are constant. Source of the problem is the delayed reaction of the plant FMU on changes in room air temperature. Whenever the room temperature crosses the setpoint temperature during the course of the communication interval, the plant loop continues calculating based on outdated information. Specifically, when room temperature increases above setpoint temperature, the heating system still provides heat to the room based on previous room air temperature information. During cooling, the heating system remains off for too long, allowing the room air temperature to drop below the setpoint temperature. As expected, reducing the communication step size also reduces magnitude of observed oscillations.

Using SimulationX as Co-Simulation master with same time steppings gave nearly identical results compared to MASTERSIM. Thus, we have confidence in correct behavior of the building and HVAC system FMUs.

5.3.2 Non-Iterating Gauss-Seidel with Fixed StepSize (only FMI v1)

An attempt at improving the solution was made by using the Gauss-Seidel algorithm, again with fixed step size and no iteration. Hereby, the building FMU is been given updated plant FMU results when integrated in the same step. Figure 7 shows a comparison between a Gauss-Seidel and Gauss-Jacobi simulation using the same communication step size.

Despite the notable improvement, even Gauss-Seidel does not provide sufficiently accurate results. Lowering the time step will of course improve results, but at the cost of reduced simulation performance. In practical applications the user would have to guess the communication interval and refine it in the case of stability/accuracy problems. Recognizing incorrect results may not always be easy, especially since for realistic application scenarios a reference solution does not exist. Therefore, it would be desirable to automatically adjust the time step such that results are within acceptable tolerances.

5.3.3 Adaptive Communication Step Size

We implemented the step-doubling technique in MASTERSIM as adaptive communication step method (Clauß et al., 2017). Clauß discusses such an approach within in context of FMI Co-Simulation. The error tests uses the weighted root mean square norm of all communicated real variables. When this algorithm is used, all FMUs must have the capability `canGetAndSetFMUstate` and formally implement version 2 of the FMI standard. The algorithm begins with storing the current FMU states, followed by a full communication step calculation. The results are cached, FMUs are set back and two subsequent communication steps of half length are computed. The result of the single step and the double-step calculation are used as a measure for the local truncation error. With this approach, each step, even if successful, requires 3 FMU evaluations compared to one FMU evaluation without error test.

Three simulations cases were run: Gauss-Jacobi and Gauss-Seidel methods, each with only one evaluation (no iteration), and the last with Gauss-Seidel allowing three iterations. All tests were done with a relative tolerance and an absolute tolerance of 10^{-5} . The latter may be important since thermal loads, the output variables of the plant FMU, can go down to zero.

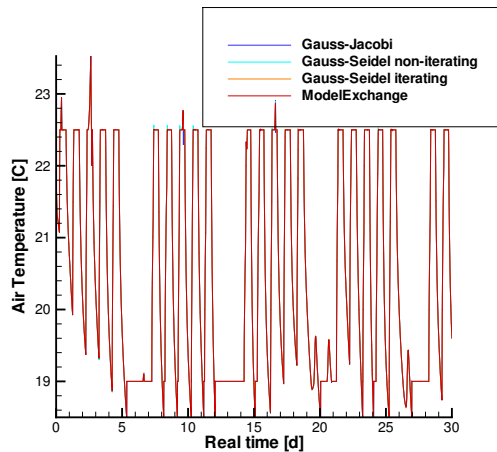


Figure 8. Comparison of non-iterating Gauss-Jacobi and Gauss-Seidel calculation variants with adaptive communication step sizes (Co-Simulation cases done with MASTERSIM)

When time step sizes fall below 1 s, iteration is disabled. This is a fallback criterion in MASTERSIM in order to avoid useless iterations in case of encountered discontinuities. Changing this value may also change performance of the simulation, but not impact accuracy of results.

Figure 8 shows the results obtained with variable step sizes for non-iterating cases. The results are now within the requested tolerance limit and are, with very few exceptions, nearly identical to the ModelExchange variant. The simulation time, however, has increased substantially compared to the incorrect fixed step variants.

Table 1 shows the statistics obtained from the three cases. For the first two cases iteration is not used, hence no convergence failures were recorded. Error test failures occurred about three times more frequent for the Gauss-Jacobi variant, which resulted in a drastic reduction of average time step sizes and similar increase of simulation time. The step sizes were sometimes reduced drastically to 10^{-7} s. Figure 9 illustrates the strong variations in time step. For the Gauss-Jacobi simulation, the time step varies permanently over several orders of magnitude. For all variants, when the heating system has been turned off at 0.75 d (6:00 pm), the time steps increase again up to the allowed maximum of 15 minutes. This is important for increasing overall simulation performance.

Interesting is the comparison between iterating and non-iterating Gauss-Seidel. Apparently, even with three iterations often a situation is encountered, that Gauss-Seidel cannot resolve. In these cases time step sizes were reduced due to convergence errors, which in turn reduced the number of error test failures. With this stability-dominated simulation case, use of the iterating Gauss-Seidel approach is not meaningful.

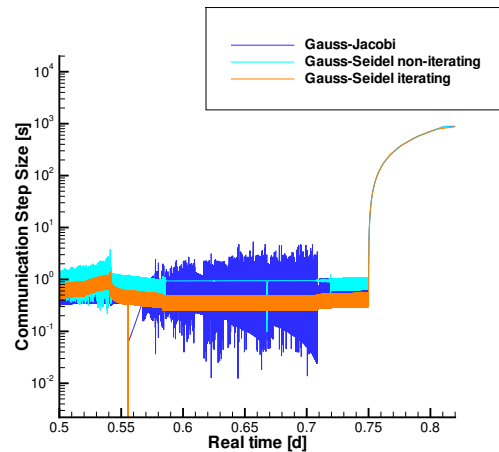


Figure 9. Illustration of step-size variation with adaptive time step methods within the first day of simulation.

6 Summary and Conclusion

We presented the tasks necessary to successfully run a coupled building energy performance simulation using the FMI standard. We discussed the physical interface between plant and building FMU, the process of generating the building FMU itself and its internal interface implementation. In realistic cases buildings may have a large number of conditioned zones, resulting in many input and output variables. Therefore, we presented an approach for improving usability by automatically generating Modelica helper components. Further, we showed one example application for a single zone model and tested different Co-Simulation algorithms for accuracy and simulation performance.

In the test case we used an ideal heating system. The strong coupling between building and plant FMU caused stability problems for fixed-step solvers. These could be controlled by use of an adaptive communication time step, based on local error estimates. The non-iterating Gauss-Jacobi method performed poorly compared to the non-iterating Gauss-Seidel method. Iteration, tested with the case of Gauss-Seidel, did not improve simulation performance. Without iteration, stability problems were detected by the error test, with iteration these stability problems often caused conversions failures. In either case communication step sizes were reduced. However, in all variants the error test and communication time step adjustment method yielded results of acceptable quality.

In our test case, the iterative Gauss-Seidel method failed frequently due to stability problems. Therefore, using Gauss-Seidel or Gauss-Jacobi iteration is not meaningful for such strongly coupled cases.

The observed behavior and conclusions drawn from the simulations are of course only an indication of general behavior. In particular, the ideal plant model and control method in conjunction with a strong thermal response of the building are definitely an extreme case. Still, success-

Table 1. Simulation statistics obtained with adaptive step simulations

<i>Method</i>	<i>Comm.</i>	<i>Steps</i>	<i>Error</i>	<i>Test Fails</i>	<i>Convergence Fails</i>	<i>Simulation Time</i>
ModelExchange	—	—	—	—	—	54 min
Gauss Jacobi non-iterating	1984803	—	590539	—	—	25 min
Gauss Seidel non-iterating	827616	—	146637	—	—	10 min
Gauss Seidel iterating	1595677	—	4003	—	809681	28 min

ful simulation was possible by use of time step adjustment, and the method and approach itself is suitable for general application.

To achieve this, the following requirements on FMU and master simulator must be fulfilled:

- the solvers within the building and plant FMU must implement an error test procedure to give consistent results,
- the FMUs must implement FMI standard version 2 with capability to set and get their states, and
- the Co-Simulation master must support communication time step adjustment based on local error estimates.

It has to be noted, though, that our conclusions are specific to the idealistic HVAC system used and observations may be different when dealing with detailed HVAC system models for modern integrated buildings.

For practical applications, overall simulation performance remains a crucial criterion. Considering the still long simulation times when applying Co-Simulation, further work is required with regard to finding suitable physical interfaces, choice of master algorithms and algorithmic parameters.

Acknowledgements

We gratefully acknowledge the support and funding received from the German Federal Ministry for Economic Affairs and Energy in the research project “En-Tool:CoSim” #03ET1215A F-002792.

References

FMI 2.0 Standard, 2014. Functional Mock-up Interface for Model Exchange and Co-Simulation.

Christoph Clauß, Kristin Majetta, and Richard Meyer. Application of Richardson Extrapolation to the Co-Simulation of FMUs from Building Simulation. In *Proceedings of the 12th international Modelica Conference*, 2017.

William.S. Dols, Wang Liangzhu, Steven J. Emmerich, and Brian J. Polidoro. Development and Application of an Updated Whole-Building Coupled Thermal, Airflow, and Contaminant Transport Simulation Program (TRNSYS/CON-TAM). *Journal of Building Performance Simulation*, 8(5): 326–337, 2014.

Alan C. Hindmarsh, Peter N. Brown, Keith E. Grant, Steven L. Lee, Radu Serban, Dan E. Shumaker, and Carol S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software*, 31(3):363–396, 2005.

Sanford A. Klein, William A. Beckman, and John A. Duffie. TRNSYS - A Transient Simulation Program. *ASHRAE Transactions*, 82(1):623–633, 1976.

Andreas Nicolai. Physikalische Grundlagen des thermischen Raummodells THERAKLES, 2013. URL <http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-102112>.

Andreas Nicolai and Anne Paepcke. Die Gebäudesimulationsplattform NANDRAD - Physikalisches Modell, Umsetzungskonzept und Technologien im Überblick. In *Proceedings of the BauSIM 2012*, 2012.

Andreas Nicolai and Anne Paepcke. Transformation der Gebäudeenergiesimulation NANDRAD mit variablem Zeitschrittlöser in eine FMU für Co-Simulation. In *Proceedings of the BauSIM 2016*, 2016.

Christoph Nytsch-Geusen, Jörg Huber, Manuel Ljubijankic, and Jörg Rädler. Modelica BuildingSystems – eine Modellbibliothek zur Simulation komplexer energietechnischer Gebäudesysteme. *Bauphysik*, 35(1):21–29, 2013.

Anne Paepcke and Andreas Nicolai. Anlagenregelung in ODE-Systemen am Beispiel der thermischen Raum- und Gebäudesimulation. In *Proceedings of the BauSIM 2014*, 2014.

Anne Paepcke, Torsten Schwan, and Andreas Nicolai. Schnittstellen für die Co-Simulationskopplung zwischen Gebäude- und Heizungsanlagensimulation. In *Proceedings of the BauSIM 2016*, 2016.

Per Sahlin, Lars Eriksson, Pavel Grozman, Hans Johnsson, Alexander Shapovalov, and Mika Vuolle. Whole-building simulation with symbolic DAE equations and general purpose solvers. *Building and Environment*, 39:949–958, 2004.

Michael Wetter. A Modelica-based model library for building energy and control systems. In *Proceedings of the 11th IBPSA Conference*, 2009.

Michael Wetter, Christoph van Treeck, and Jan Hensen. IEA EBC Annex 60: New generation computational tools for building and community energy systems. Technical report, Lawrence Berkeley National Laboratory, 2013.

Aspects of FMI in Building Simulation

Dipl.-Ing. Torsten Schwan¹ Dipl.-Ing. René Unger¹ B.A. Jörg Pipiorke²

¹EA Systems Dresden GmbH, Germany, {torsten.schwan, rene.unger}@ea-energie.de

²ESI ITI GmbH, Germany, Joerg.Pipiorke@esi-group.com

Abstract

Building physics and HVAC system simulation have become an important usage scenario of the Modelica modeling language and related simulation tools since the publication of first adequate libraries (Wetter, 2009). In 2010, the tool independent standard FMI was published in version 1.0. It enables the exchange of models between different simulation tools and even different modeling approaches. Although, automotive industry mainly initiated the FMI development, it can extensively benefit building simulation, too.

This paper describes four completely different applications of FMI in the building simulation environment which even extend the basic idea of simple model exchange. This includes the description of developed models as well as additionally required software components implementing the FMI standard.

Keywords: *Building Simulation, FMI, Model-in-the-loop, Controller Test*

1 Introduction

The versatile modeling language Modelica enables engineers to model and simulate complex multi-physical problems in a wide range of different domains. Although it has been growing in the automotive industry during the late 1990s and 2000s building engineers (mainly HVAC and building physics specialists) more and more use Modelica for their purposes as well.

Since the first publications of building physics and HVAC system related Modelica libraries in 2009, a wide range of different modeling approaches have been developed. Most of them are freely available under open-source license. Some are more commercial and only partly open-source.

One of the main open-source representatives of building modeling libraries is the Modelica *Buildings* library of LBNL (Wetter, 2009). This constantly refined library is based on Modelica *Fluid* library. It focuses on detailed modeling of heating, ventilation and air conditioning systems together with detailed thermal room models. Further open-source library examples with similar modeling approaches and objectives are the Modelica *Building* library of RWTH Aachen (Lauster, 2012) and the Modelica

BuildingSystems library of UdK Berlin (Nytsch-Geusen et. al., 2012). Today, these libraries as well as further similar derivatives are mainly dedicated to the worldwide growing academic community of building systems engineers and researchers.

Further commercial but partly open-source libraries like ESI ITI's Green Building (Unger et. al., 2012) and its latest derivative Green City (Schwan et. al., 2016) focus on integrated planning of sustainable and profitable solutions of buildings' and even whole cities' heat, cold and power supply.

Besides the Modelica language and derived libraries the Modelica Association has been supporting the MA Project Functional Mockup Interface (FMI) since its publication in 2010 as well. The FMI 1.0 standard was developed by 29 partners in the MODELISAR project between 2008 and 2011. In this ITEA 2 European project mainly the automotive industry forced the development of a tool independent model exchange standard. The FMI 2.0 standard followed these first developments with its publication in 2014.

The FMI standard in both versions enables engineers to exchange or co-simulate dynamic models of different domains. This way, FMI can extend the field of application of building and energy system simulation. It can furthermore help to overcome current and future limits of simulation.

Integrated planning processes more and more use accurate building physics and HVAC system models based on any kind of public or in-house library. Their main task is currently the simulation of different variants of complex building and energy system structures. This way, engineers make feasibility studies and profitableness analyzes of different system configurations. This application of building systems related Modelica libraries does not basically require FMI.

However, there is a wide range of further fields of application and benefits of Modelica in the building simulation environment. This paper describes four sample applications of FMI during integrated planning processes.

The first example supports the development of a high-level building control system including weather forecast and user prediction to optimize ecological footprint of a sophisticated multivalent HVAC system in a school and sports complex. This way, FMI

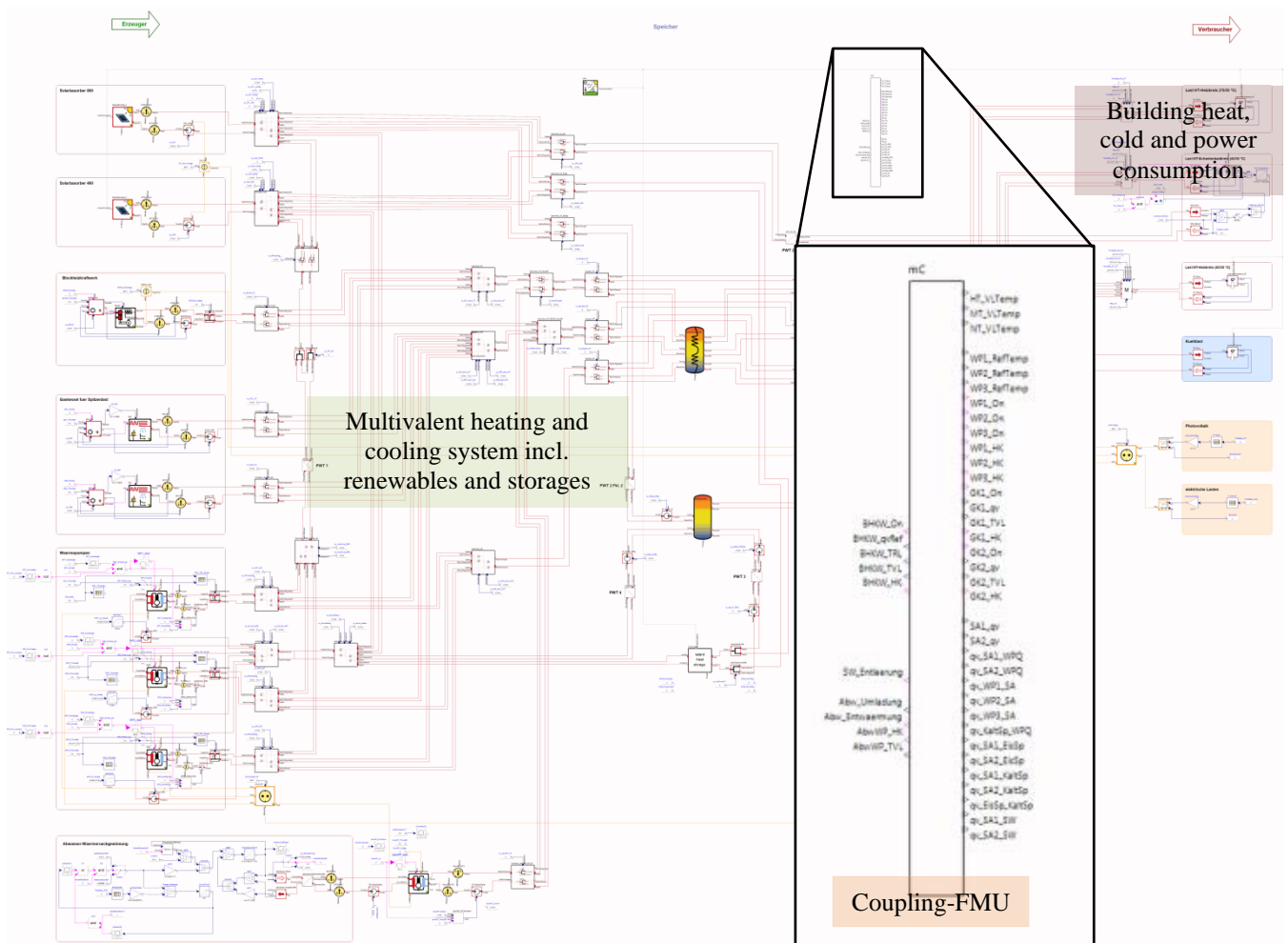


Figure 1: Modelica simulation model of the school and sports complex including coupling-FMU to external controller software (Wicke et. al., 2014)

provides the basics of a software-in-the-loop test stand of iteratively learning control software.

In the second example FMI provides the communication between a real-world HVAC system component (i.e. micro combined heat and power unit) and a complex model of the virtually connected building and hydronic heating system. This way, FMI represents the basic part of modern hardware-in-the-loop test stand.

The third example uses FMI to integrate a fast-calculating simulation model in a complex virtual power plant controller. In this model-in-the-loop structure the functional mockup unit of the Modelica model helps to identify optimal operation strategies of complex diversified power plants.

The last application uses FMI to combine advantages of different simulation platforms including individually optimized numerical solvers. This way, FMI couples highly-optimized hygrothermal multi-zone building models with easy-to-use Modelica HVAC system models. In this case, FMI helps to separate stiff ODE systems with heavily varying time constants.

2 Example 1 – Software-in-the-Loop

In a small town in northern Bavaria (Germany) a local architect wants to transfer the local 1970's school and sports complex to a future 2040's energetic level. HVAC engineers therefore planned a sophisticated multivalent heat, cold and power supply system including heat pumps, cogeneration units and backup gas-fired condensing boilers as heat supply. Main heat and power source are large-scale solar thermal absorbers and photovoltaic fields. Besides an integrated waste-water heat recovery system the integration of three different storage types (stratified heat storage, cold storage and ice storage) helps to balance daily and seasonal differences between energy consumption and production (Wicke et. al., 2014).

However, this quite complex HVAC system additionally requires smart control algorithms because of the higher degree of freedom. But such iteratively learning, optimality-based control software needs sufficient testing.

The developed controller and optimization software is based on an existing Python Framework. To provide the software engineers a suitable software-in-the-loop

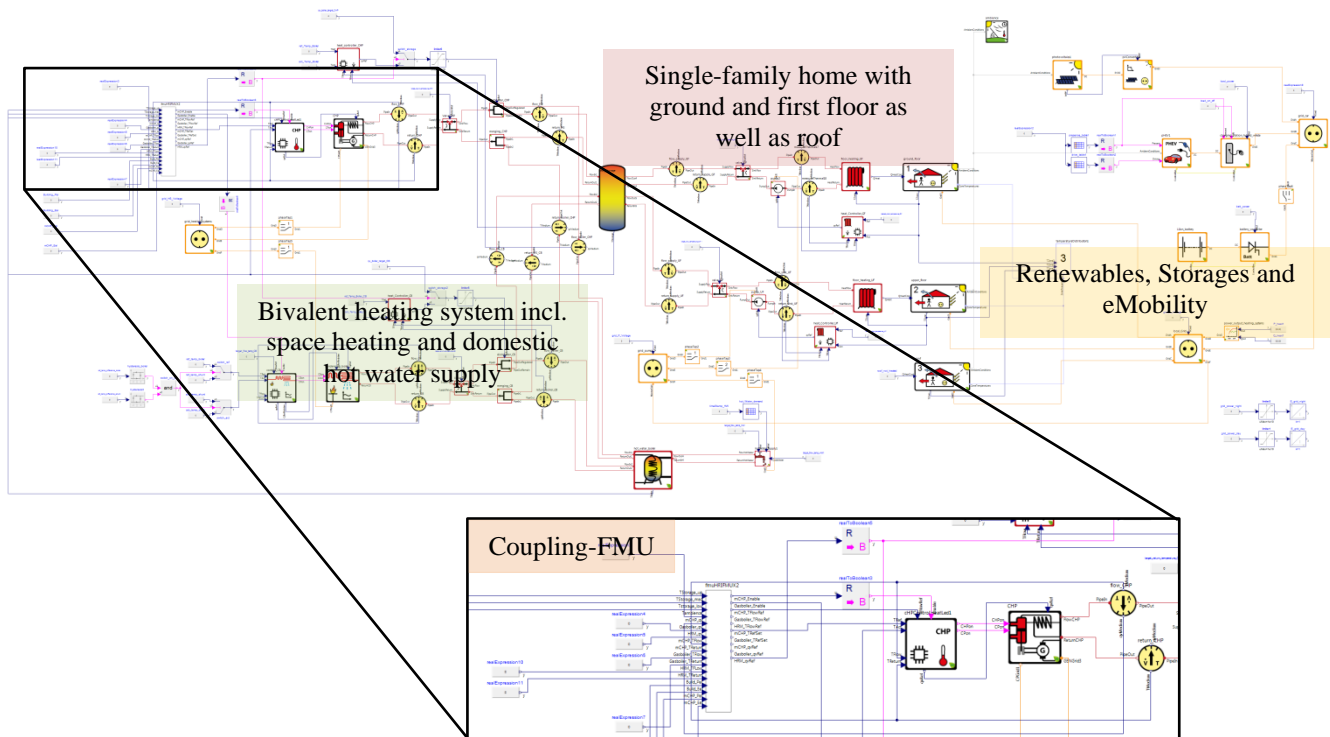


Figure 2: Modelica model the single-family-home including coupling-FMU to a real-world mCHP (Unger et. al., 2012)

test environment, a Modelica model of the developed HVAC system including the building complex's heat, cold and power demand is coupled to this Python-based controller software. This software-in-the-loop test bench uses the pyFMI-framework in the software and an automatically created coupling-FMU in the model (c.f. Figure 1) to establish a variable step size communication via TCP/IP protocol between model and software. This way, controller software and simulation model can run on different computer devices or even at different places.

The generation of the coupling FMU runs automatically using newly implemented FMU generator software in java. This software uses an external csv-file to define the required coupling interface (inputs and outputs of coupling FMU) between controller software and test model.

Software-in-the-loop tests of new controller software require extensive scrutinizing regarding internal controller timings, especially in combination with PI or PID controllers. The integrator time constants have great influence on later robustness. Building simulation models normally run (much) faster than real-time to provide sufficient results (at least one year) within adequate time periods. Those time constants therefore have to be adapted regarding communication time steps as well as a constantly synchronized real-time factor.

3 Example 2 – Hardware-in-the-Loop

Existing complex building structures, like the school and sports complex in Figure 1, require smart solutions

for energetic renovations. But also smaller buildings, like simple single-family homes, require holistic optimization approaches of heat and power supply. Photovoltaic and cogeneration units this way enable massive reductions of overall annual power consumption with manageable investment costs.

Photovoltaic (PV) modules provide renewable power and small cogeneration modules (so-called micro combined heat and power units – mCHPs) utilize synergy effects of heat and power production.

To enable a wide range of different system configurations as well as an ongoing system optimization CHP and PV manufacturers have to constantly improve their products. This includes controller software as well as hardware components. But reliable developments again require sufficient testing. Hardware tests of single system components can use simple test stand configurations (e.g. mCHP – heat storage, controlled recooling) and generically calculated heat and power load profiles. But dynamic tests of hardware and software with evaluated real-world-conditions need further expenses.

This way, Modelica models in combination with FMI can again help to provide an easy-to-use platform for hardware-in-the-loop tests. Figure 2 shows a simple example of a developed single-family home model for a hardware-in-the-loop test of a small mCHP. This single-family home model includes heat and power consumption of a small house (3-thermal zones), renewable power production and storage by photovoltaic modules, a battery and a connected eVehicle charging infrastructure. Heat is supplied by a 2.5 kW mCHP with 1.0 kW power output and a peak-

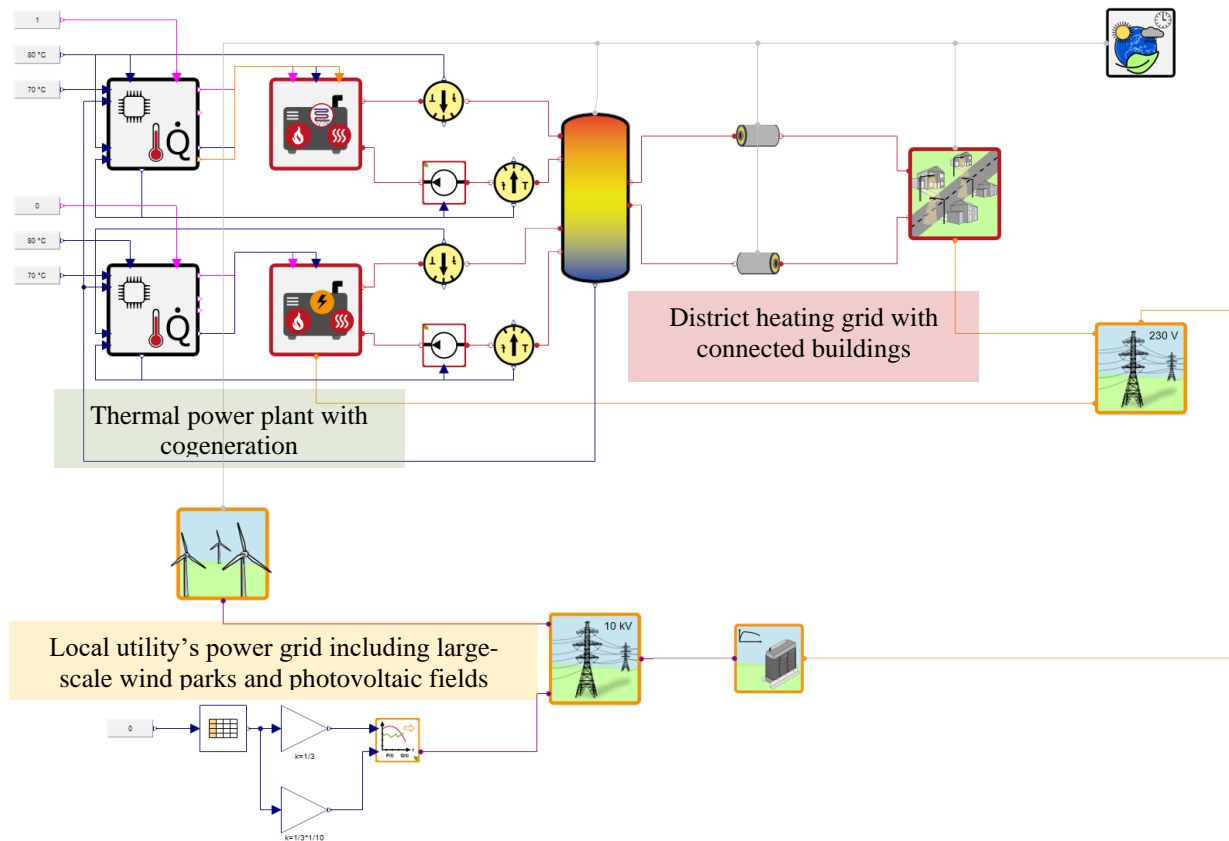


Figure 3: Modelica model of a simple district heating grid with cogeneration plant and superior electric grid connection (Schwan et. al., 2016)

power gas-fired condensing boiler (Unger et. al., 2012).

The real-world counterpart of the mCHP is connected to the model via a specific coupling FMU. This again automatically created FMU connects real-world measurement sensors (heat and power sensors of the mCHP) with the model. Measured heat and power is added as additional heat and power supply to the building. The mCHP controller gets temperature measurements of the simulated building (ground and first floor) and the heat storage to start and stop the real-world mCHP engine. Furthermore, the mCHP controller decides to run the simulated peak-power boiler if reference temperatures are underrun.

The required coupling FMU (c.f. Figure 2) couples Modelica inputs and outputs in the model with real-world digital and analogue signals of the mCHP controller. Therefore, the FMU internally converts model variables into TCP/IP protocol compatible signals. These signals are interchanged between the simulation computer unit and the local PLC (Programmable Logic Controller) of the hardware-in-the-loop test stand. This PLC directly communicates with the mCHP controller via a system specific communication bus (e.g. Modbus). Heat supply to recool and power supply to test stand's grid are measured with electronic sensors which send back the measurement data to the model.

This configuration represents a simple closed-loop hardware-in-the-loop test stand based on Modelica models and FMI. Again timing balance between real-world-time and simulation time is highly important. In opposite to the software-in-the-loop approach a hardware-in-the-loop test stand cannot easily be accelerated. It requires rigid real-time synchronization between model and device under test. This avoids the adaption of controller-internal timing constants.

Fortunately, building physics and HVAC system models mostly are (much) faster-than-real-time. Model acceleration is not necessary. The model even has to be stopped after each communication step to synchronize simulation time and real-world time.

4 Example 3 – Model-in-the-Loop

Modelica models as well as the Functional Mockup Interface standard will be a major part of future sophisticated test stands (both hardware- and software-in-the-loop) in the building sector. However, both provide more potential to improve the current situation of planners and engineers in this environment.

FMI was developed to provide an independent model exchange and co-simulation standard. Besides using FMI as a pseudo data exchange interface, Modelica models of sophisticated building and HVAC system structures can also be exported as standalone FMUs.

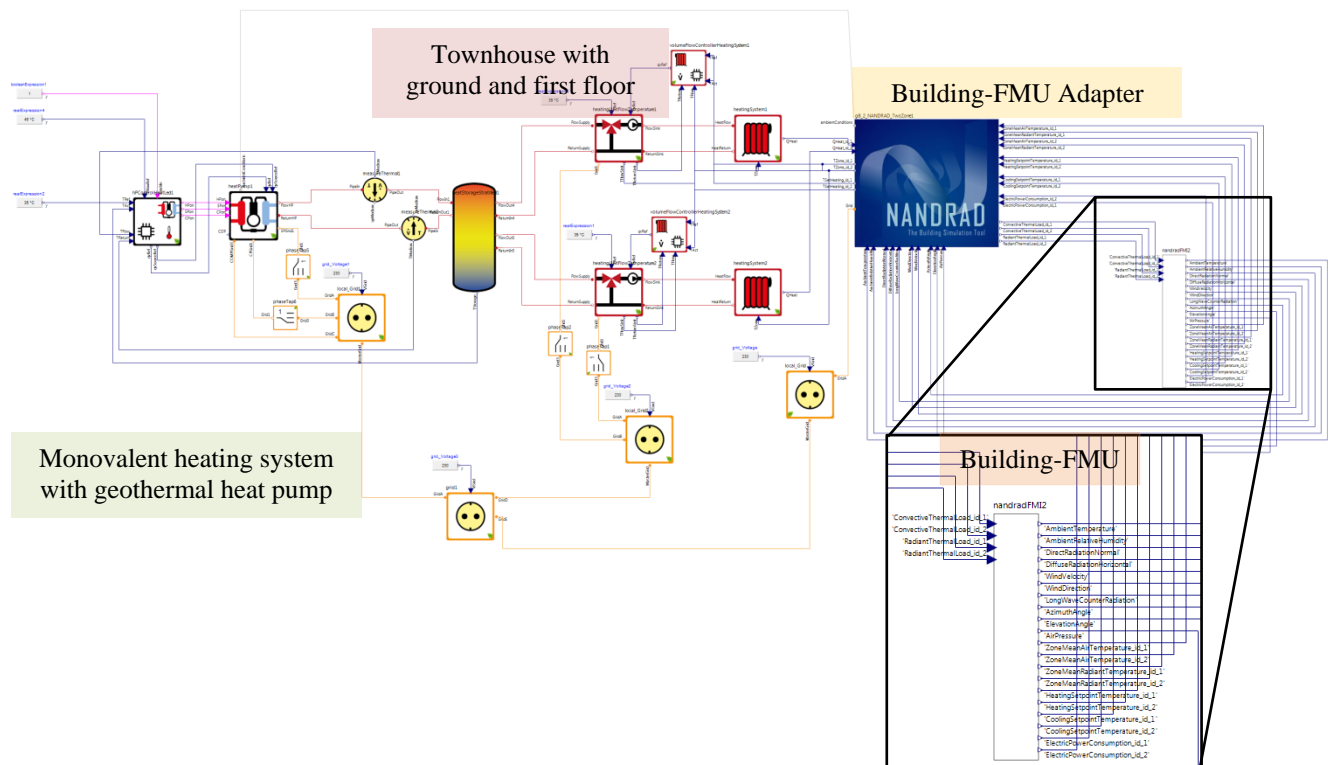


Figure 4: Coupling of a Modelica HVAC system model (SimulationX/Green City) with a TU Dresden building-FMU based on the building physics simulation tool Nandrad (Paepcke et. al., 2016)

The integration of these FMUs in upcoming model-in-the-loop controller architectures (e.g. Virtual Power Plant Controller – c.f. Schwan et. al., 2016) helps building engineers to design optimality-based controllers which can increase overall system efficiency without major extra investment costs.

Figure 3 shows such a simple Modelica model based on ESI ITI's Green City library. It represents a highly-simplified model of a cold district heating grid with decentral heat pumps, a thermal power plant with cogeneration and the local utility's power grid including wind parks and photovoltaic fields. This system is a role-model of a future part of a virtual power plant which provides balancing power to the transmission grids via energy exchange (EEX).

Available heat storage capacities in the district the heating grid provide virtual storage capacities for renewable power surplus of wind parks or photovoltaic (i.e. negative balancing power). Furthermore, decentral heat pumps can partly run as renewable energy dump as well by utilizing each building's thermal capacity. In times of grid deficits, the district heating grid reduces the power consumption of the heat pump and the cogeneration plant provides positive balancing energy.

These strategies will help to decarbonize overall energy supply of buildings in the future. However, such concepts require major investments and refinancing is not possible with diminishing energy profits. But the sale of balancing energy at the energy exchange (c.f. EEX) will help to partly compensate

resulting add-on costs. This requires though highly-accurate prediction of available storage capacities.

The integration of accurate but fast calculating simulation models of those grids in energy trading tool chains and corresponding system controllers (e.g. controller of a cogeneration plant) increases profit reliability at the balancing energy exchange. Therefore, these models are exported including a suitable numerical solver as independent FMUs.

Then, these FMUs are simulated for comparatively short time periods (i.e. few days ahead) to identify maximum profit margins regarding different upcoming weather conditions as well as expected power and heat consumption profiles.

In this model-in-the-loop configuration the models must run much-faster-than-real-time to enable the evaluation of a great number of different input parameter sets and influencing characteristics. Synchronization is not needed because the controller only uses the accumulated simulation results.

5 Example 4 – Co-Simulation

All three previously shown FMI applications in the building sector mainly use the standard to provide sophisticated test scenarios or to optimize controller functionality. However, FMI is also applicable in this environment in its inherent manner, a model exchange and co-simulation standard.

Complex building models combine a vast number of different physical components with highly diversified

time constants. Building physics mainly represent slow processes. Heating up or cooling down walls or whole buildings need several hours to several days depending on thermal inertia and available heating/cooling system configurations. Power supply of renewables, e.g. photovoltaic modules or wind power plants, can fluctuate within several seconds or few minutes depending on cloudiness and surrounding shadings. Inverter controllers of photovoltaic, batteries or even eMobility charging stations can react within a few seconds regarding available measurements (e.g. inhabitants' individual power consumption in a dwelling house).

This great variety of time constants can cause very stiff ODE systems in one Modelica model. But FMI enables to separate models regarding different time constants or to combine the strength of different simulation tools and solvers. Coupled models can co-simulate specialized building physics models with highly optimized PDE solvers (e.g. heat and moisture transport through walls) together with fast and accurate HVAC system models in Modelica and its therefore optimized ODE solvers.

Figure 4 therefore shows the coupling between Modelica HVAC system models based on ESI ITI's Green City library coupled to a townhouse building physics model in TU Dresden's Nandrad simulation environment (Paepcke et. al, 2016). This way, a Modelica simulation environment imports the Building-FMU created in Nandrad as an additional model component. The townhouse model consists of two thermal zones representing ground and first floor. The HVAC system includes a monovalent heat pump and a heat storage which provides the heat to the two hydronic heating circuits.

However, both FMI 1.0 and FMI 2.0 only allow scalar interface variables (inputs and outputs). But especially complex building physics models require a high number of common interface variables (e.g. indoor temperatures and temperature set points of all thermal zones or rooms). Therefore, Nandrad additionally provides a building-FMU adapter which automatically connects the bus interfaces in the Modelica model with the vast number of scalar inputs and outputs of the building-FMU.

On the one hand, this approach provides an interface between a Modelica HVAC system model and an imported building physics model. Co-simulation this way uses basic master algorithms of the available Modelica simulation environments, like ESI ITI's SimulationX. On the other hand, the HVAC system model including the required adapter can be exported as standalone FMU, too. This way, co-simulation between two or more FMUs can use more specialized master algorithms (Clauß et. al., 2016) to optimize simulation speed and accuracy. This can furthermore improve the tradeoff between increase of simulation

speed by usage of several individually optimized solvers and speed reduction by adding additional communication time between the FMUs.

6 Conclusion

This paper presents an overview of different fields of application of Functional Mockup Interface standard in the building simulation environment. Building and HVAC system simulations currently become one major domain of the Modelica language including an increasing number of available academic and commercial libraries.

The Modelica community also focuses on coupling of different models as well as software and hardware components to utilize synergy effects and to extend Modelica's fields of application (e.g. BCVTB - Noudui et. al., 2014). Because of its tool independency, its industrial support and its adjustability to latest network and internet technologies, the FMI standard and its upcoming updates will help to integrate Modelica in all design, test and validation processes in the building sector within the next years.

References

- M. Wetter. A Modelica-based model library for building energy and control systems, 11th International IBPSA Conference, Glasgow, 2009.
- M. Lauster. Modelica Building Library and Building Models. Symposium on Integrated Planning and Simulation in Building Physics and Technology. Dresden, 2012.
- C. Nytsch-Geusen, J. Huber, M. Ljubijunkic, J. Rädler. Modelica-BuildingSystems – A Simulation Library of complex Building Energy Systems. BauSIM, Berlin, 2012.
- R. Unger, T. Schwan, B. Mikoleit, B. Bäker, C. Kehrer, T. Rodemann. "Green Building" – Modelling renewable building energy systems and electric mobility concepts using Modelica. 9th International Modelica Conference, Munich, 2012.
- T. Schwan, R. Unger. Holistic District Heating Grid Design with SimulationX / Green City. ESI SimulationX User Forum, Dresden, 2016.
- M. Wicke, T. Schwan, R. Unger. Model-based design of control strategies for a sophisticated building energy system in a school and sports complex. 17th ITI Symposium, Dresden, 2014.
- A. Paepcke, A. Nicolai, T. Schwan. Interfaces of Co-Simulation Coupling between Building and Heating System Simulation. Central European Symposium on Building Physics, Dresden, 2016.
- C. Clauß, K. Majetta, R. Meyer. Development of Simulator Coupling Algorithms using FMI Interface for Building Simulation Applications. Central European Symposium on Building Physics, Dresden, 2016.
- T. S. Noudui, M. Wetter. Tool coupling for the design and operation of building energy and control systems based on the Functional Mock-up Interface standard. 10th International Modelica Conference, Lund, 2014.

Application of Richardson Extrapolation to the Co-Simulation of FMUs from Building Simulation

Christoph Clauß, Kristin Majetta, Richard Meyer

Fraunhofer IIS EAS, Zeunerstraße 38, D-01069 Dresden, GERMANY

christoph@clauss-it.com, {kristin.majetta, richard.meyer}@eas.iis.fraunhofer.de

Abstract

The application of the FMI technology gains ground in building simulation. As far as specialized tools support the FMI simulator coupling becomes an important option to simulate complex building models. Co-simulation needs a master algorithm which controls the communication time steps as well as the signal exchange between FMUs. Often a constant communication step size is applied chosen by the user. The Richardson extrapolation approach allows variable master step sizes. An extension of this approach is presented, and the method is applied to both academic test examples as well as examples of building simulation which co-simulate FMUs from NANDRAD and SimulationX. Although variable step size control could improve the performance this cannot be observed at the building simulation examples presented. But Richardson extrapolation turns out to guarantee finding an appropriate step size at the prize of downgraded performance.

Keywords: Building Simulation, FMI, Co-Simulation, Richardson Extrapolation, Variable Time Step Size

1 Introduction

In order to reduce the primary energy production by both reduction the consumption in buildings and growing the portion of renewable energy a much higher knowledge of the dynamic energy and mass fluxes is essential. Especially the daily and hourly fluctuations of sun and wind based energy generation require detailed dynamic considerations by simulation. Since the first publication of the FMI standard well established simulation tools have been improved to support FMI both for model exchange and for co-simulation. This allows the combination of dedicated tools as well as their model libraries which contain results of a long period of investigations. Basing on tool as well as model combination by co-simulation a big step to generate detailed simulation results was managed.

Modern buildings typically are divided into the “proper” building (walls, roof, windows ..., thermal, hygric behavior), HVAC (heating, ventilation, air conditioning devices...), and often a central acting control software (building energy management systems, BEMS). Within

the German research project EnTool:CoSim the tools NANDRAD [Nicolai, 2012] for building simulation, and the Modelica simulation tool SimulationX [ESI ITI GmbH] for mainly HVAC and control simulation were prepared to export FMUs for co-simulation. Since the FMI standard does not offer dedicated master algorithms which control the coordinated simulation of different FMUs, master algorithms have been investigated and implemented [Bastian. 2011]. So far master algorithms with a constant step size were considered mostly. FMUs generated from SimulationX for HVAC models often have a higher performance than building FMUs generated by NANDRAD. Furthermore, different “activity ranges” can be recognized (less activity at night, weekend, less heating in summer ...). The required time intervals to be simulated can be very long (years). Often the user is overstrained to define a suitable master step size, especially if the co-simulation method shall leave the research area to be applied in building practice. These issues as well as the hope for improved performance suggested the investigation of variable step size master algorithms, and furthermore asynchronous algorithms. In this paper results of the investigation of synchronous variable step size algorithms are presented. After the introduction of Richardson extrapolation methods some small academic test examples are presented, followed by three building simulations of different complexity which apply Richardson extrapolation.

2 Algorithms

The task of co-simulation of m Simulators (FMUs) can be described according to (1) with S_i , ($i = 1 \dots m$) being the simulators. Q_i and P_i are matrices which project the output values of S_i into, and the input values of S_i out of the vector of coupling values $x(t) \in [0, T] \rightarrow R^n$. The argument $P_i x$ is missing if the simulator S_i lacks input values. In [Petridis, 2015] this description is derived, and basic solution methods for cycles are presented there.

$$\begin{cases} x = \sum_{\substack{i=1 \\ Q_i \text{ exists}}}^m Q_i S_i(P_i x) & Q_i \text{ exists} \\ S_i(P_i x) & Q_i \text{ does not exist} \end{cases} \quad (1)$$

The projector P_i takes the input values of the simulator S_i out of the coupling values x . The simulator S_i calculates its output values which are written into the coupling value vector x via Q_i . Since all coupling values x are output values of exactly one simulator, and since no output value is input of the same simulator the summarizing is possible. This is described in the first line of (1). The second line clarifies that simulators without output values also have to be called.

Due to the FMI-intention the interval $[0, T]$ is divided into communication intervals $[0, T] = \cup [t_{c_k}, t_{c_{k+1}}]$ with t_{c_k} being the k^{th} communication point, and $hc_k = t_{c_{k+1}} - t_{c_k}$ the communication step size.

The communication step size can be chosen synchronously by calling each simulator with the same step size, or asynchronously by using individual step sizes for each simulator. Otherwise, the step size can be constant, or it can vary. Both properties are independent from each other. In this paper Richardson extrapolation is applied as a method of variable but synchronous step size control.

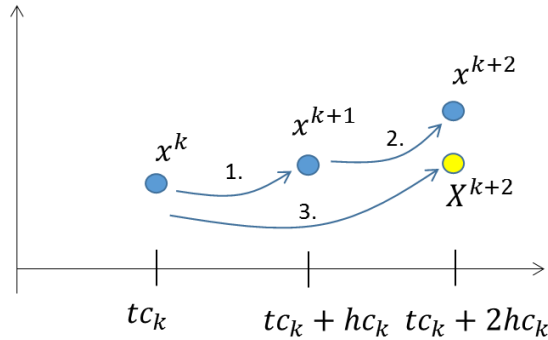


Figure 1. Richardson extrapolation

According to [Hairer, 1993], [Schierz, 2013] the **Richardson extrapolation** algorithm consists of the following steps (Figure 1):

- 1) Start at t_{c_k} , and simulate two steps using the step same size hc_k which results in the coupling variables x^{k+2} . At the first step hc_0 is provided by the user. For following steps hc_k is calculated by previous steps.
- 2) Roll back to t_{c_k} and simulate one step using the doubled step size $2hc_k$ which results in X^{k+2} .
- 3) Calculate an individual error estimation for each coupling variable ($j = 1 \dots n$) with q being the degree of the interpolation polynomial of input values:

$$errest_j = (x_j^{k+2} - X_j^{k+2}) / (1 - 2^{q+1}) \quad (2)$$

- 4) Calculate a total error estimation according to

$$ERR = \sqrt{\frac{1}{m} \sum_{j=1}^m \left(\frac{errest_j}{ATol_j + RTol_j |x_j^{k+2}|} \right)^2} \quad (3)$$

with $ATol$ and $RTol$ being absolute and relative error limits which can be chosen individually for each coupling variable.

- 5) Calculate the new step size h_{new} according to

$$h_{new} = hc_k \min \left\{ Q_{max}, \max \left\{ Q_{min}, \frac{Q_s}{\sqrt[p]{ERR}} \right\} \right\} \quad (4)$$

The heuristic values $Q_{max} \in [1.5, 5]$, $Q_{min} \in [0.2, 0.5]$, and $Q_s \in [0.8, 0.9]$ prevent too “strong” step size variations. p has to be $p = q + 1$, if there are no algebraic dependencies between inputs and outputs, otherwise $p = q + 2$ is necessary.

- 6) If $ERR \leq 1$: Both time steps are accepted, $t_{c_{k+2}} = t_{c_k} + 2hc_k$, $hc_{k+2} = h_{new}$, $k := k + 2$, go to 1)
If $ERR > 1$: Both time steps are rejected $hc_k = h_{new}$, go to 1)

Due to the steps 2) and 6) the FMUs must be able to be set back to a former communication time step. Otherwise Richardson extrapolation cannot be applied. This Richardson extrapolation algorithm assumes that the simulators S_i solve DAEs, and their output variables depend on input variables. If output variables do not depend on input variables, but on any purely time depending formula or algorithm, then intermediate time steps do not at all influence the results. Even if simulators without inputs solve DAEs their output values are in general not influenced by the master communication step size. For such components l of the coupling variables $errest_l = 0$ follows because of $x_l^{k+2} = X_l^{k+2}$. If the components l cover the vector of coupling values totally ERR becomes zero, and Richardson extrapolation cannot be applied reasonably. In such cases the step size of the output values has to be chosen such that simulators which take the output values as inputs can reconstruct the output values without losses. The step size must meet the sampling theorem [Kotelnikov, 1933]. Therefore, a maximum communication step size could be calculated if the fastest frequency component of the output values is known. Since this is not the case in general, the step size is controlled similar to classic predictor-corrector approaches [Hairer, 1993] by simply keeping the deviation from linear extrapolation small (Figure 2). The linear extrapolation of the first step is compared with the values of the second step to generate an error estimation. This algorithm called **Linear extrapolation** algorithm throughout this paper is identical to Richardson extrapolation except the error estimation formula (2) of step 3), which is replaced by

$$errest_j = x_j^{k+2} - 2x_j^{k+1} + x_j^k \quad (5)$$

The double sized step to calculate X^{k+2} is no more necessary there.

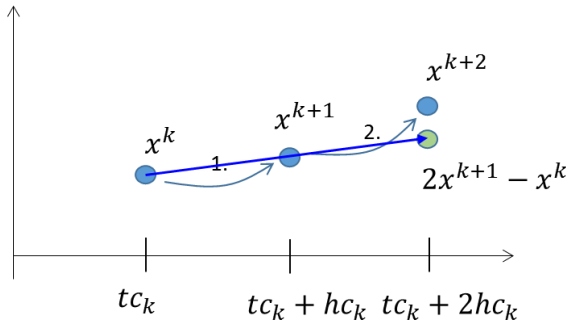


Figure 2. Linear extrapolation

This way the more communication points are inserted the more the coupling variable behavior is nonlinear. If the behavior is linear temporarily or generally then *ERR* also becomes zero. For such cases *ERR* gets a small positive minimum value. It is known that this extrapolation method on its own gives no reliable step size control if DAEs are solved. Therefore, it is combined with the Richardson extrapolation method to vanishing errors in Richardson extrapolation reasonably: At first $errest_j$ is calculated according to Richardson, formula (2) within step 3). If $errest_j$ vanishes ($|errest_j| < 1.e - 12$) then $errest_j$ is replaced by the linear extrapolation error estimation according to (5). This approach is called **Extended Richardson extrapolation** method in this paper.

In summary three methods with variable step size control are available:

- Richardson extrapolation comprising formula (2)
- Linear extrapolation algorithm comprising formula (5) instead of formula (2), without the double sized step. It does not guarantee reliable step size control in case of DAEs.
- Extended Richardson extrapolation as a combination of both of them

These algorithms are implemented in the "EAS master tool" [Petridis, 2015] which is a proprietary tool for testing master algorithms.

3 Simple Test Examples

The following simple academic examples were developed to test features of the co-simulation algorithm. They illustrate the implemented Richardson extrapolation algorithms.

3.1 Precision Test Example

The precision test example presented in [Petridis 2015] consists of three FMUs according to

Table 1. Each table line describes the equations of one FMU. The example is designed such that $y(t)$ is zero.

Table 1. Equations of the precision test example

Input	Equations	Output
x_2	$x_1 = -x_2$	x_1
x_1	$\frac{\partial x_2}{\partial t} = x_1, x_2(0) = 1$	x_2
x_2	$e^{-t} - x_2 = y$	y

The example contains one cycle which is treated by Newton's method. Since $x_2(t)$ is differentiated, and all coupling values depend on $x_2(t)$, the pure Richardson extrapolation algorithm offers correct results. Two cases of different tolerances are regarded (care for formula (3) in the algorithm):

- Case 1 (usually default values): $ATol = 1.e - 6$, and $RTol = 1.e - 4$
- Case 2 (higher precision): $ATol = 1.e - 8$, and $RTol = 1.e - 6$

The limitation of the Richardson step size variation is kept far (min 1.e-5 s, max 5 s) to not restrict the step size choice.

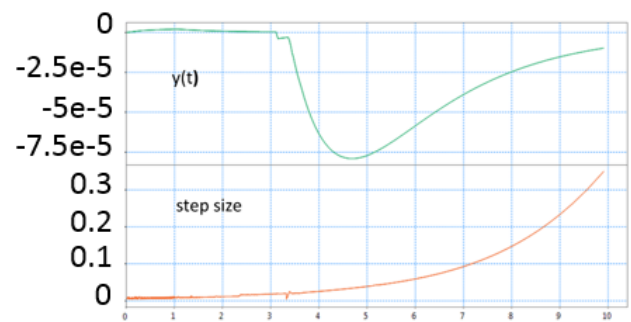


Figure 3. Precision test example results case 1

In case 1 (Figure 3) the accepted step size is growing which is expected since the solution converges to the steady state. $y(t)$ is numerically near zero.

If the tolerances demand a higher accuracy (case 2, Figure 4) $y(t)$ is closer to zero. The step size starts not significantly smaller than in case 1 but does not increase as fast as in case 1. That indicates that the smallest possible accuracy seems to be reached using the step size 0.1 s.

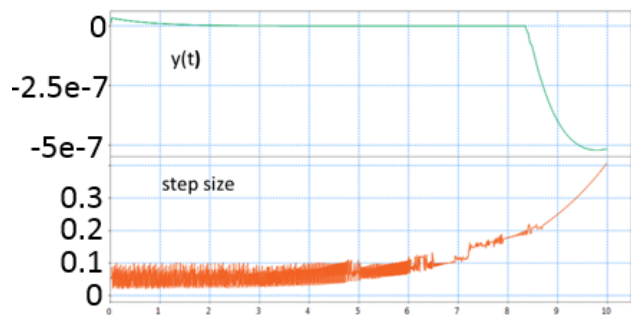


Figure 4. Precision test example results case 2

This Richardson extrapolation study advises to choose a constant step size of about 0.01 s if a constant step size algorithm should be used. Doing that Figure 5 shows the reasonable result.

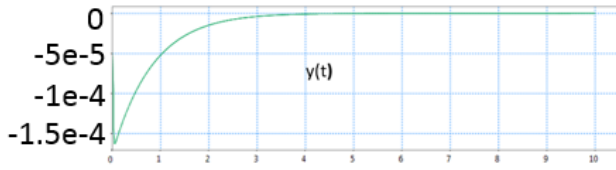


Figure 5. Precision test example using the constant step size of 0.01 s

When setting the CPU time necessary in the Richardson case 1 to be one, the normalized CPU times are listed in Table 2. Although the step size in Richardson extrapolation exceeds 0.01 s clearly in the second half of the time interval, the constant step size simulation is significantly faster than Richardson extrapolation. The reason is that Richardson extrapolation simulates the whole interval more than twice.

Table 2. Normalized CPU time comparison

<i>Richardson case 1</i>	<i>Richardson case 2</i>	<i>Constant step size 0.01 s</i>
1	1.14	0.42

3.2 Linear System of Equations

The linear system of equations with time dependent system matrix according to Table 3 was already presented in [Petridis 2015].

Table 3. Linear system of equations

<i>In</i>	<i>Equations</i>	<i>Out</i>
	$r_1 = 1, r_2 = t, r_3 = 1$	r_1, r_3, r_3
x_2, x_3, r_1	$3x_1 + (0.1 + t)x_2 + 0.2x_3 = r_1$	x_1
x_1, x_3, r_2	$0.1x_1 + 3x_2 + (0.1 + t)x_3 = r_2$	x_2
x_1, x_2, r_3	$(0.1 + t)x_1 + 0.2x_2 + 4x_3 = r_3$	x_3
x_1, x_2, x_3	$x_1 + x_2 + x_3 = y$	y

Applying Richardson extrapolation (Figure 6) the step size increases since each variable depends on the point in time only. The step size does not affect the Richardson error calculation. The values calculated at each time step are correct indeed, but there are too less time steps generated. This drawback is overcome applying the Linear extrapolation time step method instead of Richardson extrapolation (Figure 7). This example shows that the combination of both variable step size methods is necessary. The Extended Richardson extrapolation method shows the same results as the Linear extrapolation method.

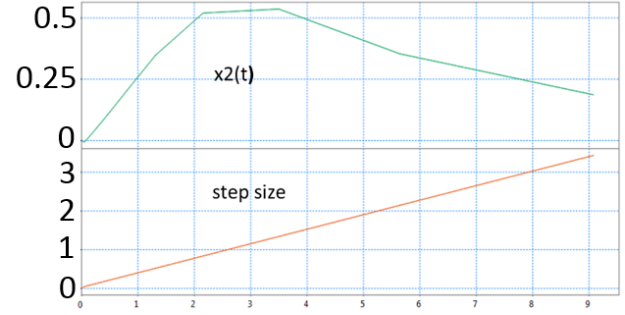


Figure 6. Linear system of equations example results applying Richardson extrapolation

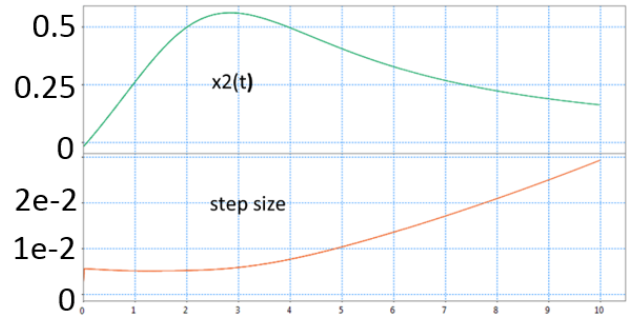


Figure 7. Linear system of equations example results applying the Linear extrapolation method

3.3 Touching Mass Example

Similar to a bouncing ball the touching mass example [Klein 2015] switches on a stiff spring as soon as the mass touches the base. This accelerates the mass into the opposite direction, and the stiff spring is switched off when the base is left.

Table 1 shows the equations of this example, separated into two parts (FMUs). The spring part contains switching as well as the calculation of the stiff spring force f .

Table 4. Equations of the touching mass example

<i>Input</i>	<i>Equations</i>	<i>Output</i>
f	$\frac{\partial s}{\partial t} = vs(0) = 10$ $\frac{\partial v}{\partial t} * 0.1 = f - 0.1f(0) = 0$	s
s	$f = \begin{cases} -1e6 * s & s < 0 \\ 0 & s \geq 0 \end{cases}$	f

All tests reported use Newton's method to calculate cyclic equations. The constant step size approach needs a very small step size to handle the reversal process correctly. Figure 8 shows that the step size of 1.e-5 s calculates a nearly correct result.

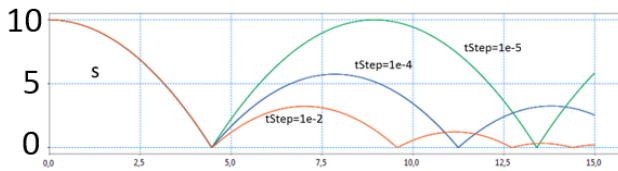


Figure 8. Course of $s(t)$ at different communication step sizes

Using Richardson extrapolation (max. step size 0.1 s, min. step size $1.e-15$ s, start step size $1.e-3$ s, absolute tolerance ATol $1.e-15$, relative tolerance RTol $1.e-8$) the maximum step size is used. Only to calculate the reversal regions (see Figure 9) the step size decreased down to about $1.e-7$ s (Figure 10).

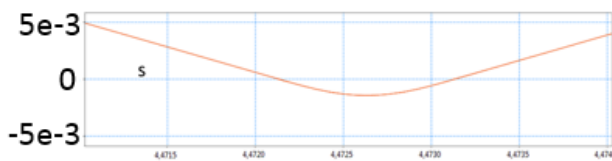


Figure 9. Reversal region calculated by Richardson extrapolation

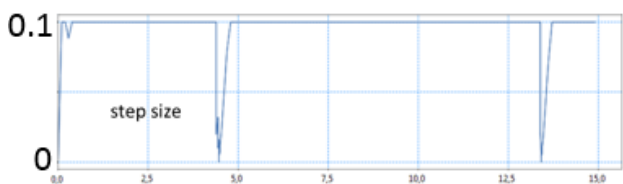


Figure 10. Varying step size using Richardson extrapolation

To calculate the reversal region correctly a small step size is necessary, otherwise the result becomes useless. The usage of the necessary small step size over the whole interval as a constant step size increases the CPU time abnormally (Table 5). The varying step size provided by Richardson extrapolation is the method of choice. The Linear extrapolation method does not succeed since the step size does not increase after decreasing. The reason is still to investigate.

Table 5. Normalized CPU time comparison for touching mass example

Step size 0.01 s	Step size $1e-4$ s	Step size $1e-5$ s	Richardson
1	56	572	1.3

If the spring constant is $1.e10$ instead of $1.e6$, the constant step size $1.e-5$ s does no more show the correct result (Figure 11). Richardson extrapolation test calculates the expected result within a short CPU time of less than 1s. The step size decreases to $1.8e-9$ s.

This example demonstrates the importance of the Richardson extrapolation method.

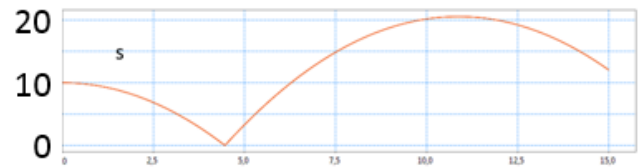


Figure 11. Course of $s(t)$ with spring constant $1.e10$, constant step size $1.e-5$ s

4 Application in Building Simulation

Three examples from building simulation are presented to study the obviousness of Richardson extrapolation at realistic use cases. The examples of different complexity consist of two FMUs each. One FMU describes both the heating facility and heating control modeled using Modelica and SimulationX [ESI ITI GmbH]. The other FMU of each example contains the building physics description as well as weather data using the NANDRAD tool [Nicolai, 2012] which solves PDEs. Table 6 shows roughly the structure of all examples.

Table 6. Macrostructure of the building examples

Input	Equations, Tool	Output
Room temperature, weather	DAEs (heating, heating control), SimulationX, Green Building, Modelica	Heat flow
Heat flow	PDEs (building physics), NANDRAD	Room temperature, weather

4.1 Single Room

The single room model is based on a small conference room (up to 20 people, Figure 12).



Figure 12. Meeting room on which the model is based

The room has a floor space of 52 m^2 , and a height of $3,3 \text{ m}$, one outer wall (west oriented), at which ambient conditions are applied. The boundary temperature of the opposite wall and the ceiling is set to constantly 18°C , for the other walls to 20°C . The four walls consist of a

heavy construction from clinker bricks and plastering, both ceiling and floor from lightweight concrete. Furthermore, an intermediate ceiling is included made of papier mâché. The room is equipped with a radiator heating operating a supply temperature of 70 °C. A valve that can operate continuously between valve position 0 and 1 regulates its volume flow.

Figure 13 shows the single room model. It consists of a thermal zone and heating facilities. The green framed part of Figure 13 contains both the thermal zone and the weather source, both modeled using NANDRAD, and therefore placed within the NANDRAD-FMU “thermal zone” which solves PDEs. The remaining model part of Figure 13 contains the heating facilities including the controller both taken from the GreenBuilding library [EA Systems Dresden, Unger et al. 2012]. This model part is written in Modelica, and exported as FMU “facility” using SimulationX 3.7.4

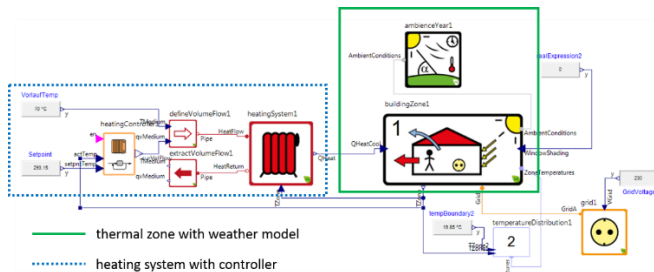


Figure 13. Single room model

This co-simulation task of both the FMUs “thermal zone” and “facility” has 17 coupling variables according to Table 7. Since one of the heating power components is zero, and the ambient variables do not depend on inputs, there are two “true” coupling values which form a cycle.

Table 7. Coupling variables of the single room example

	<i>variables</i>	<i>th. zone</i>	<i>facility</i>
2	Heating power	input	output
2	Zone temperature	output	input
10	Ambient values	output	input
2	Temperature set points	output	input
1	Electric power consumption	output	input

Using the Gauss-Jacobi method for solving the cyclic equations the three step size methods

- Constant step size 60 s
- Richardson extrapolation (tstepMax: 3600 s, tstepMin: 1 s, tstepStart: 60 s, default accuracy ATol: 1e-6, RTol: 1e-4)
- Linear extrapolation (tstepMax: 3600 s, tstepMin: 1 s, tstepStart: 60 s, default accuracy ATol: 1e-6, RTol: 1e-4)

calculate the same result (Figure 14) which does not differ from the reference solution obtained without coupling.

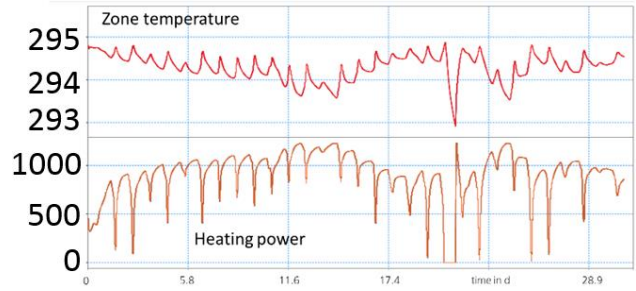


Figure 14. Room temperature and convective thermal heat load

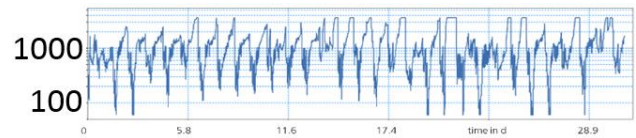


Figure 15. Step size variation in Richardson extrapolation

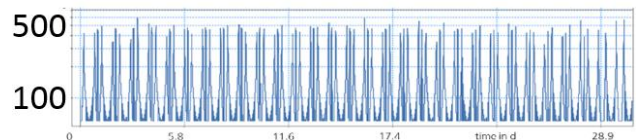


Figure 16. Step size variation in Linear extrapolation

According to Figure 15 the accepted step size in Richardson extrapolation varies considerably. The constant step size of 60 s is much smaller than most of the Richardson steps. Therefore, the Richardson method is even faster than constant step size simulation (Table 8). The Linear extrapolation time step method calculates smaller step sizes than Richardson extrapolation (Figure 16). In this example, Richardson extrapolation is the best choice since it is fast, and the user does not have to define a constant step size.

Table 8. Normalized CPU time comparison single room

<i>Step size 60 s</i>	<i>Richardson</i>	<i>Lin. Extr.</i>
1	0.7	1.6

4.2 Row House

The row house is a building according to Figure 17 with three floors. The heat to the ground floor and the first floor is provided by a volume flow controlled heating system (underfloor heating), the attic is not heated. A thermal storage buffer (50 m³) which is provided with warm water by a heat pump supplies the heating system.

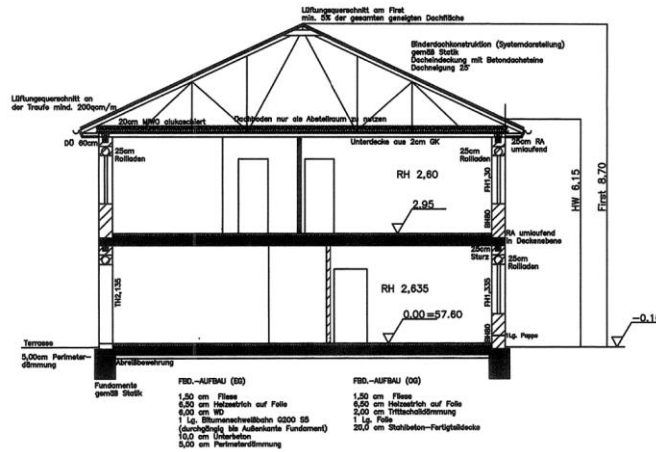


Figure 17. Row house sketch

Similar to the single room model the row house is modeled with different tools. Using NANDRAD the thermal zones including their interdependencies and additionally the weather were modeled, and exported as one FMU “thermal zones”. The facility model including heat pump, buffer, and the heating system are modeled using the GreenBuilding library. This model part is exported as “facility” FMU by SimulationX.

Figure 18 shows the graphical model representation of the row house. The green dashed frame shows the thermal zones, which form together with the weather model the “thermal zones” FMU. All other parts are within the “facility” FMU. Table 9 gives an overview on the 26 coupling variables.

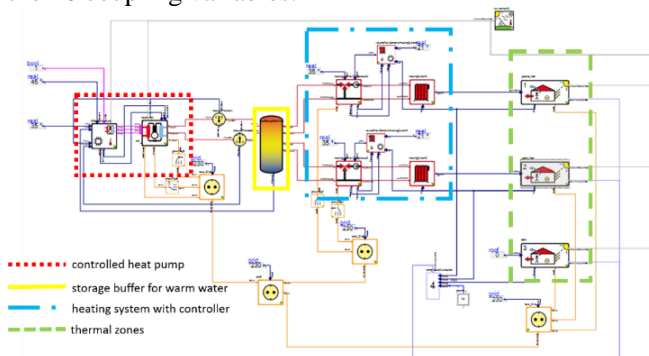


Figure 18. Row house model

Table 9. Coupling variables of the row house example

	<i>variables</i>	<i>th. zones</i>	<i>facility</i>
4	Heating power	input	output
2	Zone temperature	output	input
2	Zone mean radiant temp.	output	
9	Ambient values	output	input
1	Ambient values	output	
2	Heating setpoints	output	input
2	Cooling setpoints	output	
2	User load	output	
2	Electric power consumption	output	

The following results are based on Newton’s method for solving the cyclic equations. Figure 19 shows the zone temperatures, and Figure 20 shows the heat flow into the heated zones over a time interval of 31 days using constant step size of 60 s. This step size was chosen based on experience. The temperatures differ less than 5×10^{-3} K from reference results obtained by closed simulation via model exchange. The constant step size cannot be enlarged considerably, since already a constant step size of 300 s creates clear deviations, see Figure 21.

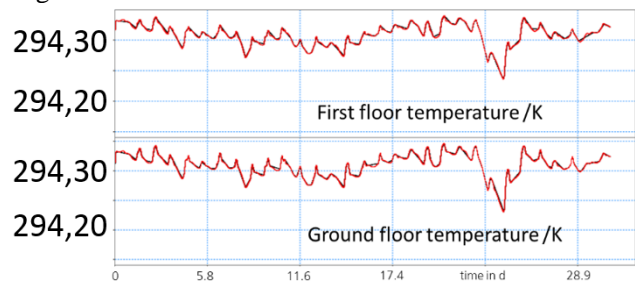


Figure 19. Row house room temperatures, constant step size 60 s

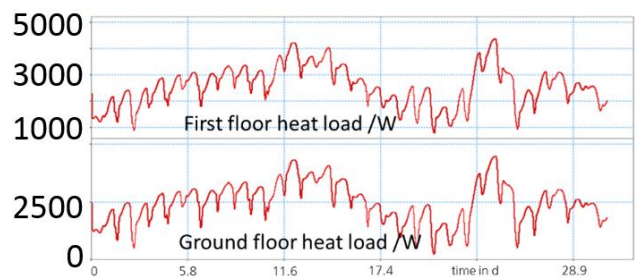


Figure 20. Row house convective thermal heat load, constant step size 60 s

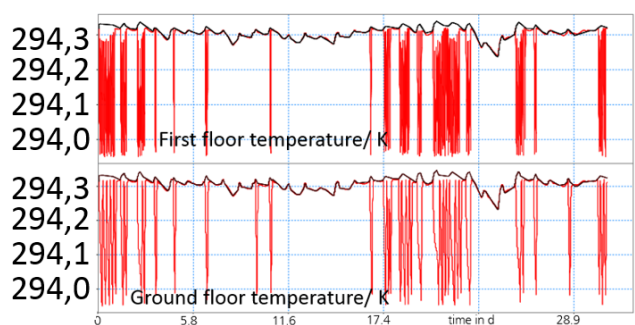


Figure 21. Row house room temperatures with deviations, constant step size 300 s

Richardson extrapolation calculates the same results as shown in Figure 19 and Figure 20, differences are negligible. Figure 22 shows the step size variation which was allowed to vary in a wide range from 0.01 s up to 3600 s, the lower limits were not reached.

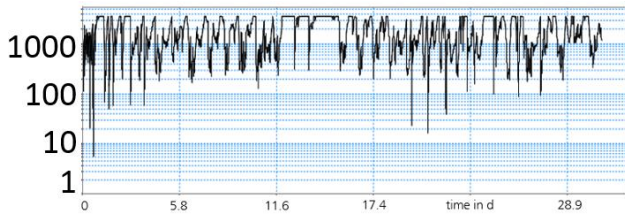


Figure 22. Row house Richardson accepted step size

According to Figure 22 the constant step size of 60 s is not a bad choice. This inspires to use a short time Richardson extrapolation for finding an appropriate constant step size. Table 10 compares different simulations. All variable step size methods calculate wrong results using the default tolerances. If a higher precision is applied correct results are achieved. Richardson extrapolation with higher precision is more twice as slow as well chosen constant step sizes. The reason is that Richardson extrapolation simulates the whole interval more than twice. Furthermore, it is to notice that some wrong simulations take much more CPU than correct ones at that example.

Table 10. Row house comparison of simulation runs

method	step size	deviation**	CPU*
Constant	60 s	0.004 K	21 min
Constant	300 s	0.4 K	20 min
Richardson	3600 s...60 s	0.4 K	1 day
Linear extr.	3600 s...60 s	0.4 K	24 min
Extended Richardson	3600 s...60 s	0,4 K	52 min
Richardson***	3600 s...60 s	0.004	1.1 h
Linear extr.***	3600 s...60 s	0.004	38 min
Extended Richardson***	3600 s...60 s	0.004	1.1 h

* Desktop-PC, SSD, Intel 2, 1 GHz, 8 GB RAM, Windows 7 (64 bit), ** max. deviation of the first floor room temperature from reference values, *** tighter tolerances (ATol=1e-8, RTol=1e-6)

4.3 Apartment Building

The apartment building has four floors, and three staircases. Per staircase and per floor there are three flats so that the building comprises 36 flats, see Figure 23.

The model consists of 168 thermal zones which are described using NANDRAD like at the row house and single room model. The thermal zones are exported altogether with one FMU2.0 “zones” for co-simulation. The heat supply of the building consists of a thermal storage buffer which is recharged by a both a block heat and power plant and a gas boiler. To keep the huge model smaller the heat supply model was simplified by prescribing the temperature of the medium that supplies the radiators. The 168 heating systems of the thermal zones comprise the model of a radiator, a controller model for the volume flow, and a controller model for the supply temperature each, see Figure 24. The heat

supply is modeled using the Green Building library (Modelica) and SimulationX, and exported as one FMU 2.0 “facility”.

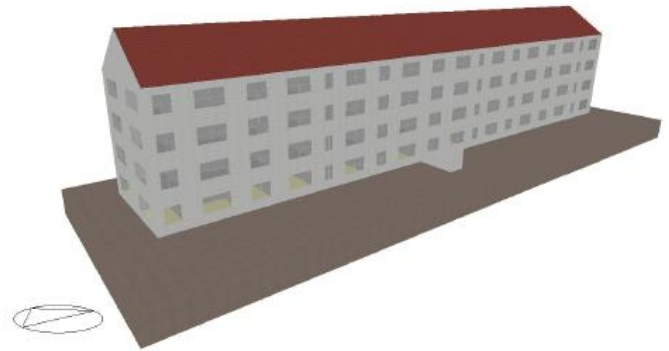


Figure 23. CAD model of the apartment building

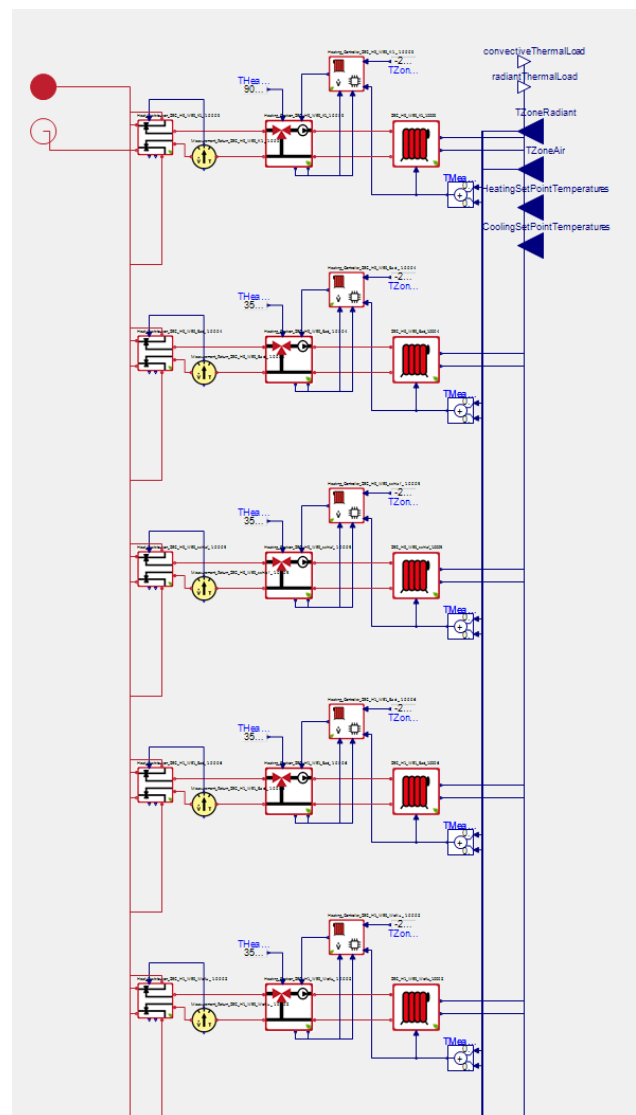


Figure 24. Heating system models of 5 thermal zones

The apartment house example has 1186 coupling variables which are roughly explained in Table 11.

Table 11. Coupling variables of the apartment building example

	<i>variables</i>	<i>th. zones</i>	<i>facility</i>
336	Heating power	input	output
168	Zone temperature	output	input
168	Zone mean radiant temp.	output	input
10	Ambient values	output	input
168	Heating setpoints	output	input
168	Cooling setpoints	output	input
168	Electric power consumption	output	input

Because of the bad performance the model is simulated over 7 days only. The following considerations are based on the Gauss-Seidel method with one iteration for cycle handling. Newton's method is not applicable due to its extremely bad performance. Figure 25 shows the mean zone temperatures as well as the thermal load of some rooms which are result of Richardson extrapolation with step sizes varying between 0.01 s and 1 hour. This result coincidences with a reference solution obtained by a co-simulation using SimulationX 3.7.4 for the facility part with included NANDRAD FMU for the thermal zones. Therefore, the result is regarded to be correct.

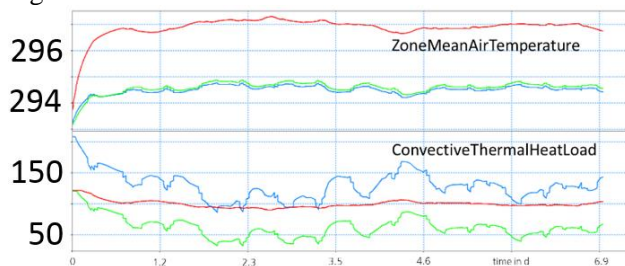
**Figure 25.** Apartment building: temperatures and heat load, Richardson extrapolation

Figure 26 shows the accepted step size variation of Richardson extrapolation. The step size varies between 2 seconds and an hour, however, only some peaks are below 100s. Therefore, a constant step size simulation was tested, which shows no visible deviation from the Richardson extrapolation result at some selected signals. A gradual increase of the constant step size up to 1 hour does not change the calculated signals clearly. At 1 hour step size the differences are about 0.05 K at some temperatures, and 0.5 W at thermal loads. A more detailed comparison is necessary. It is an advantage of Richardson extrapolation that no fixed step size needs to be defined.

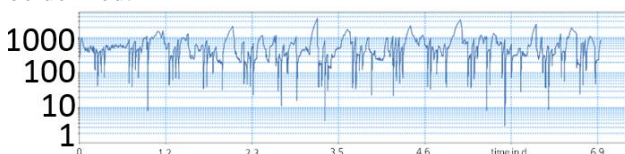
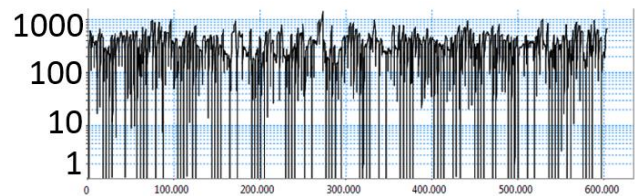
**Figure 26.** Apartment building: accepted step size variation during Richardson extrapolation, limited by 3600 s...0.01 s

Table 12 shows the performance of the different simulations. Reasonable constant step size simulations are about twice as fast as Richardson extrapolation. The Extended Richardson extrapolation as well as the Linear extrapolation approach also calculate correct results. But their performance is worse than Richardson extrapolation since it uses smaller step sizes (Figure 27).

**Figure 27.** Apartment building: step size variation during Extended Richardson extrapolation**Table 12.** Apartment building: performance comparison

<i>method</i>	<i>step size</i>	<i>CPU*</i>
Richardson	3600 s ... 0.01 s	11.4 min
Constant	100 s	7.3 min
Constant	200 s	6.4 min
Constant	400 s	6.0 min
Constant	1000 s	5.4 min
Constant	3600 s	5.4 min
Linear extrapolation	3600 s...0.01 s	14.3 min
Extended Richardson	3600 s...0.01 s	18.2 min

* Desktop-PC, SSD, Intel 2, 1 GHz, 8 GB RAM, Windows 7 (64 bit)

This example demonstrates that Richardson extrapolation seems to ensure finding the correct solution. Furthermore, it is useful for finding adequate step sizes for constant step size simulations. But it is not an approach to obtain a somewhat high performance.

5 Conclusion

Richardson extrapolation is recognized to be an important and useful approach for co-simulation. It was shown that enhancements are necessary for the cases of outputs that do not depend on inputs which control DAEs.

There are examples which need a variable step size approach in co-simulation. The touching mass example requires the Richardson extrapolation approach.

To apply Richardson extrapolation in building simulation three differently sized examples are presented. The results which are by far not yet representative to building simulation models at all, are:

- The performance of Richardson extrapolation is worse than the performance of constant step size method, although Richardson extrapolation partly uses higher step sizes.

- Richardson extrapolation with a wide step size limitation can be applied to find out a trustable constant step size. This helps the user to define the step size. This approach should be automated.
- Furthermore, the building simulation examples show that a high number of coupling variables is to be expected. This frustrated the application of Newton's method for cycles. Therefore, modifications of Newton's method should be investigated.

Acknowledgements

This paper is based on the results of the German research project „Entwicklung der Kopplungstechnologie von Komplexmodellen für Bauteil-, Raum- und Gebäudesimulation mit Modelica-basierten Anlagen-, Regelungs- und Nutzermodellen“ (EnTool:CoSim), funding reference 03ET1215C.

Gefördert durch:



aufgrund eines Beschlusses
des Deutschen Bundestages

The authors are much obliged to the contributors of the research project J. Bastian, T. Blochwitz (ESI ITI GmbH), A. Nicolai and Anne Paepke (IBK TU Dresden), Torsten Schwan, and Monika Wicke (EA Systems Dresden) as well as Kosmas Petridis and Andreas Klein for support and discussions.

References

- Andreas Klein. Private information, 2015.
- Andreas Nicolai, Anne Paepke. Die Gebäudesimulationsplattform NANDRAD – Physikalisches Modell, Umsetzungskonzept und Technologien im Überblick. BauSIM 2012, Berlin, 26.-28. September 2012.
- EA Systems Dresden: Portfolio. Die neue Generation intelligenter Energiekonzepte. Company internal document, via info@ea-energy.de, 2015.
- Ernst Hairer, Syvert Paul Norsett, Gerhard Wanner. Solving Ordinary Differential Equations. Berlin, Springer, 1993.
- ESI ITI GmbH: Simulation software SimulationX, website <https://www.simulationx.de/simulationssoftware.html>
- Jens Bastian, Christoph Clauß, Susann Wolf, Peter Schneider. Master for CoSimulation Using FMI. 8th International Modelica Conference, Dresden, March 20-22, 2011.
- Kosmas Petridis, Christoph Clauß. Test of basic co-simulation algorithms using FMI. 11th International Modelica Conference, Versailles, 2015.
- Vladimir Aleksandrovitch Kotelnikov. On the transmission capacity of the ether and of cables in electrical communications. Proc. of the first All-Union Conference on the technological reconstruction of the communications sector and low-current engineering, Moscow 1933.

Rene Unger, Torsten Schwan et alt.. "Green Building"-Modelling Renewable Building Energy Systems and Electric Mobility Concepts Using Modelica. 9th International Modelica Conference, Munich, Germany, 2012.

Tom Schierz, Martin Arnold, Christoph Clauß. Cosimulation with communication step size control in an FMI compatible master algorithm. 9th International Modelica Conference, Munich, 2012.

Tom Schierz. Modulare Zeitintegration gekoppelter Differentialgleichungssysteme in der technischen Simulation. Fortschritt-Berichte, VDI Reihe 20 Nr. 447. Düsseldorf, VDI Verlag 2013.

Development of a Thermodynamic Engine in OpenModelica

Rahul Jain¹ Kannan M. Moudgalya¹ Peter Fritzson² Adrian Pop²

¹Dept. of Chemical Engineering, Indian Institute of Technology Bombay, India,
rahjain1@gmail.com, kannan@iitb.ac.in

²Dept. Computer and Information Sciences, Linköping University, Sweden,
{peter.fritzson, adrian.pop}@liu.se

Abstract

OpenModelica, an open source equation oriented modeling environment for steady state and dynamic simulation, lacks good chemical engineering support. This problem is addressed by making available in different ways the thermodynamic library Chemsep that comes with DWSIM, an open source sequential modular steady state simulator. Only slow speeds could be achieved through a Python-C API based interface connecting OpenModelica with the thermodynamic library. A socket programming based interface helps achieve faster speeds. Best results have been achieved by porting the thermodynamic library and the calculation routines to OpenModelica, due to two reasons: (1) thermodynamic equations are solved simultaneously with mass and energy balances (2) overheads in calling the external routines of DWSIM are eliminated. Performances of the above mentioned three approaches have been validated with steady state and dynamic simulations. Benzene - toluene separation, methanol - ethanol - water distillation, and steam distillation of an n-octane - n-decane mixture, have been carried out through these simulations. This work makes available a powerful simulation platform to the chemical engineering.

Keywords: *OpenModelica, DWSIM, Chemsep, thermodynamics, modeling, simulation, chemical engineering, Python-C API, socket programming, media*

Abbreviations

API	Application programming interface
csv	Comma separated values
dll	Dynamic link library
DTL	DWSIM thermodynamics library
EOS	Equation of state
VLE	Vapor liquid equilibrium

1 Introduction

Modelica (Modelica Association, 2000) is a powerful modelling language and OpenModelica (Fritzson, 2014) is its open source implementation. In OpenModelica has an excellent interface to build models and to perform simulations. As it implements an equation oriented solution approach, models and solution methods are maintainable (Piela et al., 1992). Many engineering domains have used OpenModelica.

Unfortunately, OpenModelica does not have a library of chemical engineering models and a thermodynamic database. As a result, it is not yet of much use to the chemical engineering community. If we can add a CAPE Open thermodynamic database to OpenModelica, it can immensely increase the utility for chemical engineers.

DWSIM is a state of the art open source steady state process simulator (Medeiros, 2015). It comes with two CAPE Open thermodynamic databases, Chemsep (Kooijman and Taylor, 2001) and the native one. In this work, we describe the different methods to make the DWSIM chemical engineering library available for OpenModelica.

This paper is organized as following. We explain the Python-C interfacing approach to call the DWSIM's thermodynamic database from OpenModelica. We then explain how to instead use socket programming to connect the two simulators. The final part is devoted to the porting of thermodynamics in native mode on to OpenModelica. We conclude with a comparison of the three approaches.

2 Importing the Thermodynamic engine of DWSIM in OpenModelica

As DWSIM is based on sequential modular solution techniques (Westerberg et al., 1979), it is more suitable to solve *analysis* type of problems. One will have to resort to iterations to *design* systems, which may involve finding the value of some parameters in the output stream or in the equipment, or the building block. It is also difficult to carry out dynamic simulation in DWSIM. DWSIM has a strong thermodynamic engine. DWSIM also has a standalone thermodynamic library (DTL) which can be used externally.

The weakness of DWSIM is the strong point of OpenModelica: it is equation oriented and capable of handling unsteady state equations. Similarly, the weakness of OpenModelica is the strength of DWSIM: thermodynamic database and routines. As they complement each other, there is a good case to integrate OpenModelica with DTL.

2.1 Python-C API approach to Integrate OpenModelica with DTL

DTL consists of a file with an extension .dll (dynamic link library) which is written in VB.NET in windows en-

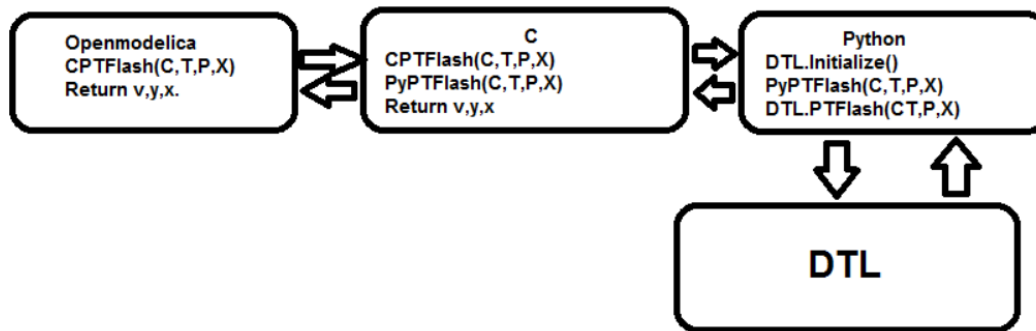


Figure 1. Structure of Python-C API approach

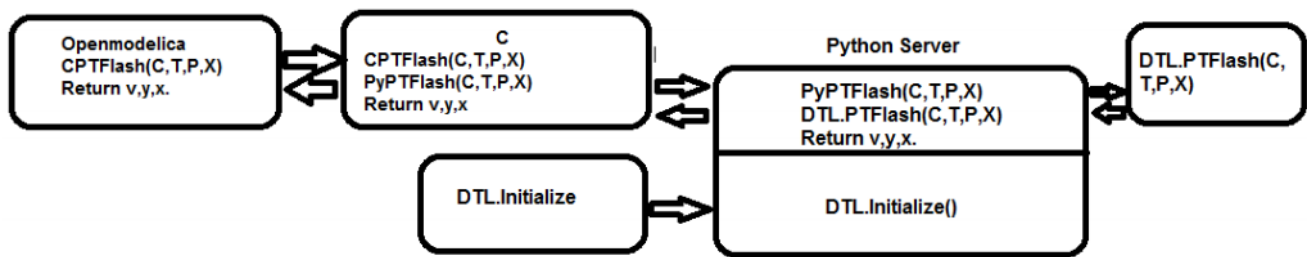


Figure 2. Structure of Python-C socket approach

vironment. This file is COM (component object model) enabled, which means that any programming language which supports COM can import this library and access the built-in thermodynamic subroutines. OpenModelica is written in C in Linux environment and it is not straight forward to call programs written in VB.NET.

We used Python as the glue language to call the COM enabled objects of DWSIM from C routines of OpenModelica. This was achieved through the package *win32com* of Python. This allowed us to access the DTL library and all the thermodynamic routines available in DWSIM from OpenModelica. Figure 1 describes the flow of the approach, which are further described below.

- DTL routines are imported to Python first through a package named *win32com.client*. This package allows Python to call routines from a dll file registered in the windows registry. Once the dll is dispatched through *win32com.client*, Python has access to all the COM enabled functions of the dll.
- Now Python functions can send input parameters to DTL routines, get the required thermodynamic properties calculated and receive them. As results of calculations are available, these Python functions can be considered to behave similar to DTL routines.
- These Python functions are now called by C through Python-C API. In computer programming, an API (Application Programming Interface) is a set of routines, protocols, and tools for building software applications. An API expresses a software component

in terms of its operations, inputs, outputs, and underlying types. This API is responsible for converting C variables to Python and vice versa.

- Finally as OpenModelica is compatible with C, the inputs are then sent to C functions through OpenModelica external C functions, which in turn calls the Python functions, which in turn calls DTL routines.

2.2 Client-Server (sockets) approach to integrate OpenModelica with DTL

Client-Server or socket approach (Rhodes and Goerzen, 2010) is another approach through which the integration is possible. Figure 2 describes the data flow of the approach, which are further explained below.

- In this approach also, firstly the the DTL routines are called in Python with the help of *win32com.client* package.
- Similar to Python-C approach, functions are written in Python which calls DTL to calculate various physical properties.
- Now a Python server which consists of all the above functions is created. This server waits for a C client to establish connection, receive inputs from it and send the calculated values back to the client.
- For every calculation (e.g. vapor pressure, equilibrium constant, etc.), a Python server is established.

Table 1. Thermodynamic routines and the procedures to call them

Thermodynamic Prop.	Thermodynamic Func.	Arguments
Vapor Pressure	VapPres	Comp,T
Enthalpy	Ent	Comp., T,P
Liquid Density	LiqDen	Comp,T
Vapor Density	VapDen	Comp,T
Pres. Temp. Flash	PTFlash	Comp,Z,T,P,Model
Pres. Volume. Flash	PVFlash	Comp,Z,V,P,Model
Pres. Enth. Flash	PHFlash	Comp,Z,H,P,Model
Liquid Viscosity	LiqVis	Comp,T
Vapor Viscosity	VapVis	Comp,T
Surface tension	SurfTen	Comp,T

- Clients are coded in C which establishes connections with the Python servers and send and receive data from them.
- Once the connection is established, a Python server receives the data from C client, contacts DTL, calculates the required property as asked by the C client and sends it back to the client.
- Finally, these C clients are called by OpenModelica external C functions giving the required inputs to the client which in turn contacts the Python servers for calculations. The C clients receive calculated values from Python servers and transfer them to OpenModelica.

2.3 Comparison of the two approaches

In this section, we compare the two approaches presented above. In both approaches, before any routine in DTL is called, one has to carry out initialization. This *Initialize* routine loads all the compounds and their properties from the database, which is a time consuming operation. In the Python-C API approach, this initialization is done every time a call is made from OpenModelica to DTL. On the other hand, this has to be done only once in the Client Server approach. As a result, the latter is far more efficient than the former.

Whenever an API is used in any program it makes it slow as there is a lot of conversions involved, such as data type conversions. As the Python-C API approach is based on API it is slow.

To verify the speeds of two approaches, we use the thermodynamic calculations presented next. Table 1 lists the thermodynamic functions and their arguments that we have implemented in OpenModelica to receive values from DTL. These functions can be used directly in any simulation. When using the socket approach, the Python server should be up and running during the execution of the simulation. We now present two case studies that helped compare the two approaches.

• Steady State Flash Separator

An equimolar mixture of Benzene and Toluene was flashed in a flash separator. The thermodynamic

package used was Raoult's law. All the pure component and mixture properties were imported from DTL. To test the capability of the integration methods, the composition of the resulting vapor stream was specified, and the temperature at which this composition was attained was left unknown. It was observed that the Python-C API approach took 30 seconds to solve the system, whereas the Client-Server method took less than 1 second to simulate.

• Dynamic Flash

A dynamic flash was simulated with the feed as equimolar mixture of benzene and toluene. The thermodynamic package used was again Raoult's Law. It was assumed that the output liquid stream was at the same composition and temperature, as the holdup inside the flash separator. Heat supplied to flash separator was kept constant. The set of equations involved were mass balance, energy balance and equilibrium equations. The mass and energy balance were differential equations. It was observed that the Python-C API approach took 30 minutes to solve, whereas the Client-Server approach took 4 minutes to solve the system.

The above two examples and others that we have not reported here show that the Client-Server approach is more efficient than the Python-C API approach.

3 Development of a native thermodynamic engine in OpenModelica

A thermodynamic engine consists of the following three components: Compound database, thermodynamic functions and phase equilibria models. In this section, we describe how we have developed a native thermodynamic engine in OpenModelica.

3.1 Development of Compound Database

A Compound database is a comprehensive database of physical and chemical properties of all compounds. It also includes constants for calculating various temperature or pressure dependent properties like vapor pressure, enthalpy, viscosity, etc.

We first describe the Chemsep (Kooijman and Taylor, 2001) database that we ported to OpenModelica. Chemsep is an open source database, written in **xml** format. It has over six hundred compounds with a comprehensive set of thermodynamic properties of each compound. It also has an extensive database of binary interaction parameters for thermodynamic packages like NRTL (Renon and Prausnitz, 1968), Peng Robinson (Peng and Robinson, 1976), UNIQUAC, SRK (Soave, 1972), etc. Most of the thermodynamic properties are calculated by empirical equations that are functions of temperature or pressure. Chemsep database includes the constants which are used in these equations. Therefore, Chemsep database is

Table 2. Independent thermodynamic properties and OpenModelica routines to call them.

Thermodynamic Property	Calling procedure
Critical Temperature	Compound.Tc
Critical Pressure	Compound.Pc
Critical Volume	Compound.Vc
Boiling point	Compound.Tb
Melting point	Compound.Tm
Molecular weight	Compound.MW
Acentric Factor	Compound.AF
Triple Point	Compound.TT
Solubility parameter	Compound.SP
Dipole moment	Compound.DP
Heat of formation	Compound.HOF
Absolute enthalpy	Compound.ABSENT

a comprehensive database that can be used to build a powerful and robust thermodynamic engine.

We now explain how we ported Chemsep to OpenModelica. First, the xml data have to be rewritten in a form understandable by OpenModelica. Therefore, each compound (including all its thermodynamic properties) is replicated as a single class in OpenModelica, as shown in Figure 3. The properties are given abbreviations (as shown in Table 2) so that they can be called conveniently. The conversion from xml to OpenModelica classes is carried out by developing a Python script which automates this process, thus making it fast and robust.

Now, one can extract any independent property of a compound by the . (dot) operator, followed by the property relevant abbreviation. For example the critical temperature (Tc) of methane can be accessed by **Methane.Tc**. Similarly, all properties of any compound can be accessed in the same way as shown in table 2.

3.2 Development of Thermodynamic Functions

Thermodynamic properties are generally calculated through empirical equations that include constants, whose values are provided by the compound database as explained above, and independent variables, such as temperature, pressure and composition. These properties are written in the form of functions in OpenModelica. Arguments to these functions are the independent variables mentioned above, and the coefficients of respective compounds whose properties have to be calculated. These coefficients can be accessed by instantiating the base compound class. The functions then return the calculated property. For example, the vapor pressure of methane at 300 K can be calculated by first instantiating the base Methane class (**parameter Methane methane**) and then calling **Pvap(methane.VP, 300)**. Where Pvp is a generic function to calculate the vapor pressure of any compound at any given temperature. The whole process is shown

Table 3. Dependent thermodynamic properties and OpenModelica functions to call them.

Thermodynamic Property	Calling procedure
Liquid density	LiqDen(Compound name,temp)
Vapor pressure	VP(Compound name,temp)
Heat of Vaporization	HOV(Compound name,temp)
Liquid heat capacity	LiqCp(Compound name,temp)
Liquid viscosity	LiqVis(Compound name,temp)
Vapor viscosity	VapVis(Compound name,temp)
Liquid thermal conductivity	LiqK(Compound name,temp)
Vapor thermal conductivity	VapK(Compound name,temp)

in Figure 4. Similarly, all other thermodynamic properties can be calculated using their respective functions as shown in Table 3.

3.3 Development of Phase Equilibria models

Phase equilibria models consist of modelling equations for Vapor Liquid Equilibrium (VLE) models like Peng Robinson, NRTL, UNIQUAC, etc. These models are used to predict the behavior of various systems.

In a mixture of phases that are in an equilibrium, the component fugacities are the same in all phases (Smith et al., 2005), that is :

$$f_i^L = f_i^V \quad (1)$$

where f_i^L and f_i^V are the liquid and vapor phase fugacities of the i th component respectively. The fugacity of a component in a mixture depends on temperature, pressure and composition. In order to relate f_i^V with temperature, pressure and molar fraction, we define the fugacity coefficient,

$$\Phi_i = \frac{f_i^V}{y_i P^*} \quad (2)$$

where Φ_i is the fugacity coefficient and P^* is the pressure of the system, which can be calculated from PVT data, commonly obtained from an equation of state (EOS). For a mixture of ideal gases, $\Phi_i = 1$. The fugacity of component i in the liquid phase is related to the composition of that phase by the activity coefficient γ_i , which by itself is related to x_i and standard-state fugacity f_i^0 by

$$\gamma_i = \frac{f_i^L}{x_i f_i^0} \quad (3)$$

The standard state fugacity f_i^0 is the fugacity of the i th component at the system temperature, i.e. mixture, and in an arbitrary pressure and composition. Here, the standard-state fugacity of each component is considered to be equal to pure liquid i at the system temperature and pressure. An Equation of State is used to calculate equilibria. The fugacity of the i th component in the liquid phase is calculated by

$$\gamma_i = \frac{f_i^L}{x_i P^*} \quad (4)$$

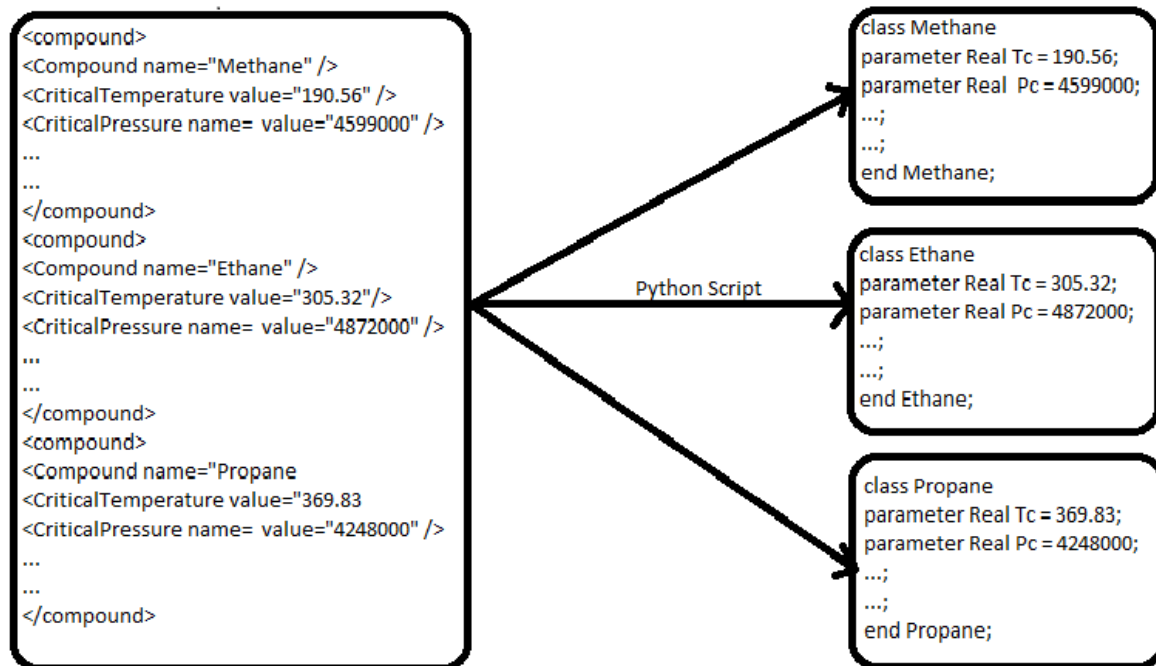


Figure 3. Porting Chemsep database in OpenModelica

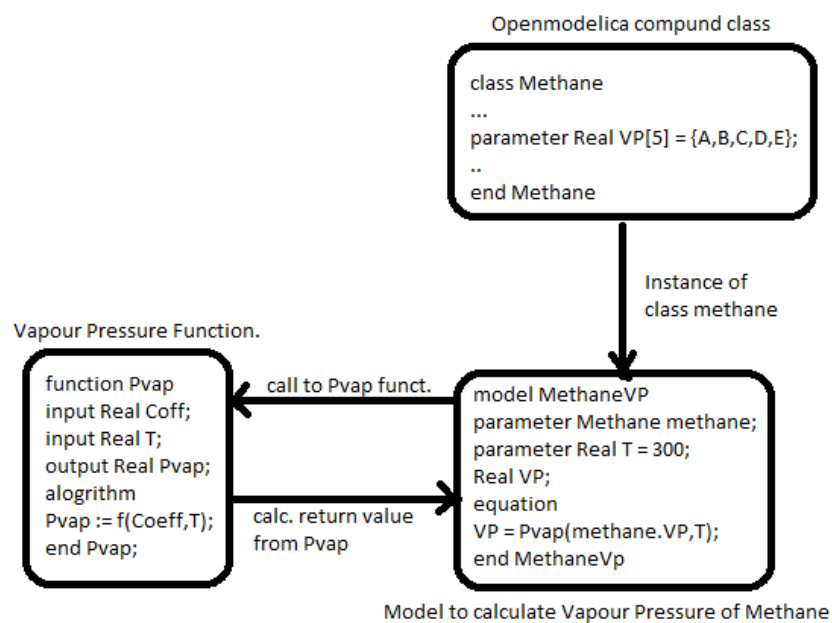


Figure 4. Using built in thermodynamic functions (Pvap)

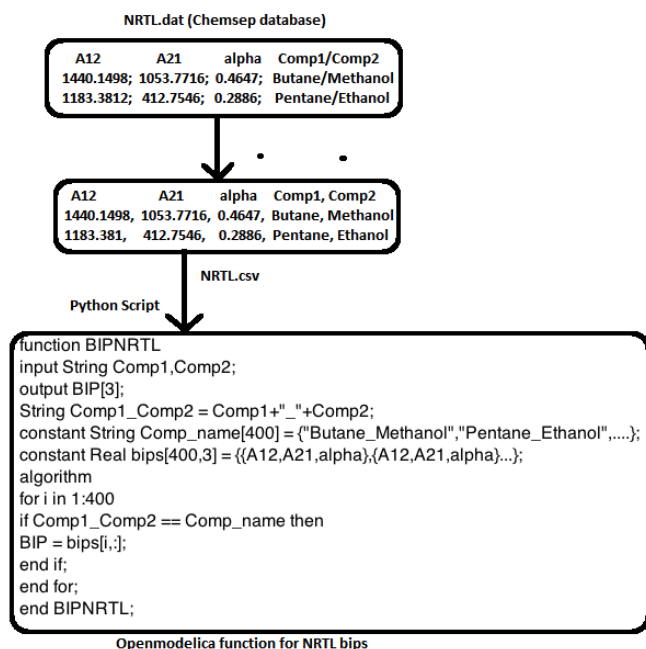


Figure 5. Porting Chemsep's binary interaction parameters in OpenModelica

with the fugacity coefficient Φ_i calculated by the EOS, just as it is done for the vapor phase.

We have implemented the following four phase equilibria models: Peng Robinson, SRK, NRTL and UNIQUAC. Peng Robinson and SRK are the most abundantly used EOS models, whereas NRTL and UNIQUAC find a wide variety of applications where activity coefficient models are required (Medeiros, 2015).

The binary interaction parameters for each of the EOS and activity coefficient models have been extracted from Chemsep database where they are stored in a .dat file. The following procedure is used to port all the binary interaction parameters to OpenModelica.

- First, the dat file is converted to a csv file, which is easier to process by Python.
- This csv file is then converted to an OpenModelica function by a Python script which converts the compound and the binary interaction parameters as an array.

Figure 5 demonstrates the above process for NRTL activity coefficient model. This code is automatically generated by the Python script. Line 6 of this code has been shortened for convenience. The actual code has 400 triplets of real numbers on the right hand side of line 6.

Figure 6 shows the NRTL model. The model acquires the required binary interaction parameters from the BIPNRTL function. The model incorporates the equilibrium relation described in equation 4. This model can now be directly extended into any model which requires calculation of phase equilibrium. All other phase equilibria models have been modeled similarly.

```
model NRTL
parameter Real BIP[3] = BIPNRTL(Comp.name); //Binary interaction parameters
Real P, T (start = 373, min = 350, max = 380); //Pressure and Temperature
Real x[2](each start = 0.5, each min = 0.0001, each max = 1); //liquid compositions
Real y[2](each start = 0.5, each min = 0.0001, each max = 1); //vapor compositions
Real gamma[2](each start = 1); //activity coefficients
equation
gamma = f(x,y,P,T,BIP); //equilibrium equations relating gamma,x,y,P,T,BIP
end NRTL;
```

Figure 6. NRTL model as written in OpenModelica 1.11.0

4 VLE curve (Txy) for a binary system through the UNIQUAC model

In this section, we explain the procedure to generate the VLE curve for a binary system and demonstrate it with results from an ethanol-water system.

We will first explain the procedure to generate the bubble point curve. Suppose γ_1 and γ_2 are the activity coefficients, y_1 and y_2 are vapor phase compositions, x_1 and x_2 are liquid phase compositions, and P_{vap1} , P_{vap2} are corresponding vapor pressures, of components 1 and 2, respectively. Then, the following equations are used to generate the bubble point curve.

$$y_1 \cdot P = \gamma_1 \cdot x_1 \cdot P_{vap1} \quad (5)$$

$$y_2 \cdot P = \gamma_2 \cdot x_2 \cdot P_{vap2} \quad (6)$$

This is known as the modified Raoult's law.

Adding the above two equations and equating the vapor phase mole fractions to one ($y_1 + y_2 = 1$), we get

$$P = \gamma_2 \cdot x_2 \cdot P_{vap2} + \gamma_1 \cdot x_1 \cdot P_{vap1} \quad (7)$$

Here γ_1 and γ_2 are complex nonlinear functions of temperature and liquid compositions and P_{vap1} and P_{vap2} are functions of temperature.

The pressure is kept constant at 1 atm. The value of x_1 is varied from 0 to 1 with an interval of 0.1 and for each value of x_1 , the corresponding value of temperature is calculated by equation 7. This is known as the bubble point.

Now we explain how the dew point curve is generated. We once again use the modified Raoult's law for this purpose. Manipulating the equations 5 and 6 and putting $x_1 + x_2 = 1$ we get

$$\frac{y_1}{\gamma_1 \cdot P_{vap1}} + \frac{y_2}{\gamma_2 \cdot P_{vap2}} = 1 \quad (8)$$

The pressure is kept constant at 1 atm. The value of y_1 is varied from 0 to 1 with an interval of 0.1 and for each value of y_1 the corresponding value of temperature is calculated by equation 8.

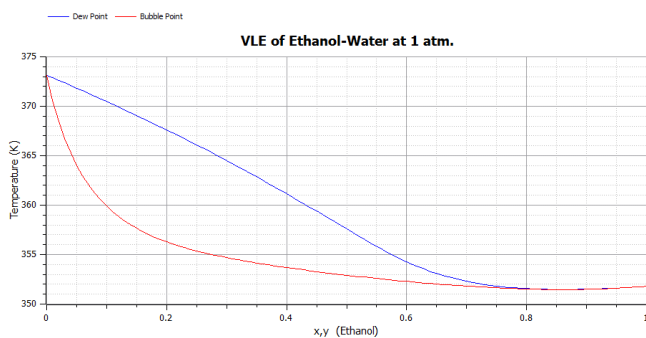
Figure 7 describes the implementation of the bubble point model in OpenModelica. The dew point model have also has been modeled similarly. As shown all the three parts of the thermodynamic engine, namely, compound

```

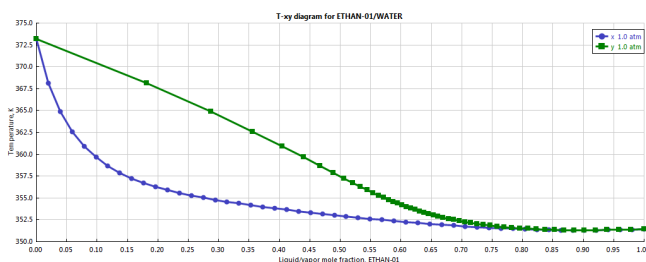
model bubblepnt
parameter Ethanol ethanol; // compound properties from database
parameter Water water; // compound properties from database
parameter Real z[2] = {0.5, 0.5};
parameter Real compound[2] = {ethanol, water}
extends UNIQUAC(Comp = compound, P = 101325); //Phase equilibria model
Real Psat[2];
algorithm
Psat:= Thermodynamic_Functions.Psat(Compound.VP, T); //Thermodynamic functions
equation
x[1] = time;
x[2] = 1 - x[1];
P = sum(gamma[i] .* x[i] .* Psat[i]);
y[1] = 0;
y[2] = 0;
end bubblepnt;

```

Figure 7. Bubble point model as written in OpenModelica 1.11.0



(a) Results from OpenModelica 1.11.0



(b) Results from Aspen Plus 8.1

Figure 8. Comparison of T-xy curve for ethanol water system using UNIQUAC VLE model

database, thermodynamic functions and phase equilibria models, have been incorporated in the model.

Using the above procedure, we have calculated the bubble point curve and the dew point curve for the ethanol(1)-water(2) system, and presented them in Figure 8(a). One can see it to be identical to the figure generated by Aspen Plus (Aspentech, 2017), presented in Figure 8(b).

Reliable azeotropic data source by American Chemical Society (Gmehling et al., 1995) says that for ethanol-water system, at 1 atm, the azeotropic composition and temperatures are 0.96 mole fraction ethanol and 351.4 K, respectively. These values are also in agreement with the OpenModelica results.

The same simulation when carried out with the imported DWSIM's thermodynamic engine in OpenModelica resulted in an execution time of about 20 minutes, whereas for the built in thermodynamic engine, the exe-

cution time was 0.58s.

5 Steady State Flash

Now that thermodynamics is available in OpenModelica, we simulate a steady state flash of a methanol, ethanol, water system, using NRTL. To check the design efficiency of the developed thermodynamic engine in OpenModelica, the output composition of the vapor product is specified, while the temperature at which this desired composition of vapor is attained is left unspecified. Thus, it is a design problem. To carry out this simulation in DWSIM, we have to use the *adjust* operation that uses a trial and error method.

We now explain the problem we propose to solve. The flow rate and the composition of the feed is specified. Pressure is kept constant at 1 atm. It is desired to calculate all other variables for three different values of methanol mole fraction, x_1 . In other words, vapor compositions of ethanol and water, temperature of all streams and all flow rates need to be calculated. The schematic of the problem statement is presented in Figure 9.

The input stream enters at 1 atm and its temperature is to be determined according to the specified input composition. The simulation is run for three different desired vapor compositions of methanol. The minimum and maximum temperatures were taken to be boiling points of pure methanol and water and the initial guess for temperature is taken to be average of these two boiling points. The following equations describe the model.

Mass balance:

$$z_i F = x_i L + y_i V \quad (9)$$

$$F = L + V \quad (10)$$

Equilibrium equation:

$$y_i = K_i x_i \quad (11)$$

Summation Equation:

$$\sum_{i=1}^2 y_i = 1 \quad (12)$$

Here, F , L , V are the feed, liquid, and vapor flow rates, respectively, in kmol/hr and z_i , x_i , y_i are the feed, liquid, and vapor compositions respectively. K_i is the equilibrium constant. UNIQUAC activity coefficient model is used as the phase equilibria model.

Figure 10 depicts the example as developed in OpenModelica. The type *compound* in the fifth line is a general class used to represent the compound class.

Results of these calculations have been presented in Table 4. One can see that these results are consistent with the general requirement that higher the mole fraction of the least volatile component, lower the temperature. All calculations got completed in OpenModelica in less than a second.

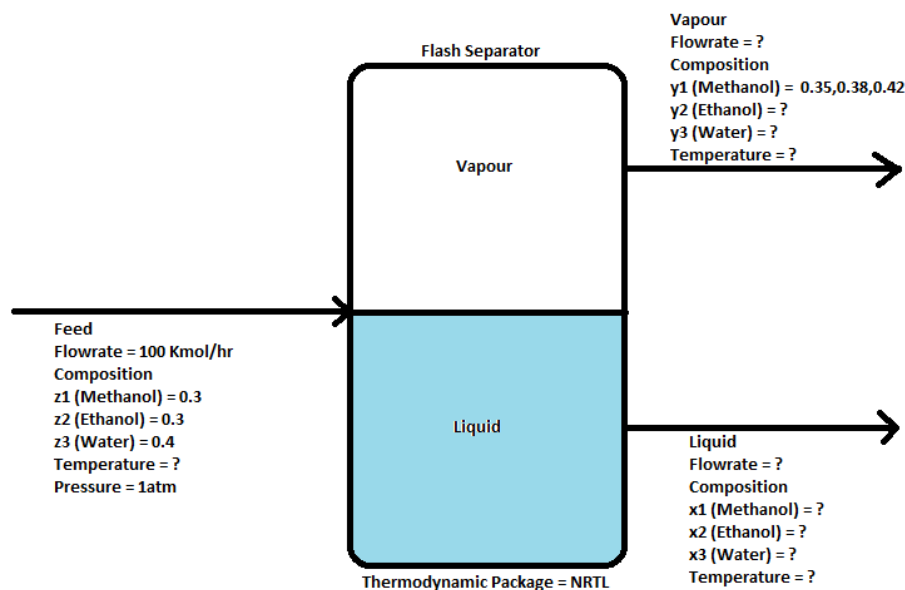


Figure 9. Model with problem statement for steady state flash of Methanol-ethanol-water.

```

model Flash
parameter Methanol methanol;
parameter Ethanol ethanol;
parameter Water water;
parameter compound compounds[3] = {methanol,ethanol,water};
parameter Real z[3] = {0.3,0.3,0.4};
Real x[3], y[3], k[3], T, Psat[3];
Real L,V,F;
extends UNIQUAC(Comp=compounds,P=101325); //
equation
y[1] = 0.35;
F = 100;
z[1]*F - (x[1]*L+y[1]*V) = 0;
y[1] = k[1]*x[1];
k[1] = {gamma[1]*x[1]*Psat[1]}/P;
Psat[1] = Thermodynamic_Functions.Psat(compounds.VP,T);
end Flash;

```

Figure 10. Flash model as written in OpenModelica 1.11.0

Same calculations are repeated in DWSIM using the *adjust* function, by trial and error, and the results are reported in the same Table. One can see the results to be comparable. It took 15 to 20 seconds to do each of these calculations in DWSIM, however.

6 Semi-Batch Steam Distillation of a Binary Organic Mixture

We now illustrate the ease with which dynamic simulation can be carried out in OpenModelica, using the semi-batch steam distillation of a binary mixture. We present the model first and then an example.

Table 4. Results of simulation in OpenModelica using the built-in thermodynamics and in DWSIM

OpenModelica 1.11.0		
Desired Vapor Comp.(Methanol)	Temperature	Liquid Comp.
0.35	351.21	0.1985
0.38	350.28	0.2354
0.425	349.24	0.274
DWSIM 3.4		
0.35	351.26	0.199
0.38	350.211	0.234
0.425	349.12	0.279

6.1 Model of the process

This illustrative example involves semi-batch steam distillation of a binary mixture (n-octane and n-decane). A schematic plot of the steam distillation apparatus is shown in Figure 11. The organic mixture is charged into the still initially, and then steam is bubbled through continuously until the desired degree of separation has been reached. There are two different periods in the operation of the still: the *heating period*, until the boiling point temperature of the organic mixture is reached, and the *distillation period*. A brief description of the mathematical models for the two periods follows (Shacham et al., 2012).

We present the model for the heating period first. A simple mass balance on the water phase yields

$$\frac{dm_w}{dt} = W_s \quad (13)$$

where W_s is the steam flow rate in kmol/s and m_w is the mass of water in the still in kmol. It is assumed that all the steam condenses in the distillation vessel and that the organic phase masses remain constant during the heating period.

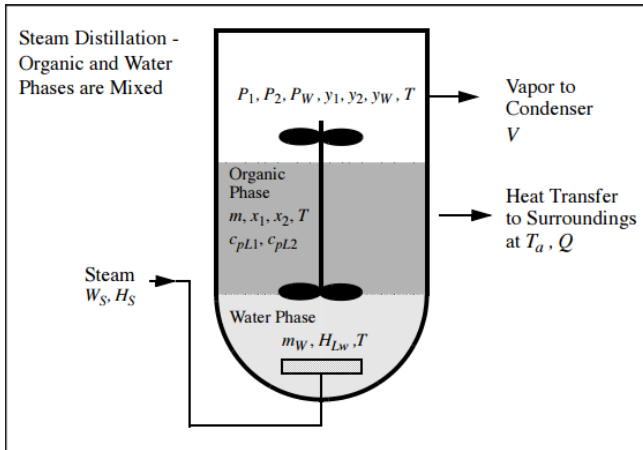


Figure 11. Schematic of steam distillation apparatus (Shacham et al., 2012).

An energy balance on the still provides the equation for the change of the temperature T in $^{\circ}\text{C}$

$$\frac{dT}{dt} = \frac{W_s(H_s - H_{lw}) - Q}{m_w c_{pLw} + m(x_1 c_{pL1} + x_2 c_{pL2})} \quad (14)$$

where H_s is the enthalpy of the steam in J/kmol, H_{lw} is the enthalpy of liquid water in J/kmol, Q is the rate of heat transfer to the surroundings in J/sec, c_{pLw} is the molar specific heat of the water in J/kmol-K, m is the mass of the organic phase in the still in kmol, x_1 and x_2 are the mole fractions, and c_{pL1} and c_{pL2} are the molar specific heats of organic compounds No. 1 and 2, respectively, in J/kmol-K. The heat transfer rate to the surroundings is calculated from the following equation.

$$Q = UA(T - T_a) \quad (15)$$

where UA is the product of the overall heat transfer coefficient U and the contact area A with the surroundings in J/s-K, T_a is the ambient temperature in K, and T is the temperature of the liquid in the still in K.

Assuming ideal liquid behavior, Raoult's law can be used to calculate the vapor mole fraction of the components in the organic phase

$$y_1 = \frac{x_1 P_1}{P} \quad y_2 = \frac{x_2 P_2}{P} \quad (16)$$

where P is the total pressure in Pa and P_1 and P_2 are the vapor pressures of the organic compounds in Pa. The mole fraction of the water which is immiscible in the organic phase is given by $y_w = P_w/P$. y_1 and y_2 are the vapor phase mole fraction of n-octane and n-decane respectively. The heating period continues until the sum of vapor pressures of the organic compounds and the water is equal to the total pressure. Thus, the bubble point equation to be satisfied can be expressed as

$$f(T) = 1 - (y_1 + y_2 + y_w) = 0 \quad (17)$$

We now present the model for the distillation period. During the distillation period, there is output of water vapor from the still.

$$\frac{dm_w}{dt} = W_s - Vy_w \quad (18)$$

where V is the outlet vapor flow rate. Material balances on the two organic compounds yield two additional differential equations

$$\frac{d(mx_1)}{dt} = -Vy_1 \quad \frac{d(mx_2)}{dt} = -Vy_2 \quad (19)$$

The organic mass in the still at any time is given by: $m = mx_1 + mx_2$. The temperature in the still changes in a manner so that the bubble point equation is satisfied. The energy balance at a particular temperature yields the momentary vapor flow rate

$$V = \frac{W_s(H_s - H_{LW}) - Q}{H_v - [y_w h_{lw} + (y_1 h_{L1} + y_2 h_{L2})]} \quad (20)$$

where H_v is the molar enthalpy of the vapor phase; h_{Lw} , h_{L1} , and h_{L2} are the liquid phase molar enthalpies of water, n-octane and n-decane, respectively. Material balances on the water and organic phases in the still can provide the amount and the mole fractions of the various components in the distillate.

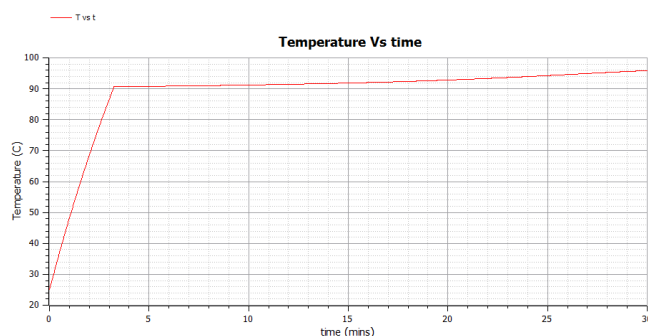
6.2 Example: n-octane, n-decane distillation

Semibatch steam distillation of a mixture containing n-octane (compound 1) and n-decane (compound 2) is to be processed. Initially $M = 0.015$ kmol of organics with composition $x_1 = 0.725$ is charged into the still. The initial temperature in the still is $T_0 = 25^{\circ}\text{C}$. Starting at time $t = 0$, steam at a temperature $T_{\text{steam}} = 99.2^{\circ}\text{C}$ is bubbled continuously through the organic phase at the rate of $M_s = 3.85 \times 10^{-5}$ kmol/s. All the steam is assumed to condense during the heating period. The ambient temperature is $T_E = 25^{\circ}\text{C}$ and the heat transfer coefficient between the still and the surrounding is $UA = 1.05$ J/s-K. The ambient pressure is $P = 9.839 \times 10^4$ Pa.

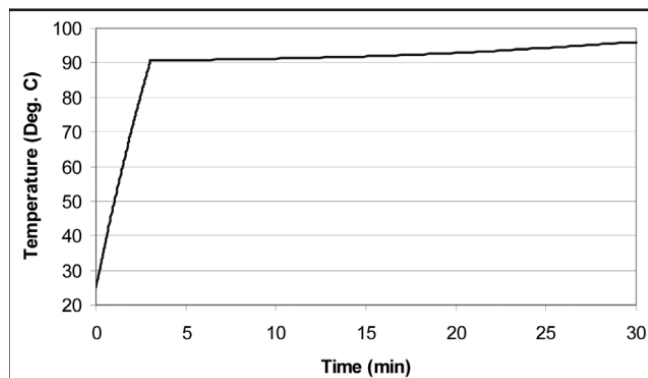
Assumptions: 1) Ideal behavior of all components in pure state or mixture; 2) complete immiscibility of the water and the organic phases; 3) ideal mixing in the boiler; and 4) equilibrium between the organic vapor and its liquid at all times. The standard state for enthalpy calculations pure liquids at 0°C and 1 atm. can be used.

We have to Calculate and plot the still temperature (T), component mole fractions inside the still (x_1 , x_2 , y_1 , and y_2), and the component mole fractions in the distillate ($x_{1\text{dist}}$ and $x_{2\text{dist}}$) using the data and the initial values provided.

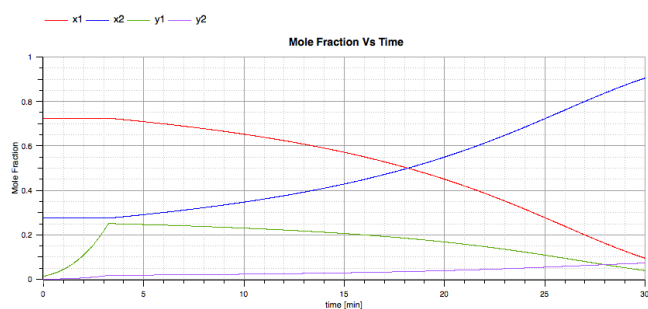
We have to determine the lowest n-octane mole fraction in the feed that can yield a distillate concentration of 90% of n-octane. Compute the percent recovery of n-octane in the distillate as function of its concentration in the feed.



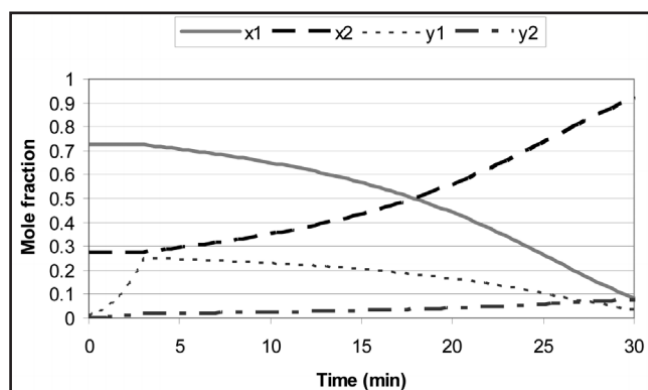
(a) Temperature profiles generated in OpenModelica 1.11.0



(b) Temperature profiles as reported in literature (Shacham et al., 2012)



(c) Profiles of vapor and liquid compositions, as generated in OpenModelica 1.11.0



(d) Profiles of vapor and liquid compositions, as reported in the literature (Shacham et al., 2012)

Figure 12. Comparison of temperature and composition change during semi-batch steam distillation

Vary the feed concentration in the range where the requirement for the n-octane concentration in the distillate is attainable.

Plots in Figure 12 shows that the results of OpenModelica are in agreement with that of (Shacham et al., 2012). It can be observed that the temperature increases during the heating period and then stays constant during the distillation period when the Bubble point is attained. The liquid phase compositions is constant during the heating period as there is no vapor formation.

7 Conclusion and Future Work

In this paper, we have implemented and compared three different ways of making available thermodynamics in OpenModelica. Each of these approaches has been illustrated with simulations of one or more chemical processes. We have found the native port to be the most efficient.

We have compared the results of OpenModelica with those from DWSIM, Aspen Plus, and published literature, and they match quite well in all calculations.

As now OpenModelica has its own thermodynamic engine, a library of steady state chemical process models could be modeled. It may also be possible to build a library of dynamic chemical process models in OpenModelica, to carry out general purpose dynamic simulation.

We hope to explore the possibility of enhancing OMEDIT's (Asgharand et al., 2011) features so that the GUI shall resemble that of established simulators, such as Aspen Plus or DWSIM, for chemical process simulation.

We propose to check the correctness of thermodynamic calculations by solving a large number of already solved flowsheets. Sources for these will include examples from books, journals, reports and sample problems from other process simulators. We hope to present these flowsheets in a way similar to what we have done for DWSIM (DWSIM-Team-FOSSEE-Project, 2017). Usefulness of such an initiative has been articulated in a similar context (Braatz, 2014).

A difficult task we face is with respect to thermodynamics in general and the thermodynamic database, in particular. This facility has to be strengthened by adding information on more chemicals and more thermodynamic calculations. We invite experts in this important area to contribute and to make OpenModelica a much better open source process simulator.

Acknowledgements

This work has been supported by Swedish Vinnova governmental agency and the Indian Department of Science and Technology governmental agency in the Indo-Swedish RTISIM project, and by the National Mission on Education through ICT, Ministry of Human Resource Development, through the FOSSEE project. The OpenModelica development is supported by the Open Source Modelica Consortium.

References

- Adeel Asgharand, Sonia Tariq, Mohsen Torabzadeh-Tariand, Peter Fritzson, Adrian Pop, Martin Sjolund, Parham Vasaiely, and Wladimir Schamai. An open source modelica graphic editor integrated with electronic notebooks and interactive simulation. *Proc. of the 8th International Modelica Conference 2011*, pp, pages 739–747, 2011.
- Aspentech. Aspen plus. <http://www.aspentech.com/products/engineering/aspen-plus/>, 2017. Last seen on 1 April 2017.
- R. D. Braatz. Scilab textbook companions. *IEEE Control Systems Magazine*, page 76, June 2014.
- DWSIM-Team-FOSSEE-Project. Dwsim flowsheets. <http://dwsim.fossee.in/flowsheeting-project/completed-flowsheet>, 2017. Last seen on 1 April 2017.
- Peter Fritzson. *Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Second edition, 2014. ISBN 9781118989166. doi:10.1002/9781118989166.
- Jürgen Gmehling, Jochen Menke, Jörg Krafczyk, and Kai Fischer. A data bank for azeotropic data - status and applications. *Fluid Phase Equilibria*, 103(1):51–76, 1995. ISSN 03783812. doi:10.1016/0378-3812(94)02569-M.
- H.A. Kooijman and R. Taylor. *The ChemSep Book*. Books on Demand, Norderstedt, Germany, 2001.
- Daniel Medeiros. Dwsim technical document. Technical report, 2015. <http://dwsim.inforside.com.br/>.
- Modelica Association. ModelicaTM - A Unified Object-Oriented Language for Physical Systems Modeling: Language Specification. *ReVision*, 2000. ISSN 09284869. doi:10.1016/S0928-4869(97)84257-7.
- Ding-Yu Peng and Donald B. Robinson. A New Two-Constant Equation of State. *Industrial & Engineering Chemistry Fundamentals*, 15(1):59–64, 1976. ISSN 0196-4313. doi:10.1021/i160057a011. URL <http://pubs.acs.org/doi/abs/10.1021/i160057a011>.
- Peter Piela, Roy McKelvey, and Arthur Westerberg. An introduction to the ascend modeling system: Its language and interactive environment. *Journal of Management Information Systems*, 9(3):91–121, 1992.
- Henri Renon and J. M. Prausnitz. Local compositions in thermodynamic excess functions for liquid mixtures. *AIChE Journal*, 14(1):135–144, 1968. ISSN 15475905. doi:10.1002/aic.690140124.
- Brandon Rhodes and John Goerzen. Foundations of Python Network Programming, 2010. ISSN 1098-6596. URL <http://www.springerlink.com/index/10.1007/978-1-4302-3004-5%5Cnhttp://it-ebooks.info/book/1796/>.
- M. Shacham, M. B. Cutlip, and M. Elly. Semi-Batch Steam Distillation Of a Binary Organic Mixture: a Demonstration of Advanced Problem-Solving Techniques and Tools. *Chemical Engineering Education*, 46(3):173–181, Summer 2012.
- J M Smith, H C Van Ness, and M M Abbott. *Introduction to Chemical Engineering Thermodynamics*, volume 27. McGraw Hill Education, 2005. ISBN 0072402962. doi:10.1021/ed027p584.3.
- G. Soave. Equilibrium constants from a modified redlich-kwong equation of state. *Chemical Engineering Science*, 27(6):1197–1203, 1972.
- A.W. Westerberg, H.P. Hutchison, R.L. Motard, and P. Winter. *Process Flowsheeting*. Cambridge University Press, Cambridge, 1979.

Integrated Process and Molecular Design with Modelica Using Continuous-Molecular Targeting

Christoph U. Gertig¹ Dominik Tillmanns¹ Johannes Schilling¹ Uwe Bau¹

Franz Lanzerath¹ Joachim Gross² André Bardow¹

¹Institute of Technical Thermodynamics, RWTH Aachen University
Schinkelstr. 8, 52062 Aachen, Germany

²Institute of Technical Thermodynamics and Thermal Process Engineering, Stuttgart University
Pfaffenwaldring 9, 70569 Stuttgart, Germany

andre.bardow@ltt.rwth-aachen.de

Abstract

The performance of many chemical and energy conversion processes depends on the choice of the molecules used, e.g. as solvents or working fluids. To capture the complex relations between the properties of the molecules used and the process conditions, the selection of suitable molecules should be directly integrated into process design. Solving the resulting challenging integrated design problem is enabled by the Continuous-Molecular Targeting – Computer-Aided Molecular Design (CoMT-CAMD) approach. Here, the combinatorial complexity of the molecular decisions is avoided by relaxing molecular parameters in a physically-based thermodynamic model. So far, implementations of CoMT-CAMD were based on procedural programming languages. This impedes reusability and the investigation of process variants as well as the design of complex processes. In order to overcome these shortcomings, we implement the CoMT-CAMD approach based on object-oriented process modeling and thus enable the integrated process and molecular design with Modelica. The resulting approach is demonstrated for the design of a process and the working fluid for a geothermal Organic Rankine Cycle application.

Keywords: *GenOpt, optimization, integrated fluid and process design, computer-aided molecular design, PC-SAFT*

1 Introduction

In order to achieve high performance, chemical as well as energy conversion processes have to be tailored to the specific applications. The key to tailoring a process is often the choice of suitable molecules. Examples are the selection of solvents for absorption processes (Adjiman *et al.*, 2014; Bardow *et al.*, 2010; Burger *et al.*, 2015; Papadopoulos and Linke, 2009), refrigerants for compression chillers (Roskosch and Atakan, 2015; Sahinidis *et al.*, 2003) and working fluids for Organic Rankine

Cycles (Linke *et al.*, 2015; Bao and Zhao, 2013; Lampe *et al.*, 2015).

Today, design methods usually separate the choice of suitable molecules and the process design (for a literature review, see e.g. Linke *et al.*, 2015): in a first step, molecular candidates are pre-selected using criteria based on heuristics. In a second step, the pre-selected molecules are used for process optimization.

However, these two-step approaches usually lead to suboptimal solutions. Heuristic selection criteria cannot capture the strong and complex relations between the properties of chosen molecules and the corresponding optimal process conditions. Therefore, the global optimum might already be excluded from the solution space when heuristics are applied. Consequently, the design of molecules should be directly integrated into the process design (Adjiman *et al.*, 2014; Linke *et al.*, 2015). The direct formulation of this integrated design problem leads to a mixed integer nonlinear program (MINLP) (Gani, 2004) where each molecule considered adds one degree of freedom. Due to the large number of potential candidate molecules, the solution of this MINLP is usually prohibitively difficult.

Thus, systematic approaches have been proposed for the approximate solution of the integrated design problem: Pereira *et al.* (2008; 2011) solve the integrated design problem based on property predictions with the statistical associating fluid theory for potentials of variable attractive range (SAFT-VR) with a search space limited to linear alkanes. For Organic Rankine Cycles, the review by Linke *et al.* (2015) summarizes the state of the art. Recently, Burger *et al.* (2015) have solved the integrated design problem utilizing a hierarchical approach and short-cut models for the process. Gopinath *et al.* (2016) have proposed an approach for the integrated design utilizing physical domain reduction. They employ tests to remove regions from the molecular and process domains where constraints, e.g., on phase behavior, are violated.

Bardow *et al.* (2010) proposed a targeting-based design approach called Continuous-Molecular Targeting – Computer-Aided Molecular Design (CoMT-CAMD) for the integrated molecular and process design. Here, the molecular properties are modeled by the Perturbed-Chain Statistical Associating Fluid Theory (PC-SAFT) equation of state (Gross and Sadowski, 2001). In PC-SAFT, each fluid is described by a set of physically-based pure component parameters. In the first step of CoMT-CAMD, the so-called Continuous-Molecular Targeting (CoMT), the discrete PC-SAFT pure component parameters are regarded as continuous degrees of freedom of the design problem and are optimized simultaneously with the process conditions (Lampe *et al.*, 2014; Stavrou *et al.*, 2014). The resulting design problem can be formulated as a nonlinear program (NLP) optimization problem. The results of this optimization are the set of optimal pure component parameters for a hypothetical target molecule and the corresponding optimal process parameters. In a second step, Computer-Aided Molecular Design (CAMD) methods can be used to design the real molecule which best matches the optimal process performance (Lampe *et al.*, 2015). The CoMT-CAMD approach has been applied successfully to the design of solvents for CO₂ capture (Stavrou *et al.*, 2014) and working fluids for Organic Rankine Cycles (Lampe *et al.*, 2014; 2015). A similar targeting approach for integrated design was presented by Roskosch and Atakan (2015). They use a cubic equation of state for property modeling and relax its parameters in a simultaneous optimization of a compression heat pump process and working fluid. Subsequently, they select suitable fluids from databanks utilizing a fitted function for COP estimation.

So far, integrated process and molecular design with CoMT-CAMD was based on process models implemented in a procedural programming language. This hinders the reusability of models and complicates the design of complex processes as well as the investigation of process variants. Furthermore, the use of procedural languages is not convenient in case dynamic processes have to be investigated.

These shortcomings can be overcome by using a language suited for object-oriented and equation-based modeling like Modelica (Fritzson, 1998) to model the process. Thus, in this work, we present the first implementation of the CoMT-CAMD approach based on Modelica process models. Thereby, we enable the integrated process and molecular design with Modelica. In order to illustrate the design approach, a case study is presented for the design of a geothermal Organic Rankine Cycle (ORC) application.

The paper is structured as follows: in Section 2, the CoMT-CAMD approach is explained. In Section 3, the implementation of CoMT-CAMD based on Modelica

models is described. The case study of the ORC application is presented in Section 4 before conclusions are drawn in Section 5.

2 The CoMT-CAMD Approach

The Continuous-Molecular Targeting – Computer-Aided Molecular Design (CoMT-CAMD) approach was introduced by Bardow *et al.* (2010). In CoMT-CAMD, the fluids are modeled by the Perturbed-Chain Statistical Associating Fluid Theory (PC-SAFT) equation of state (EOS). Thus, any fluid can be described by a set of PC-SAFT pure component parameters. The CoMT-CAMD approach comprises two main steps as shown in Figure 1.

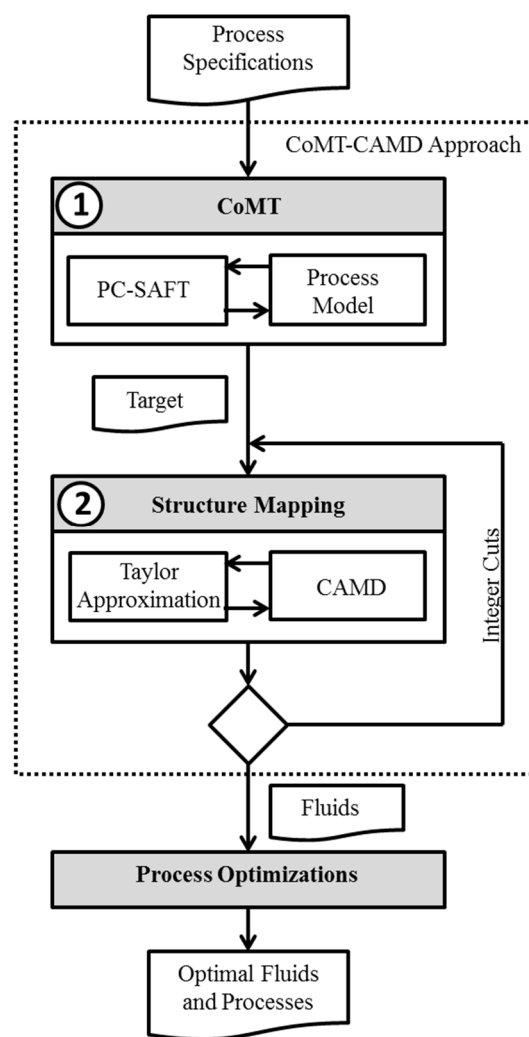


Figure 1: The workflow of the CoMT-CAMD approach.

In the first step, the so-called Continuous-Molecular Targeting (CoMT), the process conditions and the molecules are simultaneously optimized. For this purpose, the PC-SAFT pure component parameters describing the properties of the molecules are relaxed and treated as continuous variables of the optimization problem. The results of this optimization are the PC-SAFT pure component parameters of a hypothetical optimal fluid,

the so-called target, and the corresponding optimal process conditions. In the second step, real fluids are identified in the so-called structure-mapping. For this purpose, a second-order Taylor-approximation is computed around the hypothetical optimum. This Taylor approximation is used to estimate the objective function value of processes with real fluids. In the structure-mapping, real fluids can either be selected from databanks of known fluids (Lampe *et al.*, 2014) or designed using Computer-Aided Molecular Design (CAMD) algorithms (Lampe *et al.*, 2015). This CAMD algorithm employs Group Contribution (GC) methods to link PC-SAFT pure component parameters to molecule structures. This link is used to identify the optimal molecular structure by solving a mixed-integer quadratic program (MIQP) optimization problem (for details, see Section 2.3). Due to inaccuracies in the models and the method, usually, not only one fluid but a ranking of candidates is desired. Thus, the CAMD algorithm is applied repeatedly using integer cuts (see e.g. Fazlollahi *et al.*, 2012) to exclude previously found molecules in each new run. The CoMT-CAMD approach thus yields a list of real fluids which best match the target. For these fluids, individual process optimizations are performed to determine the optimal process performance.

Details of the CoMT step are explained in Section 2.1 and the PC-SAFT equation of state is described in Section 2.2. The structure-mapping step of CoMT-CAMD is discussed in more detail in Section 2.3 followed by a short section on the final process optimizations.

2.1 Continuous-Molecular Targeting

The aim of the CoMT step is the simultaneous optimization of process conditions and fluids to obtain a target for the subsequent structure-mapping (Bardow *et al.*, 2010). As mentioned before, the fluids are modeled with the PC-SAFT equation of state. The PC-SAFT pure component parameters used to describe the fluids are relaxed and thus treated as continuous variables of the optimization problem. This relaxation transforms the mixed-integer nonlinear program (MINLP) of the fully integrated design into a nonlinear program (NLP) given by problem (1) (Lampe *et al.*, 2015):

$$\begin{aligned}
 & \max_{x,y} f(x,y) \\
 & \text{s. t. } \begin{cases} g_1(x,y) \leq 0 \\ g_2(x,y) = 0 \\ h(x,y) = 0 \\ Ay \leq b \end{cases} \quad \begin{array}{l} \text{"process model"} \\ \text{"PC-SAFT"} \\ \text{"convex hull"} \end{array} \\
 & x_{\min} \leq x \leq x_{\max} \in \mathbb{R}^m \\
 & y_{\min} \leq y \leq y_{\max} \in \mathbb{R}^l.
 \end{aligned} \tag{1}$$

Here, x is the vector with the degrees of freedom of the process and y the vector containing the PC-SAFT pure component parameters of the molecules.

The objective function f is, for example, a thermodynamic measure like an efficiency. The process model is formulated in terms of inequality constraints g_1 and equality constraints g_2 . The PC-SAFT equation of state h is used to compute thermodynamic quantities based on the pure component parameters and the process conditions. Additionally, bounds are defined on the process degrees of freedom x_{\min} and x_{\max} and the PC-SAFT pure component parameters y_{\min} and y_{\max} .

Additional linear inequality constraints ($Ay \leq b$) are used which set up a convex hull around the PC-SAFT pure component parameters of real fluids (see Lampe *et al.* (2014) for details). This ensures that the CoMT step results in a hypothetical fluid which is similar to any real substance. The NLP optimization of the CoMT step results in a hypothetical optimal fluid y^* and optimal process conditions x^* .

In general, the pure component parameters y^* of the optimal hypothetical fluid are not equal to those of any real fluid. Thus, real substances with favorable performance are identified in a subsequent structure-mapping (discussed in Section 2.3).

2.2 PC-SAFT Equation of State

The Perturbed-Chain Statistical Associating Fluid Theory (Gross and Sadowski, 2001, 2002; Gross, 2005; Gross and Vrabec, 2006) is a physically-based equation of state model for the residual Helmholtz energy. The underlying molecular picture considers molecules as chains of hard spheres (segments) which interact with each other.

Both pure fluids and fluid mixtures are described based on typically 3 to 7 parameters per each pure component. In this work, we consider only non-polar and non-associative molecules, so that 3 parameters of PC-SAFT are sufficient: the segment number m , the segment diameter σ and the segment dispersion energy ε/k .

As only the residual part of the Helmholtz energy is calculated from PC-SAFT, an additional property is required to calculate absolute caloric properties. Here, the additional property is the ideal gas heat capacity. In the CoMT step, the molecules are exclusively described by the PC-SAFT pure component parameters, which should therefore also be used in order to obtain the ideal gas heat capacity (Lampe *et al.*, 2014). For this purpose, Quantitative Structure-Property Relationships (QSPR) are used with PC-SAFT pure component parameters as inputs to calculate ideal gas heat capacities $c^{p,ig}$ (for details see Stavrou *et al.*, 2014; Lampe *et al.*, 2014; 2015). Additionally, another QSPR model based on PC-SAFT pure component parameters is used to calculate molar masses (Lampe *et al.*, 2015).

By combining the QSPR methods and PC-SAFT, all thermodynamic equilibrium properties can be calculated based on 3 PC-SAFT pure component parameters in a thermodynamically consistent form.

2.3 Structure-Mapping using Computer-Aided Molecular Design

As shown in Figure 1, we use Computer-Aided Molecular Design (CAMD) in the structure-mapping step in order to design real fluids which best match the target obtained from the CoMT step. For this purpose, a measure for the expected performance of fluids in the process is needed.

A simple measure is the distance $\|y - y^*\|$ of a real fluid's pure component parameters from those of the hypothetical optimal fluid in the space of the PC-SAFT parameters. However, this simple measure is not appropriate since it neglects the different sensitivities of the objective function with respect to different PC-SAFT parameters and it depends on scaling. To overcome these limitations, a performance measure is calculated in the space of the objective function itself. For this purpose, we compute the following Taylor-approximation to estimate the performance:

$$\hat{f}(y) = f^{\text{opt}}(y^*) + \frac{df^{\text{opt}}}{dy}(y - y^*) + \frac{1}{2}(y - y^*)^T \frac{d^2 f^{\text{opt}}}{dy^2}(y - y^*). \quad (2)$$

Here, f^{opt} is the objective function of Problem (1) $f(x, y)$ rewritten such that it yields the optimal performance solely based on the PC-SAFT pure component parameters:

$$\begin{aligned} f^{\text{opt}}(y) = \min_x f(x, y) \\ \text{s.t. } g_1(x, y) \leq 0 \\ g_2(x, y) = 0 \\ h(x, y) = 0 \\ x_{\min} \leq x \leq x_{\max} \in \mathbb{R}^m. \end{aligned} \quad (3)$$

The estimated objective function value $\hat{f}(y)$ is used as assessment criterion in the structure-mapping.

In this work, a CAMD algorithm is employed to design optimal molecules. This CAMD algorithm optimizes the molecular structure with respect to the performance estimate $\hat{f}(y)$ (2). In order to evaluate $\hat{f}(y)$, the PC-SAFT pure component parameters have to be known for each molecule. Lampe *et al.* (2015) employ a Group Contribution (GC) method in order to calculate PC-SAFT pure component parameters from molecule structures. They use the homosegmented approach from Sauer *et al.* (2014) which they call GPC-SAFT. We use the same method as Lampe *et al.* (2015) and all non-polar, non-associating groups they considered.

Following Lampe *et al.* (2015), the CAMD problem is formulated as mixed integer quadratic problem (MIQP) by employing the second-order Taylor approximation (2) as objective function:

$$\begin{aligned} \min_n \hat{f}(y) \\ \text{s.t. } GC(n) = y \\ FEAS(n) \leq 0 \\ Ay \leq b_{\text{rel}} \\ n \in \mathbb{Z}^0 \end{aligned} \quad (4)$$

In this formulation, a fluid is described by a vector n representing the functional groups constituting the molecular structure. The set of PC-SAFT pure component parameters y are calculated with a GC method as described above ($GC(n) = y$). A set of constraints ($FEAS(n) \leq 0$) ensures feasible connectivity of the designed molecules (Struebing, 2011; Struebing *et al.*, 2011). A convex hull ($Ay \leq b_{\text{rel}}$) as described in Section 2.1 is also used in the CAMD optimization. In order to permit the design of novel fluids, the convex hull is relaxed compared to the one used in the CoMT step, i.e. $b_{\text{rel}} \geq b$ (Lampe *et al.*, 2015). The result of the MIQP is the optimal molecular structure of a real fluid.

A ranking of fluid candidates can be obtained by repeating the CAMD optimization with integer cuts (see e.g. Fazlollahi *et al.*, 2012) which exclude previously found molecular structures from the design space.

2.4 Final Process Optimizations

The result of the structure-mapping step of CoMT-CAMD is a ranking of candidate molecules. Since the objective function used for the structure-mapping is a second-order Taylor-approximation (2) as described in Section 2.3, process optimizations are performed with the identified real molecules in a final step in order to obtain the respective optimal process conditions and actual objective function values. These objective function values are also used to refine the ranking of the molecules.

3 CoMT-CAMD with Modelica Models

As explained in Section 1, the integrated process and fluid design with the CoMT-CAMD approach was based on process models implemented in a procedural programming language so far. The contribution of this work is to enable the utilization of CoMT-CAMD based on object-oriented process modeling with Modelica. The utilization of this language facilitates the convenient development, adaption and reusability of models. In this way, the design of complex processes is enabled. Furthermore, equation-based modeling with Modelica is suited for the investigation of dynamic processes. The implementation of all steps of CoMT-CAMD based on Modelica is described in the subsequent sections.

3.1 PC-SAFT Modelica-Package

Since the CoMT-CAMD approach is based on fluid property calculations with PC-SAFT, these calculations have to be available in Modelica to model the processes. Examples of such property calculations are the calculation of saturation pressures and temperatures as well as of caloric properties like specific enthalpies.

For this purpose, a PC-SAFT implementation written in the procedural language FORTRAN 90 is used for external property calculations. The interface with Modelica is constructed using the Modelica external function interface (Modelica Association, 2012).

According to Modelica Association (2012), however, interfaces are only supported with external FORTRAN 77 code. For this reason, FORTRAN 77 subroutines are implemented as wrappers for all relevant top-level subroutines in the FORTRAN 90 code. The wrapper subroutines are called by Modelica external functions (for details see Modelica Association (2012)). As the properties are calculated on mole basis in the external code and calculations on mass basis are desired for the process simulations in Modelica, additional functions for conversions between mole and mass basis as well as unit conversions are implemented.

In order to enable the convenient use of the PC-SAFT property calculations in Modelica, a Modelica package with the following functions is developed:

- Modelica external functions needed to call external subroutines.
- Additional functions for conversion between mole and mass basis as well as unit conversions.
- “Top-level” functions for each type of property calculation which call the external functions and functions for necessary conversions, perform any additional calculations required and return the desired thermodynamic properties.

As the focus of this work is embedding Modelica in the existing CoMT-CAMD framework, the PC-SAFT package is created independent of other libraries. The following types of property calculations are available in the package:

- Vapor-Liquid Equilibrium (VLE) calculations for pure components with specified saturation pressure or temperature.
- pT-flash calculations for mixtures.
- Bubble point and dew point calculations for mixtures.
- Property calculations for pure components and mixtures in subcooled liquid and superheated vapor states.
- Estimation of the critical point.

All these functions can be conveniently used in Modelica process models.

3.2 CoMT-CAMD Based on Modelica Models and GenOpt

In order to facilitate the integrated optimization of processes and fluids based on object-oriented modeling, the process and all its equipment models are implemented in Modelica. For any required property calculation, the functions in the PC-SAFT package described in Section 3.1 are utilized.

In order to find the hypothetical optimum in the CoMT step, an objective function is defined. Then, the Modelica process model is used to search for the point in the solution space of process conditions and PC-SAFT pure component parameters with the maximal objective function value.

One tool suited for this search is the open-source optimization program GenOpt (Generic Optimization Program) (Wetter, 2000; 2016) which provides algorithms for parametric runs as well as several optimization algorithms and can be coupled with Modelica models (Wetter, 2009).

In this work, parametric runs with GenOpt are used in the CoMT step to find the hypothetical optimal fluid and the corresponding optimal process conditions. The derivatives required for the Taylor-approximation of the objective function are approximated by finite differences. These are computed with MatLab from the results of the GenOpt calculations. As the output files of GenOpt are conveniently imported by MatLab, no special interface is required here.

The CAMD formulation described in Section 2.3 was implemented in the high-level modeling system GAMS (General Algebraic Modeling System) (Rosenthal, 2016) which is used to solve the MIQP with integer cuts to obtain a ranking of real fluids.

For the final process optimizations described in Section 2.4 parametric runs with GenOpt are used to find optimal process conditions. The workflow of the Modelica-based CoMT-CAMD is shown in Figure 2.

4 Case Study: Design of a Geothermal ORC Application

The proposed molecular and process design with the CoMT-CAMD approach based on process modeling with Modelica is applied to a case study of a geothermal application for an Organic Rankine Cycle. Organic Rankine Cycles (ORCs) are used to transform low temperature heat into electric power (Colonna *et al.*, 2015). The performance of ORCs depends strongly on the properties of the chosen working fluid. Therefore, the ORC is a very relevant case study for the integrated fluid and process design (Linke *et al.*, 2015; Bao and Zhao, 2013). In the subsequent sections, the ORC process itself as well as the process model implemented in Modelica and the specifications of the case study are described. In Section 4.4, the results of the integrated fluid and process design are presented.

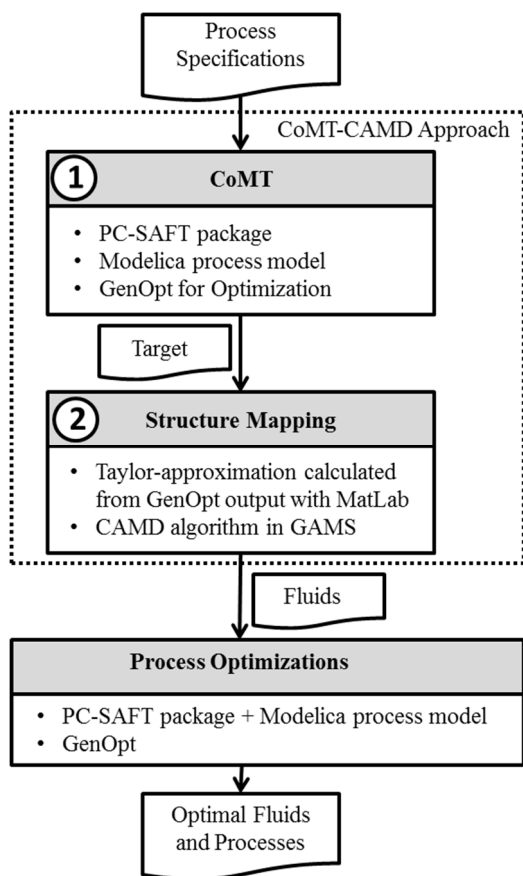


Figure 2: Workflow of the integrated molecular and process design with CoMT-CAMD including the tools used in the individual steps.

4.1 Organic Rankine Cycles

Organic Rankine Cycles (ORCs) are energy conversion processes with a sequence of process steps equivalent to the classical thermodynamic Rankine cycle (Colonna *et al.*, 2015). The flowsheet and the temperature-entropy diagram of a basic ORC process are shown in Figure 3. In contrast to classical Rankine cycles which use water as working fluid, organic fluids are utilized in ORCs. The organic fluid can be tailored to specific applications, in particular the utilization of low temperature heat and cases with low power output where water becomes unfavorable (Colonna *et al.*, 2015).

In the ORC process, the liquid working fluid (state 1 in Figure 3) is pressurized in a pump ($1 \rightarrow 2$) with power input P_p before the vaporization in an evaporator ($2 \rightarrow 3$) using heat (\dot{Q}_{evap}) from the available heat source with inlet temperature $T_{\text{HS,in}}$. At the outlet of the evaporator (State 3 in Figure 3), the working fluid can be in a saturated or superheated vapor state. The vapor is expanded in a turbine ($3 \rightarrow 4$) to gain the desired power output P_T . Subsequently, the fluid is completely condensed in a condenser ($4 \rightarrow 1$) by transferring heat (\dot{Q}_{cond}) to a cooling medium with inlet temperature $T_{\text{CW,in}}$ in order to close the cycle.

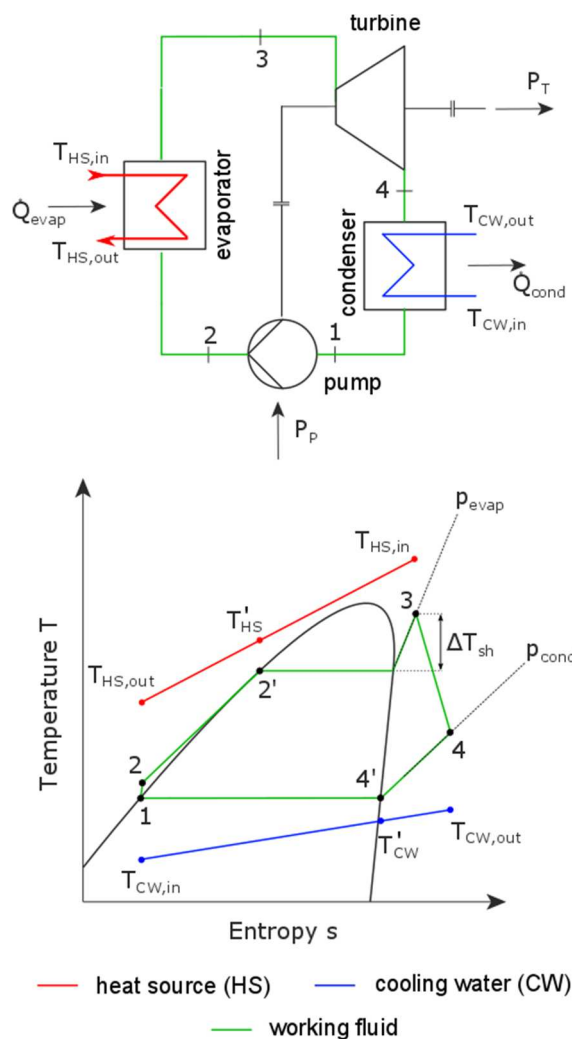


Figure 3: Upper part: Flowsheet of a basic ORC process. Lower part: T-s diagram for ORC process, heat source and heat sink.

4.2 Modelica ORC Model

To demonstrate the integrated design of working fluid and process, the ORC is modeled assuming steady state conditions. Four equipment models are implemented and connected to a process model (cf. Figure 3): a pump, a turbine and two heat exchangers, namely the evaporator and the condenser. These equipment models are connected as shown in Figure 3 via suitable connectors to assemble an ORC process model. Although the use of libraries for modeling the ORC would be possible, no existing libraries are used in this work to keep the equipment models and connectors simple. The most important model equations and assumptions are presented in the following.

4.2.1 Pump

The pump is adiabatic and modeled based on an isentropic pressure increase and an isentropic efficiency $\eta_{\text{is,P}}$:

$$\begin{aligned}
s_{p,in} &= s_{p,out}^* \\
h_{p,out}^* &= f(p_{\text{evap}}, s_{p,out}^*) \\
h_{p,out} &= h_{p,in} + \frac{h_{p,out}^* - h_{p,in}}{\eta_{is,P}}
\end{aligned} \quad (5)$$

Here, $h_{p,in} = h_1$ and $h_{p,out} = h_2$ are the specific enthalpies at pump inlet and outlet, respectively, and $h_{p,out}^*$ is the specific enthalpy at the pump outlet in case of isentropic pressure increase. $s_{p,in}$ and $s_{p,out}^*$ are the specific entropies at inlet and outlet conditions for the isentropic case. The pressure level of the evaporation is termed p_{evap} .

From the energy balance, the required power input can be obtained:

$$P_P = \dot{m}_{WF}(h_{p,out} - h_{p,in}) \quad (6)$$

where \dot{m}_{WF} is the working fluid mass flow rate and P_P the required power input.

4.2.2 Turbine

The adiabatic turbine is modeled in the same way as the pump based on an isentropic efficiency $\eta_{is,T}$:

$$\begin{aligned}
s_{T,in} &= s_{T,out}^* \\
h_{T,out}^* &= f(p_{\text{cond}}, s_{T,out}^*) \\
h_{T,out} &= h_{T,in} \\
&\quad + \eta_{is,T}(h_{T,out}^* - h_{T,in})
\end{aligned} \quad (7)$$

where p_{cond} is the pressure level of condensation. The power output of the turbine can be obtained from an energy balance as:

$$P_T = \dot{m}_{WF}(h_{T,out} - h_{T,in}) \quad (8)$$

4.2.3 Evaporator

It is assumed that pressure losses in the evaporator can be neglected such that preheating and evaporation take place on the same constant pressure level p_{evap} . The geothermal heat source is assumed to be hot liquid water

with a known inlet temperature $T_{HS,in}$, mass flow \dot{m}_{HS} and constant specific heat capacity c_{HS} .

The heat transferred is calculated from energy balances of the evaporator:

$$\begin{aligned}
\dot{Q}_{\text{evap}} &= \dot{m}_{HS}c_{HS}(T_{HS,in} - T_{HS,out}) \\
&= \dot{m}_{WF}(h_{\text{evap,out}} - h_{\text{evap,in}})
\end{aligned} \quad (9)$$

where $T_{HS,out}$ denotes the outlet temperature of the heat source and $h_{\text{evap,in}}$ and $h_{\text{evap,out}}$ the specific enthalpies of the working fluid at the evaporator inlet and outlet, respectively.

The specific enthalpy of the working fluid at the outlet of the evaporator $h_{\text{evap,out}}$ is calculated from the evaporator pressure p_{evap} and outlet temperature T_3 . The temperature T_3 is calculated from the saturation temperature of the working fluid and a degree of superheating ΔT_{sh} .

4.2.4 Condenser

The condenser is modeled based on energy balances equivalent to those shown for the evaporator. It is assumed that cooling water with known inlet temperature $T_{CW,in}$ and a known constant specific heat capacity c_{CW} is used as cooling medium. The required mass flow of the cooling water is calculated from a given temperature increase of 5 K. The outlet state of the working fluid is assumed to be a saturated liquid state. Pressure losses in the condenser are neglected.

4.3 Specifications of the Case Study and the Optimization Problem

The specifications of the case study are based on the subcritical geothermal ORC application presented by Heberle and Brüggemann (2010) and are shown in Table 1. The ORC designed in our work is medium-sized regarding its power capacity and utilizes low-temperature heat from a geothermal source with a maximum temperature of 120 °C.

Table 1: Specifications of the case study based on Heberle and Brüggemann (2010).

Parameter	Symbol	Value	Parameter	Symbol	Value
heat source mass flow rate	\dot{m}_{HS}	66 kg/s	cooling water inlet/outlet temperature	$T_{CW,in}/T_{CW,out}$	15 °C / 20 °C
heat source inlet temperature	$T_{HS,in}$	120 °C	cooling water heat capacity	c_{CW}	4200 J/(kg K)
heat source specific heat capacity	c_{HS}	4200 J/(kg K)	pinch temperature difference	ΔT^{pinch}	5 K
min. and max. absolute pressure	$p^{\text{min/max}}$	1 bar / 50 bar	min. vapor fraction at turbine outlet	$\beta^{\text{turb,min}}$	0.95
max. reduced pressure	$p^{\text{r,max}}$	0.8			

Table 2: Optimal process conditions and PC-SAFT parameters of the hypothetical optimal fluid resulting from the CoMT step as well as the achieved net power output. Values at their bounds are marked with a +.

Parameter	Symbol	Value	Parameter	Symbol	Value
condensation pressure	p_{cond}^*	7.1 bar	segment number	m^*	3.15
evaporation pressure	p_{evap}^*	25.7 bar	segment diameter	σ^*	3.45 Å
degree of superheating ⁺	ΔT_{sh}^*	0 °C	segment dispersion energy	$(\varepsilon/k)^*$	164.0 K
working fluid mass flow rate	\dot{m}_{WF}^*	70.0 kg/s	net power output	P_{net}^*	1.9 MW

The degrees of freedom x of the ORC process considered in the optimization are the pressure levels of evaporation p_{evap} and condensation p_{cond} , the mass flow rate of the ORC working fluid \dot{m}_{WF} and the degree of superheating at the evaporator outlet ΔT_{sh} . The PC-SAFT pure component parameters y considered for the working fluid optimization are the segment number m , segment diameter σ and segment dispersion energy (ε/k) (cf. Section 2.2).

Bounds on the process degrees of freedom are minimal and maximal absolute pressures p^{\min} and p^{\max} of 1 bar and 50 bar, respectively. Additionally, an upper bound of the reduced pressure $p^{r,\max}$, defined as the absolute pressure p divided by the critical pressure p^c , of 0.8 is used because only subcritical ORCs are considered. In order to avoid damage of the turbine, it is important in practice to avoid the formation of liquid droplets during expansion. Thus, a minimal vapor fraction $\beta_{\text{turb,min}}$ of 0.95 at the turbine outlet is used as a further constraint (see Table 1).

Additional constraints arise to ensure that the temperature differences between the heat source and the working fluid in the evaporator and between the working fluid and the cooling water in the condenser do not violate the specified pinch temperature difference ΔT^{pinch} of 5 K at any point. As can be seen in the T-s diagram shown in Figure 3, possible pinch points in the evaporator are at the inlet and outlet and at the point where the working fluid reaches the saturation temperature. Possible pinch points in the condenser are at the condenser inlet and, in case the working fluid is superheated at the inlet, at the point where it first reaches a saturated state.

The net power output P_{net} of the ORC is used as objective function for the optimization and can be calculated from the required power input of the pump P_{P} and the power output of the turbine P_{T} as

$$f(x, y) = P_{\text{net}} = -(P_{\text{T}} + P_{\text{P}}) \quad (10)$$

This definition of P_{net} leads to a positive objective function $f(x, y)$ which is maximized in the optimization.

4.4 Results

To identify an optimal working fluid and the corresponding optimal process conditions for the considered

ORC the CoMT-CAMD approach is used (see Figures 1 and 2). First, an optimal hypothetical fluid and corresponding process conditions are obtained as target in the CoMT step. Subsequently, fluids are designed in the CAMD step which best match the target.

The results of the CoMT step are shown in Table 2. The PC-SAFT pure component parameters shown in the table correspond to a hypothetical optimal fluid. The net power output P_{net} of the target is 1.9 MW. This target value serves as an upper bound as it represents the highest power output achievable for all hypothetical fluids represented by PC-SAFT in the considered convex hull (cf. Section 2.1).

In order to determine a ranking of real fluids which best match the hypothetical optimum, a Computer-Aided Molecular Design (CAMD) algorithm is employed as described in Sections 2.3 and 3.2. Subsequently, the optimal process parameters and corresponding net power output P_{net} for the individual fluids are obtained from process optimizations (cf. Sections 2.4 and 3.2).

The top 10 fluids of the CAMD step are presented in Table 3 together with the respective net power outputs resulting from the process optimizations.

As can be seen in Table 3, the top fluids suggested by the CAMD algorithm are all linear and branched alkanes and alkenes with a maximum of 6 carbon atoms. We identify propene as the best working fluid for the ORC with a net power output P_{net} of 1.45 MW. This value is 24 % less than the target value of the hypothetical molecule.

From Table 3, it can be seen that the net power outputs determined in the process optimizations do not perfectly match the ranking from the CAMD step. The objective function used in the CAMD step is a Taylor-approximation of the real objective function of the integrated optimization problem as explained in Section 2. As the Taylor-approximation does not match the real objective function perfectly, the ranking from the CAMD step slightly deviates from the final ranking based on optimized net power outputs. It is therefore recommended to generate a list of candidates and not only one working fluid. This behavior was also found in earlier implementations of the CoMT-CAMD approach (cf. Lampe *et al.*, 2014; 2015) and is thus not specific for the Modelica-based implementation.

Table 3: Top fluids resulting from the CAMD step and corresponding net power output determined using process optimizations. The order is according to the ranking of the CAMD step. The sorted rank according to the results of the individual process optimizations is shown in parentheses.

Rank (Sorted Rank)	Fluid	Net Power Output
1 (2)	Propane	1.44 MW
2 (6)	Neopentane	1.31 MW
3 (1)	Propene	1.45 MW
4 (9)	Propyne	1.24 MW
5 (3)	Isobutene	1.38 MW
6 (4)	2-Butene	1.38 MW
7 (7)	Butane	1.25 MW
8 (5)	1-Butene	1.33 MW
9 (10)	Neohexane	1.11 MW
10 (8)	1-Butyne	1.25 MW

Clearly, the net power output of the target given in Table 2 serves as an upper bound of the objective function. Therefore, the target is not reached by the designed real working fluids.

In order to find the target in the CoMT step with GenOpt, about $2 \cdot 10^5$ evaluations of the process model are required. On the other hand, an individual process optimization for one working fluid requires about 10^4 evaluations of the process model. Thus, the effort for the integrated molecular and process design with the Modelica-based CoMT-CAMD is similar to the effort for process optimizations for about 30 working fluid candidates. This shows the strength of CoMT-CAMD, where thousands of different molecules are considered in the structure-mapping. Additionally, there are further improvements possible compared to the current implementation. As presented in Section 3.2, parametric runs with GenOpt are used in the CoMT step to find the optimal hypothetical fluid and the corresponding optimal process conditions. Employing gradient-based optimization instead of the parametric runs has the potential to further decrease the computational effort while substantially increasing the accuracy.

5 Conclusion

In this work, we propose the Continuous-Molecular Targeting – Computer-Aided Molecular Design (CoMT-CAMD) approach based on object-oriented process modeling with Modelica. The CoMT-CAMD approach originally proposed by Bardow *et al.* (2010) enables the simultaneous optimization of processes and fluids without any fluid pre-selection by utilizing the physically-based equation of state PC-SAFT. So far, implementations of the CoMT-CAMD design approach were based on process modeling utilizing procedural programming languages. Thus, the reusability of models has been hindered and the investigation of process variants as well as complex processes has been cumbersome. For this reason, it is shown in this work how integrated process

and molecular design with CoMT-CAMD can be based on object-oriented process modeling with Modelica. The proposed implementation uses the Modelica external function interface for external property calculations with PC-SAFT and the optimization tool GenOpt for the search of optimal hypothetical fluids and process conditions. Additionally, a Computer-Aided Molecular Design algorithm implemented in GAMS is used in order to design real fluids which best match the hypothetical optimum. The Modelica-based CoMT-CAMD implementation is demonstrated for the design of a subcritical geothermal Organic Rankine Cycle (ORC). The results show that the approach efficiently identifies optimal ORC processes and working fluids. The CoMT-CAMD approach implemented in an object-oriented language for process modeling allows for convenient and efficient integrated process and fluid design for complex processes. Future work will address the use of deterministic optimization and the utilization of libraries for the Modelica-based CoMT-CAMD.

Acknowledgements

We thank the Deutsche Forschungsgemeinschaft (DFG) for funding this work (BA2884/4-1). Furthermore, the authors thank Heike Schreiber, Matthias Lampe and Christian Schulze for valuable discussions.

References

- Claire S. Adjiman, Amparo Galindo and George Jackson. Molecules Matter: the Expanding Envelope of Process Design. *Computer Aided Chemical Engineering*, 34:55–64, 2014. doi: 10.1016/B978-0-444-63433-7.50007-9.
- Junjiang Bao and Li Zhao. A Review of Working Fluid and Expander Selections for Organic Rankine Cycle. *Renewable and Sustainable Energy Reviews*, 24:325–342, 2013. doi: 10.1016/j.rser.2013.03.040.
- André Bardow, Klaas Steur and Joachim Gross. Continuous-Molecular Targeting for Integrated Solvent and Process Design. *Industrial & Engineering Chemistry Research*, 49(6):2834–2840, 2010. doi: 10.1021/ie901281w.
- Jakob Burger, Vasileios Papaioannou, Smitha Gopinath, George Jackson, Amparo Galindo and Claire S. Adjiman. A Hierarchical Method to Integrated Solvent and Process Design of Physical CO₂ Absorption Using the SAFT- γ Mie Approach. *AIChE Journal*, 61(10):3249–3269, 2015. doi: 10.1002/aic.14838.
- Piero Colonna, Emiliano Casati, Carsten Trapp, Tiemo Mathijssen, Jaakko Larjola, Teemu Turunen-Saaresti and Antti Uusitalo. Organic Rankine Cycle Power Systems. *Journal of Engineering for Gas Turbines and Power*, 137(10):100801, 2015. doi: 10.1115/1.4029884.
- Samira Fazlollahi, Pierre Mandel, Gwenaelle Becker and Francois Maréchal. Methods for Multi-Objective Investment and Operating Optimization of Complex Energy Systems. *Energy*, 45(1):12–22, 2012. doi: 10.1016/j.energy.2012.02.046.

- Peter Fritzson. Modelica – A Language for Equation-Based Physical Modeling and High Performance Simulation. *Applied Parallel Computing*, 1541:149–160, 1998. ISSN: 0302-9743.
- Rafiqul Gani. Chemical Product Design. *Computers & Chemical Engineering*, 28(12):2441–2457, 2004. doi: 10.1016/j.compchemeng.2004.08.010.
- Smitha Gopinath, George Jackson, Amparo Galindo and Claire S. Adjiman. Outer approximation algorithm with physical domain reduction for computer-aided molecular and separation process design. *AIChE Journal*, 62(9):3484–3504, 2016. doi: 10.1002/aic.15411.
- Joachim Gross and Gabriele Sadowski. Perturbed-Chain SAFT. *Industrial & Engineering Chemistry Research*, 40(4):1244–1260, 2001. doi: 10.1021/ie0003887.
- Joachim Gross and Gabriele Sadowski. Application of the Perturbed-Chain SAFT Equation of State to Associating Systems. *Industrial & Engineering Chemistry Research*, 41(22):5510–5515, 2002. doi: 10.1021/ie010954d.
- Joachim Gross. An Equation-of-State Contribution for Polar Components. *AIChE Journal*, 51(9):2556–2568, 2005. doi: 10.1002/aic.10502.
- Joachim Gross and Jadran Vrabec. An Equation-of-State Contribution for Polar Components. *AIChE Journal*, 52(3):1194–1204, 2006. doi: 10.1002/aic.10683.
- Florian Heberle and Dieter Brüggemann. Exergy Based Fluid Selection for a Geothermal Organic Rankine Cycle for Combined Heat and Power Generation. *Applied Thermal Engineering*, 30(11-12):1326–1332, 2010. doi: 10.1016/j.applthermaleng.2010.02.012.
- Matthias Lampe, Marina Stavrou, Hanns M. Bücker, J. Gross and A. Bardow. Simultaneous Optimization of Working Fluid and Process for Organic Rankine Cycles Using PC-SAFT. *Industrial & Engineering Chemistry Research*, 53(21):8821–8830, 2014. doi: 10.1021/ie5006542.
- Matthias Lampe, Marina Stavrou, Johannes Schilling, Elmar Sauer, Joachim Gross and André Bardow. Computer-Aided Molecular Design in the Continuous-Molecular Targeting Framework Using Group-Contribution PC-SAFT. *Computers & Chemical Engineering*, 81:278–287, 2015. doi: 10.1016/j.compchemeng.2015.04.008.
- Patrick Linke, Athanasios Papadopoulos and Panos Seferlis. Systematic Methods for Working Fluid Selection and the Design, Integration and Control of Organic Rankine Cycles—A Review. *Energies*, 8(6):4755–4801, 2015. doi: 10.3390/en8064755.
- Modelica Association. Modelica - A Unified Object-Oriented Language for Systems Modeling - Language Specification Version 3.3. URL=" <https://www.modelica.org/documents/ModelicaSpec33.Pdf>", 2012.
- Athanasios I. Papadopoulos and Patrick Linke. Integrated Solvent and Process Selection for Separation and Reactive Separation Systems. *Chemical Engineering and Processing: Process Intensification*, 48(5):1047–1060, 2009. doi: 10.1016/j.cep.2009.02.004.
- Frances E. Pereira, Emmanuel Keskes, Amparo Galindo, George Jackson and Claire S. Adjiman. Integrated Design of CO₂ Capture Processes from Natural Gas. In: Efstratios Pistikopoulos, Michael Georgiadis and Eustathios S. Kikkinides. Editors. *Process Systems Engineering: Energy Systems Engineering*. Weinheim: Wiley-VCH Verlag GmbH & Co. KG, pp. 231–248, 2008.
- Frances E. Pereira, Emmanuel Keskes, Amparo Galindo, George Jackson and Claire S. Adjiman. Integrated Solvent and Process Design Using a SAFT-VR Thermodynamic Description. *Computers & Chemical Engineering*, 35(3):474–491, 2011. doi: 10.1016/j.compchemeng.2010.06.016.
- Richard E. Rosenthal. GAMS – a User's Guide, GAMS Release 24.6.1. URL: <http://www.gams.com/help/topic/gams.doc/userguides/GAMSUsersGuide.pdf>. 2016.
- Dennis Roskosch and Burak Atakan. Reverse Engineering of Fluid Selection for Thermodynamic Cycles with Cubic Equations of State, Using a Compression Heat Pump as Example. *Energy*, 81:202–212, 2015. doi: 10.1016/j.energy.2014.12.025.
- Nikolaos V. Sahinidis, Mohit Tawarmalani and Minrui Yu. Design of Alternative Refrigerants via Global Optimization. *AIChE Journal*, 49(7):1761–1775, 2003. doi: 10.1002/aic.690490714.
- Elmar Sauer, Marina Stavrou and Joachim Gross. Comparison between a Homo- and a Heterosegmented Group Contribution Approach Based on the Perturbed-Chain Polar Statistical Associating Fluid Theory Equation of State. *Industrial & Engineering Chemistry Research*, 53(38):14854–14864, 2014. doi: 10.1021/ie502203w.
- Marina Stavrou, Matthias Lampe, André Bardow and Joachim Gross. Continuous Molecular Targeting—Computer-Aided Molecular Design (CoMT–CAMD) for Simultaneous Process and Solvent Design for CO₂ Capture. *Industrial & Engineering Chemistry Research*, 53(46):18029–18041, 2014. doi: 10.1021/ie502924h.
- Heiko Struebing. Identifying Optimal Solvents for Reactions Using Quantum Mechanics and Computer-Aided Molecular Design. *Ph.D. thesis, Imperial College London, London*, 2011.
- Heiko Struebing, Amparo Galindo and Claire S. Adjiman. Optimal Solvent Design for Reactions Using Computer-Aided Molecular Design. URL: <http://www.minlp.org/library/problem/mod/index.php?lib=MINLP&i=180&pi=-137>. 2011.
- Michael Wetter. Design Optimization with GenOpt. *Building Energy Simulation*, (21):19–28, 2000.
- Michael Wetter. Modelica-Based Modelling and Simulation to Support Research and Development in Building Energy and Control Systems. *Journal of Building Performance Simulation*, 2(2):143–161, 2009. doi: 10.1080/19401490902818259.
- Michael Wetter. GenOpt – Generic Optimization Program – User Manual – Version 3.1.1. URL: <https://simulationresearch.lbl.gov/GO/download/manual-3-1-1.pdf>, 2016.

Dynamic Simulations of the Post-combustion CO₂ Capture System of a Combined Cycle Power Plant

Rubén M. Montañés Lars O. Nord

Department of Energy and Process Engineering, NTNU – Norwegian University of Science and Technology,
Trondheim, Norway

Abstract

Dynamic process models of the capture unit of a 600 MW combined cycle power plant with post-combustion CO₂ capture were developed in the Modelica language. The process models were utilized to understand the transient response of the capture unit when the plant was initially operated at steady-state conditions at different power plant's loads. Simulations to characterize the open-loop response of main process variables of the process to step-change disturbances in flue gas mass flow rate, solvent circulation mass flow rate and reboiler duty were performed. It was found that the plant was slower when operated at lower loads, i.e., it required longer total stabilization times for the most relevant variables of the process. Simulations revealed that the PCC unit responded significantly faster to an increase in exhaust gas mass flow rate than to a reduction in exhaust gas mass flow rate.

Keywords: transient, carbon capture, gas liquid contactors, operational flexibility, chemical absorption.

1 Introduction

CO₂ capture and storage (CCS) comprises a group of technologies that can significantly reduce the CO₂ emissions from thermal power plants and other industrial sources (IEA, 2016). Post-combustion CO₂ capture based on the chemical absorption-desorption process using amines is a technology that has been technically proven at commercial scale from coal fired power plants in projects such as Boundary Dam in Saskatchewan, Canada, and the Petra Nova project in Texas, USA.

The introduction of large shares of variable renewable energy sources such as wind and solar in power systems is changing the operating patterns of thermal power generation units, including coal power plants and natural gas combined cycle plants (IEA, 2011). Power plants traditionally operated as base load units are operated as load-following units (Montañés, et al., 2016). Therefore, during the last years, interest has grown in the field of operating flexibility of thermal power plants with CO₂ capture technologies (IEA-GHG, 2012).

The low amount of existing commercial-scale post-combustion capture plants (PCC) and the scarcity of published transient performance data of such systems claims for an interest for the development of dynamic process models (Bui, et al., 2014). These models allow studying plant dynamic performance, analyzing various plant transient events as well as developing and implementing optimal control strategies for PCC plants integrated with thermal power plants. Dynamic process simulation provides process insight and contributes to the development of the learning curve for flexible operation of future thermal power plants with CO₂ capture.

The purpose of the study is to provide understanding of the open-loop transient performance of the main process variables of the PCC unit at different load operation points of the power plant. A thermal power plant operated as load-following unit will be operated at part-load conditions during a significant amount of hours during its lifetime (Montañés, et al., 2016). Therefore it is of importance to find out differences in the transient behavior of the process at part-load operating conditions with respect to those of nameplate capacity. In this work, a dynamic process model of the PCC unit of a 600 MW combined cycle power plant with post-combustion CO₂ capture using aqueous monoethanolamine (MEA) as chemical solvent is utilized for providing understanding of the open-loop response of key performance variables to different disturbances applied to the PCC plant. The process insight and understanding developed in this work will be valuable to develop control strategies of the process when integrated with the thermal power plant.

2 Post-combustion CO₂ capture with chemical absorption

2.1 Chemical absorption process

The process of CO₂ capture by chemical absorption is a two-steps regenerative process; one involves the absorption of CO₂ into a solvent, while the other involves the desorption or stripping of CO₂ from the solvent and the regeneration of the solvent. The process conditions change in the absorption and desorption

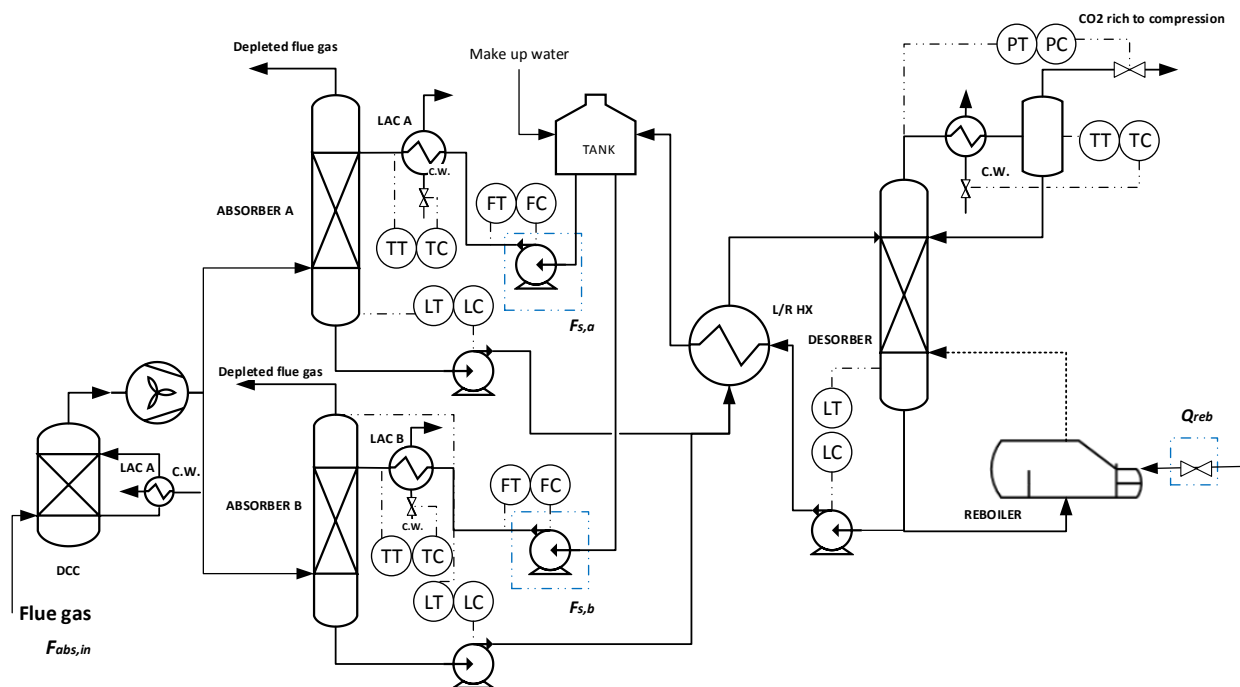


Figure 1. Process configuration of the post combustion CO₂ capture unit (PCC) of the natural gas combined cycle power plant studied in this work. Includes temperature (T), level (L), flow (F) and pressure (P); transmitters (T) and controllers (C).

process, being main changes temperature and pressure, and also solvent concentrations and pH. For absorption, low temperature and high partial pressure of CO₂ is desired, while for desorption, high temperature and low partial pressure of CO₂ is desired.

When the process is utilized for flue gas treatment from a power plant, the exhaust gases are normally cooled down by means of a direct contact cooler (DCC), that reduces the flue gas temperature and the water content. A fan allows overcome the gas pressure drop in the absorber, which is operated slightly above atmospheric conditions, and at around 40 °C, refer to Figure 1. In the absorber column, the exhaust gas flowing upwards meets the chemical solvent flowing downwards. Packing material allows having a thin film of liquid with high surface contact area for heat and mass transfer between the gas and liquid phases, and the exothermal chemical absorption process. Depleted flue gas leaves the absorber at the top through a stack, normally after flowing through a water wash section that allows keeping the water mass balance of the process and reduces chemical solvent emissions due to solvent droplets or solvent vapor carry over. The rich solvent, i.e., solvent with a lot of bounded CO₂, accumulates in the absorber sump and is then pumped towards the top of the stripper. An intermediate heat exchanger allows for heat integration between the absorber and stripper columns. The rich solvent is heated up by the lean solvent coming from the stripper bottom towards around 110 °C and then enters the stripper at the top of the column. This heat integration allows reducing reboiler and cooling duties. A mixing tank allows for

accumulation of the solvent at different operating conditions of the plant.

The desorption process normally occurs at around 100 to 130 °C. Steam supplied from the power plant provides the reboiler duty required to regenerate the solvent (endothermal desorption process), and to generate the stripping vapors flowing upwards in the stripper column, consisting mainly of H₂O and CO₂. The regenerated lean solvent is sent to the absorber inlet via the heat integration exchanger and a lean amine cooler that controls the temperature of the solvent at the inlet of the absorber to around 40 °C. At the top of the stripper there is a condenser and a cooler where the solvent and steam condenses. The condensate is conducted back to the column via a reflux. The product CO₂ rich flow the top of the stripper is conducted to the compression section where it will be conditioned for transport and storage purposes.

2.2 Process configuration

The PCC unit was designed to treat flue gas from a 611 MW combined cycle power plant. The gas turbine (GT) of the power plant was the heavy duty Mitsubishi JAC 701, and the steam cycle consisted of a three-pressure reheat (3PRH) configuration. The design of the PCC unit included the process integration with the power plant through the flue gas line from the HRSG outlet and a steam extraction from the steam turbine's IP/LP crossover. The steam extraction was utilized to feed the reboiler duty required to produce the stripping vapors needed for chemical desorption in the stripper column. The design point chosen for the post-

combustion unit was 100% GT load under ISO conditions, which, for the gas turbine, corresponded to flue gas with a mass flow rate of 887.1 kg/s with 4.33 vol % CO₂ (wet). The chemical solvent utilized was 30%wt aqueous MEA and the target capture rate was 90%. Further details on design aspects of PCC units for combined cycle power plants can be found in (Dutta, et al., 2017).

The resulting process configuration of the PCC unit consisted of a two absorbers and one stripper layout, as shown in Figure 1. Each absorber column had dimensions of 16.3 m in diameter and 23.2 m height, while the desorber had a 9.7 m diameter with 10 m height. The process equipment included absorber columns, desorber column and reboiler, overhead condenser, internal lean/rich heat exchanger, mixing tank for water and MEA makeups, direct contact coolers and circulation pumps. A fan was included in the process to overcome the pressure drop imposed by the absorber column.

3 Dynamic process model development and validation

The Modelica library Gas Liquid Contactors (GLC) (Modelon AB, 2016), from Modelon AB, was utilized as a basis to develop the dynamic process model of the PCC unit. The library contains dynamic process models of the main equipment for systems' level modeling of the absorber-desorber process with monoethanolamine (MEA) as chemical solvent. That equipment includes absorber and desorber columns, sumps, reboiler, condensers, water wash sections, pumps, valves, mixing tank, and property media packages.

The chemical absorption-desorption process within packed segments was modelled considering the two-film theory approach for heat and mass transfer. Chemical equilibrium for reactions was assumed, and mass transfer was modeled considering rate-based models with enhancement factor (Kvamsdal, et al., 2009). Detailed description of the dynamic process models included in the GLC library has been presented previously in literature (Pröhl, et al., 2011).

The dynamic process models included in the GLC library have been previously validated with large-scale experimental data by (Montañés, et al., 2017). The validation consisted of modeling the whole absorber-desorber system of the demonstration scale chemical absorption plant at CO₂ Technology Centre Mongstad (TCM DA), in Norway. The amine plant at TCM DA was configured to treat exhaust gases coming directly from the exhaust of a natural gas fueled combined heat and power (CHP) plant placed at Mongstad's refinery. The exhaust gas from two GE 9001E gas turbines contains about 3.5 % vol CO₂, and around 3% of the total exhaust gas mass flow rate is conducted to the amine plant for CO₂ absorption. The PCC plant at TCM can

treat up to 60 000 Sm³/hr of exhaust gas and can capture around 80 ton CO₂/day at nameplate capacity when configured to treat CHP gas. The experimental data utilized for validation includes steady-state data for a wide range of operating conditions and multiple transient events. The plant was operated with 30 wt % aqueous MEA. The conclusion of the work in (Montañés, et al., 2017) is that the process models can capture, with sufficient accuracy, the steady-state and transient phenomena of the process at the demonstration plant scale. In addition, it gives confidence towards using the models for simulation and analysis of the transient performance of the scaled-up process to commercial scale of 4770 ton/day CO₂ captured.

Rules for consistent inventory control (Aske & Skogestad, 2009) were applied to design the regulatory control layer of the PCC unit in Figure 1. It included level controllers for absorbers and stripper sumps, overhead condenser pressure control, lean solvent temperature at absorbers inlet, and exhaust gas temperature at absorber inlet. The controllers were tuned by means of the SIMC tuning rules.

The supervisory control layer for this process has three degrees of freedom, consisting of the two solvent mass flow rates at absorber inlet $\dot{F}_{s,a}$ and $\dot{F}_{s,b}$, and the reboiler duty \dot{Q}_{reb} .

4 Process simulations description

Generally, a combined cycle power plant is brought to part-load operating conditions by reducing the GT load and consequently the steam turbine's power output will be reduced. A GT load reduction results in reduced GT exhaust gas mass flow rate sent to the HRSG of the combined cycle and to the absorbers of the PCC unit. The open-loop transient performance of the plant is studied for three steady-state operating conditions of the power plant, corresponding to 100%, 80% and 60% GT load.

4.1 Steady-state operating conditions at 100%, 80% and 60% GT load

In order to obtain the steady-state operating conditions of the PCC unit at the three operating points, simulations were run with different flue gas mass flow rates as input boundary conditions to the dynamic process model, corresponding to different GT loads, refer to Table 1. The exhaust gas temperature and composition of the absorber was considered constant as boundary condition (input). Note that the exhaust temperature at the inlet of the absorber is normally controlled by the DCC, and that it was observed that exhaust gas composition did not change considerably for the purpose of this study, considering the part load range analyzed of 100% to 60% GT load, and for the specific GT utilized in this work. In addition, a decentralized control structure for the supervisory control layer was included. Several

studies, including the one based on self-optimizing control theory by (Panahi, 2011), suggest that keeping the capture ratio Cap and a temperature in the stripper column constant can lead to efficient operation of the process for varying loads of the absorption-desorption process. Therefore, the available degrees of freedom for operation were utilized to control these process variables. Solvent mass flow rates $\dot{F}_{s,a}$ and $\dot{F}_{s,b}$ were utilized to control the respective CO₂ capture rates Cap_a and Cap_b at the top of the absorbers to the design value of 0.9, while reboiler duty was used as manipulated variable to control reboiler temperature T_{reb} to the value 119 °C. CO₂ capture rates are calculated for each absorber column at the top, by using Equation (1), where $\dot{F}_{abs,in}$ is the exhaust flue gas at the inlet of the absorber column, $X_{abs,in}$ is the CO₂ mass fraction in the exhaust gas at the absorber inlet, $\dot{F}_{abs,out}$ is the depleted flue gas mass flow rate at the absorber stack and $X_{abs,out}$ is the CO₂ mass fraction in the flue gas at the absorber stack. The resulting operating conditions of the PCC at different GT loads are shown in Table 1 and Table 2.

Table 1. Values of PCC unit input variables at different power plant's load operating conditions. Note that both absorber columns were operated in parallel, so $\dot{F}_{s,a}$ was equal to $\dot{F}_{s,b}$.

GT load [%]	$\dot{F}_{abs,in}$ [kg/s]	$\dot{F}_{s,a}$ [kg/s]	\dot{Q}_{reb} [MW]
100	887.1	613.3	205.9
80	765.1	535.2	176.2
60	653.5	464.1	149.6

$$Cap_a = \frac{\dot{F}_{abs,in} \cdot X_{abs,in} - \dot{F}_{abs,out} \cdot X_{abs,out}}{\dot{F}_{abs,in} \cdot X_{abs,in}} \quad (1)$$

Table 2. Values of most relevant process variables of the PCC unit at different operating conditions of the power plant. Note that both absorber columns were operated in parallel, so Cap_a was equal to Cap_b (in the table shown as Cap). It also resulted in same value of solvent loading at absorbers inlets ($L_{i,abs}$).

GT load [%]	$L_{i,abs}$	$L_{i,str}$	Cap	$Prod$ [kg/s]
100	0.280	0.501	0.9	55.2
80	0.280	0.497	0.9	47.6
60	0.279	0.493	0.9	40.7

4.2 Open-loop step response simulations

The simulations consisted of step-changes of $\pm 10\%$ of main PCC inputs, or disturbances, when the plant was at steady-state operating conditions at the three GT operating points. Step-changes were applied to each process input at a time, keeping the remaining process inputs constant. The output in main process variables was recorded and dead times and 10% settling times were calculated.

- Dead time θ describes how long it takes before a process variable begins to respond to a change in the process input. With begins to respond it is meant that the trajectory of the process variable moves out of the band defined by the initial steady-state value of the process variable y_0 , and a $\pm 1\%$ change in the process variable Δy , i.e.: $-0.01 \Delta y + y_0 < y_0 < 0.01 \Delta y + y_0$, for the first time.
- The 10% settling time t_s is the time it takes from the instant in which the process variable begins to respond to the input change, until it remains within an error band described by the final steady-state value of the process variable y_∞ , and 10% of the change in the process variable Δy , i.e.: $-0.1 \Delta y + y_\infty < y_\infty < 0.1 \Delta y + y_\infty$.
- The resulting total stabilization time t_{sta} is the sum of the dead time and the settling time. In addition, the relative change RC in the process variable is calculated as in Equation (2), where y_0 is the initial steady-state value of the process variable.

$$RC (\%) = 100 \cdot \frac{y_\infty - y_0}{y_0} \quad (2)$$

The main inputs/disturbances applied to the process in this analysis were:

- Flue gas mass flow rate $\dot{F}_{abs,in}$. Note that the flow was split and the absorber columns were operated in parallel. This means that each absorber column treated an exhaust gas mass flow rate of $\dot{F}_{abs,in}/2$.
- Solvent mass flow rates at absorbers inlets $\dot{F}_{s,a}$ and $\dot{F}_{s,b}$.
- Reboiler duty \dot{Q}_{reb} .

The responses of the main process variables of interest in this analysis were:

- Solvent lean CO₂ loading at absorbers inlet $L_{i,abs}$.
- Solvent rich CO₂ loading at stripper inlet $L_{i,str}$.
- CO₂ capture rate at absorbers stacks Cap_a and Cap_b .
- CO₂ product mass flow rate $Prod$, at the outlet of the overhead condenser of the desorber. This is the CO₂ rich product flow of the PCC unit sent to conditioning, compression and transport.
- Temperature at stripper column bottom T_{reb} .

The difference in solvent loading at inlet and outlet of the absorber determines the capability of the solvent to carry CO₂. This in turn depends on the absorber size, operating conditions, regeneration of the solvent and CO₂ partial pressure. Solvent CO₂ loading L is defined as the ratio between moles of CO₂ and moles of solvent (mol/mol) in Equation (3).

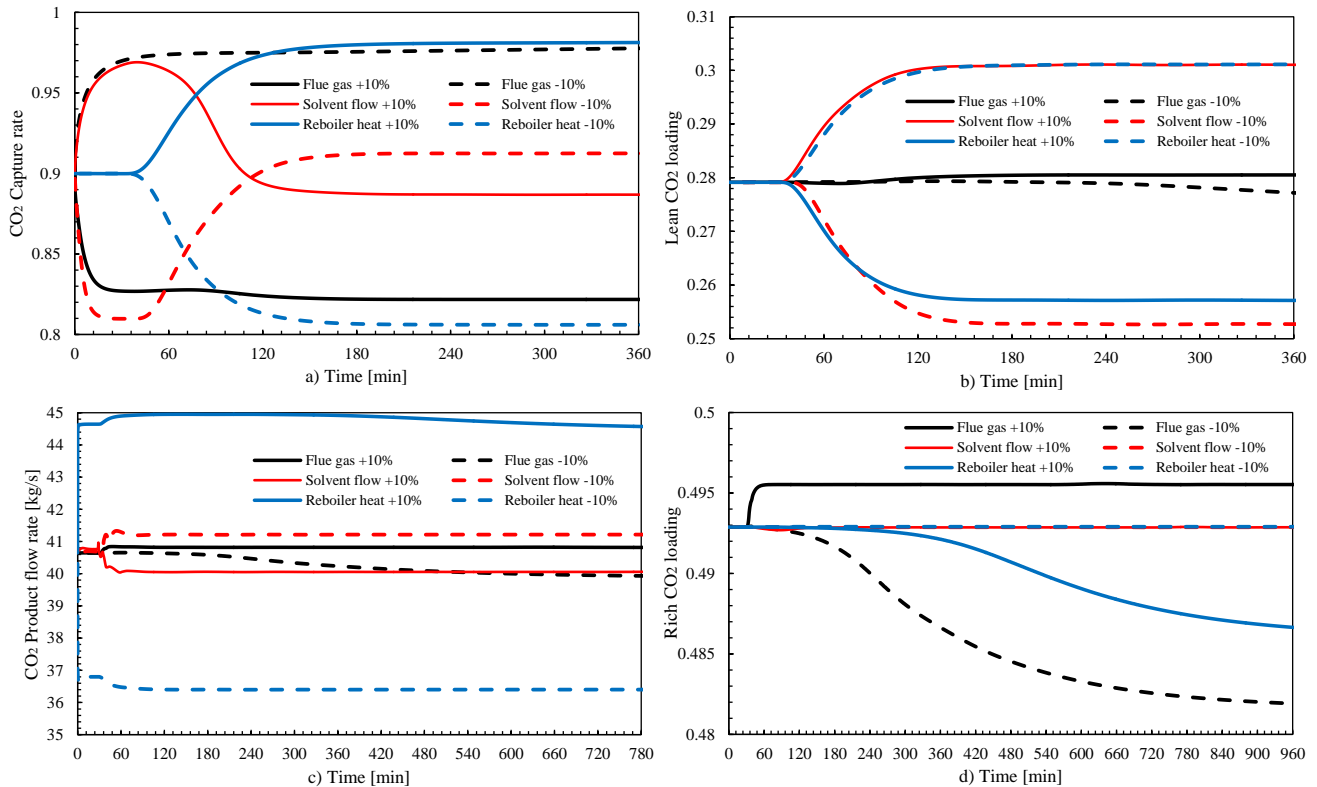


Figure 2. Transient responses of the relevant process variables to different step-changes in process inputs. These simulations correspond to the initial steady-state operation of the PCC unit for 60% GT load. Step-changes were applied at $t = 0$ min.

$$L = \frac{\text{mols of } CO_2}{\text{mols of solvent}} \quad (3)$$

5 Results and discussion

5.1 Response to step changes in flue gas mass flow rate $\dot{F}_{abs,in}$

The resulting response times of the PCC unit's main process variables to step-changes in flue gas mass flow rate are shown in Table 3 and Table 4. Figure 2 shows the transient response of the main process variables for the different step changes studied in this work. In addition, Figure 3 shows trends of total stabilization times t_{sta} for the main variables of the process when operating the plant at different loads.

It can be observed that CO_2 capture rate Cap stabilized relatively fast, within 1 h, after a disturbance in flue gas mass flow rate. The CO_2 capture rate decreased for increased flue gas mass flow rate (+10%). A faster response in Cap was observed when the flue gas flow rate was increased (+10%) than when it was decreased (-10%), showing the non-linear performance of the PCC system. This behavior was consistent at the different operating points of the PCC plant. The dead time of this response was negligible, since the flue gas mass flow rate was included in the calculation and naturally changes when a step change is applied.

The CO_2 product flow rate $Prod$ required larger stabilization times than Cap . This shows the differences in performance of the absorbers and desorber columns during transient conditions when a disturbance is applied to the PCC unit. The dead times observed in the CO_2 product mass flow rate can be explained by the residence time imposed by the solvent hold-ups in the cold side of the internal heat exchanger's piping and rich solvent piping. These residence times resulted in dead times in convectively transported variables of the liquid solvent from absorber outlet to stripper inlet, including rich solvent loading at the stripper inlet $L_{i,str}$. Note that the dead times of $L_{i,str}$ and $Prod$ responses are similar in Table 3 and Table 4. Stabilization of the $Prod$ was significantly faster when increasing flue gas mass flow rate (around 1 h) than when flue gas mass flow rate was decreased (9 to 11 h). It can also be observed that the $Prod$ response was slower at lower power plant loads, refer to Figure 3.

For flue gas flow rate increase (+10%), the relative change in solvent loadings was small. This is because the solvent capacity was close to the limit under these operating conditions. In general, it was found that lean solvent loading at the inlet of the absorber $L_{i,abs}$ required larger stabilization times t_{sta} than rich loading at stripper inlet $L_{i,str}$. This can be explained by the buffering effect introduced by the mixing tank placed in the recycle loop (from stripper sump to absorber liquid inlet). In addition, larger dead times to this specific disturbance were found for $L_{i,abs}$ than for $L_{i,str}$, due to the additional

Table 3. Response to +10% step increase in flue gas mass flow rate $\dot{F}_{abs,in}$ at various GT loads. Dead times Θ , settling times t_s and total stabilization times t_{sta} are shown.

GT load [%]		10 %			
		Θ [min]	t_s [min]	t_{sta} [min]	RC [%]
100	$L_{i,abs}$	74	53	127	0.45
	$L_{i,str}$	26	11	36	0.45
	Cap	0	12	12	-8.79
	$Prod$	26	5	31	0.29
	T_{reb}	0	127	127	-0.05
80	$L_{i,abs}$	36	105	141	0.47
	$L_{i,str}$	27	12	39	0.50
	Cap	0	15	15	-8.74
	$Prod$	28	45	72	0.35
	T_{reb}	0	54	54	-0.05
60	$L_{i,abs}$	68	88	156	0.48
	$L_{i,str}$	34	13	46	0.54
	Cap	0	17	17	-8.70
	$Prod$	34	28	62	0.42
	T_{reb}	0	60	60	-0.06

Table 4. Response to -10% step decrease in flue gas mass flow rate $\dot{F}_{abs,in}$ at various GT loads. Dead times Θ , settling times t_s and total stabilization times t_{sta} are shown.

GT load [%]		-10 %			
		Θ [min]	t_s [min]	t_{sta} [min]	RC [%]
100	$L_{i,abs}$	47	578	626	-2.50
	$L_{i,str}$	26	538	564	-2.62
	Cap	0	55	55	8.55
	$Prod$	27	556	583	-0.98
	T_{reb}	3	572	575	0.30
80	$L_{i,abs}$	50	639	689	-2.31
	$L_{i,str}$	29	592	621	-2.53
	Cap	0	58	58	8.84
	$Prod$	30	603	633	-1.90
	T_{reb}	0	625	625	0.27
60	$L_{i,abs}$	129	619	748	-1.99
	$L_{i,str}$	131	529	661	-2.26
	Cap	0	39	39	8.88
	$Prod$	163	503	666	-1.85
	T_{reb}	5	667	672	-2.83

residence time introduced by liquid hold-ups in desorber packed segments and sump, lean amine piping and hot side piping of the integral heat exchanger, mixing tank and lean amine cooler. Again, the plant response in

solvent CO₂ loadings was faster when flue gas mass flow rate was increased for all power plant loads studied, refer to Figure 3. It must be mentioned that the relative change in process variables to step-changes is more significant the step-down than step-up of the flue gas flow rate. This can be explained by the fact that the solvent rich loading at the steady-state operating conditions is close to the solvent limit CO₂ loading capacity, which is limited by stoichiometry.

5.2 Response to step-changes in solvent mass flow rate $\dot{F}_{s,a}$ and $\dot{F}_{s,b}$

The resulting response times of the PCC unit's main process variables to step-changes in solvent circulation mass flow rates are shown in Table 5 and Table 6.

Table 5. Response of the main process variables to 10% step increase in solvent circulation mass flow rate $\dot{F}_{s,a}$ and $\dot{F}_{s,b}$ at the inlet of the absorbers, for different GT loads. Dead times Θ , settling times t_s and total stabilization times t_{sta} are shown.

GT load [%]		10 %			
		Θ [min]	t_s [min]	t_{sta} [min]	RC [%]
100	$L_{i,abs}$	27	50	77	8.04
	$L_{i,str}$	25	118	143	-0.02
	Cap	0	131	131	-1.46
	$Prod$	21	40	60	-1.48
	T_{reb}	0	25	25	-1.04
80	$L_{i,abs}$	31	113	144	8.19
	$L_{i,str}$	37	112	149	-0.01
	Cap	0	137	137	-1.77
	$Prod$	25	21	46	-1.78
	T_{reb}	0	35	35	-1.05
60	$L_{i,abs}$	35	67	102	7.85
	$L_{i,str}$	29	813	842	0.00
	Cap	0	161	161	-1.47
	$Prod$	31	22	52	-1.46
	T_{reb}	0	39	39	-0.99

Solvent circulation mass flow rate step changes resulted in inverse responses in CO₂ capture rates, refer to Figure 2. This can be explained by the coupled operation of the absorbers and desorber columns via the recycle loop. When increasing the solvent circulation flow rate (10%), the Cap increases during the first part of the transient. However, since the reboiler duty is kept constant, the lean loading at the inlet of the absorber $L_{i,abs}$ will increase (more solvent being circulated for the same regeneration energy introduced in the process \dot{Q}_{reb}), resulting in a reduction of Cap , with a delay imposed by solvent hold-ups (residence time) through piping and mixing components in the recycle loop.

Observe the large dead time in $L_{i,abs}$ in Figure 2. An analog explanation could be used for the inverse response observed when solvent circulation mass flow rate was reduced. Larger stabilization times were required when the plant was operated at lower loads, see Figure 3.

For these disturbances, CO₂ product mass flow rate $Prod$ stabilizes relatively faster (around 1 h) than CO₂ capture rate Cap (2–3 h). Similar stabilization times t_{sta} were noted when increasing (10%) and when decreasing (-10%) the solvent circulation mass flow rates \dot{F}_s .

The relative change in stripper inlet rich solvent loading $L_{i,str}$ was very small, so it can be considered constant when changing the solvent circulation rate by 10%. It shows that the solvent's capacity was working at the limit. However, lean loading $L_{i,abs}$ relative change was large. A large dead time was observed in $L_{i,abs}$ (27–47 minutes), due to the large amount of solvent inventory within the plant (residence time), and in the recycle loop. In addition, a settling time of 1 to 2 hours was observed, this is likely due to the buffering effect introduced by the absorber tank and other mixing components, such as, desorber and absorber sumps.

Table 6. Response of the main process variables to -10% step decrease in solvent circulation mass flow rate $\dot{F}_{s,a}$ and $\dot{F}_{s,b}$ at the inlet of the absorbers, for different GT loads. Dead times Θ , settling times t_s and total stabilization times t_{sta} are shown.

		-10 %			
GT load [%]		Θ [min]	t_s [min]	t_{sta} [min]	RC [%]
100	$L_{i,abs}$	35	53	88	-10.26
	$L_{i,str}$	29	150	180	0.00
	Cap	0	118	118	2.05
	$Prod$	26	29	55	2.03
	T_{reb}	0	8	8	1.09
80	$L_{i,abs}$	35	74	108	-10.60
	$L_{i,str}$				0.00
	Cap	0	125	125	2.11
	$Prod$	30	33	64	2.85
	T_{reb}	0	37	37	1.13
60	$L_{i,abs}$	43	71	115	-9.47
	$L_{i,str}$	37	788	825	0.00
	Cap	0	166	166	1.38
	$Prod$	35	28	63	1.40
	T_{reb}	0	9	9	0.99

5.3 Response to step-changes in reboiler duty \dot{Q}_{reb}

Simulations in which reboiler duty \dot{Q}_{reb} was changed with step-changes by $\pm 10\%$ were performed. Flue gas conditions and solvent circulation flow rates were kept

constant at each operating point of the plant. The resulting response times of the PCC unit's main process variables are shown in Table 7 and Table 8.

Table 7. Response of the main process variables to 10% step increase in reboiler duty \dot{Q}_{reb} , for different GT loads. Dead times Θ , settling times t_s and total stabilization times t_{sta} are shown.

		10 %			
GT load [%]		Θ [min]	t_s [min]	t_{sta} [min]	RC [%]
100	$L_{i,abs}$	28	384	412	-10.00
	$L_{i,str}$	172	526	697	-1.80
	Cap	31	69	100	9.54
	$Prod$	0	322	322	9.70
	T_{prod}	0	335	335	1.09
80	$L_{i,abs}$	33	419	451	-9.42
	$L_{i,str}$	247	531	778	-1.52
	Cap	35	67	102	9.06
	$Prod$	0	332	332	9.67
	T_{reb}	0	353	353	1.02
60	$L_{i,abs}$	37	457	494	-8.91
	$L_{i,str}$	335	539	874	-1.34
	Cap	40	87	126	9.24
	$Prod$	0	606	606	9.44
	T_{reb}	0	368	368	0.96

Table 8. Response of the main process variables to -10% step decrease in reboiler duty \dot{Q}_{reb} , for different GT loads. Dead times Θ , settling times t_s and total stabilization times t_{sta} are shown.

		-10 %			
GT load [%]		Θ [min]	t_s [min]	t_{sta} [min]	RC [%]
100	$L_{i,abs}$	28	56	85	8.448
	$L_{i,str}$	44	694	739	0.008
	Cap	29	52	81	-10.81
	$Prod$	0	24	24	-10.84
	T_{reb}	0	11	11	-1.10
80	$L_{i,abs}$	29	66	96	8.1519
	$L_{i,str}$	92	685	777	0.0058
	Cap	34	65	99	-10.62
	$Prod$	0	27	27	-10.63
	T_{reb}	0	13	13	-1.05
60	$L_{i,abs}$	37	72	109	7.88
	$L_{i,str}$	93	403	496	-0.006
	Cap	39	75	113	-10.44
	$Prod$	0	33	33	-10.44
	T_{reb}	0	12	12	-1.01

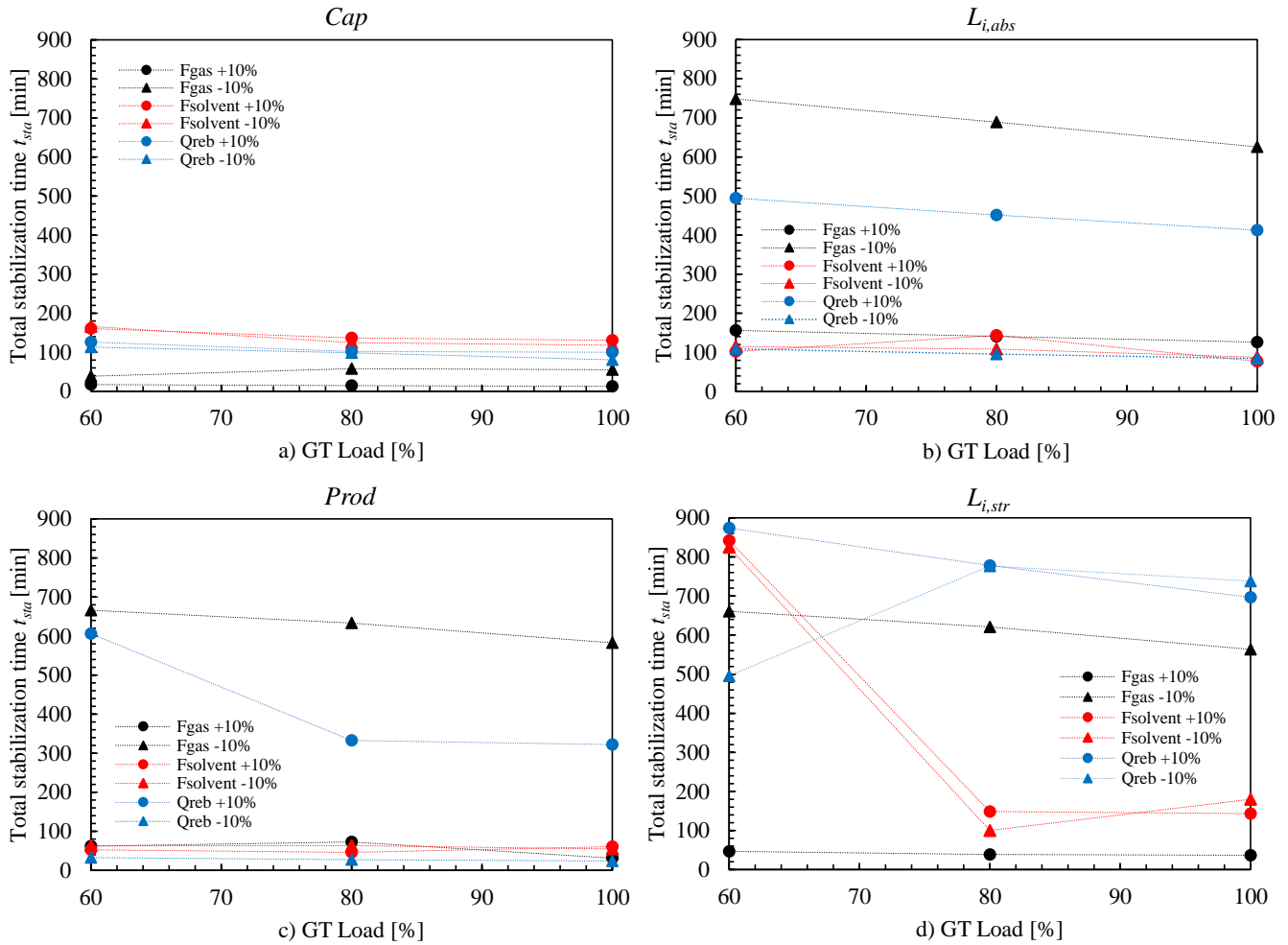


Figure 3. Trends in total stabilization times of main process variables of the PCC unit, when disturbed by the different plant input step changes, at different GT loads. a) CO₂ capture rate Cap ; b) Solvent CO₂ loadings at absorbers inlets $L_{i,abs}$; c) Product CO₂ mass flow rate $Prod$; and d) solvent CO₂ loading at stripper inlet $L_{i,str}$.

Increasing the reboiler duty will result in increased CO₂ capture rate Cap due to the lower resulting lean loading at the inlet of the absorber $L_{i,abs}$. Reducing reboiler duty will result in reduced Cap due to the increase in $L_{i,abs}$. A relatively large dead time in the Cap response of 28–37 min was found. This dead time was larger when the plant was operated at lower power plant loads. This is because at lower power plant loads solvent circulation rates are smaller (refer to Table 1), resulting in larger residence time though piping and mixing tank in the recycle loop.

The relative change in CO₂ product mass flow rate $Prod$ was also large, but with practically no dead time. This is because the reboiler duty introduced in the reboiler is physically closer to the overhead of the stripper. However, the recycle loop and coupled operation of the absorber and desorber makes the total stabilization time t_{sta} of the $Prod$ longer than for Cap . Observe the slow response in $L_{i,str}$ in Figure 3. In general, longer total stabilization times were found for

both Cap and $Prod$ when the plant was operated at lower loads, refer to Figure 3.

The relative change was also significant for lean loading at absorber inlet $L_{i,abs}$ with a large dead time, as previously mentioned. The dead times were even larger for rich loading at the inlet of the stripper $L_{i,str}$, and longer total stabilization times than for $L_{i,abs}$ were observed.

6 Conclusions

The open-loop transient performance of the main process variables of the plant were studied when the plant was operated at different power plant's load conditions, and for different disturbances to the PCC unit. In general, it is found that the plant was slower when the plant was operated at lower loads, i.e., it required longer total stabilization times for the main variables of the process. In general, CO₂ capture rate stabilized relatively faster (1–3 h) than other process variables (1–11 h).

In addition, it was found that the PCC unit responded significantly faster to the increase in flue gas mass flow rate than to reductions in flue gas mass flow rate. This could have significant implications on efficient operation of the PCC unit when ramping down the power plant's load, due to long stabilization times required of the process and the resulting inefficient operation during transient conditions, if a suitable control structure cannot be implemented.

Process variables respond differently to different disturbances. For the same process disturbance and process variable, the response was different when increasing or decreasing the input. This shows the non-linear behavior of the process. The recycle loop in the process from desorber outlet to absorber inlet connects the operation of the absorbers units and the stripper, and the resulting dynamic interaction between the absorption and desorption unit resulted in long stabilization time of main process variables, up to 11 h.

Current and future work includes the integration of the PCC unit with a dynamic process model of the power plant. That will allow the study of dynamic interactions between the power plant and the PCC unit under transient events of the power plant, and to analyze optimal control structures and operation of the integrated process for efficient flexible operation.

References

- Aske, E. M. B. & Skogestad, S., 2009. Consistent inventory control. *Industrial engineering chemistry research*, Volume 48, pp. 10892-10902.
doi: <http://pubs.acs.org/doi/abs/10.1021/ie801603j>
- Bui, M., Gunawan, I., Verheyen, V., Feron, P., Meuleman, E., Adeloju, S., 2014. Dynamic modeling and optimisation of flexible operation in post-combustion CO₂ capture plants - A review. *Computers and Chemical Engineering*, Volume 61, pp. 245 - 265.
doi: <http://dx.doi.org/10.1016/j.compchemeng.2013.11.015>
- Dutta, R., Nord, L. O. & Bolland, O., 2017. Selection and design of post-combustion CO₂ capture process for 600 MW natural gas fueled thermal power plant based on operability. *Energy*, Volume 121, pp. 643-656.
doi: <http://dx.doi.org/10.1016/j.energy.2017.01.053>
- Flø, N. E., 2015. *Doctoral Thesis: Post-combustion absorption-based CO₂ capture: modeling, validation and analysis of process dynamics*. Trondheim (Norway): Doctoral Theses at NTNU, 2015:244.
doi: <http://hdl.handle.net/11250/301562>
- IEA, 2011. *Harnessing Renewable Energies: A guide to the balancing challenge*, 9, rue de la Fédération, 75739 Paris Cedex 15, France: International Energy Agency.
- IEA, 2016. *20 years of carbon capture and storage - Accelerating future deployment*, Paris, France: IEA.
- IEA-GHG, 2012. *Operating Flexibility of Power Plants with CCS*.
- Kvamsdal, H. M., Jakobsen, J. P. & Hoff, K., 2009. Dynamic modeling and simulation of a CO₂ absorber column for post-combustion CO₂ capture. *Chemical Engineering Process*, Volume 48, pp. 135-144.
doi: <http://dx.doi.org/10.1016/j.cep.2008.03.002>
- Modelon AB, *Post-combustion capture with amine solutions*. Montañés, R. M., Flø N. E., Dutta, R., Nord, L. O., Bolland, O., 2017. Dynamic process model development and validation with transient plant data collected from an MEA test campaign at the CO₂ Technology Center Mongstad. *Energy Procedia*. (accepted for publication).
doi: [10.1016/j.egypro.2017.03.1284](https://doi.org/10.1016/j.egypro.2017.03.1284)
- Montañés, R. M., Korpås, M., Nord, L. O. & Jaehnert, S., 2016. Identifying operational requirements for flexible CCS power plant in future energy systems. *Energy Procedia*, 86(TCCS-8), pp. 22-31.
doi: <https://doi.org/10.1016/j.egypro.2016.01.003>
- Panahi, M., 2011. *Ph.D. Thesis: Plantwide control for economically optimal operation of chemical plants - Applications to GTL plants and CO₂ capturing processes*. Trondheim: Norwegian University of Science and Technology. doi: <http://hdl.handle.net/11250/248272>
- Prölb, K., Tummerscheit, H., Velut, S. & Åkesson, J., 2011. Dynamic model of a post-combustion absorption unit for use in a non-linear model predictive control scheme.. *Energy Procedia*, 4(GHGT-11), pp. 2620-2627.
doi: <https://doi.org/10.1016/j.egypro.2011.02.161>

Optimizing the start-up process of post-combustion capture plants by varying the solvent flow rate

Thomas Marx-Schubach¹ Gerhard Schmitz¹

¹Institute of Engineering Thermodynamics, Hamburg University of Technology, Germany,
{thomas.marx,schmitz}@tuhh.de

Abstract

This paper presents an optimization of the start-up process of a post-combustion carbon capture plant (PCC-plant) by varying the solvent flow rate. In a first optimization run the start-up time is minimized. In a second optimization run the overall carbon capture rate during the start-up process is maximized. The results show the great potential of the optimization, as the start-up time can be reduced from $\Delta t = 4650$ s in the reference case to $\Delta t = 2840$ s in the optimized scenario.

Keywords: *optimization, start-up, absorption process, carbon capture, process engineering*

1 Introduction

To stop global warming, ambitious goals have to be set. At the United Nations Climate Change Conference in Paris in 2015 a limit for the global temperature increase was set to 1.5 °C above pre-industrial levels (United Nations, Framework Convention on Climate Change, 2016). To achieve this goal the carbon dioxide emissions have to be reduced significantly. One possibility is the usage of the Carbon Capture and Storage (CCS) technology, which means that the carbon dioxide is removed from the flue gas and can be stored in underground formations. As reported by the International Energy Agency (IEA) it is estimated that CCS can have a 17 % share on the CO₂ reduction in the year 2035 (IEA (International Energy Agency), 2013).

Furthermore, the increasing amount of renewable energies will lead to more fluctuations of net load in the power grid (Montañés et al., 2016). This will impose new challenges and operational requirements on the flexibility of thermal power plants as start-up and shutdown sequences and the operation at partial load will become still more important in the future.

The start-up process of a power plant is a time consuming and complex operation. When a carbon capture plant is coupled to a power plant the start-up procedure becomes even more complex. As power plants will start up and shutdown even more frequently in the future, the minimization of the start-up time is desirable. Hence, there is a requirement of reducing the start-up time. This can be achieved by optimizing the start-up time using dynamic models. As a first step an approach for the optimization

of a post-combustion carbon capture plant is given in this paper. At the moment the common start-up procedure of carbon capture plants is only based on experience.

Many different studies focus on the dynamic simulation of post-combustion carbon capture plants (Bui et al., 2014). Some studies present the optimization of a capture plant but most of them focus only on the steady state and full time operation. Since the full time operation of carbon capture plants might not be economically feasible, also the optimization of dynamic operation periods will become important in the future (Bui et al., 2014). There are also studies available dealing with the optimal control of the process, e.g. (Panahi and Skogestad, 2011), (Åkesson et al., 2012), (Lin et al., 2011), (Luu et al., 2015) and (Mechleri et al., 2017). However, to the authors knowledge the studies do not concentrate on modelling start-up and shut down procedures of the whole plant. In order to close this gap, a model which can describe the start-up process is developed and a first approach for an optimization of the process is presented.

2 Process description

Several processes are available for the post-combustion capture (PCC) of CO₂ from power plant flue gases such as membrane processes, adsorption and absorption processes. One possibility is to remove the CO₂ in a gas scrubbing unit using aqueous solutions of different alkylamines as a solvent, also known as the amine gas treating process. A process flow diagram of this process is shown in figure 1. The flue gas first enters a flue gas cooler and is compressed in a blower. The cooled flue gas enters the absorption unit at the bottom of the column. The solvent flows counter-current to the flue gas down to the column sump. 90 % of the CO₂ is chemically absorbed by the solvent. Afterwards the rich solvent is preheated in a heat exchanger and pumped to the stripper. The solvent is evaporated in a reboiler using steam from the power plant to provide the required energy for solvent regeneration. Desorption takes place in the stripper and the almost pure CO₂ (up to 99.9 %) leaves the plant through a condenser. The regenerated solvent is pumped back to the absorption unit. Many different solvent additives, primary, secondary and tertiary amines, can be used. In this case the used solvent is a 30 wt.-% monoethanolamine (MEA) aqueous solution.

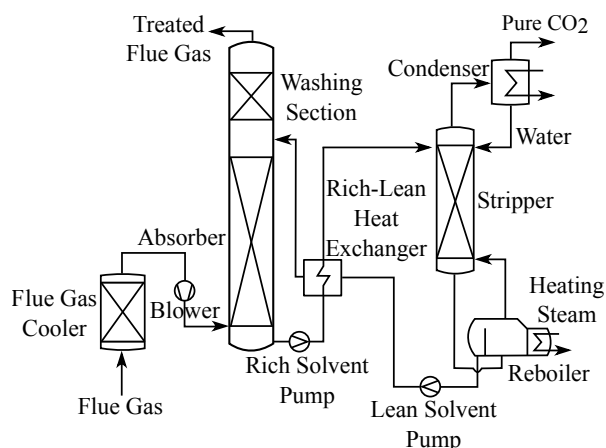


Figure 1. Process flow diagram of a post-combustion capture plant (Wellner et al., 2016).

The columns are usually filled with structured or random packings of different types.

The start up process of the plant is described in the following. At the beginning, the columns are in a cold and empty state. The process is started with the activation of the solvent pumps. Simultaneously, the other pumps for the washing sections, flue gas cooler and condenser are switched on. When the packing units inside the columns are wetted the heating steam can be supplied in the reboiler and the flue gas compressor can be switched on. The start up process is completed when the capture rate in the absorber reaches its nominal value of 90 %. To achieve this goal, the thermal and chemical equilibrium in the absorption unit has to be reached and the stripper has to attain his operation temperature of approximately 120 °C to ensure a sufficient regeneration of the solvent. The heat up process of the stripper takes the most time and is therefore the limiting factor for the start up time. The stripper heats up in two different ways.

1. The solvent is evaporated in the reboiler and the vapour enters the stripper at the lower part of the column.
2. The solvent which remains in the stripper sump is pumped through a counter flow heat exchanger and heats up the rich solvent that enters the stripper at the top.

A reduction of the start-up time can be performed by selecting the right solvent flow trajectory, which is not trivial, since the variation of the solvent flow rate has counteracting effects on the heat up rate. When the solvent flow rate is increased the temperature in the reboiler is reduced and it takes more time for the solvent to reach the boiling point. On the downside, the residence time of the solvent in the stripper sump decreases leading to a faster increase of the solvent temperature in the stripper sump. Decreasing the solvent flow rate leads to the opposite result. Therefore, the optimal trajectory of the solvent flow rate has to be

found numerically using a model of the post combustion carbon capture plant.

3 Model description

In this section the developed model and the used model libraries are described briefly. A detailed model for the dynamic simulation of the described process is developed within the *ThermalSeparation* library in Modelica (Dietl, 2012; Joos et al., 2009). The Optimization is performed with the commercial *Optimization Library* developed by DLR (A. Pfeiffer, 2012).

3.1 Model libraries

The *ThermalSeparation* library is a free Modelica library intended to describe separation processes such as absorption and rectification processes in Modelica. It can be used for the dynamic simulation of tray and packed columns with different levels of detail. More information about the library can be found in (Dietl, 2012; Joos et al., 2009).

The *Optimization* library is a commercial library for many different optimization tasks such as Trajectory Optimization, Realtime Optimization and Model Optimization. The library is released with Dymola and works only in this simulation environment. In this article the Trajectory Optimization is used. Further information about the library is given in (A. Pfeiffer, 2012).

3.2 Start-up model

A model of a post-combustion CO₂ capture plant, that can describe the start-up process, is developed at the Institute of Engineering Thermodynamics in Modelica using the *ThermalSeparation* library and validated with data of a pilot plant.

The pilot plant is located in Heilbronn, Germany and can handle a nominal flue gas stream of 1500 m³/h. The absorber has a height of 40 m and the stripper of 30 m. Both columns have a diameter of 0.6 m and are filled with the random packing type VSP-25 (VFF GmbH, 2016). The most relevant parameters of the nominal operation point are listed in table 1. More information about the pilot plant can be found in (Rieder and Unterberger, 2013).

The pilot plant is modelled using a first principle approach where the columns are axially divided into stages. The vapour and liquid mass and energy balances are solved separately in each stage. For the heat- and mass transfer across the phase boundary a equilibrium approach is used. The chemical reaction of carbon dioxide with the solvent is considered. It is assumed that the reaction takes place only in the liquid phase. More information about the model and the underlying assumptions can be found in (Wellner et al., 2016).

The most important input variables of the pilot plant are the solvent flow rate \dot{V}_{liq} , the molar flue gas flow rate \dot{N}_{fg}

Table 1. Main parameters of pilot plant.

Parameter	Value
Flue gas volume flow rate	1500 m ³ /h
CO ₂ concentration flue gas	12.3 mol-%
Flue gas temp. absorber inlet	40 °C
Solvent flow rate	5.3 m ³ /h
Solvent temp. absorber inlet	35 °C
MEA concentration	30 wt.-%
Stripper pressure	2 bar
CO ₂ product temperature	23 °C
CO ₂ target capture rate	90 %

and the heat flow rate to the reboiler \dot{Q}_{reb} .

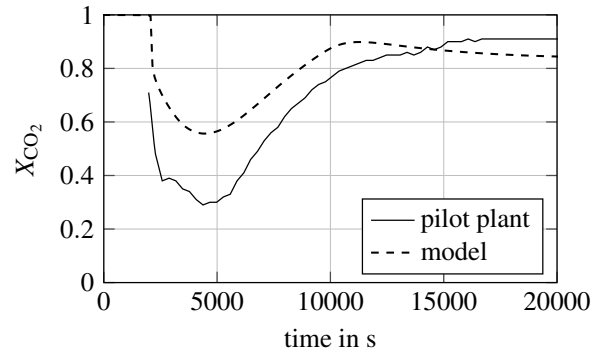
While the solvent flow rate and the heat flow rate are controllable within their limits, the flue gas flow rate depends on the firing output of the power plant. The flue gas flow rate is only controllable when a part of the flue gas is bypassed. In an usual start up scenario the solvent flow rate and the heat flow in the reboiler are set to their steady state values during the whole start-up process. However, this results in a very high start-up time. Hence, there is a high potential for optimization.

Simulation and validation results of the validation start-up scenario are shown in the following. At the beginning of the start-up procedure, the solvent pumps are switched on until the CO₂ concentration in the solvent is homogenized in the whole plant and until the columns are wetted. This step was not included in the optimization, since the solvent pumps should be operated at maximum flow rate to wet the columns as fast as possible. Hence, the focus of the optimization lies on the second part of the process when the flue gas flow enters the absorber.

In Figure 2 the carbon capture rate during the second part of the start-up process in the model and in the pilot plant is illustrated. The CO₂ capture rate is defined in the following way, where $\dot{N}_{CO_2,Abs,out}$ is the molar flow rate of CO₂ after the absorption unit and $\dot{N}_{CO_2,Abs,in}$ the molar flow rate of CO₂ before the absorption unit.

$$X_{CO_2} = 1 - \frac{\dot{N}_{CO_2,Abs,out}}{\dot{N}_{CO_2,Abs,in}} \quad (1)$$

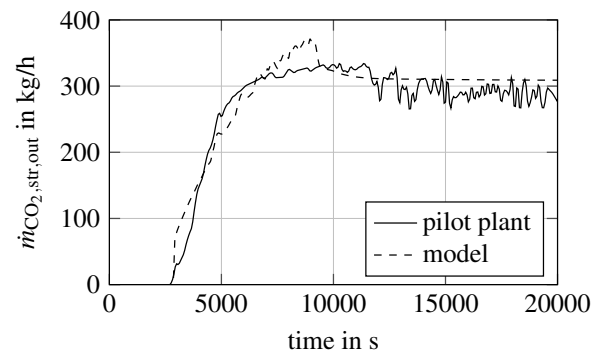
At $t = 0$ s the flue gas flow to the absorber is started. The steam supply in the reboiler starts later at $t = 240$ s due to the time delay in the steam generator and the steam cycle of the power plant. All incoming CO₂ is absorbed until the solution is saturated. Therefore, the CO₂ capture rate is nearly 100 % at the beginning and drops quickly after approximately 2000 s since the solvent in the absorption unit is saturated with CO₂ and is not regenerated in the stripper yet. The capture rate in the pilot plant can only be calculated from this point on because the CO₂ concentration at the absorber outlet has not been measured correctly before. Due to the increasing temperature in the stripper, the CO₂ loading of the solvent in the stripper decreases

**Figure 2.** Validation start-up strategy - CO₂ capture rate

and the capture rate rises slowly to its steady state value.

After $t = 2000$ s the results of the model show a significant deviation from the measurements of the pilot plant. One reason is that the columns are modelled based on an equilibrium approach which means that the heat and material transport equations in each stage are neglected. Another reason is the deviation of the measuring instruments, since some measured values, e.g. the CO₂-concentration at the absorber outlet, are far from their nominal values. However, the agreement of the dynamic behaviour between the model and the pilot plant is very good except for the small overshoot after $t = 10000$ s. The start-up process is completed when the plant is capable of keeping a capture rate of 90 %.

Figure 3 shows the transient behaviour of the stripped CO₂ mass flow rate downstream the condenser for the same time period. The measured values are also compared with the simulated ones.

**Figure 3.** Validation start-up strategy - CO₂ mass flow downstream the stripper

The simulation data shows a very good agreement with the measurement data except for the overshoot in the simulation data at the end of the start-up process. The overshoot can be explained by the model assumption that the steam in the stripper condenses in every stage where the boiling temperature is not reached yet. In reality the CO₂ leaving the stripper is loaded with steam especially at high temperatures close to the boiling point. This energy loss leads to a smaller amount of stripped CO₂ in the pilot plant at

Table 2. General criteria in optimization runs

Variable name	Constraint	Type
Reboiler temperature	$T_{Reb} \leq 125^{\circ}\text{C}$	at any time
Liquid hold-up	$V_{liq,Reb} \geq 0.35 \text{ m}^3$	at any time
Solvent flow rate	$0.0002 \text{ m}^3/\text{s} \leq \dot{V}_{liq} \leq 0.002 \text{ m}^3/\text{s}$	at any time

the end of the start-up process. It can be seen that the amount of stripped CO_2 in the stripper is nearly the same in the model as in the pilot plant in the timespan between approximately $t = 2000 \text{ s}$ and $t = 8000 \text{ s}$. However, the amount of captured CO_2 is much higher in the model in the same timespan. Therefore, if one assumes a correct initial amount and loading of the solvent, the overall CO_2 mass balance in the pilot plant is not fulfilled, which leads to the assumption of a certain measuring error. Furthermore, the time limiting component during the start-up process is the stripper. The model can therefore be used for optimization purposes.

According to the simulation, the start-up time in the validation case is $\Delta t = 9440 \text{ s}$, which is very high.

4 Optimization setup

In this section the simplification of the model and the implementation of the optimization run is presented.

The model used for validation is simplified in a first step in order to reduce the computation time and improve the robustness of the model. The following simplifications of the model were made:

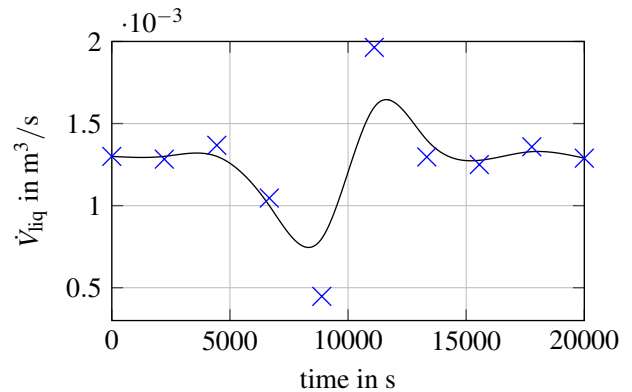
- The intermediate cooling unit in the absorber was neglected.
- The flue gas scrubbing units before and after the absorber were removed

Furthermore, the total amount of solvent in the pilot plant is specifically high compared to other post-combustion capture plants (Wellner et al., 2016). Therefore, the total amount of solvent in the pilot plant is reduced by 50 % to get more commonly results. Thermal stresses were neglected in the reboiler as the start-up process is most of all limited due to the high amount of solvent in the stripper sump. Nevertheless, thermal stresses should be taken into account in the future to prove this assumption. An overview of the model used for the optimization is shown in Figure 5. The gas streams are marked orange, the solvent streams are marked blue. The flue gas stream entering the absorber and the heat flow rate are implemented using a ramp from the Modelica Standard library as source signal. The solvent flow rate can be set by using an input connector which is connected to the lean

solvent pump. The input connector is used for the optimization. The rich solvent pump is controlled keeping the filling level of the absorber sump at a setpoint of 2 metres.

As optimization method a *Single Shooting Technique* approach is used. The trajectories are approximated with B-splines of order 3. For the construction of the splines 10 equidistant knots, so called de Boor points, are used in the optimization runs. They can be used as tuner variables in the optimization method. Using more knots would lead to a more accurate solution but the optimization would take more time. Optimizations with different amounts of knots were carried out. The result is that 10 knots are a good compromise between accuracy and simulation time. An example of the construction of the trajectory using B-splines is shown in figure 4. The optimization problem is solved by using the gradient based Sequential Quadratic Programming (SQP) algorithm. The algorithm is effective for solving nonlinear optimization problems with linear constraints. The constraints are essential for the optimization problem. Therefore an algorithm which is capable of solving optimization problems with constraints has to be used.

Generally, different optimization algorithms for nonlinear optimization with constraints can be used, such as Random search, interior-point method. A overview of possible optimization algorithms can be found in (Rao, 2009). In this paper a gradient based method was used as these algorithms converge faster in general. The SQP algorithm is sufficient for solving the optimization problem. A disadvantage is that the algorithm can only find local optima.

**Figure 4.** Example of the construction of B-splines

A general optimization problem with equality and in-

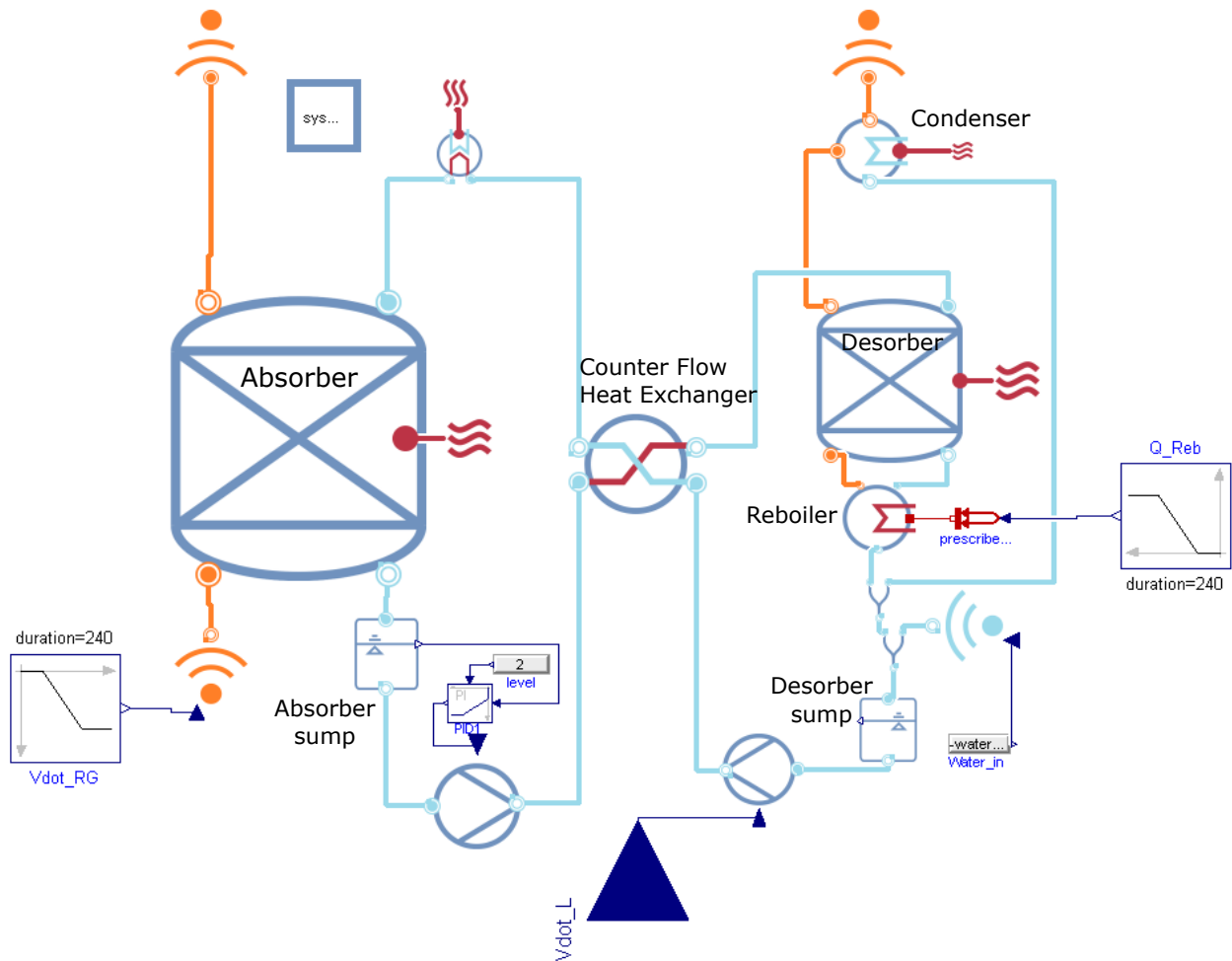


Figure 5. Simplified Modelica model of the carbon capture plant in Dymola simulation environment.

equality constraints can be defined in the following way.

$$\begin{aligned} & \text{minimize } f(x) \\ & \text{in subject to} \\ & \quad g_i \leq a \\ & \quad h_i = b \end{aligned} \quad (2)$$

Before the optimization is started, the minimization criteria and the different constraints have to be defined. In the first optimization the start-up time is minimized. In a second optimization the mean value of the CO₂ capture rate during the start-up process is maximized.

The optimal trajectory of the solvent flow rate has to satisfy many constraints to ensure that the plant is operated within the permissible operation range. Two of them are used in both optimization runs. First, the temperature in the reboiler may not exceed 125 °C at any time since the degradation of the solvent increases significantly at higher temperatures. Second, the hold-up in the reboiler may not fall below 0.35 m³ to provide that the heat pipes are covered with solvent.

The SQP solver can only handle inequality constraints in the following form.

$$g_i \leq b \quad (3)$$

As the hold-up in the reboiler may not fall below a specific value, the inverted hold-up is used as constraint. For the reboiler temperature and the hold up a *Maximum-Block* of the *Optimization-Library* is used to make sure that the temperature or liquid hold-up do not exceed their limits during the whole simulation time as the optimization algorithm only evaluates the last value of the criteria. The general constraints are summarized in table 2.

The other two input variables, the flue gas flow and the heat flow rate to the reboiler, are kept constant during the optimization runs at their steady state values.

The optimization runs are executed on a windows server with two Intel® Xeon E5-2650 v3 CPU and 128 GB memory. To improve the simulation time, the integration of the model is parallelized by the *Optimization-library*. The maximum number of threads is set to 8.

5 Reference start-up scenario

As mentioned in section 3.2 the implementation of the start-up process is based on experience. The current start-up process is performed by setting all manipulated variables to their steady-state values, which results in a high

start-up time. As presented in section 4 the model structure was changed slightly and the total amount of solvent was reduced. Therefore, the validation scenario shown in section 3.2 can not be used as a reference for the optimization. The start-up time decreases significantly due to the reduction of the total amount of solvent. For this reason a new reference scenario has to be defined by simulating the reduced model. The carbon capture rate in the reference scenario is shown in figure 6.

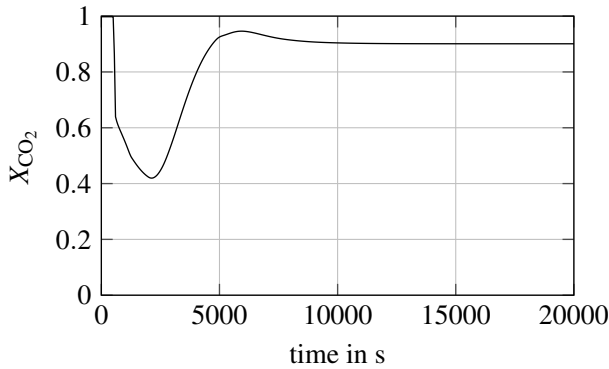


Figure 6. Reference start-up scenario - capture rate

All manipulated variables are kept at their steady state values during the whole start-up process in the reference case. The start-up time in the reference scenario is $\Delta t = 4650$ s, which is reduced in the following optimization scenario.

6 Minimizing the start-up time

In this section the minimization of the start-up time should be achieved by finding the optimal solvent flow trajectory (optimization 1). For the optimization run additional constraints and criteria have to be set.

First, the end of the start-up process has to be specified by using constraints. Therefore, it is defined that the carbon capture rate has to be at least 90 % at the end of the optimization. However, this constraint does not guarantee that the plant can keep a stable capture rate of 90 %. To make sure that the solvent is sufficiently regenerated at the end of the start-up process, a constraint for the solvent loading in the stripper sump is added. The solvent loading is defined in equation 4 and is the ratio of the amount of CO_2 and MEA in the solvent.

$$\alpha = \frac{N_{\text{CO}_2}}{N_{\text{MEA}}} \quad (4)$$

The solvent loading in the stripper sump has to reach the steady-state value of $\alpha = 0.184 \text{ mol}_{\text{CO}_2} / \text{mol}_{\text{MEA}}$ to ensure a stable carbon capture rate of 90 %. As the library only handles criteria in the form presented in equation 3 the capture rate was also inverted in the model. The additional criteria are shown in table 3.

Table 3. Additional criteria in optimization run 1

Variable name	Criteria	Type
Capture rate	$X_{\text{CO}_2} \geq 0.9$	end point
Solvent loading	$\alpha_{\text{str, sump}} = 0.184 \frac{\text{mol}_{\text{CO}_2}}{\text{mol}_{\text{MEA}}}$	end point
Start-Up time	t	minimize

The resulting solvent flow rate trajectory of the first optimization run is shown in Figure 7.

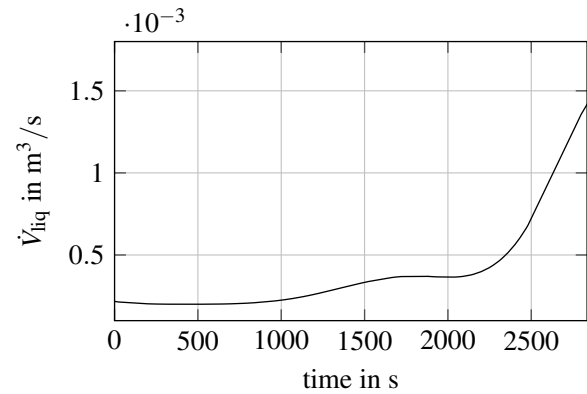


Figure 7. Minimized start-up time - solvent flow rate

As in the reference start-up scenario presented in chapter 3.2 the flue gas flow to the absorber starts at $t = 0$ s. The steam supply in the reboiler starts also later at $t = 240$ s due to the time delay in the steam generator and the steam cycle of the power plant. The solvent flow trajectory during the start-up process can be split up into three phases. At the beginning of the optimal start-up process the solvent flow rate is at the minimum flow rate of $0.0002 \text{ m}^3/\text{s}$. The minimum solvent flow rate leads to a maximum of the reboiler heat up rate. The reboiler temperature is shown in figure 8.

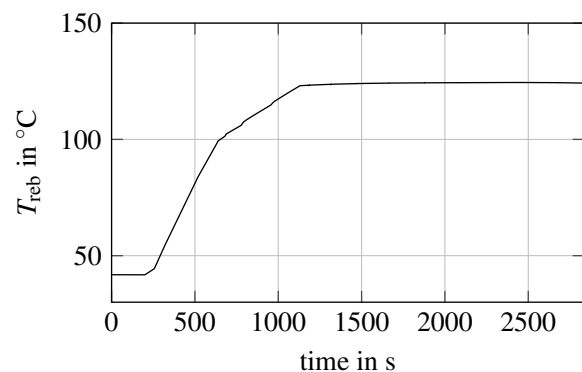


Figure 8. Minimized start-up time - Reboiler temperature

When the reboiler reaches its operation temperature of approximately 120°C the solvent flow rate rises to $0.00037 \text{ m}^3/\text{s}$. This happens for two reasons. On the one

hand the solvent flow rate has to be increased to prevent that the temperature in the reboiler exceeds the limit of 125 °C. On the other hand the increasing solvent flow rate leads to a smaller residence time in the stripper sump in order to achieve a higher heat up rate of the solvent in the stripper sump. As soon as the stripper reaches the operation temperature the solvent flow rate increases continuously to increase the capture rate. At a capture rate of 90 % a PID controller is switched on. The controller uses the capture rate of 90 % as a setpoint and the solvent flow rate as the manipulated variable. This results in a small oscillation of the solvent flow rate after the controller is switched on. The start-up process is finished at this point.

Figure 9 shows the carbon capture rate during and after the start-up process.

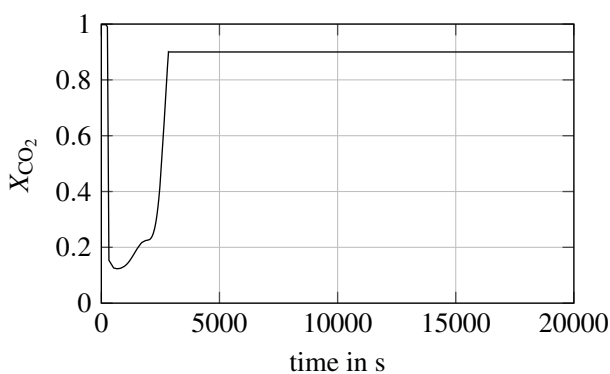


Figure 9. Minimized start-up time - capture rate

The capture rate drops quickly to a very small value of 12 % and increases with increasing solvent flow rate and stripper temperature until a capture rate of 90 % is reached. The start-up time is significantly reduced from $\Delta t = 4650$ s to $\Delta t = 2840$ s. As illustrated in figure 9, the plant can keep the capture rate of 90 % after the start-up process.

The result of the optimization is a specific optimal solvent flow trajectory for the pilot plant. As the trajectory cannot be applied directly to other plants a more generalized approach for the solvent flow trajectory should be given. Furthermore, the optimal solvent flow trajectory is quite complex since many control actions are required.

To solve this problem a simplified approach based on the optimal solvent flow trajectory is developed. As already mentioned, the solvent flow trajectory can be divided into three phases. Based on this segmentation the recommendations for an optimal start-up scenario are:

1. Set the solvent flow rate to its steady state or maximum possible value to wet the columns.
2. When the columns are wetted and the steam can be supplied from the power plant reduce the solvent flow rate to the lowest possible value.
3. As soon as the reboiler temperature reaches the operation temperature of 120 °C, increase the solvent flow rate to a certain value (in the case of the pilot plant:

$\dot{V}_{liq} = 0.00037 \text{ m}^3/\text{s}$). The value has to be found by optimizing a model of the certain plant. If this is not feasible another possibility is to control the reboiler temperature with the solvent flow rate by using a PID controller.

4. When the top of the stripper is heated up to 100 °C set the solvent flow rate to the optimal steady state value (in the pilot plant: $\dot{V}_{liq} = 0.001289 \text{ m}^3/\text{s}$).
5. Just as the capture rate reaches 90 % switch on the PID controller to keep a constant capture rate.

When applying the simplified start-up strategy to the model, the start-up time increases slightly. However, the implementation of the start-up strategy is a lot simpler. The simplified solvent flow rate derived from the optimization case is presented in figure 10.

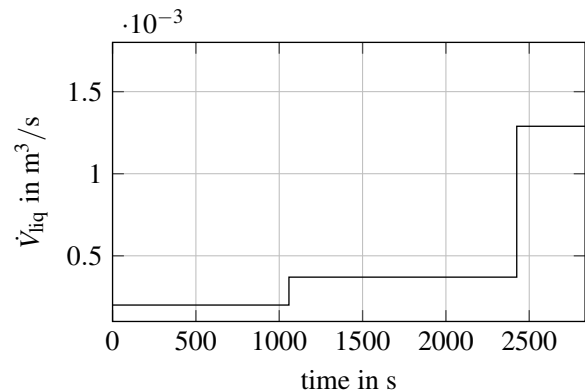


Figure 10. Minimized start-up time - solvent flow rate (simplified)

The result for the carbon capture rate is shown in figure 11. The carbon capture rate increases as expected steeply with increasing solvent flow rate. Between the steps of the solvent flow rate the capture rate increases slightly as the CO₂ loading in the stripper sump decreases slowly over time.

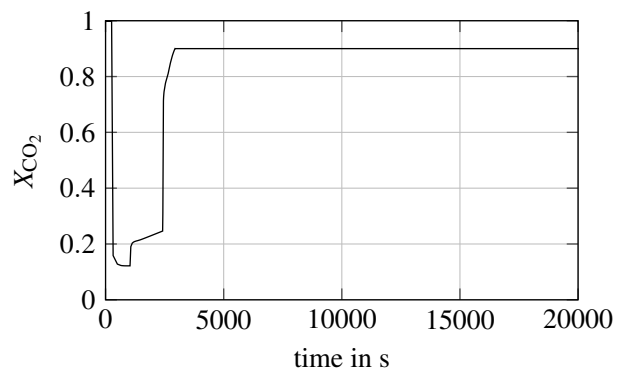


Figure 11. Minimized start-up time - capture rate (simplified)

The start-up time in the simple case increases marginally from $\Delta t = 2840$ s to $\Delta t = 2939$ s. However, the

implementation of the strategy requires only three control actions of the solvent flow rate during the whole start-up process. The simple approach also offers a high reduction of the start-up time while the constraints are still fulfilled. The reboiler temperature and the liquid hold-up in the reboiler do not exceed their limits at any time.

7 Maximizing the capture rate

In a second approach, the main goal is to maximize the mean value of the capture rate during the start-up process (optimization 2). This start-up strategy is useful when low CO₂ emissions are more important than a minimal start-up time.

For this optimization run additional constraints have to be determined. The maximum CO₂ capture rate is limited to $X_{CO_2} = 0.95$ after $t = 1000$ s since the carbon capture efficiency is decreasing steeply at very high capture rates. The reason is the very low CO₂ partial pressure in the gas phase at very high capture rates. The accuracy of the model decreases in this point of operation. The optimal stationary solvent flow rate is $\dot{V}_{liq} = 0.001289 \text{ m}^3/\text{s}$ which is defined as the end point of the trajectory. As the library only handles minimization criteria as presented in equation 3 the mean value of the negative carbon capture rate in a time period of $t = 20000$ s is chosen. The additional constraints are shown in table 4.

The resulting solvent flow rate trajectory of the second optimization run is shown in Figure 12. The trajectory starts at $0.001 \text{ m}^3/\text{s}$ and decreases to a minimum at approximately $0.00064 \text{ m}^3/\text{s}$. Afterwards the solvent flow rate rises till $0.0016 \text{ m}^3/\text{s}$ and drops again until the optimal stationary solvent flow rate of $0.001289 \text{ m}^3/\text{s}$ is reached.

Table 4. Additional constraints in optimization run 2

Variable name	Criteria	Type
Capture rate limit	$X_{CO_2} \leq 0.95$	anytime
Solvent flow rate	$\dot{V}_{liq} = 0.001289 \text{ m}^3/\text{s}$	end point
Mean capture rate	X_{CO_2}	maximize

The CO₂ capture rate in the second optimization run is depicted in figure 13. In comparison with the reference start-up strategy, the capture rate drops quickly to a lower value at the beginning because of the lower solvent flow rate. However, the lower solvent flow rate leads to a faster heat-up in the stripper and regeneration of the solvent. When the solvent flow rate is increased the carbon capture rate increases as well. As a side effect, the start-up time is also in this scenario reduced significantly to $\Delta t = 3520$ s.

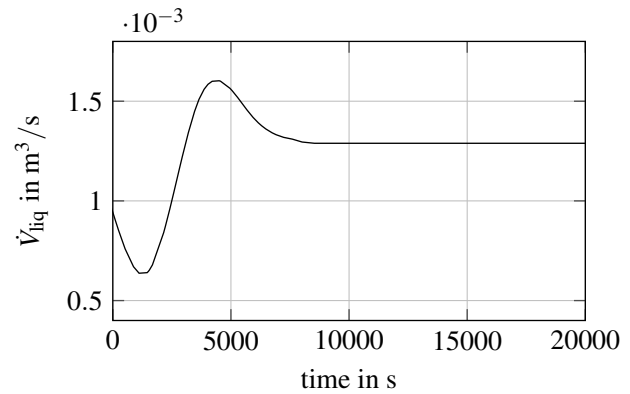


Figure 12. Maximized amount of captured CO₂ - solvent flow rate

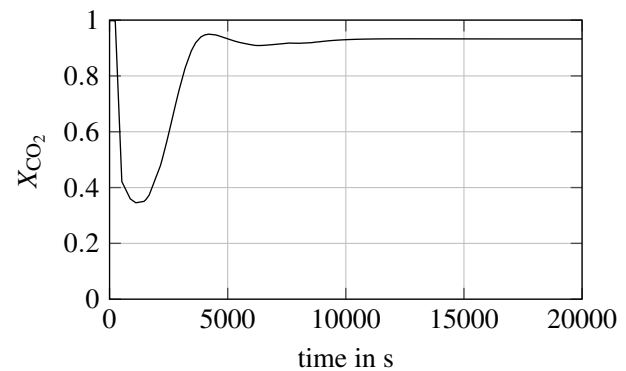


Figure 13. Maximized amount of captured CO₂ - capture rate

In table 5 a comparison of the mean capture rate in the different start-up scenarios over the same time period of $t = 20000$ s is shown. The minimization of the start-up time is labelled as *Optimization 1*, the maximization of the mean capture rate is labelled as *Optimization 2*. The reference strategy is presented in section 5.

Table 5. Mean value of capture rate in different start-up scenarios

Reference	Optimization 1	Optimization 2
0.836	0.805	0.856

As expected the highest mean capture rate is achieved in the second optimization run, but the enhancement of the mean capture rate in comparison with the reference start-up strategy is quite small. However, the second optimization run leads to a very good compromise between a relatively fast start-up process and a high amount of captured carbon dioxide during the start-up process as the mean capture rate is comparatively small in the first optimization run. In the future, also a multicriteria optimization with the combination of both optimization criteria is conceivable.

8 Performance

The optimization model consists of 29833 equations. As the optimization algorithm can only find local optima one can not guarantee that the best solution has been found. Therefore, the optimization is repeated several times and different start values for the solvent flow rate are used. The result is that the solutions actually are different but they vary only slightly when different start values are used. Sometimes the algorithm does not converge. This is mostly due to the fact that the optimization algorithm cannot evaluate a new solution since the integration of the model fails in some cases. One optimization run takes approximately an average of 18 hours.

9 Conclusion and Outlook

The results of this paper show that there is a high optimization potential for the start-up process of a post-combustion capture plant. The start-up time can be significantly reduced by varying the solvent flow rate only.

The results of the model optimization should be applied to the real operation of a carbon capture plant for validation purposes in the future. Unfortunately, the pilot plant used for the validation of the model is no longer in operation.

The optimization does not take thermal stresses in the reboiler into account. Additionally, the reboiler should be discretized in space in order to calculate local temperatures in the reboiler. Both options should be added to the model in the future.

This work focusses on the start-up process at full load. Future work could concentrate on the start-up process at partial load. Furthermore, future optimization should include the variation of other parameters of the plant that influence the start-up process as for example the total amount of solvent. It could be also performed with the used *Optimization Library*.

The technical design of the pilot plant was not changed during the optimization. Future work should also include the improvement of the technical design. A possible option would be for example the implementation of the lean vapour compression (Fernandez et al., 2012).

The SQP solver used in the optimization does not guarantee that the global optimum is found. The optimization should be also performed by using a genetic algorithm to confirm the results of this paper in the future.

Nomenclature

α	solvent loading (mol _{CO₂} /mol _{MEA})
\dot{m}	mass flow rate (kg/s)
\dot{N}	molar flow rate (mol/s)
\dot{Q}	heat flow rate (W)

\dot{V}	volume flow rate(m ³ /s)
N	amount of substances(mol)
T	temperature (°C)
t	time s
X	CO ₂ capture rate (-)

Abbreviations

CO ₂	carbon dioxide
CCS	carbon capture and storage
MEA	monoethanolamine
PCC	post-combustion carbon capture

Subscripts

abs	absorber
fg	flue gas
in	incoming stream
liq	liquid
out	outgoing stream
reb	reboiler
str	stripper
sump	column sump

References

- A. Pfeiffer. Optimization Library for Interactive Multi-Criteria Optimization. In *Proceedings of the 9th International Modelica Conference*, pages 669–680. Modelica Association, 2012. doi:10.3384/ecp12076669.
- A. Bui, I. Gunawan, V. Verheyen, P. Feron, E. Meuleman, and S. Adeloju. Dynamic modelling and optimisation of flexible operation in post-combustion CO₂ capture plants-a review. *Computers and Chemical Engineering*, 61:245–265, 2014.
- Karin Dietl. *Equation-Based Object-Oriented Modelling of Dynamic Absorption and Rectification Processes*. PhD thesis, Hamburg University of Technology, Hamburg, Germany, 2012.
- Eva Sanchez Fernandez, Egbertus J. Bergsma, and Thijs J.H. Vlugt Ferran de Miguel Mercader. Optimization of lean vapour compression (lvc) as an option for post-combustion CO₂ capture: Net present value maximisation. *International Journal of Greenhouse Gas Control*, 11:114 – 121, 2012.
- IEA (International Energy Agency). World energy outlook special report: Redrawing the energy-climate map. Technical report, OECD/IEA, France, 2013.
- Andreas Joos, Karin Dietl, and Gerhard Schmitz. Thermal Separation: An Approach for a Modelica Library for Absorption, Adsorption and Rectification. In *Proceedings of the 7th International Modelica Conference*, pages 804–813. Modelica Association, 2009.

Yu-Jeng Lin, David Shan-Hill Wong, Shi-Shang Jang, and Jenq-Jang Ou. Control strategies for flexible operation of power plant with CO₂ capture plant. *AIChE J*, 58:2697–2704, 2011. doi:10.1002/aic.12789.

Minh Tri Luu, Norhuda Abdul Manaf, and Ali Abbas. Dynamic modelling and control strategies for flexible operation of amine-based post-combustion CO₂ capture systems. *International Journal of Greenhouse Gas Control*, 39:377–389, 2015. doi:10.1016/j.ijggc.2015.05.007.

Evgenia Mechleri, Adekola Lawal, Alfredo Ramos, John Davison, and Niall Mac Dowell. Process control strategies for flexible operation of post-combustion CO₂ capture plants. *International Journal of Greenhouse Gas Control*, 57:14–25, 2017. doi:10.1016/j.ijggc.2016.12.017.

Rubén M. Montañés, Magnus Korpås, Lars O. Nord, and Stefan Jaehnert. Identifying operational requirements for flexible CCS power plant in future energy systems. *Energy Procedia*, 86:22 – 31, 2016.

Mehdi Panahi and Sigurd Skogestad. Economically efficient operation of CO₂ capturing process. Part II. Design of control layer. *Chemical Engineering and Processing*, 52:112–124, 2011. doi:10.1016/j.cep.2011.11.004.

Johan Åkesson, Carl D. Laird., Geoffry Lavedan., Katrin Pröhl, Hubertus Tummescheit, Stephane Velut, and Yu Zhu. Non-linear Model Predictive Control of a CO₂ Post-Combustion Absorption Unit. *Chem. Eng. Technol.*, 35:445–454, 2012. doi:10.1002/ceat.201100480.

Singiresu S. Rao. *Engineering Optimization: Theory and Practice*. Wiley-VCH, 2009. doi:10.1002/9780470549124.

Alexander Rieder and Sven Unterberger. EnBW's post-combustion capture pilot plant at Heilbronn - Results of the first year's testing programme. *Energy Procedia*, pages 1553–1571, 2013.

United Nations, Framework Convention on Climate Change. Report of the Conference of the Parties on its twenty-first session, held in Paris from 30 November to 13 December 2015, 2016.

VFF GmbH. Manufacturer of random packings for pilot plant, 2016. URL <http://http://www.vff.de/en/products/random-packings>.

Kai Wellner, Thomas Marx-Schubach, and Gerhard Schmitz. On the dynamic behaviour of coal fired power plants with post-combustion CO₂ capture. *Industrial & Engineering Chemistry Research*, 55(46):12038–12045, 2016. doi:10.1021/acs.iecr.6b02752.

Framework for dynamic optimization of district heating systems using Optimica Compiler Toolkit

Gerald Schweiger¹ Håkan Runvik² Fredrik Magnusson^{3,2} Per-Ola Larsson² Stéphane Velut²

¹AEE INTEC, 8200 Gleisdorf, Austria, gerald.schweiger@aee.at

²Modelon AB, SE-223 70 Lund, Sweden, {per-ola.larsson, hakan.runvik, fredrik.magnusson, stephane.velut}@modelon.com

³Lund University, SE-221 00 Lund, Sweden, fredrik.magnusson@control.lth.se

Abstract

Recent studies show that district heating infrastructures should play an important role in future sustainable energy systems. Tools for dynamic optimization are required to increase the efficiency of existing systems and design new ones. This paper presents a novel framework to represent, simplify, simulate and optimize district heating systems. The framework is implemented in Python and is based on Optimica Compiler Toolkit as well as Modelon's Thermal Power Library. The high-level description of optimization problems using Optimica allows flexible optimization formulations including constraints on physically relevant variables such as supply temperature, flow rate and pressures. The benefit of new algorithms for symbolic elimination in Optimica Compiler Toolkit is also investigated. The framework is applied on a test case, which is based on a planned city district located in Graz, Austria. The results demonstrate the generality of the representation as well as the accuracy of the simplification for dynamic optimization of temperature supply and pressure control. *Keywords:* district heating, dynamic optimization, symbolic elimination

1 Introduction

A major challenge for future energy systems is the design of systems that integrate large shares of fluctuating renewable inputs while improving the overall system efficiency. There are a number of options for increasing energy system flexibility, including the combination of different energy domains, increasing supply and demand flexibility or the integration of energy storage technologies. Previous research has shown that district heating infrastructure has the potential to play a key role in sustainable energy systems (Lund et al., 2014; Schweiger et al., 2017b). The new generation of district heating systems (called 4th generation district heating) plays an integral part of smart energy systems. Among others these systems will be characterized by intermittent operations and highly fluctuating supply temperatures. As reported by (Allegrini et al., 2015), there is much to be done to explore the full benefit of innovative district energy systems. They argue that a shift to fully dynamic

models and sophisticated control design would be supportive. Limitations of standard methods rely often on simplified models, static relationships and single-domain approaches. Therefore standard approaches are restricted and thus unsuitable for investigating many issues. The presented framework is based on the previous work of some authors (Velut et al., 2014; Runvik et al., 2015; Schweiger et al., 2017a).

The main contributions of this paper are (i) a demonstration of the capabilities of Modelon's Thermal Power Library that in version 1.14 will have out-of-the box models for dynamic thermo-hydraulic optimization of district heating systems, (ii) a demonstration of a framework for creating and manipulating district heating networks in Python as well as translating networks into executable Code for simulation and optimization and (iii) an investigation of the impact of the new algorithm for symbolic elimination available in JModelica.org and Optimica Compiler Toolkit (OCT).

2 Tools and languages

Three environments were used within the framework: (i) The unified network representation and the aggregation algorithms are implemented in Python; (ii) Dymola is used to simulate the complex models and (iii) JModelica.org and OCT were used to solve the dynamic optimization problem.

2.1 OCT and JModelica.org overview

2.1.1 JModelica.org

JModelica.org (Åkesson et al., 2010) is an open-source platform developed for simulation, optimization and analysis of complex dynamical systems. It utilizes the open Modelica and FMI (Functional Mock-up Interface) standards and has a Python-based user interface. It is developed in collaboration between Modelon and several academic institutions, such as the Department of Automatic Control and the group of Numerical Analysis at the Centre for Mathematical Sciences at Lund University.

Of special interest in this project is the dynamic optimization capabilities of this tool. An extension of the

Modelica language called Optimica (Åkesson, 2008) is used for this purpose. The model dynamics of the optimization models are described using Modelica. To add the extra information necessary to describe the optimization formulation Optimica is used. This means that constraints, objective function and optimization time horizon all can be collected in one easily understandable model. More details of how optimization problems are solved in the JModelica.org toolchain are presented in the following sections.

2.1.2 Optimica Compiler Toolkit

Optimica Compiler Toolkit (OCT) is based on JModelica.org technology, but has several additional features (Modelon, 2016). One of these is the support for encrypted libraries, which is of special interest in relation to this project. This makes it possible to combine the usage of commercial libraries, here Modelon's Thermal Power Library, with the optimization framework. This lowers the difficulty for users to solve their own optimization problems, when predefined components and media models from the library can be used.

3 District Heating Network Models

A district heating network model for short-term production planning must capture the following: (i) transport delays depending on mass flows, (ii) pressure losses and (iii) heat losses.

3.1 General model properties

The presented framework is based on the physics-based modeling language Modelica and a high-level, large-scale dynamic optimization method available in OCT.

High-fidelity models of district heating networks often have high computation cost and some model properties like events or non-differentiability make them even unusable in dynamic optimization. Hence, there is a need to design simpler models, in particular regarding size and differentiability, that can be used for online optimization. There is also a need to design accurate models that can be used for dynamic simulation to validate the optimal inputs computed based on the simpler model. Models of both types will be available in Modelon's Thermal Power Library 1.14.

Pipes are the central components in district heating systems. The pipe model for simulation is implemented based on a plug-flow approach as the solution of the following one dimensional energy balance:

$$\frac{\partial T}{\partial t} + v(t) \frac{\partial T}{\partial x} + \frac{1}{S\rho c_p} q(T(x)) = 0$$

where v is the fluid velocity, S the cross-section area, ρ the fluid density, c_p the specific heat capacity of the fluid and q the heat loss to the surroundings of the pipe. The Modelica built-in operator `spatialDistribution` provides a robust method to approximate the solution of such partial differential equations when there is no heat loss, i.e.

$q = 0$ (Modelica Association, 2014). The operator keeps track of the spatial distribution via suitable sampling, interpolation and shifting of the stored distribution and it also supports flow reversal. Assuming positive flow and a heat loss q that depends linearly on $T_{boundary} - T(x)$, the difference between the surrounding temperature and the fluid temperature, the temperature at the pipe outlet is

$$T(x = L, t) = T_{boundary} + (T(x = 0, t - \tau) - T_{boundary})e^{-\frac{\tau}{T_p}}$$

where L is the pipe length, τ the time-varying transport delay and T_p a temperature decay constant. From the previous equation, it can be seen that the pipe model with heat loss can be implemented using two `spatialDistribution` operators, one to keep track of the temperature distribution inside the pipe and therefore $T(x = 0, t - \tau)$, and one to calculate the time-varying delay τ that is needed to compute the impact of the heat loss given by $e^{-\frac{\tau}{T_p}}$.

The `spatialDistribution` operator can however not be used in the optimization framework because of insufficient differentiability of the involved equations. The pipe model for optimization (see Figure 1) contains a combination of a fixed time delay and a discretized dynamic volume. The goal is to compute the main characteristics of the pipe without having to use a model with a large number of segments which would increase model complexity. The fixed delay is dependent on the range of the mass flow for each pipe and corresponds to the minimal time delay. The dynamic volume must capture the flow-dependence of the varying transport delay. The number of segments within the dynamic volume depends on the geometry of the pipe.

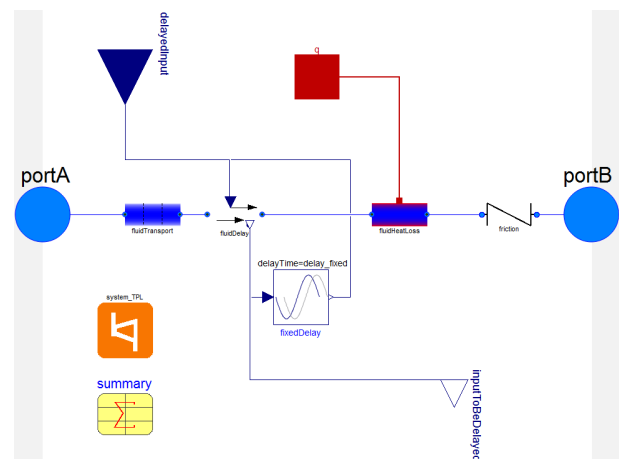


Figure 1. Pipe model for optimization consisting (from left to right) of a dynamic volume model (`fluidTransport`), a model that captures the fixed delay (`fluidDelay`), a model that calculates the heat losses (`fluidHeatLoss`) and a model that calculates the pressure drop (`friction`).

3.2 Case study

The case study represents a district heating network in a planned city district in Graz/Austria that consists of one

production unit, 16 consumers and a total length of about 4200 m, see Figure 2. We assume a perfect load prediction over the entire optimization horizon. Work has also been done on non-perfect load prediction (Rantzer, 2015), but was omitted here to focus on other parts of the framework.



Figure 2. The scheme of the district heating system. The production unit is seen on the right side; the black/blue circles represent the 16 consumers.

Optimica files that extend the optimization models are used to describe the optimization problems. The dynamic optimization problem used for all optimization cases has the general form

$$\begin{aligned} \min. \quad & \int_{t_0}^{t_f} (\alpha T_{\text{prod}} + \beta dp_{\text{prod}} + \gamma \dot{Q}_{\text{prod}}^2 + \delta \dot{p}_{\text{prod}}^2) dt, \\ \text{s.t.} \quad & \text{model dynamics,} \\ & m_{\text{prod}}(t) \leq m_{\text{prod}}^U \quad \forall t \in [t_0, t_f], \\ & T_{\text{customer}}^L \leq T_{\text{customer}}(t) \quad \forall t \in [t_0, t_f], \\ & dp_{\text{customer}}^L \leq dp_{\text{customer}}(t) \quad \forall t \in [t_0, t_f], \end{aligned}$$

where T_{prod} is the supply temperature, dp_{prod} the differential pressure at the production unit, Q_{prod} the load and α, β, γ as well as δ are weights. The load derivative \dot{Q}_{prod} and the pressure derivative \dot{p}_{prod} are the degrees of freedom in the optimization formulation. These are squared in the cost function to penalize fast control signal changes. m_{prod}^U is the upper limit of the mass flow at the production unit and it was set to 65 kg/sec; it is representing the pump limitations. T_{customer}^L is the lower limit of the supply temperature for all customers and it was set to 60 deg.C. The lower limit of the differential pressure for all customers (dp_{customer}^L) was set to 0.5 bar. Minimum

supply water temperatures and pressure differences for all customers were introduced based on real network limits to satisfy the customers' demand. The minimization of supply water temperature and pressure difference for the production unit mimics the situation in a real plant where low temperatures and differential pressures are desirable in order to reduce heating and pumping costs. The weights are chosen such that a low temperature is given a higher priority than pressure minimization, as this is the relatively larger cost in reality. The optimization constraints are inequality constraints defined using the `min` and `max` variable attributes.

4 Framework

4.1 Overview

A schematic view of the framework is presented in Figure 3 and each step is explained below.

- Step 1: The network is created using a unified network representation that includes data of the network, demand and boundary conditions.
- Step 2: The unified network representation is translated into executable Modelica code, including graphics annotations (can be read by any Modelica authoring tool). A dynamic simulation with fixed nominal control signals is performed, to get a nominal operation conditions where the aggregation will be done (Loewen, 2001).
- Step 3: The original network is aggregated to a size suitable for optimization. The aggregation depth is flexible and certain consumers can be excluded from the aggregation.
- Step 4: The aggregated network is simulated to get initial trajectories for the dynamic optimization.
- Step 5: The dynamic optimization problem is solved.
- Step 6: The optimal trajectories are applied to the original network.

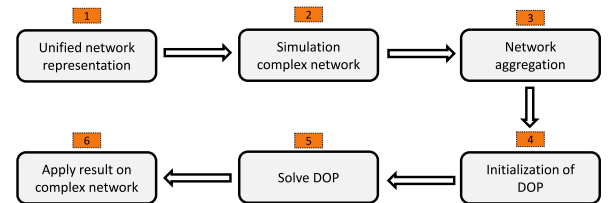


Figure 3. Schematic view of the framework showing the different steps for network creation, simulation and optimization.

4.2 Modeling

4.2.1 District heating network representation

The core of the framework is a unified network representation implemented in Python. In general, the representation is applicable for all kinds of network-based energy systems including district heating systems, gas systems and power systems. Typically such systems consist of edges and nodes. Edges could be transmission or distribution lines, gas pipes or pipes within a district heating system. Nodes could represent consumers and producers in any energy domain, storage or hybrid technologies. Such a unified network representation is required for two reasons. Firstly, an automatically generated simulation or optimization network model based on a unified network representation reduces the effort for modeling as well as the liability for errors. Secondly, several steps require detailed information of the network topology and other steps change the topology of the network. The unified network representation consists of three central modules: network representation, aggregation of the network and translation into executable simulation/optimization code. The first two modules are independent of the actual simulation and optimization language. The library is implemented in the Python module `networkX` (Hagberg et al., 2008) that is suitable for the creation and manipulation of complex networks. The network representation is in this paper combined with models from Modelon's Thermal Power Library suitable for dynamic simulation and optimization of district heating systems.

4.2.2 Aggregation method

Dynamic thermo-hydraulic optimization of large-scale district heating systems is very complex and numerically challenging. Several concepts approach the problem by simplifying (some) models (Olsthoorn et al., 2016; Orehounig et al., 2015), others by simplifying the network topology using aggregation methods (Larsen et al., 2004; Grosswindhager et al., 2012). Two methods have been developed in Denmark and Germany (Larsen et al., 2004); they are called "the Danish" and "the German" method. The idea behind the aggregation is (i) to change the tree structure of a network into a line structure and (ii) to remove short branches. The German method can handle network topologies with loops as well; this was the reason why we implemented the German method in our framework. Both methods were originally defined for steady state operation. The methods have different starting points: The German method conserves volume, mass flow and temperatures in all nodes. Thus, heat losses from the original and the aggregated networks are not exactly the same. The Danish method conserves heat losses. Thus, the node temperatures of the original and the aggregated networks are not exactly the same. Previous works on aggregation methods show that networks can be aggregated up to a very high level even in dynamic operations without losing significant accuracy (Loewen,

2001; Larsen et al., 2004).

4.2.3 Generation of Modelica models for simulation and optimization

The network representations are translated into Modelica models using Python functions. Based on the information in the network, corresponding Modelica code is generated, complete with annotations to enable visual inspection of the resulting model. In Figure 4, a generated network for the Graz network is visualized in the diagram view in Dymola. This method allows for the creation of complete models for simulation or optimization with components, connect statements and parameter values defined by the network model. Apart from the actual network, the generated Modelica models intended for optimization also contain input and output connectors, to handle the control signals and delay modeling in the optimization setup.

The components of the Modelica models which are used for optimization in this project come from Modelon's Thermal Power Library 1.14.

Both the complete district heating network described in Section 3.2 and aggregated versions of this are translated into Modelica models. The complete models are used for simulation, while the aggregated models are used for optimization and for creating initial trajectories for optimization. Different aggregation levels are evaluated in optimization, as explained in Section 5.1.

4.3 Optimization

The OCT toolchain that is used to solve the dynamic optimization problems starts by transferring the generated Modelica and Optimica code to CasADi Interface (Lennernäs, 2013), which has a flattened and symbolic representation of the model and optimization problem based on CasADi (Andersson, 2013). This representation is then propagated to the dynamic optimization algorithm implemented in JModelica.org (Magnusson and Åkesson, 2015). This algorithm implements direct collocation (Biegler, 2010) to transcribe the problem into a nonlinear program (NLP), which is then solved by IPOPT (Wächter and Biegler, 2006). CasADi is used to compute first- and second-order sparse derivatives using algorithmic differentiation (Griewank and Walther, 2008).

The dynamic optimization framework has recently been extended to treat delay differential-algebraic equations where the delay is fixed a priori, which is needed for the pipe models discussed in Section 3.1. Methods based on direct, local collocation are well-suited for handling such models (Betts et al., 2016).

4.3.1 Symbolic elimination

Before the model is transferred to CasADi Interface, the OCT compiler performs alias elimination, variability propagation and index reduction. The flattened, fully implicit differential-algebraic equation (DAE) is then transferred to CasADi Interface and later exposed to the direct

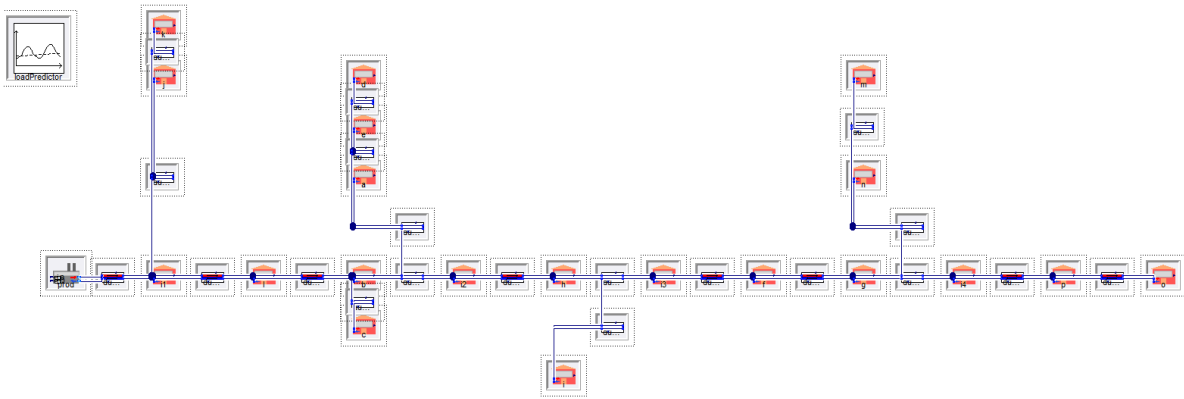


Figure 4. Modelica model of Graz network generated in Python from network description. The customers are represented by the orange red house icons, the pipe models are red and blue and the production unit is the gray model located furthest to the left.

collocation.

This approach leads to very large and sparse NLPs because of the multitude of algebraic variables in the network model. There has been recent work carried out (Magnusson and Åkesson, 2016) to address this problem in general by applying a block-lower triangular (BLT) transformation of the DAE to identify algebraic variables that only depend affinely on the corresponding block variables. This allows symbolic elimination of such variables by forward substitution. Further variables can be eliminated by applying tearing (Meijer, 2011; Baharev et al., 2016) to handle nonlinear dependencies. The majority of algebraic variables are thus eliminated prior to discretization by direct collocation, drastically reducing the size of the NLP. However, although the number of variables and equations are reduced, the resulting NLP Jacobian and Hessian tend to become more dense as a result, potentially crippling the performance of the sparse numerical linear algebra. A novel heuristic, similar to local minimum fill-in (Duff et al., 1986), is used to identify algebraic variables that should not be eliminated in order to preserve the sparsity of the NLP, typically leading to faster solution times.

In Modelica tools it is common to “eliminate” *all* algebraic variables by embedding Newton iterations in the right-hand side of an explicit ordinary differential equation, which is the foundation of FMI. In the spirit of simultaneous discretization, this approach is not used in OCT to avoid the long evaluation times that may result from solving implicit equations in each iteration and also the increased problem density resulting from elimination.

As demonstrated in (Magnusson and Åkesson, 2016), and as we will also see is the case in this work, the symbolic elimination not only reduces the solution time, but also improves convergence robustness, that is, probability of successfully solving an optimization problem in a timely manner.

5 Results

The production planning formulation described in Section 3.2 was solved for different optimization and model se-

tups. The goal is to understand the impact of the aggregation level on the production plans and of the symbolic elimination on the convergence and robustness of the optimization problem.

5.1 Optimization setups

The optimization problem was solved for three different aggregation levels resulting in two, five and seven customers. Very little difference in the optimal trajectories could be observed (data not shown). This indicates that aggregating the network to just two customers is sufficient to describe the current network with good accuracy.

The convergence of the optimization algorithm was also analyzed in detail for each aggregation level, to investigate the scalability of the current approach. All optimization cases were run on a laptop with 8 GB RAM and four 2.6 GHz CPUs, with convergence results and optimization model statistics displayed in Tables 1 and 2. The results show that the main benefit of the elimination occurs for larger network models, when both the time per iteration and the number of iterations is significantly reduced, resulting in a much better overall performance. The total time for running the entire script is reduced by more than a factor 2 and the optimization convergence is also significantly more robust, as indicated by the number of iterations and by manual inspection of the output from IPOPT. For fewer customers, the comparison between the two methods give less clear results. The overall time for running the script is approximately the same, as is the robustness of the convergence. The reason for the similar performance is that the time gained by eliminating variables is lost from the extra time needed to perform the elimination.

5.2 Optimal trajectories

The optimized trajectories are studied for the aggregated system with five customers. The network aggregation is displayed in Figure 5, and the trajectories are displayed in Figure 6. It is clear from the temperature plot that the heat loss and transport delay are correctly captured in the optimization model: the customer at the network periphery

Table 1. Optimization model statistics.

Nbr of customers	2	5	7
Nbr of states	12	22	30
Nbr of algebraics	136	298	413
Nbr of algebraics a.e	17	45	66
Nbr of variables in NLP	61354	130412	150159
Nbr of variables a.e	18395	39079	45712

Table 2. Optimization convergence results.

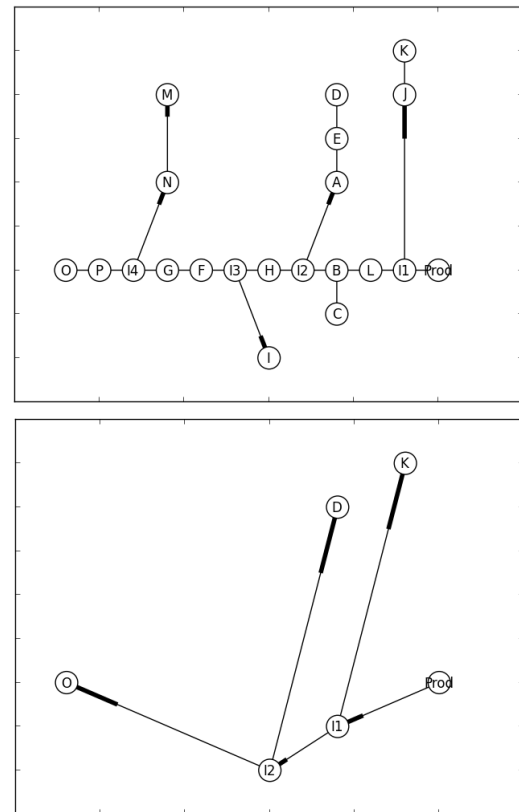
Nbr of customers	2	5	7
Without symbolic elimination			
Nbr of iterations	29	42	88
IPOPT CPU time [s]	13	163	1103
NLP function eval time [s]	24	60	161
IPOPT total time [s]	103	422	1516
Script total time [s]	120	448	1548
With symbolic elimination			
Nbr of customers	2	5	7
Nbr of iterations	31	43	44
IPOPT CPU time [s]	7	31	60
NLP function eval time [s]	25	68	71
IPOPT total time [s]	75	240	314
Script total time [s]	121	431	685

received a slightly colder water and with some delay. It is also visible that the optimization minimizes pump cost, i.e. discharge pressure at the producer, while respecting the differential pressure constraint across the customers' valve: the customer O, furthest away from the producer has its dp constraint active most of the time. During high load, the distribution pump of the producer is at its maximum capacity and the mass rate saturates. As a consequence the supply temperature is increased to fulfill the heat demand of all customers. The temperature increase is done in advance to compensate for the mass flow dependent delays in the network.

Another interesting phenomenon can be seen when customer O, far away from the producer is operating at maximum valve opening, at about $t=2.5h$ and $9.5h$. Figure 7 displays this phenomenon around the first load peak. It shows that the increase in the supply temperature at the producer propagates with the flow in the network and results in valve closing at the customers close to the producer, in the figure illustrated with customer I1, which is closest to the producer. This shifts the mass flow rate from the close customers to customer O that gets its higher load fulfilled. The increase in the producer's supply temperature propagates quicker than the speed of the hot water.

5.3 Verification in simulation

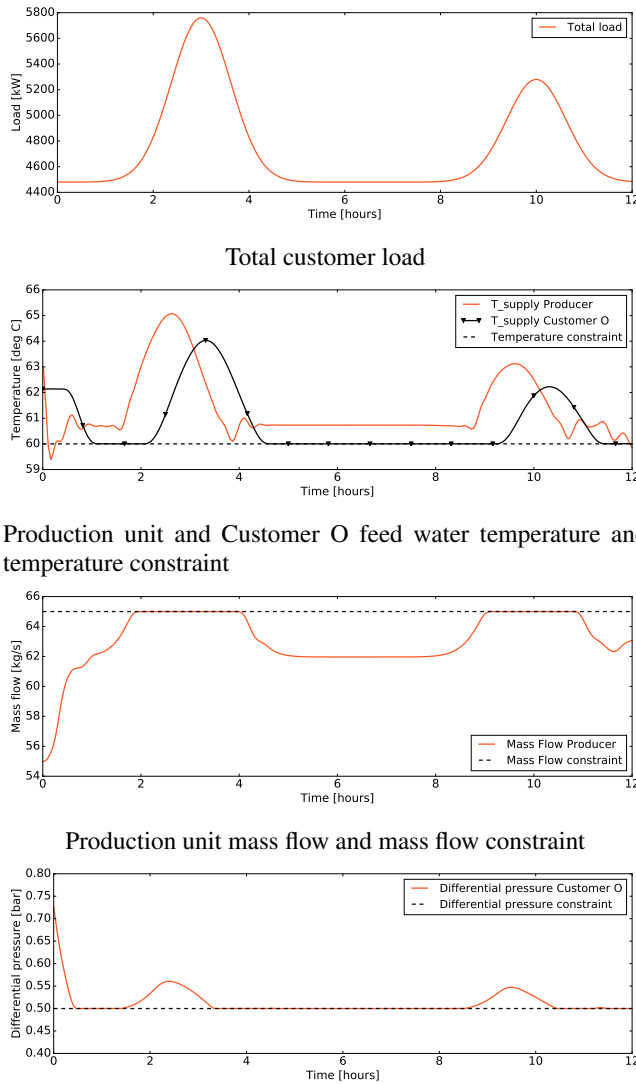
The previous section demonstrates that the optimization method is able to generate optimal trajectories for temperature and pressure that fulfill the operational constraints from the customers and the distribution network. The net-


Figure 5. Complete and aggregated network models.

work model used for optimization differs however from the original one as it has been simplified by the aggregation method described in Section 4.2.2. The idea is now to validate the optimization results and the aggregation method by applying the optimal trajectories on the complex model with 16 customers. As the pressure profile would not be applied in reality, the supply pressure at the production unit is instead manipulated by a controller that maintains a minimum pressure difference over all customers. Only the supply temperature trajectory is applied to the complex network model. The results are shown in Figure 8. The supply temperature at the customer furthest away from the plant is very similar when the optimization and simulation results are compared. The mass flowrate computed by the differential controller is also very similar to the optimized trajectory. Some differences in the differential pressure can be seen as the optimization does not always operate at the minimum value but sometimes at a higher level to minimize the overall cost. The results indicate in general that the aggregation to two customers is good enough for this 16 customers network model.

6 Discussion

This paper presents new features of Modelon's Thermal Power Library 1.14 in the field of dynamic optimization of district heating systems as well as the impact of the new algorithm for symbolic elimination available in Optimica Compiler Toolkit. The new features together with



Customer O differential pressure and differential pressure constraint

Figure 6. Optimal trajectories for an aggregated network model with five customers.

a unified representation of network-based energy systems make it possible to analyze, simulate and optimize small and larger district heating or cooling systems. It is also possible to include physical constraints based on operational limitations into the optimization formulation. Based on the results it can be concluded that the aggregation method achieves accurate results at an aggregation depth of about 90 %. Furthermore it can be concluded that the main benefit of the elimination occurs for larger models where the computation time could be reduced by more than a factor 2. Enabling the elimination yields an overall computation time for seven remaining customers of about 11 minutes and a solution time of about 5 minutes. In the context of model predictive control the solution time is sufficiently low for a real-time application and it could be further reduced by initializing the optimization with the results of the latest iteration. In an offline opti-

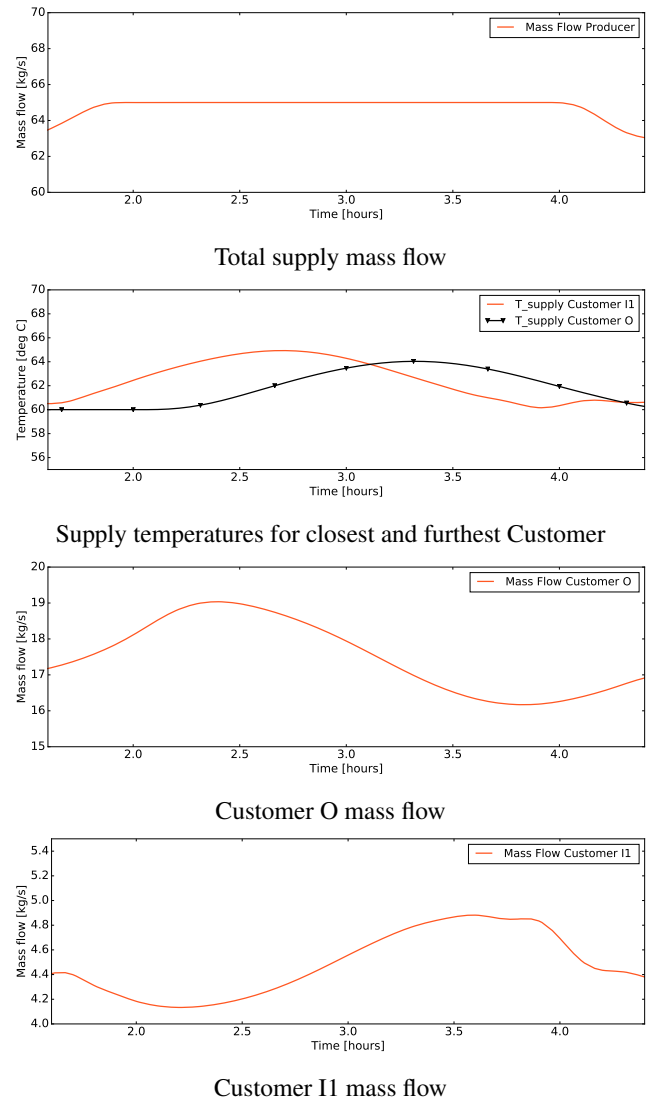


Figure 7. Heat to customer O based on mass flow change for remaining customers.

mization context, the overall computation time could also be reduced by using the optimization results for lower aggregation levels as initial guesses for the optimization of more complex networks. The next stage of our research will include scale-up studies and the integration of the unit commitment problem in the overall framework.

7 Acknowledgements

Fredrik Magnusson acknowledges support from the LCCC Linnaeus Center and eLLIIT Excellence Center at Lund University. Gerald Schweiger acknowledges the Austrian Federal Ministry of Science, Research and Economics for funding the project "FlexEnergySys (848346)". Modelon AB acknowledges support from PiiA – Processindustriell IT och Automation.

References

Johan Åkesson. Optimica—an extension of Modelica supporting dynamic optimization. In *Proceedings of the 6th Interna-*

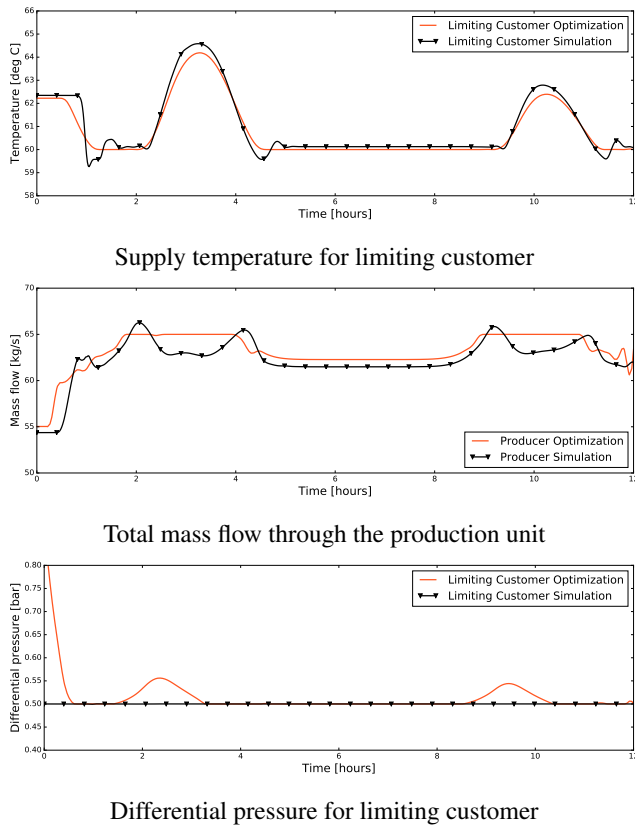


Figure 8. Comparison between optimization results for two customers and simulation results for the complete network with optimal inputs.

tional Modelica Conference, pages 57–66, 2008.

Johan Åkesson, Karl-Erik Årzén, Magnus Gäfvert, Tove Bergdahl, and Hubertus Tummescheit. Modeling and optimization with Optimica and JModelica.org—languages and tools for solving large-scale dynamic optimization problems. *Computers & Chemical Engineering*, 34:1737–1749, 2010.

Jonas Allegrini, Kristina Orehounig, Georgios Mavromatidis, Florian Ruesch, Viktor Dorer, and Ralph Evins. A review of modelling approaches and tools for the simulation of district-scale energy systems. *Renewable and Sustainable Energy Reviews*, 52:1391–1404, 2015. URL <http://dx.doi.org/10.1016/j.rser.2015.07.123>.

Joel Andersson. *A General-Purpose Software Framework for Dynamic Optimization*. Ph.D. thesis, Arenberg Doctoral School, KU Leuven, Belgium, 2013.

Ali Baharev, Hermann Schichl, and Arnold Neumaier. Decomposition methods for solving sparse nonlinear systems of equations. Submitted for publication. Available online: http://reliablecomputing.eu/baharev_tearing_survey.pdf, 2016.

John T. Betts, Stephen L. Campbell, and Karmethia C. Thompson. Solving optimal control problems with control delays using direct transcription. *Applied Numerical Mathematics*, 108:185–203, 2016.

Lorenz T. Biegler. *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. MOS-SIAM, Philadelphia, PA, 2010.

Iain S. Duff, Albert. Erisman, and John K. Reid. *Direct Methods for Sparse Matrices*. Clarendon Press, Oxford, United Kingdom, 1986.

Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia, PA, 2nd edition, 2008.

Stefan Grosswindhager, Andreas Voigt, Martin Kozek, and A Varying-coefficient Models. Predictive Control of District Heating Network using Fuzzy DMC. In *International Conference on Modelling, Identification and Control*, 2012.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. *Proceedings of the 7th Python in Science Conference (SciPy 2008)*, (SciPy):11–15, 2008.

Helge V Larsen, Benny Bøhm, and Michael Wigbels. A comparison of aggregated models for simulation and operational optimisation of district heating networks. *Energy Conversion and Management*, 45:1119–1139, 2004.

Björn Lennernäs. A CasADi based toolchain for JModelica.org. M.Sc. thesis, Department of Automatic Control, Lund University, Sweden, 2013.

Achim Loewen. *Entwicklung eines Verfahrens zur Aggregation komplexer Fernwärmenetze*. Ph.D. thesis, Fraunhofer UMSICHT, Germany, 2001.

Henrik Lund, Sven Werner, Robin Wiltshire, Svend Svendsen, Jan Eric Thorsen, Frede Hvelplund, and Brian Vad Mathiesen. 4th Generation District Heating (4GDH): Integrating smart thermal grids into future sustainable energy systems. *Energy*, 68:1–11, 2014. URL <http://www.sciencedirect.com/science/article/pii/S0360544214002369>.

Fredrik Magnusson and Johan Åkesson. Dynamic optimization in JModelica.org. *Processes*, 3(2):471–496, 2015.

Fredrik Magnusson and Johan Åkesson. Symbolic elimination in dynamic optimization based on block-triangular ordering. *Optimization Methods and Software*, 2016. Accepted for publication.

Patrik Meijer. Tearing differential algebraic equations. M.Sc. thesis, Centre for Mathematical Sciences, Lund University, Sweden, 2011.

Modelica Association. Modelica® - A Unified Object-Oriented Language for Systems Modeling Language Specification Version 3.3 Revision 1. 2014. URL <https://www.modelica.org/documents/ModelicaSpec33Revision1.pdf>.

Modelon. OPTIMICA Compiler Toolkit, 2016. URL <http://www.modelon.com/products/optimica-compiler-toolkit/>.

- Dave Olsthoorn, Fariborz Haghighat, and Parham A Mirzaei. Integration of storage and renewable energy into district heating systems : A review of modelling and optimization. *Solar Energy*, 136:49–64, 2016. URL <http://dx.doi.org/10.1016/j.solener.2016.06.054>.
- Kristina Orehounig, Ralph Evins, and Viktor Dorer. Integration of decentralized energy systems in neighbourhoods using the energy hub approach. *Applied Energy*, 154:277–289, 2015. URL <http://dx.doi.org/10.1016/j.apenergy.2015.04.114>.
- Jonatan Rantzer. Robust production planning for district heating networks. M.Sc. thesis, Centre for Mathematical Sciences, Lund University, Sweden, 2015.
- Håkan Runvik, Per-Ola Larsson, Stéphane Velut, Jonas Funquist, Markus Bohlin, Andreas Nilsson, and Sara Modarrez Razavi. Production Planning for Distributed District Heating Networks with JModelica.org. In *11th International Modelica Conference*, pages 217–223, 2015.
- Gerald Schweiger, Per-Ola Larsson, Fredrik Magnusson, Patrick Lauenburg, and Stéphane Velut. District heating and cooling systems – framework for modelica-based simulation and dynamic optimization. *Energy*, 2017a. ISSN 0360-5442. doi:<https://doi.org/10.1016/j.energy.2017.05.115>. URL <http://www.sciencedirect.com/science/article/pii/S0360544217308691>.
- Gerald Schweiger, Jonatan Rantzer, Karin Ericsson, and Patrick Lauenburg. The potential of power-to-heat in swedish district heating systems. *Energy*, 2017b. ISSN 0360-5442. doi:<http://dx.doi.org/10.1016/j.energy.2017.02.075>. URL <http://www.sciencedirect.com/science/article/pii/S0360544217302499>.
- Stéphane Velut, Per-Ola Larsson, Johan Windahl, Linn Saarinen, and Katarina Boman. Short-term production planning for district heating networks with JModelica.org. In *Proceedings of the 10th International Modelica Conference*, pages 959–968, 2014.
- Andreas Wächter and Lorenz T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

Optimal Control of District Heating Systems using Dynamic Simulation and Mixed Integer Linear Programming

Loïc Giraud Massinissa Merabet Roland Baviere Mathieu Vallée

Univ. Grenoble Alpes, INES, F-73375 Le Bourget du Lac, France
CEA, LITEN, 17, Rue des Martyrs, F-38054 Grenoble, France, roland.baviere@cea.fr

Abstract

This paper presents the development of a new advanced control method suitable for variable temperature District Heating Systems (DHS). The proposed controller determines optimal planning for the on/off status and power of the heat generators as well as for the supply temperature and differential pressure at the production plant level. Compared to existing methods, the original features of the developed solution are to fully exploit the thermal storage capacity of the network and to determine the best compromise between pumping costs and heat losses. A numerical case study based on a representative DHS is used to evaluate the method over a heating season (5 months). Results show that our method reduces production costs up to 8.3 % when compared to a more classical controller. Moreover, the observed computing time is compatible with the requirements of the receding time horizon principle, ensuring that the method is tractable on real DHS.

Keywords: *District Heating, Model Predictive Control, Dynamic Simulation, Mixed Integer Linear Programming*

1 Introduction

Nowadays, many research works devoted to District Heating Systems (DHS) are performed on low and very low temperature systems, mainly because of their energy performance and their ability to use renewable energy sources. However, High Temperature District Heating Systems (HTDHS) represent an important share of the existent systems in Europe. For instance, systems with temperature over 110 °C account for more than 50 % of the heat delivered by French DHS (SNCU, 2013).

The energy load of HTDHS is generally supplied by numerous generators and fuels. On the other hand, their distribution network usually bears large variations in temperature and differential pressure. Thus, HTDHS are affected by non-constant production costs yet they natively comprise important thermal storage capacities, i.e. network storage, if an adequate supply temperature control is used. Additionally, HTDHS are subject to complex dynamic behaviors originating both from the variability of the demand and the significant temperature transportation delay. Finally, heat can be

delivered to the consumers using various combinations of temperature and mass flow rate. Lowering network temperature would limit the thermal losses; however, the mass flow rate shall increase in order to maintain the same heat flow, and this will cause pumping work to rise. Contrary to what is generally considered, in many practical situations, particularly recurrent in HTDHS, the optimal balance between pumping work and heat losses may be obtained with high supply temperature and low differential pressure.

As pointed in (Lund, 2014), the intelligent control for optimal operation is a future challenge for the improvement of DHS. Due to its complexity and high parameters combinatorics, the determination of optimal production and distribution plans in DHS is difficult, if not impossible, when solely based on empirical laws and/or expert judgement. In this context, decision support/making tools relying on a Model Predictive Control (MPC) scheme are very useful. Despite significant progress, there is still an important room for improvement in this domain.

This paper focuses on the optimal control of pressurized water DHS. For this application, we have developed and tested an algorithm that optimizes, given a load prediction, the use of production means as well as supply temperature and differential pressure. Compared to existing methods, the original features of the developed solution are, firstly, to fully exploit the thermal storage capacity of the network. Secondly, our controller is suitable for determining the best possible combination between electrical costs for pumping and heat production costs compensating distribution losses. Though generic, the proposed control method is particularly adapted to existing HTDHS.

Our controller is based on a MPC scheme. At each time step, a dynamic non-linear model of the distribution network is simulated over a finite time-horizon. In the present work, this model is built using the equation-based object-oriented language Modelica along with the simulation platform Dymola and an in-house component model library named *DistrictHeating* (Giraud b), 2015). The simulation results are then used to formulate a linearized model of the distribution network. Combined with other linear constraints representing the technical limitations of the different

pieces of equipment and with a linear cost function, the model forms a Mixed Integer Linear Program (MILP). In a last step, this program is solved yielding the optimal trajectories for the various control variables of the problem.

In section 2, we present a literature review on advanced control for DHS. Section 3 firstly describes the algorithmic aspects of the proposed controller, secondly presents the non-linear network model used for dynamic simulation and finally details the formulation of the linear optimization problem. A case study consisting of a virtual yet representative HTDHS is then described in section 4. In section 5, the simulation results obtained for various controllers over a heating season are presented and discussed. Section 6 includes the conclusions and perspectives of our study.

2 Existing DHS Control Methods

The approach currently used to determine the control variables of DHS (e.g. supply temperature, differential pressure, load distribution between different heat production units ...) often relies on heuristic methods, i.e. a formalization of common sense, simple logic or expert judgement. As an example, we can recall here the determination of supply temperature using a single or even multi-variable heating curve. Though simple to implement and robust with respect to production or demand uncertainties, the efficiency of such control methods is always limited when applied to a system comprising several sources, variable energy purchase prices and energy storage capacity. This is partly due to the fact that anticipative control strategies are difficult if not impossible to formulate in this framework. Another difficulty is that production goals may be multiple and conflicting (e.g. power and heat generation in combined heat-and-power units).

To overcome the aforementioned difficulties, an MPC scheme is an interesting alternative. The MPC approach consists of a load prediction module and an optimization procedure used to determine the best possible path for control variables, i.e. the one minimizing an objective function while meeting different technical and operational constraints. Depending on the formulation of the quantitative objective to minimize, operating costs and/or CO₂ content of the delivered heat may be significantly improved.

However, building an MPC scheme to control a DHS is a complex task. This explains why most previous works done on this subject only consider some parts of the problem. On the one hand, numerous studies deal with production optimization only, i.e. they address the unit commitment and load dispatch problems. In (Eriksson, 1994), the author determines the heat power planning for each production unit considering starting costs, minimal load of each generator and bounds for heat power ramp rates. This approach is mostly used in

studies interested in combined heat and power plants since electricity must be produced when it is the most profitable. On the other hand, several works only consider the supply temperature determination. Important features here are to reduce heat losses and to use the network capacity for heat storage. For instance, the supply temperature is optimized in (Nielsen, 2005) using the Finite Impulse Response method to linearize a dynamic distribution network model and then to solve the linear optimization problem. Few works study both the load dispatch problem and the supply temperature determination. The integer variables, representing for instance the heat generators' statuses, are then not taken into account in these cases in order to reduce the combinatory.

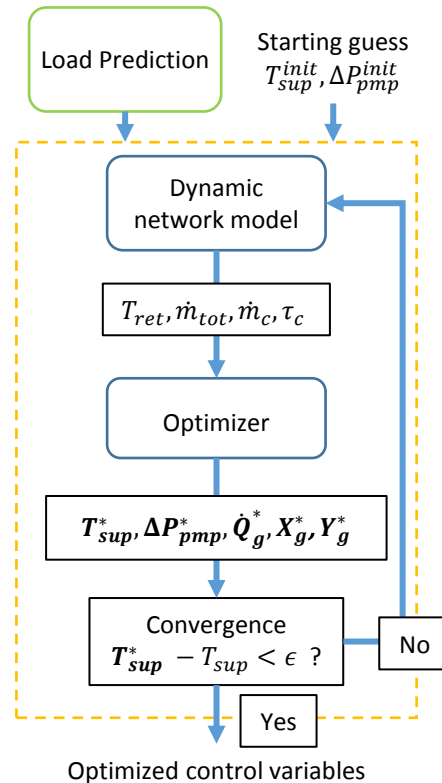


Figure 1: Proposed optimal control algorithm for DHS.

More recently, both the supply temperature and heat power planning have been determined in order to minimize the production costs yet without considering the time delays in the network (Fang, 2015). Another possible approach, quite popular in the Modelica community, consists in modeling, formulating and solving a dynamic optimization problem using the JModelica.org tools (Akesson, 2011). Following this method, (Runvik, 2015) also solve a short-term production planning problem for a DHS using a two-steps optimization procedure including production and distribution variables. However, due to prohibitive computational costs, the network representation only includes three customers.

In conclusion of this review, no method has been identified in the literature that adequately optimizes the production and the distribution considering network storage for DHS. In the following, we present such a method and study its benefits on a representative HTDH.

3 Optimal Control of DHS

In this section, we describe the optimal control method we designed. The objective is to determine the optimal trajectory for the control variables over a defined anticipation horizon. Taking into account the relatively slow dynamics of a DHS requires anticipation, i.e. using load predictions and optimizing of control variables so that the system produces the desired effects in the future. Additionally, periodically revising the optimization is mandatory in order to cope with load prediction uncertainties.

To address these challenges, we combine a dynamic nonlinear model of the DHS with linear optimization methods, similarly to other works like those of (Benonysson, 1991) and (Sandou, 2006). For each anticipation horizon, our system encompasses both production and distribution optimization and controls heat powers, generators' statuses, supply temperature and differential pressure. To guarantee the applicability of the control trajectories, the optimization is constrained by the technical limits of the DHS's pieces of equipment.

This section first gives a general description of the algorithm, then details the dynamic nonlinear model, which we illustrate more specifically in the case study. It ends with a description of the linear optimization problem's formulation.

3.1 General Description

Figure 1 depicts the proposed algorithm. The dynamic nonlinear network model is first simulated using initial guesses for the distribution control variables, namely the supply temperature and differential pressure at production plants. We then extract relevant input data for the optimization problem and we formulate a linear relaxation of the optimization problem using the MILP formalism. The optimizer finally computes a new set of control variables. Iterations between the dynamic network model and the MILP optimizer are conducted until convergence is reached, using a criteria defined by a threshold on the supply temperature increment.

We then periodically revise the optimization using the receding time horizon. At time t , the optimization procedure is performed for the predictive horizon $t+N_t$ yet only the first output values for time slot $[t:t+Pr]$ are applied to the system. At time $t+Pr$, the calculation is repeated for the optimization horizon $t+Pr+N_t$. This

algorithm is illustrated in Figure 2 in a situation where the receding horizon Pr is chosen equal to 1.

3.2 The dynamic distribution network model

The proposed algorithm requires the simulation of a dynamic model representing the distribution network.

Figure 3 depicts the mesh-free layout of the distribution network considered in the case study. Since we study our control strategy on a mid-scale DHS, our dynamic network model is based on a detailed physical representation of the system by gathering component models that we previously developed and validated. The components models are taken from an in-house Modelica library named *DistrictHeating* and presented in (Giraud b), 2015). The heat generators are represented considering equivalent heat and momentum sources. Moreover, the model enables the control of supply temperature and differential pressure at the production plant level. The distribution pipe model that we use is based on the method of characteristics, also called node method in the specific DHS related literature see (Benonysson, 1991). This model accounts for heat propagation delays, heat losses, tube thermal inertia and pressure losses. Concerning the substation representation, we use an explicit model comprising a

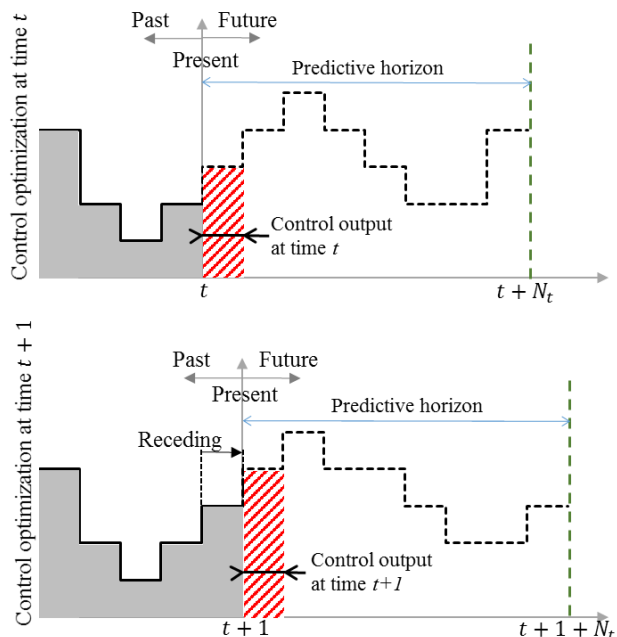


Figure 2: Receding horizon principle

heat exchanger, a regulation valve and an ideal controller (see (Giraud a), 2015) for details).

3.3 Formulation of the Linear Optimization Problem

The linear optimization problem is an approximation, or relaxation, of the complete, strongly nonlinear problem that would take into account all the physical and technical constraints of the system. However,

formulating the linear problem using appropriate assumptions yields usable results in a limited computing time. This section details the formulation of the linear problem we adopted, which includes the MILP variables, the cost function to minimize, the technical limits of the DHS, the critical curve describing consumer's constraints and a linearized model of the distribution network.

3.3.1 MILP variables

The optimization variables of the problem are identified by an asterisk and typed in bold to better readability. All of them are time discretized and considered constant over one time step.

Continuous variables

$\dot{Q}_g^*(t)$: Represents the heat power produced by each generator g at the time t .

$T_{sup}^*(t)$: Represents the supply temperature in the network at the time t .

$m_{tot}^*(t)$: Represents the total mass flow rate at the production plant at the time t .

$\Delta P_{pmp}^*(t)$: Represents the differential pressure at the production plant at the time t .

$W_{pmp}^*(t)$: Represents the pump work at the time t .

Binary variables

$Y_g^*(t)$: Represents the on/off status of each generator, i.e. it is equal to one when generator g is on and to zero otherwise.

$X_g^*(t)$: Identifies the timing of each generator start-up, i.e. it is equal to one only when Y_g^* switches from 0 to 1 and to zero otherwise.

3.3.2 The cost function

The function to be minimized, presented in expression (1), reflects the integral of operational costs over a finite time-horizon. The first term in equation (1) represents the fuel consumption and it is thus proportional to the

C_g^{th} parameters standing for fuel prices. The second term accounts for specific costs linked to generator start-up and they are therefore proportional to fixed monetary amounts denoted $C_g^{off/on}$. The last term accounts for electricity consumption due to pump's operation. It is therefore proportional to a time variable electricity purchase price hereafter denoted $C^{el}(t)$. Pumping and heat generation efficiencies are accounted for respectively using the η_{pmp} and η_g constant parameters.

$$\sum_{t..t+Pr} \left(\sum_g \left(\frac{C_g^{th}}{\eta_g} \cdot \dot{Q}_g^*(t) \cdot dt + C_g^{off/on} \cdot X_g^*(t) \right) + \frac{C^{el}(t)}{\eta_{pmp}} \cdot W_{pmp}^*(t) \cdot dt \right) \quad (1)$$

To guarantee the applicability of its outcomes, the minimization calculation must be performed in the presence of linear constraints on the optimization variables. Inequality constraints will be presented first. In a second step we present the equality constraints accounting for the mass, energy and momentum balance equations governing the relations between the operating variables.

3.3.3 Inequality constraints

Several continuous variable are considered with lower and upper bounds representing physical limitations of DHS components as presented in inequalities (2), (3) and (4). Each parameter noted with a *min* or *max* superscript is a known and fixed parameter of the problem.

$$T_{sup}^{min} \leq T_{sup}^*(t) \leq T_{sup}^{max} \quad (2)$$

$$\Delta P_{pmp}^{min} \leq \Delta P_{pmp}^*(t) \leq \Delta P_{pmp}^{max} \quad (3)$$

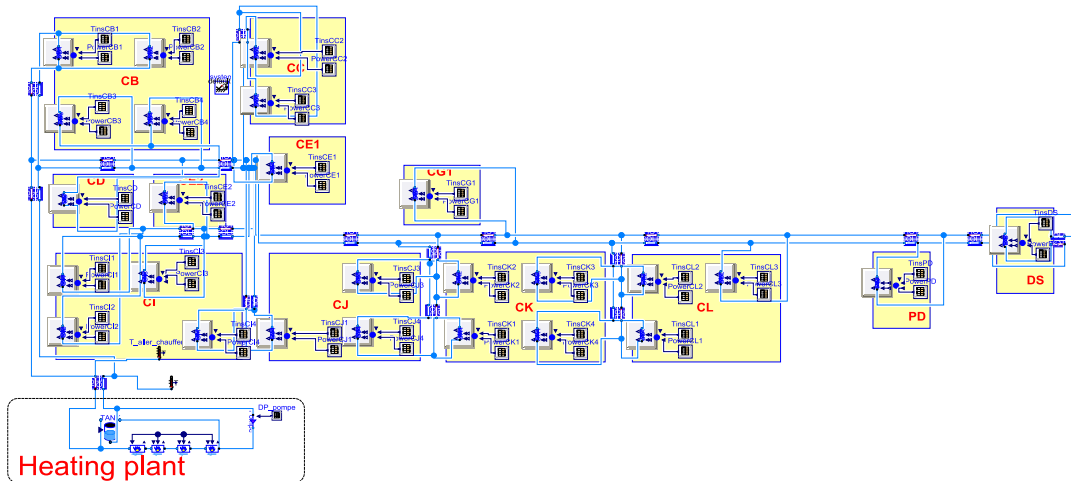


Figure 3: Layout of the distribution network case study in Modelica/Dymola.

$$\dot{m}_{tot}^{min} \leq \dot{m}_{tot}^*(t) \leq \dot{m}_{tot}^{max} \quad (4)$$

To limit thermal fatigue, we bound supply temperature and heat power variations with expressions

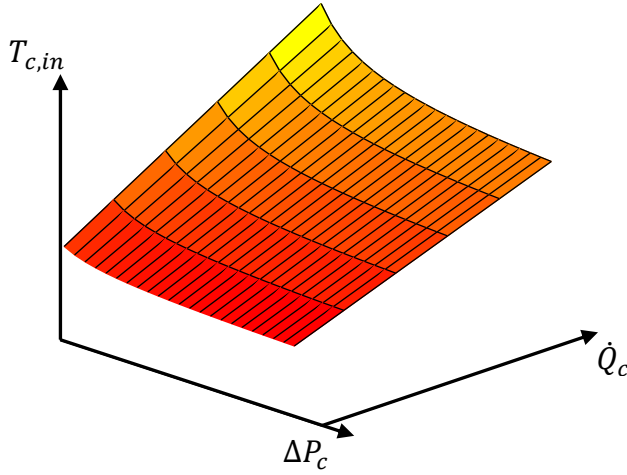


Figure 4: Consumer critical temperature as a function of the differential pressure and heat demand (left) – Cut-plane for a fixed heat demand (right).

of the following type, written for a sample variable \mathbf{Z}^* :

$$-\Delta Z^{max} \leq \mathbf{Z}^*(t) - \mathbf{Z}^*(t-1) \leq \Delta Z^{max} \quad (5)$$

Inequalities (6) are used to confine \dot{Q}_g^* between \dot{Q}_g^{min} and \dot{Q}_g^{max} when generator g is started. Otherwise, \dot{Q}_g^* is set to 0.

$$\mathbf{Y}_g^*(t) \cdot \dot{Q}_g^{min} \leq \dot{Q}_g^*(t) \leq \mathbf{Y}_g^*(t) \cdot \dot{Q}_g^{max} \quad (6)$$

Finally, inequalities (7) are considered to define the timing of each generator start-up, i.e. variable \dot{X}_g^* .

$$\mathbf{Y}_g^*(t) - \mathbf{Y}_g^*(t-1) \leq \dot{X}_g^*(t) \leq \mathbf{Y}_g^*(t) \quad (7)$$

3.3.4 Critical conditions to supply heat demand

Supplying the requested heat demand to a DHS customer is only possible when the local network temperature exceeds a threshold called the critical temperature and hereafter denoted $T_{c,in}^{crit}(t)$. The present section firstly discusses how to derive the formula used to evaluate $T_{c,in}^{crit}(t)$ and secondly presents the linear inequality constraints necessary in the MILP problem to guarantee that heat demand is fulfilled.

We consider in this study that the substations are of the indirectly connected type and that they are composed of one counter-flow heat-exchanger, free of any by-pass, and a primary control valve used to regulate the building heating system temperature at a requested level. For such system, as long as the consumer heat demand \dot{Q}_c is fulfilled, a static energy balance applied on the primary side of the heat exchanger yields the mathematical

expression (8) relating \dot{Q}_c , the primary mass flow rate \dot{m}_c , the primary inlet and outlet temperatures ($T_{c,in}$ and $T_{c,out}$) and the fluid specific heat capacity Cp .

$$\dot{m}_c(t) = \frac{\dot{Q}_c(t)}{Cp \cdot (T_{c,in}(t) - T_{c,out}(t))} \quad (8)$$

On the other hand, \dot{m}_c cannot exceed the value reached when the primary control valve is fully open. Such maximal value, denoted $\dot{m}_c^{max}(t)$, can be calculated assuming a quadratic dependency between the local differential pressure of the network, namely ΔP_c , and the mass flow rate. This is shown in equation (9) used for valve modeling. In this equation, \dot{m}_c^{nom} and ΔP_c^{nom} are nominal values of the primary mass flow rate and the differential pressure.

$$\dot{m}_c^{max}(t) = \dot{m}_c^{nom} \cdot \sqrt{\frac{\Delta P_c(t)}{\Delta P_c^{nom}}} \quad (9)$$

The critical temperature is obtained by assuming that the demand is fulfilled while \dot{m}_c equals \dot{m}_c^{max} . Thus, combining equations (8) and (9) leads to expression (10) for the critical temperature:

$$T_{c,in}^{crit}(t) = T_{c,out}(t) + \frac{\dot{Q}_c(t)}{Cp \cdot \dot{m}_c^{nom} \cdot \sqrt{\frac{\Delta P_c(t)}{\Delta P_c^{nom}}}} \quad (10)$$

At this stage, it is worth mentioning that an additional heat-exchanger model is requested to evaluate formula (10) since the primary outlet temperature, namely $T_{c,out}(t)$, has not been determined yet. In our study, the Logarithmic Mean Temperature Difference method is used for this purpose.

As a consequence, heat demand of consumer c will only be satisfied if local network conditions in terms of temperature and differential pressure are above critical values shown in Figure 4, i.e. if inequality (11) is verified:

$$T_{c,in}(t) \geq T_{c,out}(t) + \frac{\dot{Q}_c(t)}{Cp \cdot \dot{m}_c^{nom} \cdot \sqrt{\frac{\Delta P_c(t)}{\Delta P_c^{nom}}}} \quad (11)$$

This last expression is adapted to our problem using piecewise linear approximations compatible with the MILP formalism:

$$T_{c,in}(t) \geq flin_{c,l}(\Delta P_c(t)) \quad l = 1, \dots, L \quad (12)$$

In expression (12), the $flin_{c,l}$ functions are a set of L linear functions approximating the critical temperature calculated from the right hand-side of expression (11).

Bearing in mind that the algorithm aims at producing optimal planning for the supply temperature and the differential pressure, we then consider linear relations between the variables at the consumers' sides and those at the production plant level where the control variables are applied. The heat propagation equation (13), detailed in (Benonysson, 1991), is used to express the consumers inlet temperature $T_{c,in}$ as a function of the supply

temperature T_{sup}^* . It considers a propagation time delay τ_c and heat losses using the $\tau_{therm,c}$ thermal time constant and the T_{ext} parameter representing the ambient temperature surrounding the distribution pipes.

$$T_{c,in}(t) = T_{ext} + \left(T_{sup}^* (t - \tau_c(t)) - T_{ext} \right) \cdot e^{-\frac{\tau_c(t)}{\tau_{therm,c}}} \quad (13)$$

At this point, one can note that temperature propagation delay τ_c introduces nonlinearities into the optimization problem. Thus, propagation delays τ_c are not handled directly in the MILP problem but are provided by the dynamic model presented in section 3.2.

For the differential pressure losses, a linearized relation, well verified on the Grenoble DHS, is considered as shown in expression (14):

$$\Delta P_c(t) = \Delta P_{pmp}^*(t) - K_c \cdot \dot{m}_{tot}^*(t) \quad (14)$$

Expression (12) is then re-written using (13) and (14) to introduce the T_{sup}^* , \dot{m}_{tot}^* and ΔP_{pmp}^* optimization variables. This yields the linear inequality constraints (15) that are used in our MILP problem to guarantee that consumer satisfaction is not sacrificed.

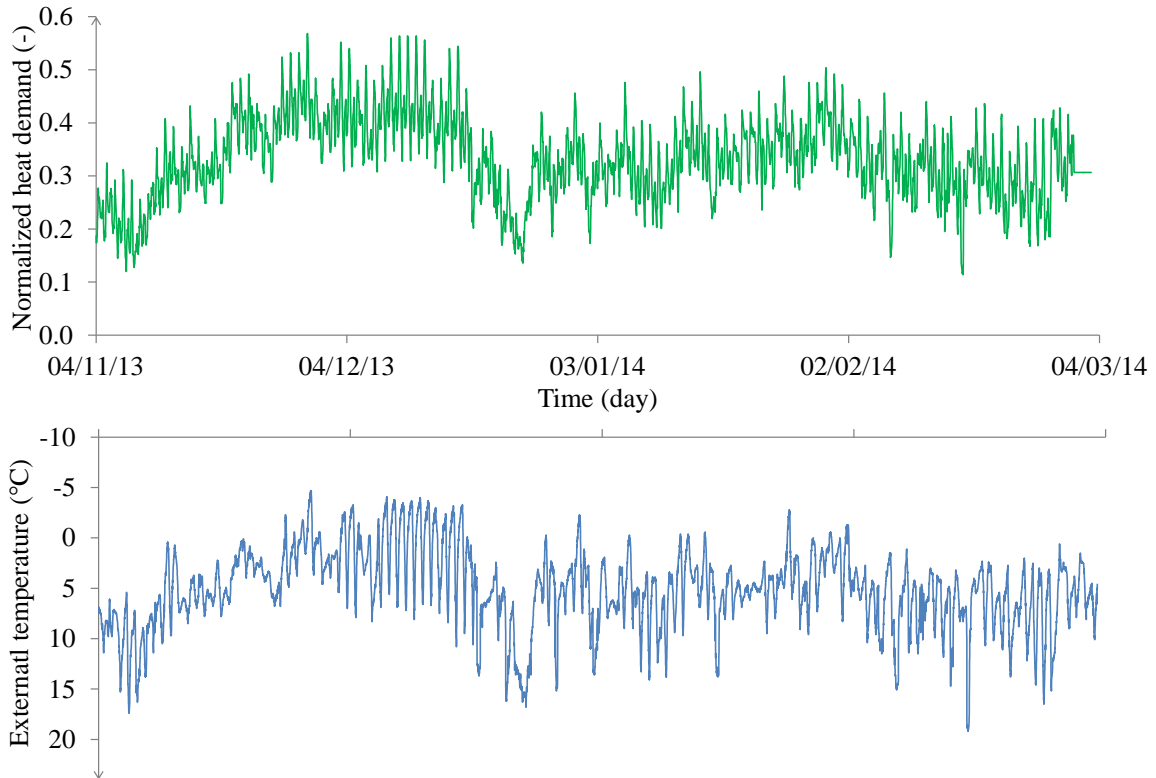


Figure 5: Evolutions over the heating season of the normalized heat demand (top) and external temperature (bottom).

$$\begin{aligned}
T_{ext} + \left(T_{sup}^* (t - \tau_c(t)) - T_{ext} \right) \cdot e^{-\frac{\tau_c(t)}{\tau_{therm,c}}} \\
\geq \text{flin}_{c,l} \left(\Delta P_{pmp}^*(t) - K_c \right. \\
\left. \cdot \dot{m}_{tot}^*(t) \right) \quad l = 1, \dots, L
\end{aligned} \quad (15)$$

3.3.5 The linearized distribution network model

The relations between the optimization variables accounting for distribution network mass, momentum and energy conservation laws are considered thanks to a set of linearized equality constraints that are presented in this section.

First, the pumping work \dot{W}_{pmp}^* appearing in the cost function (1) is expressed linearly using a first order Taylor expansion limited to differential pressure and mass flow rate variations. The fluid density ρ are therefore assumed constant, which leads to expression (16):

$$\begin{aligned}
\dot{W}_{pmp}^*(t) \\
= \frac{\dot{m}_{tot}(t) \cdot \Delta P_{pmp}^*(t) + \dot{m}_{tot}^*(t) \cdot \Delta P_{pmp}(t)}{\rho} \\
- \frac{\dot{m}_{tot}(t) \cdot \Delta P_{pmp}(t)}{\rho}
\end{aligned} \quad (16)$$

In this last expression, $\Delta P_{pmp}(t)$, $\dot{m}_{tot}(t)$ and η_{pmp} are provided by the dynamic simulation model described in section 3.2.

Second, a linearized version of the network energy balance is obtained. This point is crucial if one wants to benefit from the possibility to store heat directly in the distribution network. By following the derivation presented in (Giraud, 2016), the subsequent expression can be obtained:

$$\begin{aligned}
\sum_g \dot{Q}_g^*(t) = \sum_c \dot{Q}_c(t) \cdot [1 + F_T^*(t)] \\
- \dot{W}_{pmp}^*(t)
\end{aligned} \quad (17)$$

In expression (17), $F_T^*(t)$ is a modulation term that depends on past and present values of supply temperature.

3.3.6 Summary

Our MILP DHS production and distribution optimizer is composed of a linear cost function (see equation (1)) subject to linear equality (see equations (16) and (17)) and inequality (see equations (2)-(7), (15)) constraints representing the physical conservation laws and the technical limitations of the DHS.

4 The case study

The operation of our advanced controller has been evaluated by simulation means relying on a

representative case-study. This section explains how the case-study has been designed and it details the models composing it.

The CCIAG company and our research group are currently involved in a joint research program devoted to the development of advanced decision support/making tools for operational management of DHS. CCIAG operates the second largest DHS in France in the city of Grenoble. This system yearly delivers 900 GWh of heat using 225 km of distribution pipes and liquid pressurized water as heat carrier fluid. This system is actually managed using variable supply temperatures and differential pressures respectively ranging from 110 °C to 180 °C and 5 to 15 bars. These features are similar to systems in other French and European cities (e.g. Metz, Chambéry, Vienna, Warsaw...). Therefore, we have built the case-study used in the present paper upon the Grenoble HTDH system. However, in order to limit the modelling work during the first stages of our research project we considered only a portion of the Grenoble distribution network.

On the production side, we have considered 15 heat production units as it is representative of the installed capacity in Grenoble. All the 15 heat production units form a unique production plant represented in the dynamic model by equivalent heat and momentum sources. The sample network serves 26 heat consumers modelled using load profiles taken from an historical database (15 min sampling period) provided by CCIAG. The substation models are parametrized using the dimensioning rules applied on the Grenoble DHS.

To increase the relevance of the model, the main parameters used to model the production units in the MILP model have been proposed by CCIAG, our industrial partner in the project. For confidentiality purposes, the fuel prices denoted C_g^{th} , the cost of a generator start up denoted $C_g^{off/on}$ and the daily electricity price profile used for pump operation, denoted C^{el} are not reported here.

4.1 Simulation settings

The simulation period covers the full 2013/2014 French heating season, i.e. from November 2013 to mid-April 2014. The heat demand normalized by the installed heat power capacity (designed for an external temperature of -11 °C) and the external temperature over that period are shown in Figure 5. Outside this period, low cost heat from the waste incineration unit is produced in excess of demand thereby limiting the usefulness of advanced control strategy. Moreover, the heating season represents over 80 % of annual production costs.

The simulations were conducted with an elementary time step of 15 minutes. The optimization time-horizon is set to 24 hours and the receding horizon is fixed at 6 hours. For control in real conditions, affected by many sources of uncertainty (e.g. load prediction errors ...),

the receding horizon would be chosen equal to the 15 min elementary time step to increase robustness and stability. A complementary study on this topic is currently under investigation.

A 1 K threshold on the supply temperature increment is used to decide that convergence between the dynamic model and the MILP problem is reached. We set the relative MIP gap tolerance to 0.03. That instructs CPLEX to stop as soon as it has found a feasible integer solution proved to be within three percent of optimal. This tolerance is less than the error due to the uncertainties of the predictions. There is no relaxation in our resolution method since no approximation scheme is used, so we always find a solution within 3% from the optimal which is a global minima. It could be interesting to compute a relaxation of the MIP and compare the exact and the approximate resolutions, but the good performance of our model in terms of computational time makes the exact resolution compatible with the on-line application of our controller on a real DHS.

4.2 Implementation

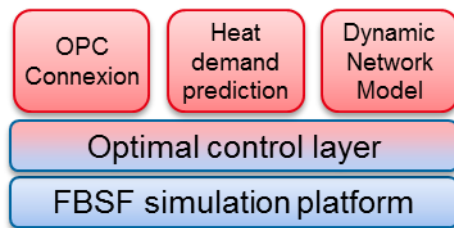


Figure 7: Architecture of the PEGASE optimal control framework

The overall algorithm is programmed using an in-house optimal control framework named PEGASE. PEGASE is based on the FBSF platform developed by the L3S company (L3S, 2016). FBSF enables multi-models simulation based on the FMI 2.0 co-simulation standard.

Figure 6 illustrates the architecture of the PEGASE optimal control framework. The lower layer is the FBSF simulation platform, which provides the basic services for running multi-model simulations. The middle layer is the optimal control layer, which performs the optimal control algorithm presented in the previous sections. The upper layer contains the dynamic simulation models used by the optimal control layer, as well as other prediction models and generic OPC connectivity services.

For the application described in this paper, we use only one dynamic simulation model, which consists of the Modelica model of the distribution network. This model is converted into a 2.0 co-simulation FMU by DYMOLA 2017. For other applications, several dynamic simulation models can be used together, either in the form of FMU or with FBSF-specific C++ code.

Within the optimal control layer, we express the MILP optimization problem using an in-house C++ code and solve it relying on CPLEX (IBM, 2009).

CPLEX can easily and quickly solve numerous problems with high combinatory owing to parallelization and application of the branch and bound method to reduce the search space (Brah, 1991). As a result, the computational time is generally lower than with other MILP solvers.

5 Results and Discussion

The simulation results obtained over the heating season are presented and discussed in this section. The numerical performance of the controller are also described. The current limitations of the proposed controller are finally presented at the end of this section.

For evaluation purposes, we compared the performance of our advanced controller to a more classical controller based on expert laws. This controller is still a popular method used in many existing systems owing to its simplicity and robustness. The standard controller is based on the piling method for the production planning. On the distribution side, the supply temperature is determined using a static heating curve while the differential pressure is maximized. To limit a chattering effect on the on/off status of heat production units, a hysteresis time-dependence is considered in the determination of generators starts and stops.

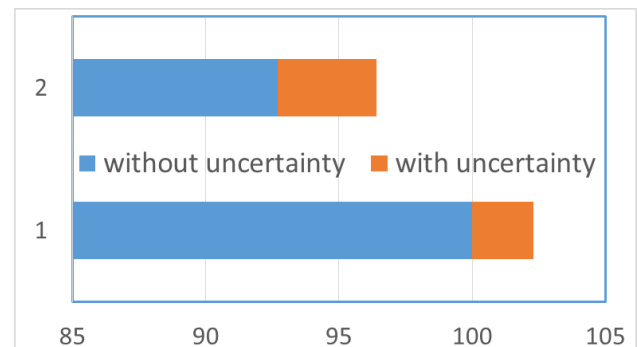


Figure 6: Normalized production costs over the 2013/2014 heating season for a standard (1) and an optimal controller (2).

As displayed in Figure 7, results point that our method significantly reduces production costs both with and without the consideration of uncertainties. The production cost's decrease is explained in the following section.

Table 1. Computational time and iterations for a 24 hours predictive horizon.

Computational time		Number of iterations	
Mean	Max	Mean	Max
37.8 s	273.7 s	3	29

On the production side, the optimal controller often keeps several peak generators at minimal load to anticipate future peak demands and avoid start-up costs when the produced heat is low (particularly during nights). As a consequence, the number of generator

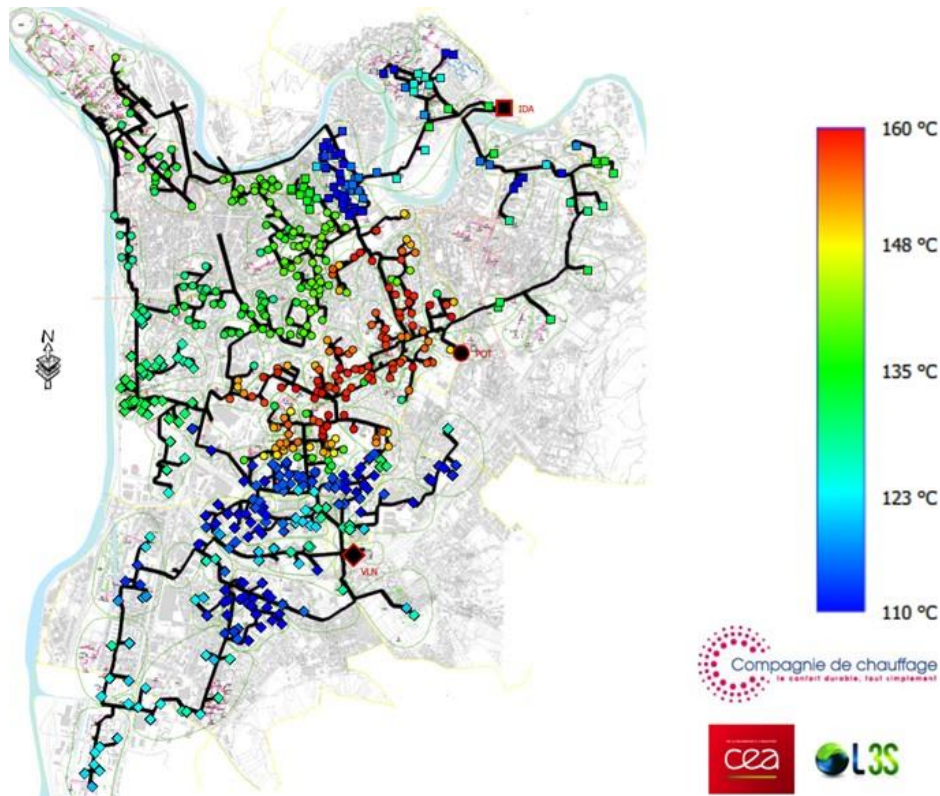


Figure 8: Snapshot of a dynamic simulation of the Grenoble DHS within the PEGASE

startups over the season is significantly reduced between the expert law control and the optimal controller.

On the distribution side, due to the optimization of T_{sup} and ΔP_{pmp} , results are very specific for the optimal controller. On the one hand, the supply temperature is often minimized and the differential pressure is maximized. As a result, the heat losses are reduced and the pumping work is increased for a global energy consumption reduction. This is due to the low pumping costs compared to the heat production costs encountered in this case study. On the other hand, our optimal controller is able to use the storage capacity of the network to anticipate future peak demands thereby increasing the use of base generators and avoiding the use of additional and expensive heat generators. To benefit from the network storage capacity, our control strategy increases the supply temperature prior to a peak demand. Accordingly, differential pressure is decreased without impacting the supplied heat demand. As a consequence of using the network storage capacity, the number of generators' startups is further decreased.

Table 1 presents the mean and maximal values of computational costs and iterations for a 24 hour predictive horizon. Using a 15 min time step, the optimization problem contains 5430 variables including 2460 binary variables and 5530 constraints. The mean and maximal computational time are respectively less than 40 s and about 4 minutes. These figures are

compatible with the on-line application of our controller on a real DHS.

The controller, as described in the present paper, is currently restricted to DHS comprising one single heating plant feeding a non-meshed network. Application of the method to a multiple supply points DHS and a meshed network is the subject of ongoing work. Another point worth mentioning is that the number of constraints of the MILP problem grows linearly with respect to the number of consumers (see inequality (15)). Thus, the application of the proposed method to large-scale DHS impose to consider a set of critical consumers in the network. As suggested in (Nielsen, 2005), such consumers may be selected so that if the T - ΔP (see inequality (15)) requirements for them are satisfied then the requirements for all consumers are satisfied. It has been verified that the current CPLEX MILP solver could handle problems comprising a set of several hundreds of critical consumers. This testing can be considered positive with respect to the scalability of the proposed controller.

6 Conclusions

In this paper, we present a new control method for DHS management which simultaneously optimizes the production and the distribution variables. For each anticipation horizon, an optimized planning for the status and power of each generator as well as for the

supply temperature and the differential pressure is proposed. Based on the heat demand prediction of each consumer or group of consumers, our controller determines autonomously the combination of supply temperature and differential pressure necessary to supply the heat demand. The method is based on an algorithm minimizing the production costs and respecting a family of constraints representing the conservation laws and the physical limitations of the generators and the distribution network. We consider the nonlinearities of the distribution network thanks to an iterative method between the dynamic network simulation and the optimizer. Once implemented on a DHS, this generic control strategy will autonomously select the best compromise among the control variables to minimize the production costs.

We also compared the proposed method to a more classical controller based on expert law. The comparison is based on the simulation over a heating season of a virtual DHS representative of the Grenoble case. Results show that our global optimization method improves the seasonal production costs by more than 8 % compared to empirical methods. The proposed controller decreases the production costs by taking advantage of the network storage capacity. The use of expensive peak heat generators is then minimized whereas base heat generators operation is maximized.

The distribution network dynamic model used in the present study was built by gathering components taken from the Modelica *DistrictHeating* modelling library. However, due to efficiency issues well described in (Casella, 2015), such modelling approach is currently not suitable for the representation of large-scale DHS comprising thousands of consumers and encompassing several hundreds of kilometers of distribution pipes. For such systems, we developed a dedicated C++ simulation code, not detailed in the present paper, and applied it to the Grenoble DHS (see Figure 8). This last development paved the way to the application and testing of our optimal controller on the 400 MW_{th} Grenoble DHS during the 2016-2017 heating season.

Acknowledgements

The authors sincerely wish to thank Elise Le Goff, Nicolas Giraud and Philippe Clotot from CCIAG, our industrial partner in the project, for the many stimulating exchanges and for providing real-life data from the Grenoble network. We would also like to acknowledge the financial support of CCIAG for the joint research program and of ADEME for the PhD of Loïc Giraud.

References

Akesson, J., Faber R., Laird C.D., Prolb K. Tummescheit H., Velut S., Zhu Y., "Models of a post-combustion absorption unit for simulation, optimization and non-linear model predictive control schemes", Proc. 8th Modelica Conference, Dresden, Germany, March 20-22, 2011

Brah S. A. and Hunsucker J. L., "Branch and bound algorithm for the flow shop with multiple processors," *Eur. J. Oper. Res.*, no. 51, pp. 88–89, 1991.

Casella F., "Simulation of Large-Scale Models in Modelica: State of the Art and Future Perspectives", Proc. 11th International Modelica Conf., Sept 21-23 2015, Versailles, France

Donald C. Augustin, Mark S. Fineberg, Bruce B. Johnson, Robert N. Linebarger, F. John Sansom, and Jon C. Strauss. The SCi Continuous System Simulation Language (CSSL). *Simulation*, No 9, pp. 281–303, 1967.

Benonysson A., Bøhm B., and Ravn H.F., "Operational optimization in a district heating," *Energy Convers. Manag.*, vol. 36, no. 5, pp. 297–314, 1995

Eriksson H., "Short Term Operation of District Heating Systems: An Application of Mathematical Programming" Doctoral thesis, Chalmers University of Technology, 1994.

Fang T. and Lahdelma R., "Genetic optimization of multi-plant heat production in district heating networks," *Appl. Energy*, vol. 159, pp. 610–619, Dec. 2015.

Giraud L., Bavière R., Paulus C., Vallée M., and Robin J.-F., "Dynamic Modelling, Experimental Validation and Simulation of a Virtual District Heating Network," Proc. 28th Int. Conf. on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems (ECOS), Pau, France, 2015.

Giraud L., Bavière R., Vallée M., Paulus C. "Presentation, Validation and Application of the *DistrictHeating* Modelica library", Proc. 11th International Modelica Conf., Sept 21-23 2015, Versailles, France doi 10.3384/ecp1511879

Giraud L., "Modélisation Dynamique et Gestion Avancée de Réseaux de Chaleur", Doctoral Thesis, Université Grenoble Alpes, 2016.

IBM, *User's Manual for CPLEX - IBM ILOG CPLEX v12.1*. International Business Machines Corporation, 2009.

Lund H., Werner S., Wiltshire R., Svendsen S., Thorsen J. E., Hvelplund F., and Mathiesen B.V., "4th Generation District Heating (4GDH)," *Energy*, vol. 68, pp. 1–11, Apr. 2014.

Nielsen T. S. and Madsen H., "Control of Supply Temperature in District Heating Systems with Multiple Supply Points," presented at the 18th Int. Conf. on Efficiency, cost, Optimization, Simulation, and Environmental Impact of Energy Systems (ECOS), Trondheim, Norway, 2005, vol. 2, pp. 1071–1079.

Runvik H., Larsson P.-O., Velut S., Funquist J., Bohlin M., Nilsson A., Modarrez Razavi S., "Production Planning for Distributed District Heating Networks with JModelica.org", Proc. 11th International Modelica Conf., Sept 21-23 2015, Versailles, France doi 10.3384/ecp15118217

Sandou G., "Modélisation, Optimisation et commande de parcs de production multi-énergies complexes," Doctoral thesis, Université Paris XI Orsay, Paris, France, 2006.

SNCU, "Enquête nationale sur les réseaux de chaleur et de froid - Restitution des statistiques portant sur l'année 2011," 2013

<https://www.l-3s.fr/>, accessed in december 2016

Rapid development of an aircraft cabin temperature regulation concept

Alexander Pollok^{1,2} Daniel Bender¹ Ines Kerling¹ Dirk Zimmer¹

¹Institute of System Dynamics and Control, DLR German Aerospace Center, Wessling, Germany,
{alexander.pollok,daniel.bender,ines.kerling,dirk.zimmer}@dlr.de

²Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy

Abstract

The air in aircraft cabins is controlled for pressure, temperature and humidity. The number of temperature zones is generally kept low, for reasons of necessary ducting space. We devise a new ducting concept, which enables a large number of temperature zones. Controllability of the system is however predicted to be a potential obstacle. For a quick resolution of this question, a Modelica model is created. Model creation is focused on a short development time as well as usefulness for controller synthesis. A workflow is presented that enables a quick iteration time between controller synthesis in Matlab and controller testing in a Modelica environment. Finally, the impact of this new concept on the energy consumption of the air generation unit is discussed.

Keywords: *Modelica, energy, exergy, control, modelling*

1 Introduction

In modern passenger aircraft, temperature control is realized using a small number of temperature zones. For instance, the Airbus A320 features two fixed-size temperature control zones for the cabin, plus one additional zone for the flight deck. A typical cabin temperature regulation system is illustrated in Figure 1. Fresh air is delivered by the air conditioning packs and ducted into the mixing chamber (M). There it is mixed with filtered and recirculated air from the cabin underfloor volume. It is split up into two mass flows. For each mass flow, very hot (around 200 °Celsius) trim air is added to increase the temperature to the desired value. The air is then ducted into the cabin volume. Displaced air is ducted into the underfloor, where some of it is recirculated, the remaining air is vented overboard.

Airlines like to customize their aircrafts with variations in travel classes, seat configurations, and availability of onboard entertainment systems. Generally, the demarcations of travel classes do not conform to the borders of cabin temperature zones. Imagine an expensive and therefore sparsely populated first class, followed by the business class, densely packed with business people producing hot air. This can lead to discrepancies with regard to the heat load per cabin length, which cannot be compensated by the control system, if they belong to the same tempera-

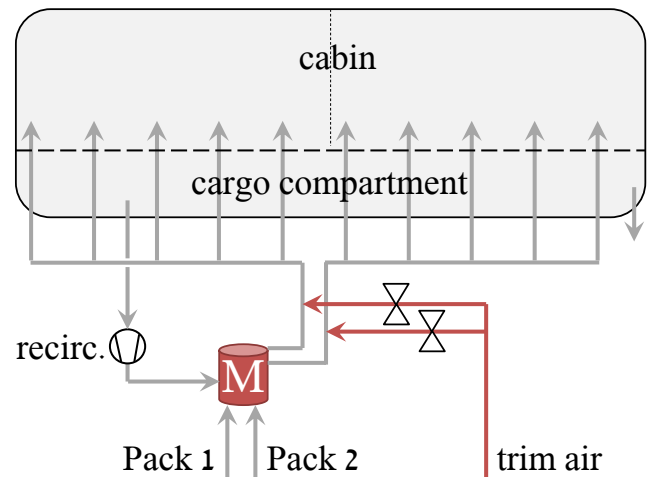


Figure 1. Conventional cabin temperature regulation system

ture zone.

Some ideas have been proposed to remedy this problem. In (Jacobs and De Gids, 2006) a concept is presented, where each passenger receives his own air outlet and individual temperature zone. However, the resulting size of the necessary ducting system within the crowded installation space of a modern aircraft is not considered. Similar concepts are mentioned in (Gao and Niu, 2008) or (Zhang et al., 2012), but the focus of the work is not on the control or feasibility side, but on air contamination reduction.

We propose an alternative cabin temperature regulation concept, which is illustrated in Figure 2. This concept is based around two main ducts, each one spanning the complete cabin length. One of them ducts air at a relatively cold temperature, the other one at a relatively hot temperature. At each cabin temperature zone, the air from both pipes is locally mixed using a small actuator, then ducted into the cabin.

This concept realises a variable number of temperature control zones. For a large number of zones, the amount of necessary ducting space and weight is lower than that of a conventional architecture, as a simple spreadsheet calculation for a typical single aisle aircraft shows, see Figure 3. Main reason for this is that only 2 pipes of cabin length L have to be fitted, instead of N pipes of average length $L/2$. System weight and volume even goes down

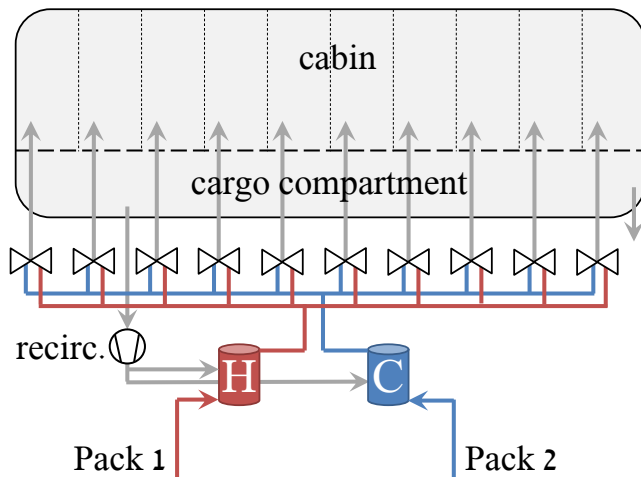


Figure 2. Proposed cabin temperature regulation concept

for a larger number of temperature zones, as less distribution ductwork between control valves and riser ducts is needed.

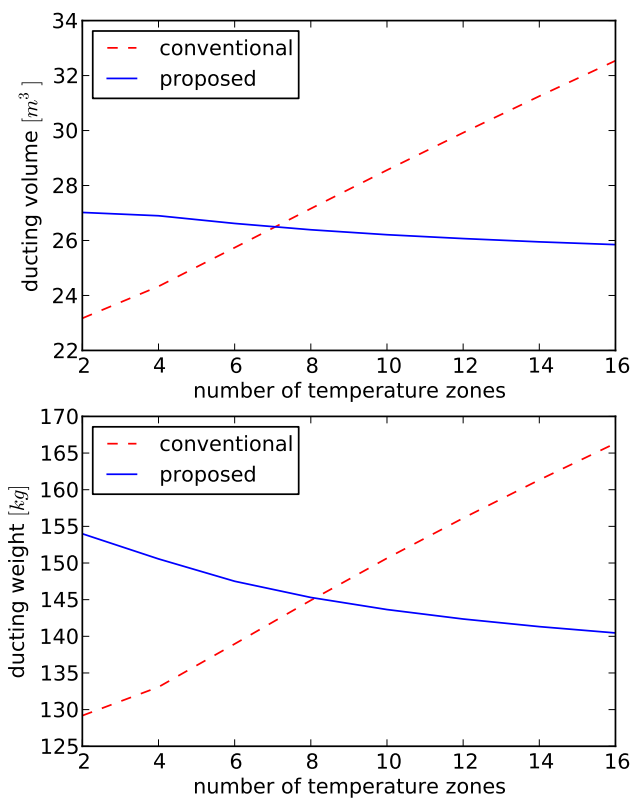


Figure 3. comparison of total ducting weight and ducting volume for conventional and proposed concept

This is bought at the expense of a potentially much more involved control system. Pneumatic and thermal interactions between temperature zones may be strong enough to prohibit the use of decentralized control. Also, the energy offtake of such a concept compared to a conventional architecture is unclear.

Given the high cost of testing facilities, a model-based

approach is needed for an early design evaluation. The corresponding modeling environment must thus be able to cover all relevant aspects of the proposed concept. Beside the physical processes belonging to the pneumatic and thermal domain, this also contains the control design and of the temperature regulation system. Modelica is a well-established choice for the physical modelling (Sielemann, 2012; Schlabe and Zimmer, 2012) but provides also sufficient to represent and evaluate the control models (Baur et al., 2009; Bonvini and Leva, 2012). The control design can then be achieved in interaction with Matlab.

The goal of this paper is to show how Modelica can be used for accelerated feasibility studies using the example of a new cabin temperature regulation concept. It is structured as follows: Section 2 presents the design requirements for a suitable model as well as the taken approach. Section 3 shows the controllability of the temperature regulation concept, based on the developed model. Section 4 treats energy considerations of the proposed architecture. Interesting points that came up during the development of the project are discussed in Section 5. Section 6 concludes the paper.

2 Modelling

A good simulation model is the basis for all subsequent development steps. The following requirements hold in the context of this work (from most to least important):

Development time The time needed to plan, develop, and test the model shall be short.

Unity If possible, all requirements shall be met in a unified model. Having several versions of a model often results in a significant increase in development time as well as project complexity.

Linearity Small perturbations around the design point shall result in linear model behavior. Model inputs that are connected to saturating actuators shall be scaled symmetrically around zero.

Accuracy The simulation model shall include all major physical effects. Deviations from reality should be small enough to be irrelevant for the subsequent development steps.

Robustness The model should predict accurate transient responses for boundary conditions that are far from the design conditions.

Simulation Speed Simulation of the model shall be fast. Numerical Stiffness shall be avoided if possible.

Size Total size of the model shall be small. This includes several metrics like the number of variables, number of states and lines of code. A small model decreases development time and increases comprehensibility.

These requirements partially contradict each other. Some balancing can be done using multi-objective optimization like shown in (Pollok and Bender, 2014), but ultimately, some arbitrariness remains. We decided to keep the model as simple as possible, with the exception of two physical effects that posed challenges for the control design: the first effect is the thermal interaction of air between the different cabin zones. The second effect is the pneumatic interaction of air in the asymmetric ducting system. The complete model structure is shown in Figure 4 and presented in the following.

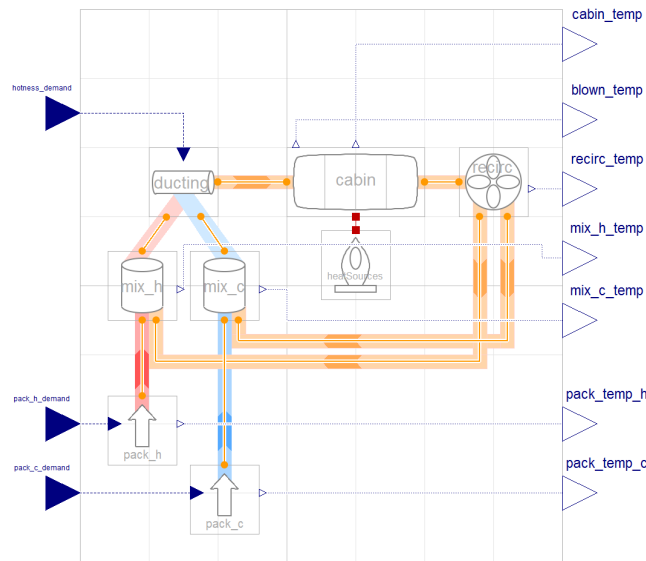


Figure 4. Top level view of temperature regulation model

Six subsystems were created for air conditioning packs, mixing chambers, ducting system, cabin, heat sources and recirculation system. These subsystems were in turn composed from simple components like flow resistances or volume elements. The model structure was limited to those three layers of system, subsystem and components. Inheritance was avoided, based on the results as described in (Pollok and Klöckner, 2016). The Modelica Standard Library (Modelica-Association, 2008) was used as much as possible to save additional modelling time.

All subsystems included an integer parameter n , denoting the number of discretized volume elements in the cabin. In this way, the scalability of the concept can be tested later without additional modelling effort.

Of those subsystems, ducting and cabin are especially interesting from a modelling perspective:

2.1 Ducting

As illustrated in Figure 2, air is ducted from the mixing chambers into the cabin via a network of ducts. This network is asymmetrical and interactive with regard to the cabin temperature zones. If a large amount of cold air is needed for the center temperature zones, the effective hydraulic resistance from the cold mixing chamber to the outer temperature zones increases. For controller synthe-

sis and concept validation, this effect has to be modelled.

This was done using vectorized flow elements together with customized connect-statements in a short amount of code and development time. The implementation is shown in Figure 5. Not shown are the parameters for the individual air resistance components. These are also dependent on the discretization parameter, since for example the length per pipe is not constant.

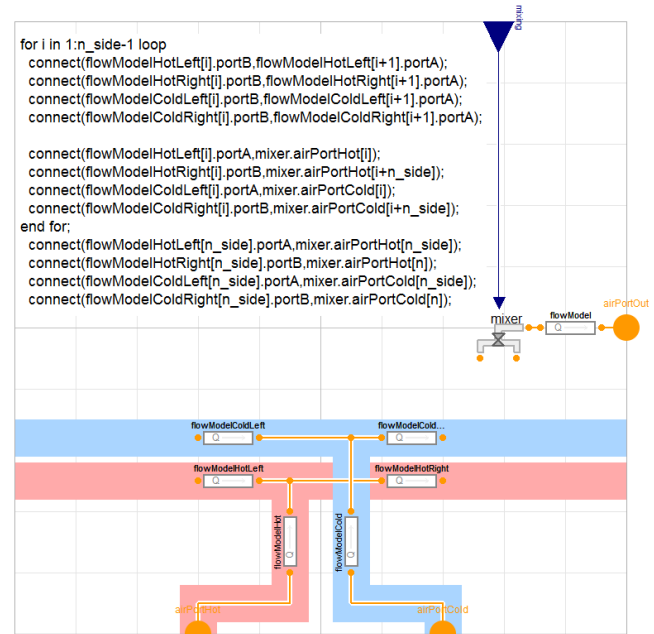


Figure 5. ducting subsystem model

2.2 Cabin

The flow configuration inside the aircraft cabin is complex and can only accurately be determined by experiments or CFD-calculations. However, for the evaluation of the presented concept, a low-order approximation suffices. The cabin is divided lengthwise into n volume elements. These elements are directly connected to enforce pressure equalization. Fluid volume elements can directly be connected in Modelica, at the cost of nonlinear systems of equations. This cost is however preferable to the alternative, where the very small flow resistances between cabin volumes leads to a very stiff simulation model. If no equalization mass flows occur, there is still some amount of thermal equalization caused by diffusion. This is modelled using thermal resistance elements, coupled between the volume elements. They were parameterized according to empirical experience.

Again, the subsystem was realized using a combination of vectorized elements and customized connect statements. The implementation is shown in Figure 6.

3 Control

A sufficient way to demonstrate controllability of a system is to find a stabilizing controller. For a demonstration of

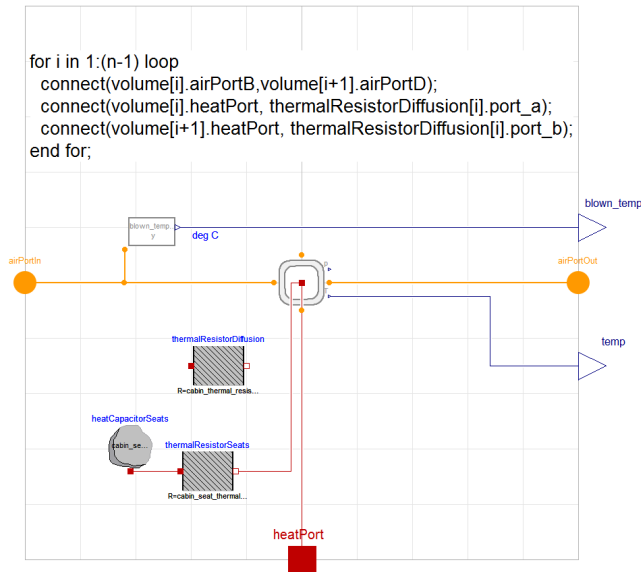


Figure 6. cabin subsystem model

robust performance, the response of the controlled system with regard to noise and nonlinearities can be shown.

We used linear quadratic Gaussian (LQG) control to find an optimal controller for the problem. The method is simple and often satisfactory in the development of multivariate, or MIMO-controllers for linear time-invariant (LTI)-systems. LQG controllers consist of a linear quadratic estimator (LQE, also known as Kalman filter) to estimate non-measured states, and a linear quadratic regulator (LQR), essentially an optimal state regulator. The regulator uses the estimated states to compute a control signal, the estimator uses the measured states as well as the control signal to estimate the states. This is illustrated in Figure 7. Both components can be designed independently, this will not compromise stability of the controlled system, but it can affect stability margins (for reference, see the very interesting abstract of Doyle (1972)), so robustness properties have to be verified after controller synthesis.

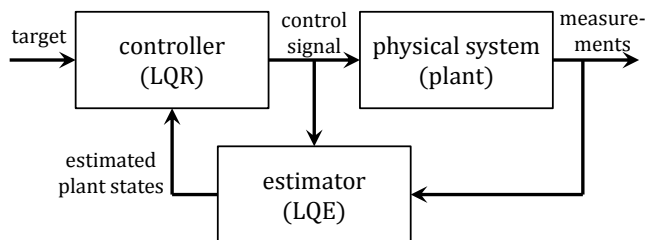


Figure 7. Structure of an LQG regulator

For LQG-synthesis, a linearized model is necessary. We used the linear systems library as presented in (Baur et al., 2009) as implemented in Dymola 2016 to linearize the model around the steady state. The model was instantiated with a pack temperature spread of 20Kelvins and 10 cabin temperature zones. Before linearization, the model

was simulated for 1000s to come close to the steady state solution. Keep in mind that all model inputs are set to zero at linearization. Therefore, the valid range for all model inputs has to include zero and some buffer in both directions. This can be a problem for instance when a model input is connected to a valve opening with a valid range of 0 to 1. In this work, we scaled all such inputs to a valid range of -1 to +1. All other in- and outputs were scaled according to typical orders of magnitude. The linearized system was exported as a .mat-file using the writeMatrix-command.

Computation of the LQG controller was done in Matlab, based on the script as described in (Skogestad and Postlethwaite, 2007, p. 348). This formulation adds integrators to the plant outputs, ensuring zero steady state error for the controlled system. The model was reduced from 32 to 23 states, based on the results of the Hankel singular value decomposition as presented in Figure 8. Also, the 1-dof¹ variant was used, since setpoint changes are not a major concern in climate control systems.

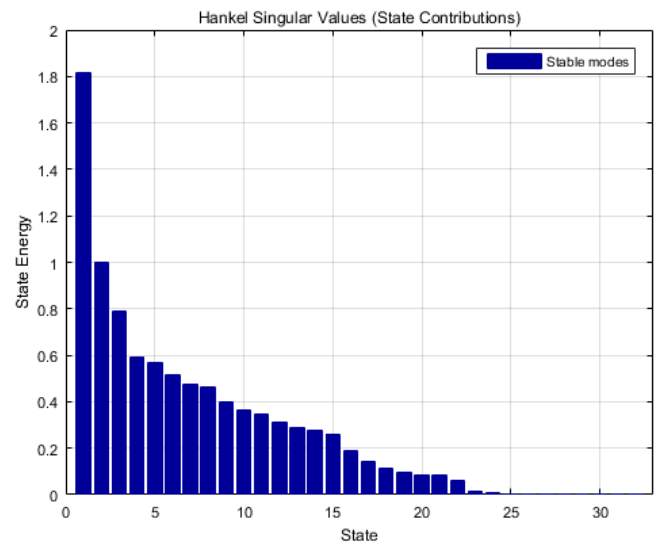


Figure 8. Hankel singular value decomposition of temperature control system

A Matlab-script was developed to automatically generate Modelica-code of the controller. This script is shown in Listings 1 and 2. In this way, a candidate controller can be tested in a Modelica environment in a few seconds.

Listing 2. Matlab code for the generation of Modelica controller code

```
function [ ] = fun_changeMatrixFormat ( M )
% Changes the Matrix to a format like
% it is needed for an Matlab- Input
% f.e. M = 1 0
```

¹In a 1-dof (one degree of freedom) controller, setpoint changes and disturbances are handled equally. In a 2-dof controller, setpoint changes can be handled far more aggressively, as the setpoint is not part of the feedback-loop and therefore has no impact on system stability.

Listing 1. Matlab code for the generation of Modelica controller code

```

function [] = fun_dymola( sys )
%input: statespace object sys, representing combined estimator and controller
sys_x = size(sys.a,1);    % = m_A, n_A, m_B, n_C
sys_y = size(sys.d,1);    % = m_C, m_D
sys_u = size(sys.d,2);    % = n_B, n_D

% declarations
fprintf('\n\nmodel Controller "1-DOF LQG controller"\n\n'); %Name ""
fprintf('// import \n');
fprintf('import Modelica.Blocks; \n\n');
fprintf('// parameters \n'); %A,B,C,D...
fprintf('parameter Real A_controller[%0.f, %0.f] = ', sys_x, sys_x);
fun_changeMatrixFormat( sys.a );
fprintf('; \n');
fprintf('parameter Real B_controller[%0.f, %0.f] = ', sys_x, sys_u);
fun_changeMatrixFormat( sys.b );
fprintf('; \n');
fprintf('parameter Real C_controller[%0.f, %0.f] = ', sys_y, sys_x);
fun_changeMatrixFormat( sys.c );
fprintf('; \n');
fprintf('parameter Real D_controller[%0.f, %0.f] = ', sys_y, sys_u);
fun_changeMatrixFormat( sys.d );
fprintf('; \n');

% variables
fprintf('// variables \n');
% blocks
fprintf('Blocks.Continuous.StateSpace statespace_controller');
fprintf('(A = A_controller, B = B_controller, C = C_controller, D = D_controller) \n');
fprintf('annotation (Placement(transformation(extent={{36,-2},{56,18}}))); \n');
fprintf('Blocks.Math.Feedback sum_controller [%0.i] \n', sys_u);
fprintf('annotation (Placement(transformation(extent={{-36,-2},{-16,18}}))); \n');
% inputs/outputs
fprintf('Blocks.Interfaces.RealInput command[%0.f] "command signal" \n', sys_u);
fprintf('annotation (Placement(transformation(extent={{-120,40},{-80,80}}))); \n');
fprintf('Blocks.Interfaces.RealInput feedbacksignal[%0.f] "sensor/feedback signal" \n', sys_u);
fprintf('annotation (Placement(transformation(extent={{-120,-78},{-80,-38}}))); \n');
fprintf('Blocks.Interfaces.RealOutput outputsignal[%0.f] "driver/output signal" \n', sys_y);
fprintf('annotation (Placement(transformation(extent={{100,-10},{120,10}}))); \n\n');

% equations
fprintf('// equations \n');
fprintf('equation \n');

% connecting the blocks and in/outputs
fprintf('for i in 1:%0.i loop\n', sys_y);
fprintf('connect(statespace_controller.y[i], outputsignal[i]) \n');
fprintf('annotation (Line(\npoints={{57,8},{110,8}},\nncolor={0,0,127})); \n');
fprintf('end for; \n\n');
fprintf('for i in 1:%0.i loop\n', sys_u);
fprintf('connect(statespace_controller.u[i], sum_controller[i].y) \n');
fprintf('annotation (Line(\npoints={{34,8},{-17,8}},\nncolor={0,0,127})); \n');
fprintf('connect(command[i], sum_controller[i].u1) \n');
fprintf('annotation (Line(\npoints={{-100,52},{-64,52},{-64,8},{-34,8}},\nncolor={0,0,127})); \n'
);
fprintf('connect(feedbacksignal[i], sum_controller[i].u2) \n');
fprintf('annotation (Line(\npoints={{-100,-58},{-26,-58},{-26,0}},\nncolor={0,0,127})); \n');
fprintf('end for; \n\n');

% creating symbol for the block
fprintf('annotation (...)\n');

fprintf('end Controller; \n'); % Name;

```

```

%           0 1 to [1, 0; 0, 1]
% for as much information as possible
format long;
mnbigness = size(M);
countm = 1;
countn = 1;
fprintf(' ');
while (countm <= mnbigness(1))
    while(countn <= mnbigness(2))
        if (countn < mnbigness(2))
            fprintf('%f, ', M(countm, countn));
        else
            if (countm < mnbigness(1))
                fprintf('%f; ', M(countm, countn));
            else
                fprintf('%f', M(countm, countn));
            end
        end
        countn = countn+1;
    end
    countn = 1;
    countm = countm+1;
end
fprintf(' ');
end

```

On the first try, equal disturbance and measurement noise was assumed, and a unity matrix was used for the input weight matrix R . The state weight matrix Q was calculated so that the projected plant output as well as the artificial integrator vector were also weighted with a unity matrix, using the Matlab code shown in Listing 3.

Listing 3. Matlab code for projection of the plant outputs to the state vector

```

R=weight_input*eye(n_u);
Q=blkdiag(weight_output.*transpose(C)*
eye(n_y)*C,weight_integrator*eye(n_y));

```

Since the actual system contains hard nonlinearities such as actuator saturation, compliance to those limits has to be tested using simulations of the controlled system. These simulations showed that the resulting controller outputs were exceeding the actuator limits. The variable weight-output was increased to 10.000, resulting in improved behavior². Note that no actual optimization with regard to some performance criterion took place. The response of the controlled system to target temperature steps (from 20 to 21 °Celsius) on each temperature zone is shown in Figure 9. Overshoot³ is generally low, but temperatures are still somewhat affected by temperature steps on neighboring zones. Rise time is at 68 to 102 seconds (M^4 : 85s, SD: 11.3s).

Robustness with regard to sensor noise was validated using the Noise library as presented by Klöckner et al.

²Using LQR/LQG, it is quite typical that small changes in controller behavior necessitate large changes in the weighting matrices.

³Overshoot describes the peak of the response to a step input, rise time describes the time it takes the output to increase from 0.1 to 0.9.

⁴ M denotes the mean value, SD denotes the standard deviation.

(2014). The qualitative behavior of the system remains unchanged. An illustration of the overall workflow can be seen in Figure 10.

4 Efficiency

The concept presented within this work requires a cold (C) and a hot (H) air reservoir (see Figure 2). Each of these reservoirs is supplied by a separate air conditioning pack. Conventional architectures duct the cooled air from the air packs to a common mixer unit (see Figure 1). The packs therefore condition the fresh bleed air to the same pressure and temperature. The proposed concept now claims an asymmetrical air conditioning in terms of temperature. The air pack would then run in different conditions compared to conventional in-service air packs. About 2-3% of the whole energy consumption of a conventional civil aircraft applies to the ECS (Bender, 2016). Thus an energy analysis of the deviating pack operation is necessary.

4.1 System Description

The key part of the ECS is the air generation unit (also called the pack). The pack conditions the air flow in terms of temperature, pressure and humidity. Usually there are two packs installed in an aircraft. Conventional systems use engine bleed air as the power source. The bleed air is drawn off from the compressor stages upstream the combustion chamber. Provided at high temperature (around 220 °C) and high pressure (around 2.5bar), the air must be conditioned before it is distributed into the cabin. First the air flow is lead to the air pack where it is cooled down and dehumidified. It passes several heat exchangers, a compressor, a turbine and valves before the flow has reached the right condition to be lead to the mixing unit. The ram air enters the pack from the ambient and passes a water injector, two heat exchangers and a fan. All of them are installed in the ram air channel. The ram air is used as a heat sink.

Figure 11 illustrates the Modelica diagram layer schematic of the air generation unit that is used for the energy analysis in this work. It includes a conventional three-wheel bootstrap-cycle, driven by bleed-air. Three different flows are considered: The bleed air arises from the compressor stage at the engine, passing at first the pneumatic distribution device before it enters the ozone converter. Inside the primary heat exchanger (PHX) the hot air is cooled down against the ram air flow. Before entering the compressor stage (CMP), a part of the air flow is separated and bypassed through the temperature control valve (TCV). Downstream the compressor stage the heated and compressed air is cooled down a second time inside the main heat exchanger (MHX) against the cold ram air flow. Here the most intense heat exchange takes place due to large temperature differences. The air flow now enters the hot side of the reheater and is cooled down again before its temperature is further decreased inside the condenser in order to dehumidify the air flow and prevent downstream conditions from reaching the saturation point.

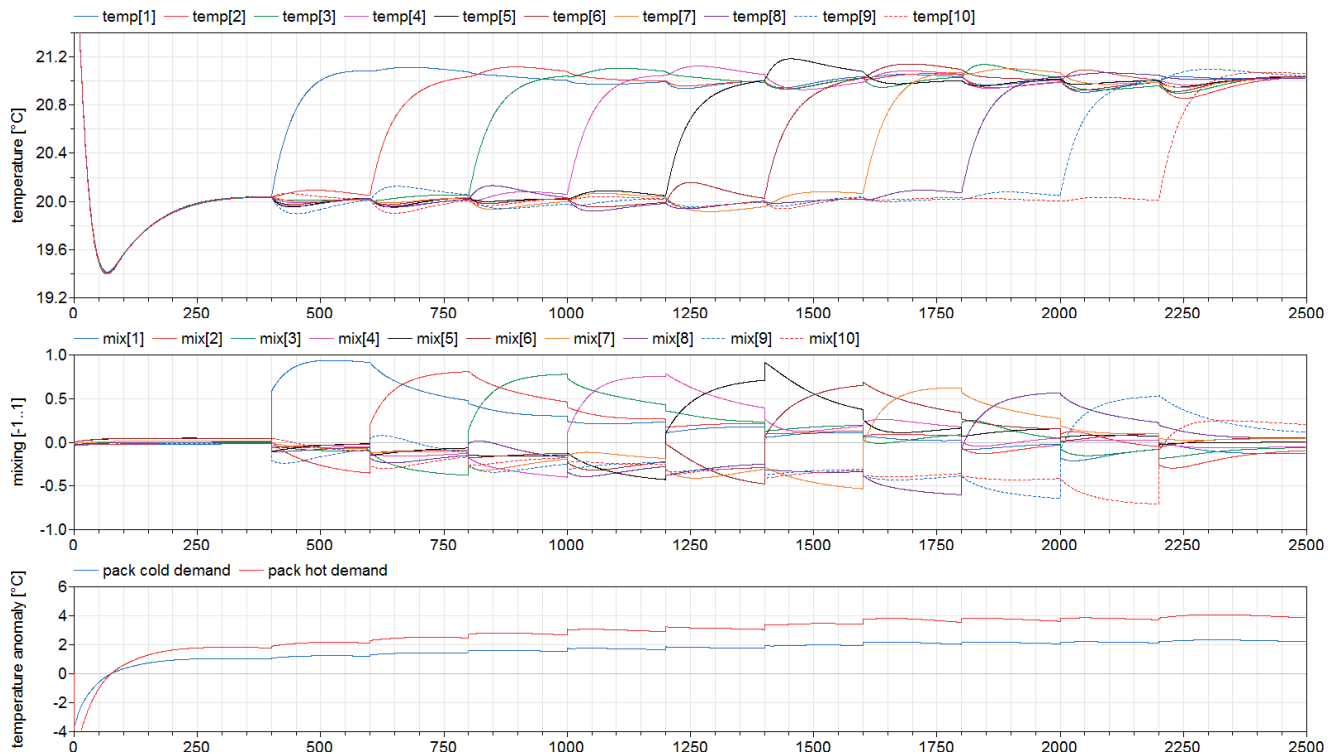


Figure 9. Response of controlled system to target temperature steps

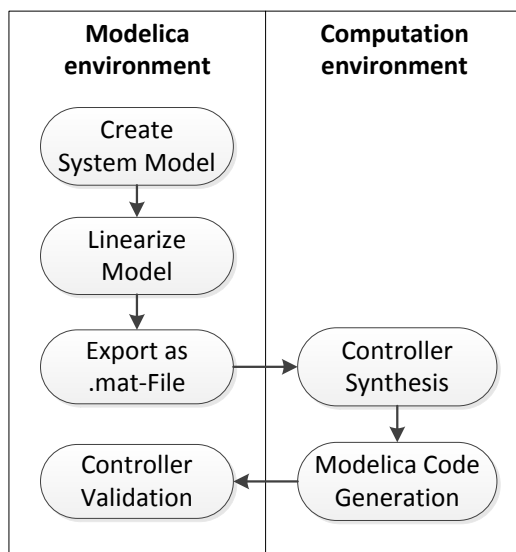


Figure 10. Workflow for controller synthesis

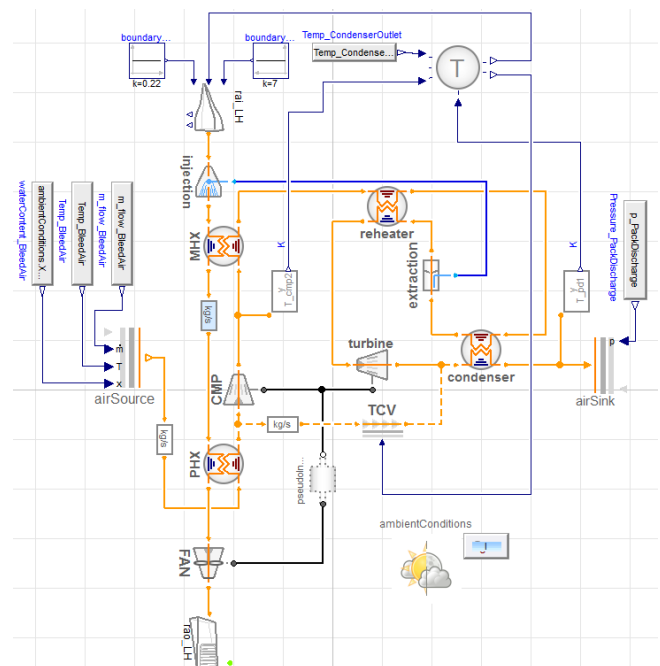


Figure 11. Diagram layer of air pack system model with three wheel bootstrap cycle

This configuration of the three wheel bootstrap cycle uses the concept of high pressure water separation. In case of condensation, the free water is separated in the water extractor and carried to the injector located at the beginning of the ram air channel. The dehumidified air flow now passes the reheater a second time, this time at the cold

side where it is reheated against its upstream air flow. Inside the turbine the air is expanded to a sufficient pressure level. Concurrently the temperature decreases significantly below ambient conditions. At this point the air reaches its coldest condition. Meeting the separated air from the temperature control valve, the flow gains a higher

temperature und finally is heated up inside the condenser again before it leaves the air pack to the mixing unit.

The second flow occurring is the ram air flow. It functions as a heat sink and enters the aircraft through inlets outside the aircraft's fuselage. The amount of air flow can be controlled by flaps installed at the inlet and outlet of the ram air channel. Water from the water extractor is now injected into the ram air flow where it evaporates and subsequently the temperature of the ram air flow is decreased. The cool air passes successively the main heat exchanger and primary heat exchanger before it leaves the ram air channel to the ambient. In ground operation the ram air flow is driven by the ram air fan that is mounted on the same shaft as the compressor and turbine. The components shown in Figure 11 are taken from an ECS library (Sielemann et al., 2007).

4.2 Energy analysis

The proposed cabin temperature regulation concept is based on asymmetrical pack discharge temperatures for each pack. Therefore the energy analysis was performed for a wider range of discharge temperatures. Two identical air packs were assumed so that simulations were carried out using the air pack Modelica model shown in Figure 11 for a range of discharge temperatures varying from $-30\text{ }^{\circ}\text{C}$ to $20\text{ }^{\circ}\text{C}$. The mass flow and the discharge pressure were kept constant. Two control laws are implemented in the model. One that keeps the discharge temperature at the defined value by regulating the bypass mass flow through the TCV and another law that limits the compressor outlet temperature by regulating the ram air mass flow.

The three wheel bootstrap cycle is a self-containing air generation unit, i.e. it does not need any additional power source to run the turbo components. This is realized by the turbine that is driven by pneumatic power of the bleed air. It is assumed that the bleed air is constantly provided by the engine, independent of the operation of the air pack. However, the ram air flow changes due to different operating points and causes different amounts of aerodynamic drag. It is therefore directly linked to the pack discharge temperature. For the energy analysis, the variation of occurring drag caused by the ram air is calculated for each operating point. The drag of both air packs is summed up and displayed with the average temperature of both packs and their anomaly in temperature. Temp average denotes the average discharge temperature of both packs, Temp anomaly denotes the deviation of both packs from the common average. For example, if one pack discharges air at $10\text{ }^{\circ}\text{C}$ while the other discharges air at $20\text{ }^{\circ}\text{C}$, temp average is $15\text{ }^{\circ}\text{C}$, and temp anomaly is $\pm 5\text{ }^{\circ}\text{C}$.

Figure 12 shows the result of the simulations for the different discharge temperatures. The model was simulated for a cruise flight phase at 39.000 feet altitude. The horizontal axis shows the average temperature that can be achieved of the two packs. All possible combinations for a temperature range from $-20\text{ }^{\circ}\text{C}$ to $30\text{ }^{\circ}\text{C}$ were considered. For each combination, the temperature anomaly to the av-

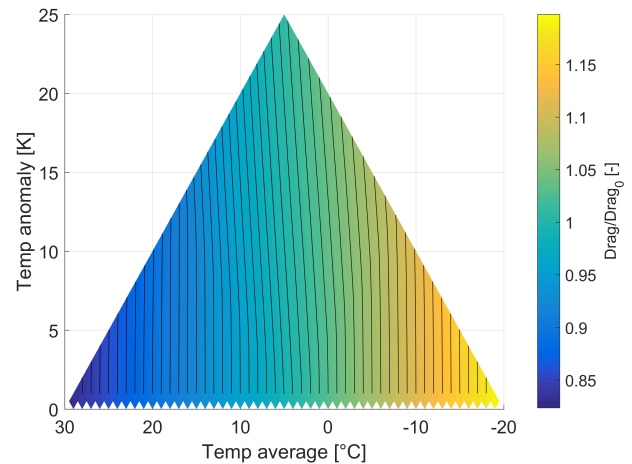


Figure 12. Drag caused by Ram air vs. average temperature and temperature anomaly

erage temperature was determined and presented by the vertical axis. The graphic shows a triangular shape what is related to the fact that e.g. an average temperature of $-10\text{ }^{\circ}\text{C}$ could be achieved by a maximum range of $0\text{ }^{\circ}\text{C}$ and $-20\text{ }^{\circ}\text{C}$ what leads to an anomaly of 10 K . The coloring represents the total drag of both packs and the black lines represent lines of constant drag. Due to confidential reasons, the values are normalized to an average drag value.

The results show that the drag for the border regions of high and low average temperature remains constant with increasing anomaly. However, for the medium temperature range, the lines of equal drag tip to the left with increasing anomaly. That means, the combined drag of both packs is slightly higher for packs operating with large temperature differences.

5 Discussion

The resulting concept with its workflow depicted in Figure 10 proved to be effective for the analysis and optimization of a nonlinear MIMO control problem with a complex plant model. Yet, the concept of this paper for the temperature regulation of aircraft cabin represents only one item of a more general problem set. In this case, the control design was an integral part needed to evaluate the overall system design in an early phase. A rapid LQG controller provided a sufficient solution. Interfaces to and from Matlab eased the control design whereas Modelica served as main modeling and evaluation environment.

The long-term goal of this work however goes beyond this use case. Since many sub-systems are already highly optimized, further system optimization requires a higher level of sub-system integration and also more centralized control approach. This represents a higher level of integration and it often implies a low availability of corresponding test examples or rigs. Principal questions of controllability, performance of controllers and of the system as a whole need to be evaluated at an early design phase.

To this end, it is necessary to bring together the different software platforms that engineers use for control design (such as Matlab) and the modeling environments for system dynamics (such as Modelica). This is not the first paper addressing this problem. Typical attempts use the S-function standard to import the plant model to Matlab. Alternative approaches use the FMI-Standard to export the controller from Matlab to a Modelica environment. The presented work shows that code generation is also a feasible technique to achieve the desired result. It has the advantage that the final result is pure Modelica and does not require any further tools or interfaces.

Having the final result in pure Modelica is more suited when many variations of the plant model shall be created in order to test for various kinds of robustness. These changes may go beyond normal parameter changes since a variety of failure scenarios have to be modelled and simulated. In this application, typical faults are the malfunction of one pack and the malfunction of one or more valves in the ducting. Also the controller might be tested against Modelica models of higher fidelity. For all this work, having the controller in Modelica with all the internal controller signals openly available is the most convenient approach.

The aforementioned robustness tests still have to be performed to a large extent. These will hopefully not only further validate the temperature regulation concept but also the foreseen work-flow.

6 Conclusion

Modelica can be used to quickly generate models for validation studies of new concepts. However, the design of controllers based on this models makes additional tools necessary. Integrated modelling and computation environments such as Modia (Elmqvist et al., 2016) could be a remedy.

Acknowledgements

We thank Trey and Matt for inspiration.

References

- Marcus Baur, Martin Otter, and Bernhard Thiele. Modelica libraries for linear control systems. In *Proceedings of 7th International Modelica Conference, Como, Italy, September*, pages 20–22, 2009.
- Daniel Bender. Exergy-based analysis of aircraft environmental control systems - integration into model-based design and potential for aircraft system evaluation. In *ECOS 2016 - 29th International Conference on Efficiency, Cost, Optimisation, Simulation and Environmental Impact of Energy Systems*, 2016.
- Marco Bonvini and Alberto Leva. A modelica library for industrial control systems. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 076, pages 477–484. Linköping University Electronic Press, 2012.
- John Doyle. Guaranteed margins for lqg regulators. 1972.
- Hilding Elmqvist, Toivo Henningsson, and Martin Otter. Systems modeling and programming in a unified environment based on julia. In *International Symposium on Leveraging Applications of Formal Methods*, pages 198–217. Springer, 2016.
- NP Gao and JL Niu. Personalized ventilation for commercial aircraft cabins. *Journal of aircraft*, 45(2):508–512, 2008.
- P Jacobs and WF De Gids. Individual and collective climate control in aircraft cabins. *International journal of vehicle design*, 42(1-2):57–66, 2006.
- Andreas Klöckner, Franciscus LJ van der Linden, and Dirk Zimmer. Noise generation for continuous system simulation. In *Proceedings of the 10th International Modelica Conference-Lund, Sweden-Mar 10-12, 2014*, number 96, pages 837–846. Linköping University Electronic Press, 2014.
- Modelica-Association. The Modelica Standard Library. Online, URL: <http://www.modelica.org/libraries/Modelica>, 2008.
- Alexander Pollok and Daniel Bender. Using multi-objective optimization to balance system-level model complexity. In *Proceedings of the 6th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pages 69–78. ACM, 2014.
- Alexander Pollok and Andreas Klöckner. The use of ockham's razor in object-oriented modeling. In *Proceedings of the 7th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pages 31–38. ACM, 2016.
- Daniel Schlabe and Dirk Zimmer. Model-based energy management functions for aircraft electrical systems. Technical report, SAE Technical Paper, 2012.
- Michael Sielemann. *Device-Oriented Modeling and Simulation in Aircraft Energy Systems Design*. PhD thesis, Hamburg University of Technology, 2012.
- Michael Sielemann, T Giese, B Öhler, and Martin Otter. A flexible toolkit for the design of environmental control system architectures. In *Proceedings of the First CEAS European Air and Space Conference*, 2007.
- Sigurd Skogestad and Ian Postlethwaite. *Multivariable feedback control: analysis and design*, volume 2. Wiley New York, 2007.
- Tengfei Tim Zhang, Penghui Li, and Shugang Wang. A personal air distribution system with air terminals embedded in chair armrests on commercial airplanes. *Building and Environment*, 47:89–99, 2012.

Investigation of the Influence of Controller Types on Room Thermal Behaviour – A Simulation Study

Kristin Majetta¹, Christoph Clauß¹, Christoph Nytsch-Geusen²

¹Fraunhofer IIS EAS, Zeunerstraße 38, D-01069 Dresden, GERMANY

kristin.majetta@eas.iis.fraunhofer.de

christoph@clauss-it.com

²Fachgebiet für Versorgungsplanung und Versorgungstechnik, Berlin University of the Arts, Hardenbergstr. 33, D-10623 Berlin, GERMANY

nytsch@udk-berlin.de

Abstract

To control the indoor temperature of rooms two kinds of approaches are common. The first one is to use standard PI-controllers with a set of default parameters, which often leads to insufficient performance, waste of energy and unacceptable comfort violations [Rahmati, 2003]. The other approach is to use specifically developed and adapted controllers [Seidel et al., 2015], which have the drawback in a time-consuming and expensive development. Therefore, this paper investigates on finding rules and guidelines to find a suitable controller for a given room without the need of expensive controller adaption via simulation. To provide those rules a simulation study will be performed. This paper presents the first preparatory steps of this investigation, which includes the choice and development of four different room models equipped with different heating systems, which are an electrical radiator, a floor heating system, and a water supplied radiator. The authors present five types of controller models of different controller types to control the operative temperature of a room. Simulations of well-defined scenarios analyze the eligibility of the controller models regarding net energy consumption and comfort for the considered room models. First optimization results to improve the quality of the controllers are shown and further steps are explained.

Keywords: Building Simulation, Room Controller, Room Thermal Behavior, Optimization

1 Introduction

The German government plans the reduction of the primary energy consumption by 20% by the year 2020 compared to 2008 and, even more ambitious, by 50% by 2050. Many actions are taken to achieve this goal. One is the foundation and support of research projects to develop energy saving technologies. In the buildings sector one of these technologies are advanced control strategies to regulate e.g. indoor climate, energy consumption or the choice of an energy source out of different energy supplies. Normally, those controller strategies are developed and adapted to a specific room for which they work efficiently. Often those controllers can only be sufficiently adapted to other buildings under

application of relatively large simulation effort which includes the model development and parameterization of the considered room, the comparison of the room behavior with measurements and the parameterization as well as an optimization of the controller model. Therefore, the adaption to other rooms or buildings is expensive and often the energy reduction is in no relation to the effort of adjusting the controller. This is why nowadays often basic controllers with default parameters are used which is not sufficient to achieve the control goals regarding room temperature and net energy consumption. During the last year's research in the field of indoor room temperature controllers, several example control algorithms have been developed. In [Lauenburg, 2014] for example, the control of a radiator heating system is optimized. A similar approach is presented in [Carlon, 2014] where the energetic performance of a low-energy house is analyzed and two possible control strategies of the biomass boiler heating systems are investigated. Very high research effort during the last years was done in the field of model predictive control methods where the future behavior of the room is predicted by simulating and optimizing a room model to calculate the in the future needed set points to ensure comfort and energy optimality. Deputizing for the abundance of research activities and literature in this field the following references are named: [Afram, 2014; Parisio, 2013; Oldewurtel, 2010; Ma, 2009]. A drawback of this method is that the needed prediction model must be developed which is normally done from measurements, and that this model needs to be updated after each optimization run. Rules would be helpful that support the choice of indoor room temperature controllers including a suitable set of parameters that fit best for the considered room. Therefore, a methodology is needed which provides those rules and guidelines for typical use cases of temperature controllers with regard to given rooms and their installed HVAC technology. This paper presents first steps of this investigation, which is the development of four different, representative room models as well as five controller models of important controller types. By means of simulations of defined scenarios suitable for each type of room (e.g. office room, class room), the eligibility of the controller

models is investigated regarding the net energy consumption and the adherence by temperature comfort boundaries. First optimization results to improve the quality of the controllers are shown and further necessary steps are explained.

2 Modeling

2.1 Building models

To investigate the influence of different controller models on room behavior, four room models were built, that differ in size, building materials, heating system and usage. The chosen rooms are an office room, a class room a meeting room and a summerhouse (also called room since it consists of a single thermal zone). All models are equipped with different heating systems (electrical radiator, radiator flown through by water, floor heating) so that the typically range of room heating supply techniques is covered. That way it is assured that the methodology, which will be developed for choosing a suitable controller, is universally applicable and can easily be transferred to other problems.

The models of the four example rooms are built from model components of the Modelica BuildingSystems library, which is developed by the University of Arts (UdK), Berlin [Nytsch-Geusen, 2013]. This library can describe the behavior of complex building systems which consist of thermal and hygrothermal models of single buildings or districts in combination with the corresponding energetic supply techniques. The technical building services can contain thermal, hydraulic or electrical models for solar heat, photovoltaic, and HVAC systems. The room models developed for the purpose of this investigation are based on the Building1Zone1DBox-template that describes a single thermal zone with six opaque boundaries that can contain windows. The template is equipped with connectors compatible to the ambient model of the BuildingSystems library, with thermal ports to supply the building zone with heat and with a connector for the air change rate. The air temperature T_{Air} of the zone is supplied by the model via an output connector. Figure 1 shows the graphical representation of the Building1Zone1DBox template connected to the ambient model.

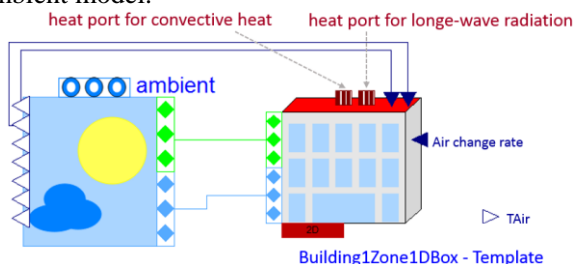


Figure 1: Template Building1Zone1D connected to ambient model

The ambient model provides the outside air temperature, the relative humidity of the ambient air, the wind speed and the solar radiation. The influence of the solar radiation on the operative room temperature depends mainly on the orientation of the windows, which are modelled within the Building1Zone1DBox template.

To assure the comparability of results, all room models receive the same ambient conditions from the TMY (typical meteorological year) [Deutscher Wetterdienst, 2014] of Chemnitz, a city in the east of Germany with approximately 250.000 inhabitants. For the sake of simplicity internal loads are not part of this study.

Summerhouse

The model of the summerhouse has a floor space of 30 m², a height of 3.5 m and, deviating from Figure 2, it is modeled with six boundaries, which represent four walls, the ceiling and the floor.



Figure 2: 3D representation of the summerhouse model

The summerhouse is modeled as a free-standing room, which means that the adjacent conditions of the boundaries (except for the floor) of the thermal zone are the ambient conditions. The adjacent boundaries are modeled as heavy construction from plastering, Styrofoam and bricks from concrete. The summerhouse model is equipped with a model of a 2kW electrical heating system. The actuating mechanism of the heating system is discrete which means it can either be switched on or off completely. Therefore, the control signal provided by the heating controller must also be discrete.

Single office room

The model of the office room is suitable for one person. It has a floor space of 15 m² and a height of 2.7 m. It is enclosed by six boundaries of which the west oriented boundary adjacent conditions are the ambient conditions. The other boundaries border on neighbour rooms and have constant adjacent temperatures of 20 °C. The west oriented boundary is modeled as heavy construction. The other materials are wood for the ceiling two boundaries (walls) that border on neighbour rooms and concrete for the wall that separated the office room from the floor. Figure 3 shows a picture of the office room that was used as a basis to develop the room model.



Figure 3. Picture of the office room on which the model is based

The office room model is equipped with a floor heating system that works with a supply temperature of 35 °C. The model of the heating system calculates a heating power Q using (1). In (1) T_F is the temperature of the floor surface, T_R is the room temperature and A is the floor area (Recknagel, 2012). T_F is calculated from the supply temperature which is given to the heating system as input signal.

$$Q = 8.92 * (T_F - T_R)^{1.1} * A_F \quad (1)$$

The heating power Q is given to the room as input signal. The heating model has an input connector for its control signal. The required *control signal* must be Boolean. In case of *control signal* = true the heating system provides a heating signal of a certain amount of heat to the room model. If *control signal* = false the amount of heat provided by the heating system is zero.

Meeting room

The model of the meeting room is based on a small conference room at Fraunhofer IIS/EAS. It is designed for meetings and workshops for about 20 people (Figure 4).



Figure 4. Picture of the meeting room on which the model is based

The room model has a floor space of 52 m² and a height of 3.3 m. it has one outer wall, the west oriented wall, at which ambient conditions are applied. The boundary temperature of the opposite wall and the ceiling is constant 18 °C, and for the other walls, it is 20 °C. The four walls are modeled as heavy construction

from clinker bricks and plastering. The ceiling and the floor are modeled from lightweight concrete. Also an interior ceiling is included which is made of papier-mâché. The room model is equipped with a water heating systems (Figure 5) that gives radiation and convective heat via a radiator to the room model. The model of the water heating system, which is taken from the *BuildingSystems* library, is modelled as fluidic system. It contains a pump, pipes, a valve to regulate the volume flow, a radiator, an expansion vessel as well as a boiler. To regulate the volume flow of the water running through the radiator model, the valve model is controlled by its actuator position according to the valve characteristic, which specifies the volume flow rate depending on the valve position. Accordingly, the control signal calculated by a controller has to take values between 0 and 1.

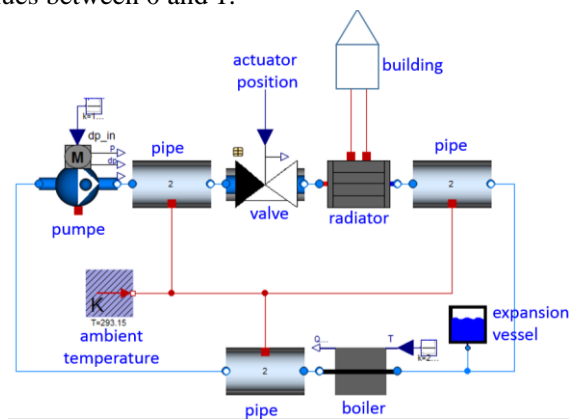


Figure 5. Water heating system

Classroom

The classroom model is based on data of the plus-energy primary school in Hohen Neuendorf near Berlin which was built as part of the research program *Energieoptimiertes Bauen (EnOB)* founded by the German Federal Ministry of Economic Affairs and Energy [Sick, 2015]. Figure 6 shows a picture of one classroom of the school, that was the basis for the classroom model.

The model has a floor space of 94 m² and a height of 4 m. It contains one outer wall that's boundary conditions are the ambient conditions.



Figure 6. Picture of the classroom on which the model is based

It is modeled as lightweight construction from wood, mineral wool and plasterboard. The inner walls are also lightweight and modeled from the materials wood and mineral wool. The floor consists of concrete, cement, bitumen, ethafoam and linoleum. The ceiling is modeled from concrete, cement and linoleum. The boundary temperatures of the inner walls are constant at 17 °C. The water based heating system is the same as described for the meeting room model.

2.2 Controller models

On the way of developing the methodology for choosing a suitable controller for a given room, the conformability of different controller models to the four introduced room models is investigated. Therefore, five common control strategies were chosen and modeled in Modelica. Partly they were developed in research projects at Fraunhofer IIS/EAS, partly they are taken from the Modelica Standard Library. The choice of the controller models can easily be extended. For example, the application of a model predictive controller is planned.

Two-Point Controller

The model of the two-point controller compares the actual room temperature with a required set point temperature. It provides a heating signal of 1 if the room temperature is below the set point temperature, which means the heating system should be turned on. The controller provides a heating signal of 0 if the room temperature is above the set point temperature. A hysteresis parameter prevents the controller from switching on and off permanently. This parameter influences the span between the given temperature set point and the actual temperature for turning the heating system on or off. Therefore, the height of the hysteresis influences the user comfort and the effective heating energy.

Forward-looking Switching (FS)

The aim of this controller is to determine the right moment to turn the heating system of a room model on and off to reach a desired target temperature at a specific point in time [Majetta, 2015]. Under the assumption of a given ambient temperature $T_A(t)$, a given supply temperature of a water heated heating system, respectively heating power in case of an electrical heating system $T_S(t)$, given temperatures of the adjacent rooms $T_N(t)$ and a given start value of the room temperature $T_R(t=0)$, the idea of this controller is to approximate the room temperature $T_R(t)$ by the response of a first order system $f(t)$ to a step change as first approximation. This is easily possible because the heating and cooling characteristics of single rooms are known neglecting disturbances and providing it with constant power. Figure 7 shows exemplarily the heating

of the office room model under constant ambient and neighbour room temperatures.

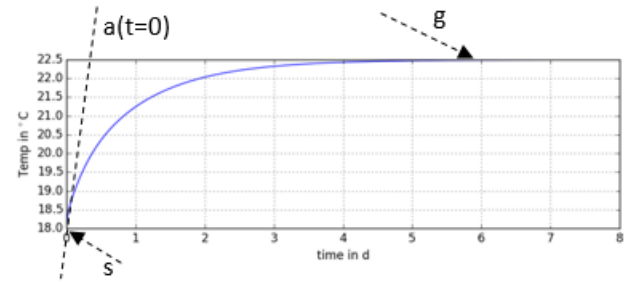


Figure 7. Heating up process of office room model

$f(t)$ is characterized by its steady-state value g , its start value s and its slope of the temperature change $a(t=0)$ at time $t=0$ and can be described by (2).

$$f(t) = g - (g - s)e^{\frac{-a}{g-s}t} \quad (2)$$

The variables g and a depend on $T_S(t)$, $T_R(t=0)$, $T_A(t)$ and $T_N(t)$. Internal loads are not considered. For simplicity reasons the temperatures of the neighbour rooms are chosen to be identical. The variables g and a are identified using results from particular simulations with defined constant values of $T_S(t)$, $T_R(t=0)$, $T_A(t)$, and $T_N(t)$. In the following, the identification process of g is shown exemplarily. To identify the dependency of the steady-state value g from $T_A(t)$, $T_N(t)$ and $T_S(t)$, the linear approach

$$g = T_A \cdot x_1 + T_N \cdot x_2 + T_S \cdot x_3 \quad (3)$$

was chosen. To identify the unknowns (x_1, x_2, x_3) in (3), n simulations under defined conditions were undertaken and the linear system

$$Ax = b \quad (4)$$

with

$$A = \begin{bmatrix} T_{A,1} & T_{N,1} & T_{S,1} \\ \vdots & \vdots & \vdots \\ T_{A,n} & T_{N,n} & T_{S,n} \end{bmatrix} \text{ and } b = \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}$$

is solved applying the least squares method [Isermann, 1991]. To do so, a solution x^* shall be calculated that minimizes the quadratic error $err = |Ax^* - b|^2$. x^* is calculated by solving the normal equation $A^T Ax^* = A^T b$. Using (2) for each value of T_A , T_N , and T_S , g can be calculated. A similar approach is applied for identifying the parameter a . The parameter s does not have to be identified since it is the start value of the room temperature, which simply can be taken from the simulation. Knowing g , s and $a(t=0)$, (2) is parameterized during the whole simulation of the room model and calculates the points in time for turning on or off the heating system online by transforming (1) to

$$t_{on,off}(t) = -\frac{g-s}{a} \ln\left(-\frac{w-g}{g-s}\right) \quad (4)$$

where w is the desired target temperature in the room. Since the heating and cooling process of the room model cannot be described exactly by an exponential function, two parameters were introduced to the controller model to optimize the point in time for turning the heating system on and off.

Combination of Two-point controller and Forward-looking Switching – Combi-Controller

Both of the introduced controllers have properties that might be considered as drawbacks. The two-point controller responds to a change of the set point not before the change happened, i.e. in case of a desired temperature that is warmer than the actual set point, the heating system starts to warm up the room at that moment the desired room temperature should be already reached. That means that it probably, especially in the case of a slow working heating system will be too cold in the room for some time. Once the desired target temperature is reached, the two-point controller works well within the tolerance given by the hysteresis.

The forward-looking switching has the ability to turn the heating system on and off at the right moment in order to achieve a desired target temperature in the future. However, once it is turned on, no mechanism is available to prevent the room from overheating. In the upper part of Figure 8 the room temperature is shown that results from controlling the room temperature with the two-point controller (red continuous line marked with dots) and the forward-looking switching controller (blue dashed line). The desired target temperature is pictured as the green continuous line. The temperature related heating signal is pictured in the lower part of Figure 8.

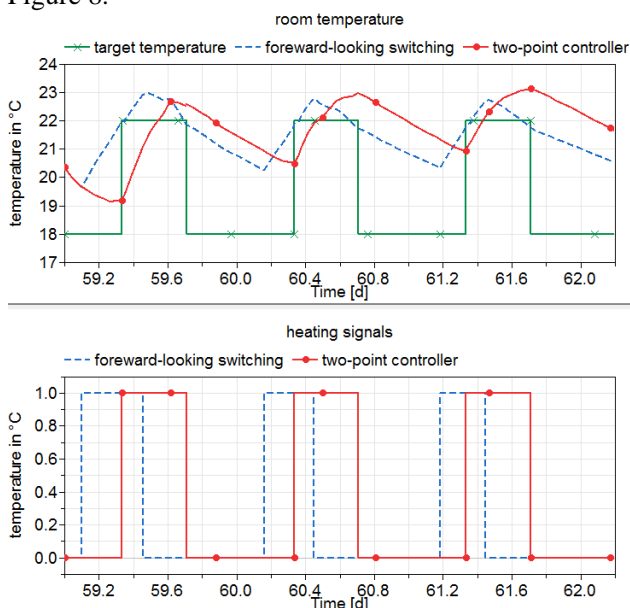


Figure 8. Room temperature and heating signal for two-point controller and forward-looking switching

Since both of the mentioned controllers have their described properties which might be drawbacks in some cases, the two controller approaches are combined and considered as the third type of room temperature controller within this study called combi-controller.

Statechart controller

Related to defined conditions (e.g. desired temperatures) certain states occur naturally (e.g. actual temperature too high or low). Those are identified as system states that require certain actions (e.g. heating) and represented as finite state machines (statechart). The statechart controller [Clauß, 2014] presented here controls the required target temperatures due to the occupancy of the room and it calculates the set points for the heating system to achieve those target temperatures. To realize those control actions two statecharts were developed that work together. Figure 9 shows one of those statecharts. It calculates the target temperatures dependent on the occupancy situation in the room

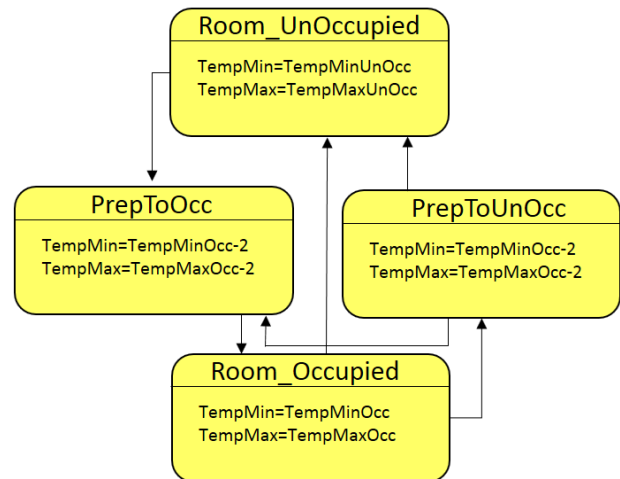


Figure 9. Occupancy statechart to calculate target temperature

The occupancy statechart contains the four states *Room_Unoccupied*, *PrepToOcc* (preparation state for oncoming occupancy), *Room_Occupied* and *PrepToUnOcc* (preparation state for oncoming leaving). Within those states different parameters are calculated, e.g. minimal (*TempMin*) and maximal (*TempMax*) temperatures and set points for heating and cooling for each state.

Values of control variables that have to be determined and decision to be taken, which are normally represented by transitions are calculated by a parameterized function approach which combines the available sensor values by a physically motivated equation. For example, the length of the preparation time (*PrepTimeOcc*) the room is pre-heated in order to reach the target temperature when persons enter or leave the room. With Tr – room temperature, $Tout$ – ambient temperature and $Tmin/Tmax$ – admissible minimal/

maximal room temperature the following heuristic and parameterized function approach to calculate the preparation time $PrepTimeOcc$ is used

$$PrepTimeOcc = c_0 + c_1(below(Tr, Tmin) + below(Tout, Tmin))^2 + c_2(above(Tr, Tmax) + above(Tout, Tmax))^2 \quad (5)$$

with

$$below(a, b) = \begin{cases} b - a & \text{if } a < b \\ 0 & \text{else} \end{cases} \quad (6)$$

$$above(a, b) = \begin{cases} a - b & \text{if } a > b \\ 0 & \text{else} \end{cases} \quad (7)$$

Equation (5) was chosen so, that the preparation time increases if the room temperature or the ambient temperature are outside the interval $[Tmin, Tmax]$. Other parameterized function approaches calculate the set point for the heating system respectively in the occupied or unoccupied state of the room. The parameters c_0, c_1, c_2 in (5) and other parameters of the statechart controller are determined by optimization.

The statechart controller is implemented in Modelica using if-then-else constructions.

P-controller

The fifth controller used in this study is a well-known p-controller with limited output. It is taken from the Modelica Standard Library (Modelica.Blocks.Continuous.LimPID).

3 Simulation Study

The aim of the work presented in this paper is to analyze the suitability of different control strategies to ensure a desired room temperature with possibly less net energy consumption. Therefore, each room model is simulated with each controller model. To ensure the comparability of the results, for each combination of room model and controller simulation scenarios were defined. Those guarantee the same simulation conditions e.g. ambient and boundary conditions of the room or the number of people entering the room at specified moments in time. The scenarios represent the usage of the rooms during heating periods (mid seasons, winter) since heating is the only action that can be done actively in the rooms (no cooling facilities are regarded).

In the following, one scenario is chosen to demonstrate the functioning of the controllers and further steps like their evaluation and optimization are discussed.

3.1 Scenario "Working Period" for the office room

This scenario was developed for the single office room. It comprises three working weeks from February 28 to March 21 including weekends. The daily working time from Monday to Friday is from 8.00 am to 5.00 pm. During this time, the desired target temperature is 22 °C. During absence of people the target temperature is 18 °C (Figure 10).

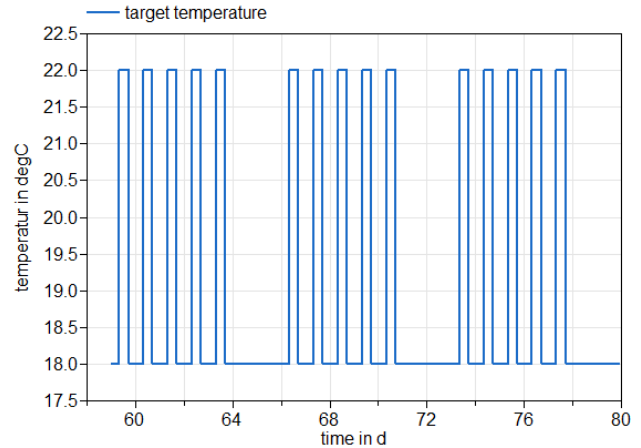


Figure 10. Target temperature of scenario "working period" for the office room

The aim of the controller models is to keep the office room temperature as close as possible to the set point while minimizing net energy consumption. To measure that the deviation between target and room temperature is calculated during occupancy and distinguished between *too warm* for times when the room temperature is more than 1 K higher than the target temperature and *too cold* if the room temperature is more than 1 K below the target temperature. To calculate the total times (*too warm total* and *too cold total*) *too warm* and *too cold* are integrated over the simulation time period. Looking at the example of the room temperature controlled by the two-point controller (Figure 11), the signal *tooCold* shows that at the beginning of every working day, it is too cold. Especially on Mondays, it is too cold nearly half of the day since the room temperature decreased a lot on the weekends before. The energy consumption is 159.6 kWh, the total time where it was too cold is 16.8 h and the total time where it was too warm in the room is 0.1 h.

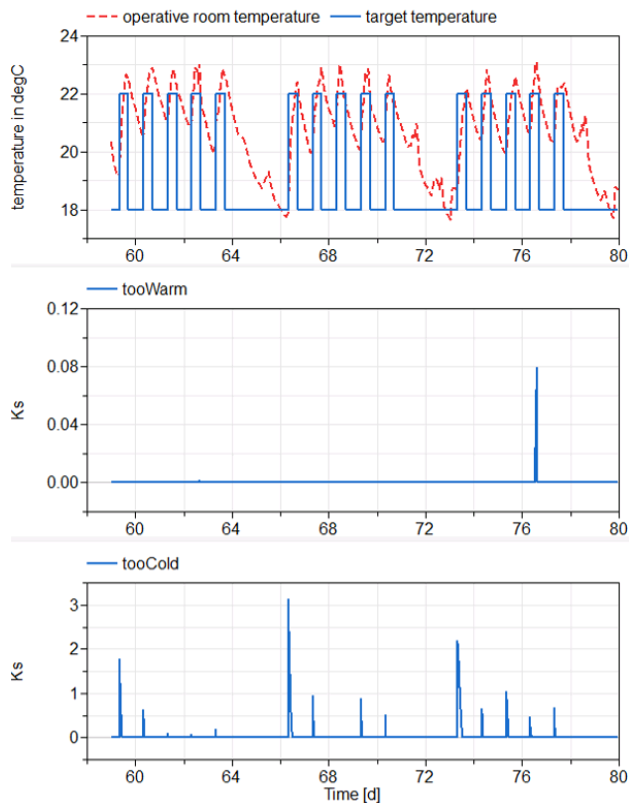


Figure 11. Room temperature and signals *tooWarm* and *tooCold* achieved by two-point controller

In comparison to the two-point controller, the statechart controller meets the required target temperature better however more net energy consumption is required. The reason for that is that before the new week begins, the room is being preheated in order to meet the high target temperature on Monday morning. Figure 12 shows the operative room temperature caused by the statechart controller as well as the signals *tooWarm* and *tooCold*. The energy consumption is 168.3 kWh and the total time where it has been too cold is 7.4 h. It is never too warm in the room.

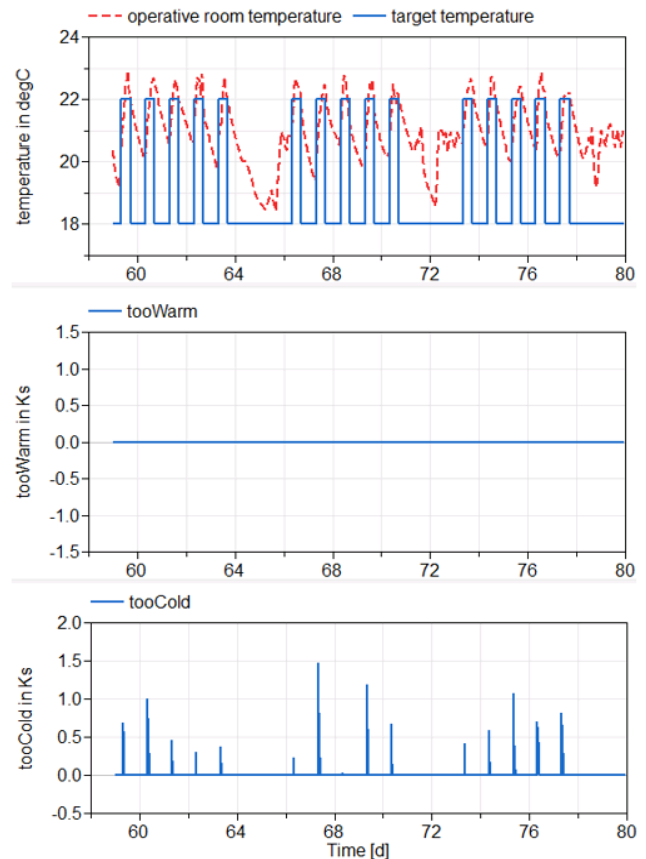


Figure 12. Room temperature and signals *tooWarm* and *tooCold* achieved by statechart controller

Table 1 shows the net energy consumption and the aggregated times when it was too warm or too cold in the room caused by the five introduced controllers. It can be seen that the combi controller containing the forward-looking switching controller and the two-point controller, needs the most net energy, however it meets the required target temperature best whereas the p-controller needs least energy at the price of the strongest comfort violation with regard to the times when it is too cold in the room. Figure 13 shows the room temperatures caused by the five different controllers as well as the desired target temperature

Table 1. Comparison of controllers regarding net energy consumption and violation of comfort boundaries

Controller	Energy in kWh	tooWarm (total) in h	tooCold (total) in h
two-point	161.1	2.1	8.3
FS	163.9	0.9	0
combi	175.5	0	0
statechart	168.3	0	7.4
p	137.31	0	32.4

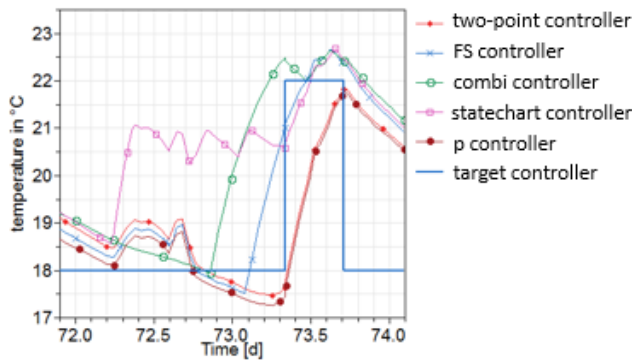


Figure 13. Room temperatures caused by the five controllers

The so far investigated eight simulation scenarios for each combination of the four room models and the five controller models show, that all of the controller models work different and fit better to one or another room models. Therefore, it is promising to take the type of the room and its HVAC equipment into consideration when choosing a suitable room temperature controller.

3.2 Further Steps

Up to now, the introduced controller models are parameterized from experience of the modeler. To achieve less energy consumption by sticking to the temperature comfort conditions, optimization of the controller parameters is considered. Optimization will be done by using the generic optimization program GenOpt [Wetter 2000]. GenOpt can be used with simulation programs that support textual based input/output functionality like EnergyPlus [EnergyPlus 2016], TRNSYS [TRNSYS, 2016] or Dymola [Dymola, 2016]. For the optimization, a cost function is needed that characterizes if a controller model is “good”. This evaluation will be done by rating the net energy consumption and the violation of the comfort specifications regarding the room temperature. Equation (8) shows the considered cost function in principle.

$$\text{cost} = \text{energy} + \text{penalty1} \cdot \text{tooWarmTotal} + \text{penalty2} \cdot \text{tooColdTotal} \quad (8)$$

In (8) *energy* denotes the net energy for heating the room, *tooWarmTotal* and *tooColdTotal* are the total times the room has been too warm or too cold respectively. The two *penalty* terms are weighting factors.

The principal optimization procedure was so far tested for the scenario *working period* for the office room controlled by the two-point controller using Dymola. The two-point controller has one free parameter, the hysteresis parameter that can be tuned in order to optimize the cost function. This parameter was allowed to vary within the boundaries of 0.1 and 5. The minimal cost function value was reached at a hysteresis parameter of 1.04. To verify this result, a parameter variation using the *mos-script* functionality in Dymola was operated which showed the same result. The net energy consumption decreased from 161.1 kWh for the

default hysteresis parameter value of 2 to 159.5 kWh for the optimized hysteresis parameter value.

After optimizing the controller parameters for all scenarios, sensitivity analyses of the optimized parameters regarding different locations (including different weather) of the rooms, different HVAC systems and other, still to be defined parameters, will be performed. Aim of this analysis is to figure out how robust the controller parameters are.

The subsequent step will be to find and establish criteria to assess the quality of the controller and hence to deduce rules for choosing a specific controller to a given room and decide if the controller needs special parameter adaption or if the default parameter will be sufficient.

4 Conclusion

The work presented in this paper are the first steps of a broad investigation with the aim to develop a methodology to provide rules and guidelines for choosing a suitable room temperature controller with regard to the given room and its installed HVAC technology. To achieve this goal a simulation study is performed. The instrument of simulation instead of measured data is used in this study for several reasons. First, one is considerably faster than real-time. Second, the investigations can be done under specified conditions and third, several scenarios can be elaborated and easily compared to each other. This paper presents the first steps of this investigation which is the development of four different representative room models with different heating systems like floor heating, electrical heating and radiator heating as well as five controller models of important controller types. Within the simulation of defined scenarios suitable for the type of the room (e.g. office room, classroom), the eligibility of the controller models is investigated. In addition, an outlook to further steps is given which will be the optimization of controller parameters including the definition of a cost function, a sensitivity analysis to study the robustness of the optimized controller parameters and the definition of criteria to evaluate the quality of the controller.

Acknowledgements

This paper is based on the results of the research project „Entwurfsverfahren für ganzheitliche Energiemanagementsysteme in Gebäuden (ENERMAT)“, funding reference 03ET1084A.



The authors are much obliged to the contributors of the research project Torsten Blochwitz, Eva Fordran,

Matthias Franke, Jeanette Floss, Jürgen Haufe, Jörg Hohlfeld, Edgar Liebold, Richard Meyer, Paul Pinther, Stephan Seidel and Jens Wurm.

References

- Abdolreza Rahmati, Farzan Rashidi, Mehran Rashidi: A hybrid fuzzy logic and PID controller for control of nonlinear HVAC systems. *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pp. 2249-2254, 2003.
- Abdul Afram, Farrokh Janabi-Sharifi. Theory and application of HVAC control systems – A review of model predictive control (MPC). *Building and Environment* 72, pp. 343-355, 2014. <http://dx.doi.org/10.1016/j.buildenv.2013.11.016>
- Alessandra Parisio, Marco Molinari, Damiano Varagnolo, et al.: A Scenario-based Predictive Control Approach to Building HVAC Management Systems. *IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 428-435, 2013.
- Christoph Clauß, Eva Fordran, Matthias Franke, et al. Entwicklung und Optimierung von Gebäude-Management-Systemen. Fifth German-Austrian IBPSA Conference, pp.166-173, Aachen, 2014.
- Christoph Nytsch-Geusen, Jörg Huber, Manuel Ljubijankic, Jörg Rädler. Modelica BuildingSystems – eine Modellbibliothek zur Simulation komplexer energietechnischer Gebäudesysteme. *Bauphysik* 35, Heft1, Ernst & Sohn Verlage für Architektur und technische Wissenschaften GmbH & Co. KG, 2013
- Deutscher Wetterdienst. Testreferenzjahre von Deutschland für mittlere, extreme und zukünftige Witterungsverhältnisse. Handbuch, 2014. Website: www.dwd.de
- Dymola Website: <http://www.3ds.com/products-services/catia/products/dymola>
- Elisa Carlon, Markus Schwarz, Christoph Schmidl, et al. Low Energy Houses Heated By Biomass Boilers: Optimization Of The Heating System Control Strategy By Means Of Dynamic Simulation. *3rd International High Performance Buildings Conference*, pp.3303/1 – 3303/8, 2014.
- EnergyPlus Website: <https://energyplus.net/>
- Frauke Oldewurtel, Alessandra Parisio, Colin N. Jones et al.: Energy efficient building climate control using stochastic model predictive control and weather predictions. *Proceedings of American control conference*, 2010
- Friedrich Sick, Sebastian Dietz, Gustav Hillmann, Margarethe Korolkow, Susanne Rexroth. Monitoring Plusenergie-Grundschule Hohen Neuendorf und IEA Task 41 (Solar Energy and Architecture). *Schlussbericht*, Berlin, 2015, Website: http://www.enob.info/fileadmin/media/Publikationen/EnBau/Projektberichte/27_Projektbericht_EnBau_0327430M_-_Monitoring_Schule_Hohen-Neuendorf.pdf
- Kristin Majetta, Christoph Clauß, Jürgen Haufe, et al. Design and Optimization of an Energy Manager for an Office Building. *ASIM/GI-Section Workshop – Simulation of Technical Systems & Methods in Modeling and Simulation*, pp. 289-296, 2015.
- Michael Wetter. Design optimization with GenOpt. *Building Energy Simulation User News* 21, p. 200, 2000.
- Patrick Lauenburg, Janusz Wollerstrand. Adaptive control of radiator systems for a lowest possible district heating return temperature. *Energy and Building* 72, pp. 132-140. 2014. <http://dx.doi.org/10.1016/j.enbuild.2013.12.011>
- Stephan Seidel, Christoph Clauß. Eva Fordran, et al.: Design and Optimization of Building Energy Management Systems. *Proceedings of 18th ITI Symposium*. pp. 329-337, November 9-11, Dresden, 2015.
- Rolf Isermann. Identifikation dynamischer Systeme 1, Springer-Verlag, 2. Auflage, pp. 189-195, 1991.
- TRNSYS website: <http://www.trnsys.com/>
- Yudong Ma, Francesco Borrelli, Brandon Hancey, et al.: Model predictive control of thermal energy storage in building cooling systems. *Proceedings of the 48th IEEE conference on decision and control*, pp. 392-397, Shanghai, China, 2009.
- Hermann Recknagel, Eberhard Sprenger, Ernst-Rudolf Schramek.: Taschenbuch für Heizung+Klimatechnik Oldenbourg Industrieverlag GmbH; p.804, 2011/2012.

Powertrain and Thermal System Simulation Models of a High Performance Electric Road Vehicle

Massimo Stellato¹ Luca Bergianti² John Batteh³

¹Dallara Engineering, Italy m.stellato@dallara.it

²Dallara Engineering, Italy l.bergianti@dallara.it

³Modelon Inc., Ann Arbor, Michigan USA john.batteh@modelon.com

Abstract

Performance and range optimization of electric vehicles are challenging targets in the design of contemporary automobiles. This paper illustrates that the thermal system and the development of the related control logic are key factors in achieving these targets. Both subjects benefit from the support of modeling and simulation. The paper describes our approach applied to a real case study.

The activity is the result of a cooperation between Dallara, responsible for the case study, and Modelon, developers of the libraries used to build the simulation model.

Keywords: electric vehicle, thermal system, control logic, powertrain, battery, cooling, range, derating.

1 Introduction

The goal is to evaluate the potential of the simulation models to define the thermal system architecture of an electric vehicle. This approach should allow the maximum degree of freedom for the control logic to be optimized later in the project, in order to optimize both vehicle range and performance.

Thanks to new technologies in the automotive field in general and for electric vehicles, a multi-physics approach to analyze the interactions between complex subsystems becomes necessary to evaluate the vehicle performance (Bouvy *et al.*, 2012). This need has led to the construction of a multi-physics model developed in the Modelica environment with components taken from four different application libraries: “Vehicle Dynamics”, “Liquid Cooling”, “Vapor Cycle” and “Heat Exchanger” (Modelon, 2016). The models are developed using Dymola (Dassault Systèmes, 2016).

2 Case Study Description

The activity reported in this paper supports the design of the thermal system of a real case high performance Sedan class electric vehicle which features three inboard motors (one at the front and two at the rear).

The thermal system architecture, the components and all the car data in this analysis reflect the real case.

3 Model Description

Figure 1 shows the top level of the “Systems Model”, comprising of the following sub-models using the libraries noted:

- Driver
- Powertrain (*Vehicle Dynamics Library*)
- Brakes (*Vehicle Dynamics Library*)
- Thermal System (*Liquid Cooling, Vapor Cycle and Heat Exchanger Libraries*)

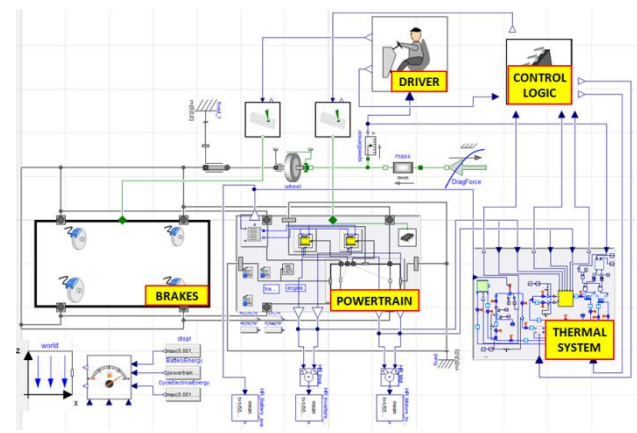


Figure 1. Systems model top view

The aim of the Systems Model is to match the reference speed profile (input) while considering the thermal limits of battery and powertrain together with the longitudinal vehicle performance (speed and acceleration) in order to calculate the range.

Figure 2 shows the main input and output of the model.



Figure 2. Model input and output

It's also possible to develop different driving cycles to represent typical real driving styles in order to evaluate their effect on the range.

The Systems Model can be interfaced with the multi-body vehicle model for vehicle dynamics analysis (Figure 3).

This approach allows both the refined analysis of the vehicle performance and the study of the cooling system at the same time.

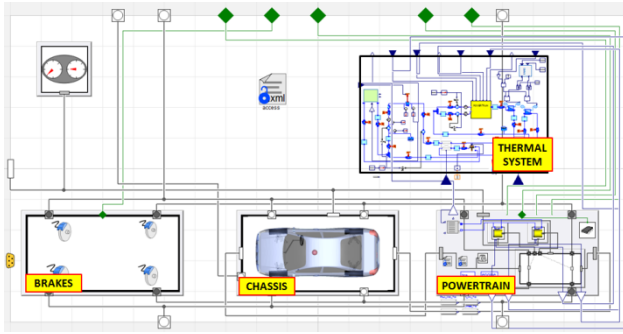


Figure 3. Full vehicle multibody model

3.1 Driver Model

The driver model, (see Figure 4), tries to match the speed profile (input) by varying the accelerator and brake pedal positions, which are respectively connected to the powertrain and brake models. If the coolant temperatures reach the limits, the control logic applies a progressive power derating as needed.

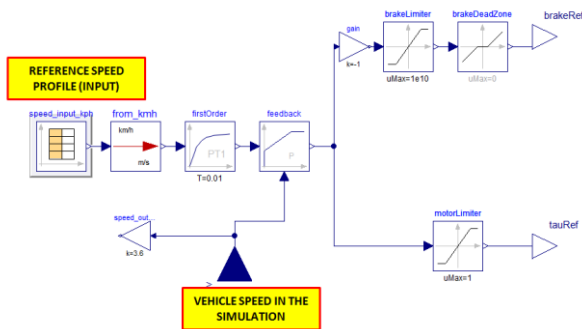


Figure 4. Driver sub-model

3.2 Powertrain Model

The powertrain model, illustrated in Figure 5, is composed by the following elements:

- Battery
- 3x Motors
- Driveline

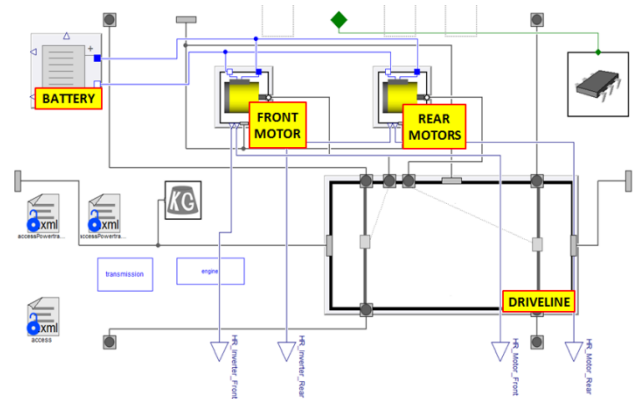


Figure 5. Powertrain sub-model

The battery model has both electric and thermal features with variable internal resistance and open circuit voltage as function of state of charge and cell temperature. The cell heat capacity is modeled with a lumped thermal element storing heat. The physical interaction between the cell and the coolant is modeled with a lumped thermal element transporting heat (Figure 6).

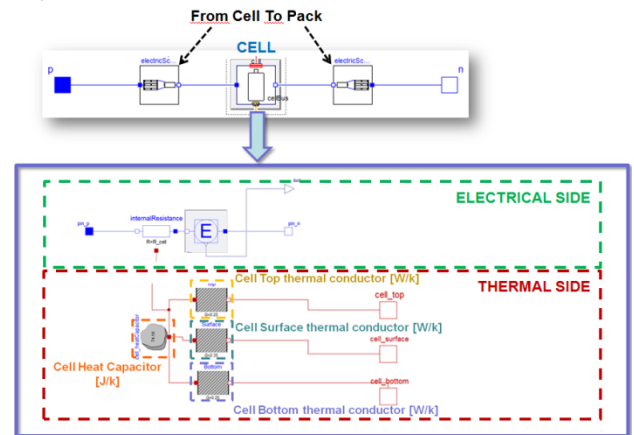


Figure 6. Battery sub-model

The model provides interfaces to the thermal system at the cell bottom, cell top or cell surface, as shown in Figure 7.

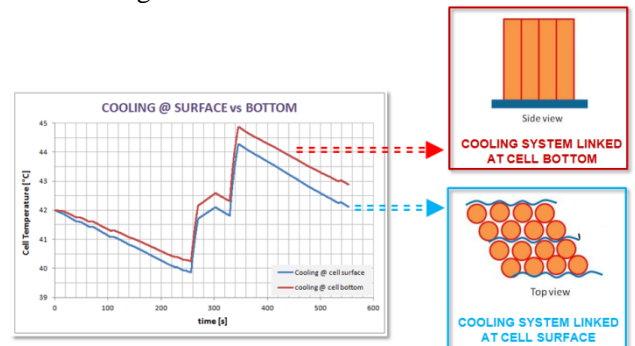


Figure 7. Cell temperature in the case of the thermal system linked to cell surface vs cell bottom

This approach looks at the irreversible heat generation due to the ohmic thermal loss caused by the battery's internal resistance, in the real case there's

also the effect of the reversible heat due to the chemical reactions in the battery electrodes (Schmitke *et al*, 2015).

A finite volume and finite element method would not represent a preferable alternative for system simulations as they slow down the simulation (Krüger *et al*, 2012) and still does not consider the effect of the reversible heat due to chemical reactions.

The battery energy consumption is used to predict the range via the state of charge calculated by integrating the generated current over time.

The motors are characterized in terms of both peak and continuous power curves, which depend on the available voltage at the inverter inlet. This voltage depends on the generated current, open circuit voltage and internal resistance of the battery. The motor and inverter efficiencies vary with torque and rpm. The model includes both winding and stator core temperatures. The winding is not interfaced with the thermal system. The equation describing the relationships between winding temperature and peak power is provided by the motor supplier. When the winding temperature reaches the limit value, the motor available power switches gradually from peak to continuous (Figure 8). The stator core is interfaced with the thermal system through coolant ducts (Figure 9); when the coolant temperature at the stator outlet reaches its limit, the control logic applies the power derating, as described later in the paper.

The stator core modeling consists of a thermal capacitance and a thermal conductance between stator and coolant which reflect the real geometry and material property.

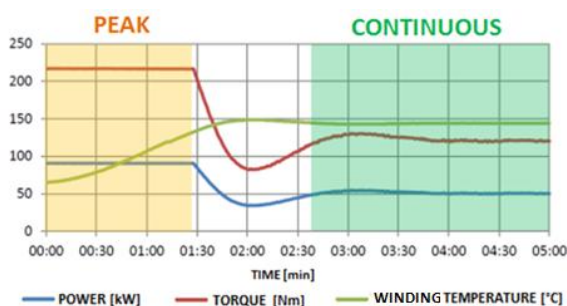


Figure 8. Winding temperature as function of peak and continuous power in a typical electric motor for automotive applications

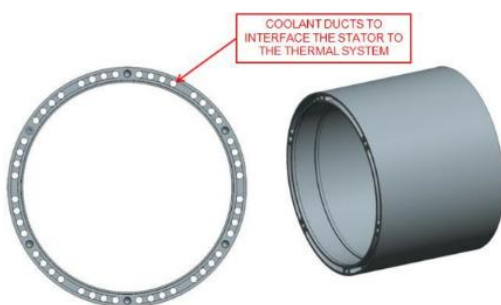


Figure 9. Stator of the electric motor

The energy recovery under braking is also considered: the motors assist the mechanical brakes by providing a torque of up to 10% that available in normal driving conditions; the energy recovery under braking affects both range and heat rejection.

An ABS / traction control model is included to avoid front and rear wheel spin during acceleration and braking.

3.3 Thermal System Model

The thermal system is the most innovative block of the systems model; its aim is to cool the battery and the powertrain as needed.

All the thermal components considered in the model are calibrated to match the behavior of the actual components. For example the radiator characteristic (Figure 10) has been provided by the supplier and validated by Dallara with experimental tests in the cooling rig (Figure 11).

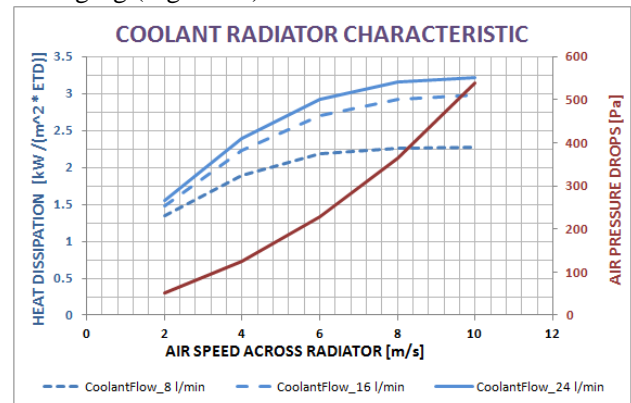


Figure 10. Coolant radiator characteristic: Heat dissipation normalized and air pressure drops

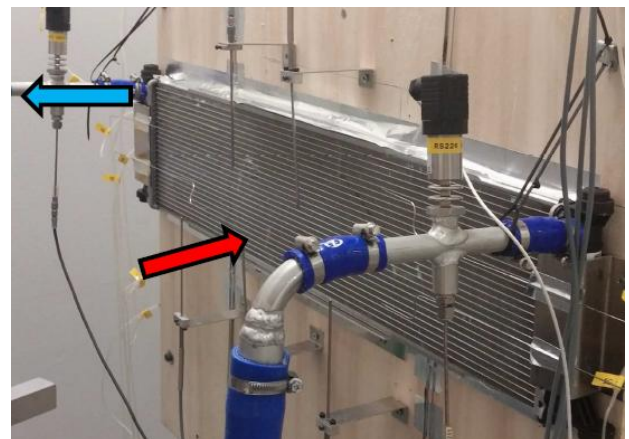


Figure 11. Radiator testing at the Dallara cooling rig

The thermal system is composed of multiple coolant radiators, complete with fans, and one chiller (plate heat exchanger).

The chiller utilizes the air conditioning refrigeration power to assist the radiators in cooling batteries and powertrain. The compressor's electrical power,

required to activate the chiller, affects the range calculation as well.

Through one or more 4-way valves, different architectures can be studied and allow the reconfiguration of the thermal system into multiple loops in order to cool both the battery and the powertrain as a single system or as separate systems (see Figure 12).

The modularity of the thermal system model allows the analysis of different architectures to select the best solution subject to the vehicle design constraints.

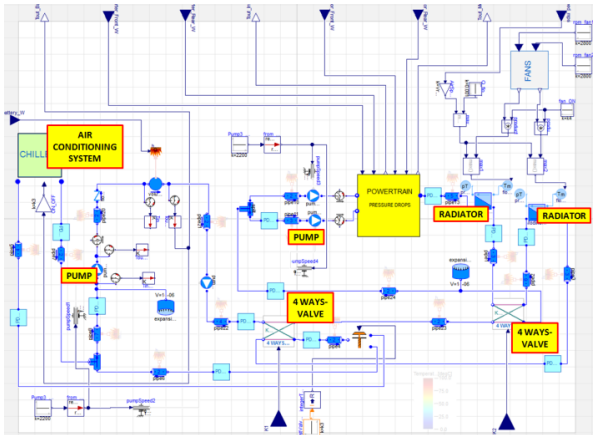


Figure 12. Thermal system sub-model

The model allows the performance of weight sensitivity analysis on the range; the effect of the weight in the configurations under investigation can be considered in the choice of the thermal system architecture.

The radiator cooling efficiency is a function of the air flow across radiator, which varies with vehicle speed and fan performance.

The air flow across the radiator is calculated by considering the maximum available between the effect of the vehicle speed and the performance of the fan (Figure 13). At low vehicle speeds, the airflow due to the fan is dominant; at higher vehicle speeds the air flow is essentially a function of the vehicle speed alone. In the real world these two effects interact and provide an even higher air flow rate.

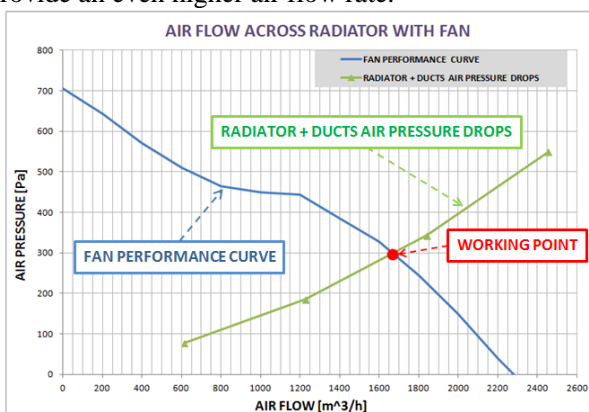


Figure 13. Fan performance curve vs radiator + duct air pressure drops

Figure 14 shows the thermal system components: chiller, radiator, battery, inverter and motor.

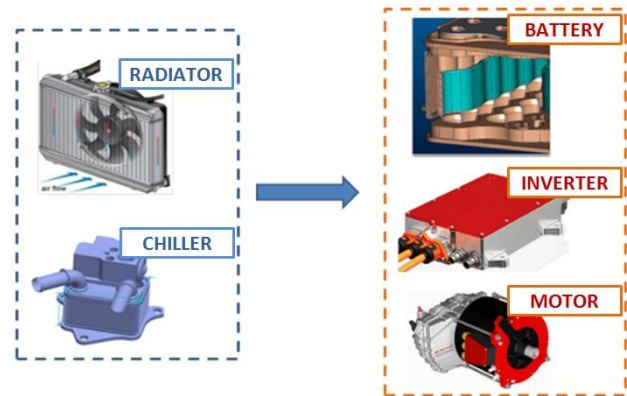


Figure 14. Thermal system components

The coolant flow rate has been calculated by considering the pump characteristic, the coolant pressure drops for each component and pipes' geometry (Figure 15). The heat exchange between pipes and the environment is also considered.

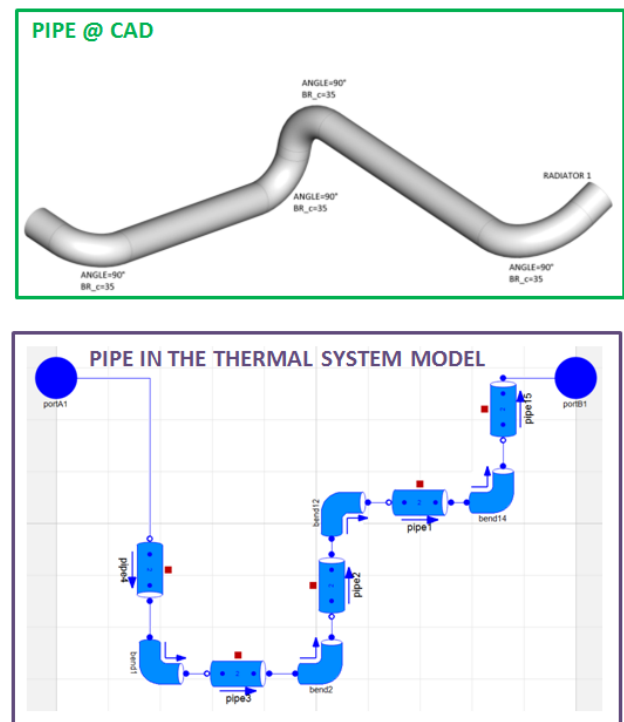


Figure 15. Pipes modeling in the thermal system, both distributed and concentrated pressure drops are considered for each pipe

The 4-way valve model, shown in Figure 16, has been developed with the Liquid Cooling Library starting from the model of the 3-way valve.

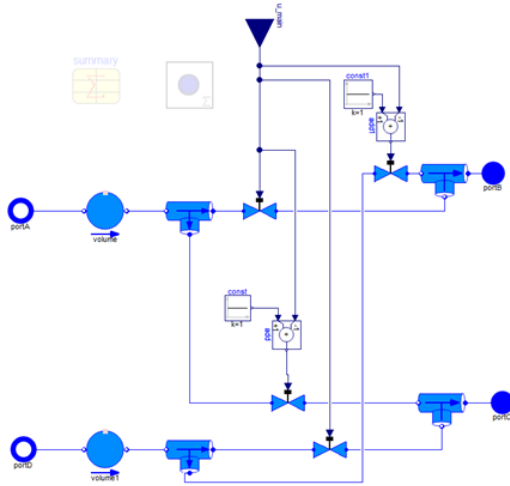


Figure 16. 4-way valve sub-model

3.4 Control Logic Model

The control logic analyzes the battery and powertrain coolant temperatures and continuously switches to the most efficient cooling loop configuration among those available through the thermal system model.

If the coolant temperatures reach the limits, then the control logic applies power derating.

4 Approach

The main purpose of designing the thermal system of an electric vehicle is to optimize the vehicle range and minimize power derating.

The chiller reduces the battery power which then makes it necessary to minimize its use to optimize the range. Moreover, as the chiller power demand to cool the battery increases, less power is available for the air conditioning of the vehicle interior, with negative implications on passenger comfort (Krüger *et al.*, 2012).

The battery and powertrain cooling requirements vary throughout the simulation, as they depend on both the instantaneous power required to match the reference speed profile (Krüger *et al.*, 2012), and battery and powertrain efficiency; in some conditions the powertrain requires more cooling than the battery while the opposite holds true in other conditions.

For this reason, a variable thermal system architecture is more efficient than a fixed layout in both the case of low heat rejection values to minimize the chiller use and with high heat rejection values to minimize the power derating. This variable architecture is configurable during vehicle operation in order to favor battery cooling over powertrain cooling or vice versa, depending on the instantaneous cooling requirements.

Following this approach, a variable thermal system layout has been designed to switch between three different configurations.

While the specifics of the variable system configuration are confidential, the general architecture, which shows how the coolant flow path is arranged in each configuration, is reported in Figure 17. The grey blocks represent radiators, chiller, battery and powertrain.

An outline of the three configurations is given below:

- **Config 0** → All components in series to cool battery and powertrain in a single system (Figure 17a). This configuration is suitable for low heat rejection requirements of both battery and powertrain with moderate ambient temperature;
- **Config 1** → Components in separate loops (Figure 17b) to independently cool battery and powertrain. This first “two-loops” configuration caters for high battery heat rejection and medium powertrain heat rejection requirements with medium and high ambient temperatures;
- **Config 2** → Components in separate loops (Figure 17c) to cool independently the battery and powertrain. This second “two-loops” configuration caters for medium battery heat rejection and high powertrain heat rejection requirements with medium and high ambient temperatures.

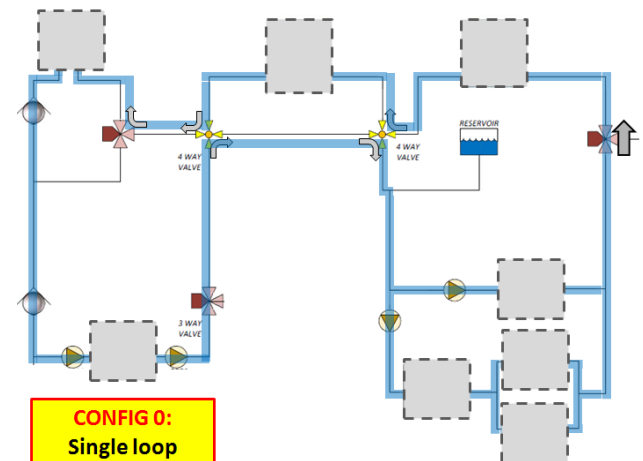


Figure 17a. Thermal system configuration 0

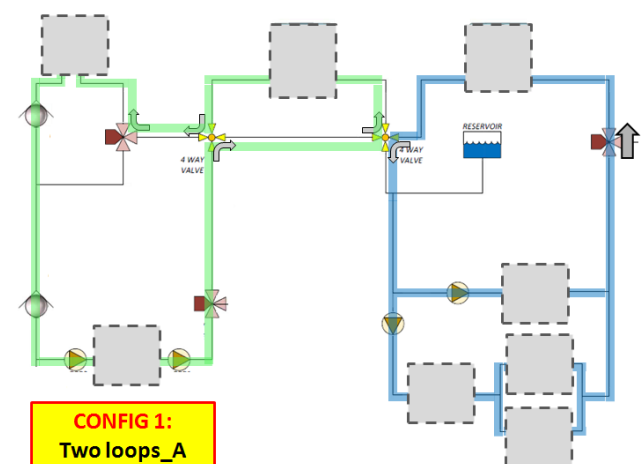


Figure 17b. Thermal system configuration 1

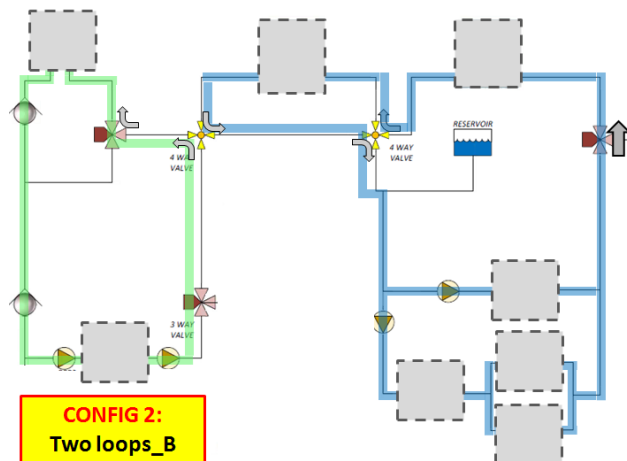


Figure 17c. Thermal system configuration 2

By monitoring the battery and powertrain coolant temperatures, the best configuration among the three choices is continuously selected by the control logic that switches from one configuration to another through two 4-way valves.

When either the battery or the powertrain coolant temperature approaches the limit values, a gradual power derating is applied.

The amount of coolant in the system plays an important role as it affects the thermal inertia and therefore the time before reaching the maximum temperature values. More coolant in the system, allows running in unstable conditions (i.e. extreme acceleration) for longer periods of time, before the control logic starts to degrade power.

This effect is shown in Figure 18, where the temperatures (considering coolant quantities of 8l and 24l in the system) are calculated for the same heat rejection profile.

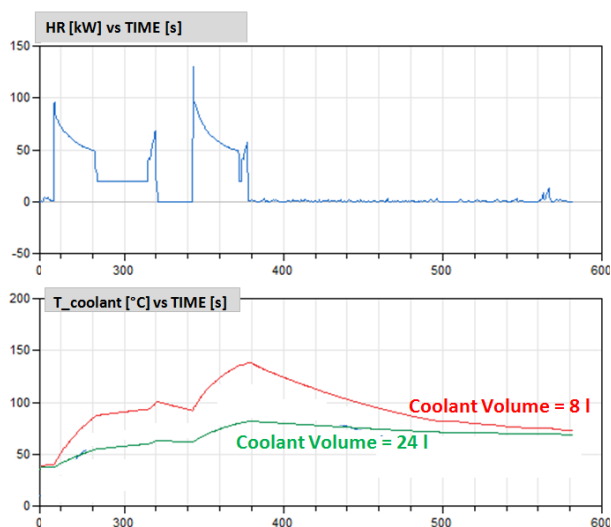


Figure 18. Coolant temperature sensitivity with coolant volume.

5 Results

The results reported in Figures 20-22 show the comparison between the variable thermal system architecture defined by this activity and three different fixed architectures (config 0, 1, 2 of Figure 17) with the same radiators and chiller of the variable thermal system. The driving cycle considered (Dimensioning Cycle) is confidential; it was developed to represent an aggressive use of the vehicle in terms of cooling requirements, Figure 19 summarizes the main input and output. A maximum available cooling power of 3 kW is considered for the chiller. The vehicle range is calculated with and without considering the power consumption due to the thermal components (fans, pumps and compressor). The impact of the thermal considerations of the system results in a roughly 8% decrease in vehicle range.

DIMENSIONING CYCLE								
INPUT					OUTPUT			
CONFIG	Tamb [°C]	T coolant start [°C]	T Battery cell start [°C]	Max Chiller Power [kW]	Range with thermal loads [km]	Range without thermal loads [km]	Compressor Energy [kWh]	TOP SPEED [kph]
variable thermal system	33	42	42	3	188	205	0.50	220

Figure 19. Dimensioning Cycle, input and output, variable thermal system architecture

The vehicle speed profile achievable with the variable thermal system architecture matches the Dimensioning Cycle speed profile (input) much better than the one provided by a fixed architecture, which needs more power derating (Figure 20-21). The performance gain of the variable thermal system architecture could be increased by optimizing the control logic.

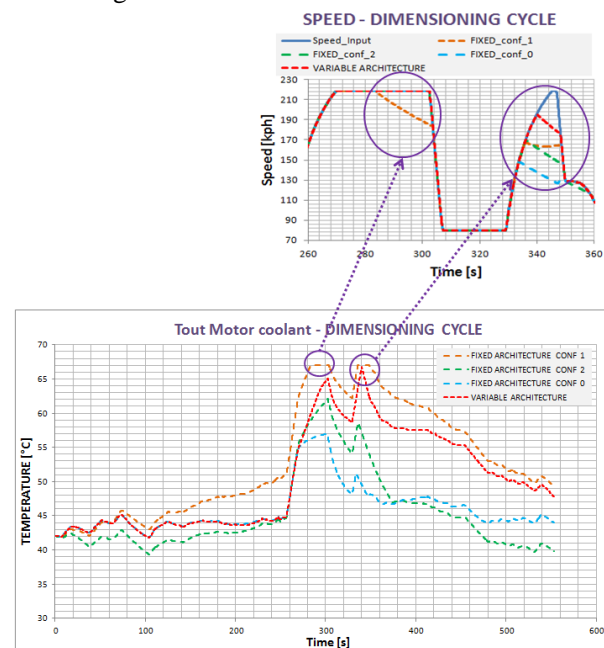


Figure 20. Dimensioning Cycle, Tout motor coolant, variable vs fixed thermal system architecture

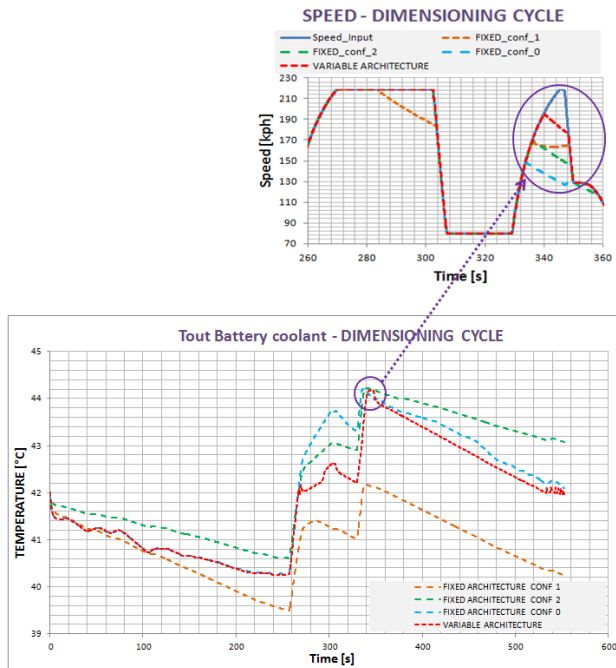


Figure 21. Dimensioning Cycle, Tout battery coolant, variable vs fixed thermal system architecture

Figure 22 shows the battery cell's temperature profile, achieved with the variable thermal system architecture in comparison with the profiles achieved with the fixed architectures.

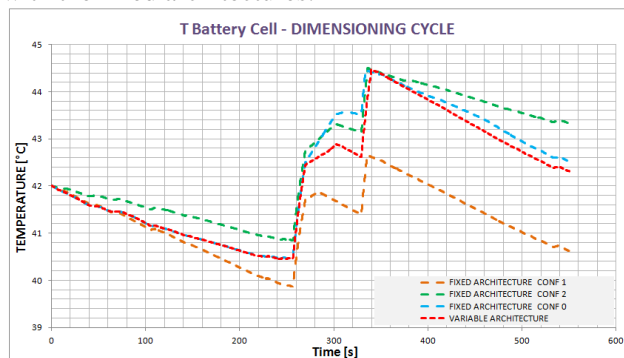


Figure 22. Dimensioning Cycle, T battery cell, variable vs fixed thermal system architecture

Figure 23 shows that the battery cell temperature is higher than the coolant temperature because of the thermal conductance of the cell.

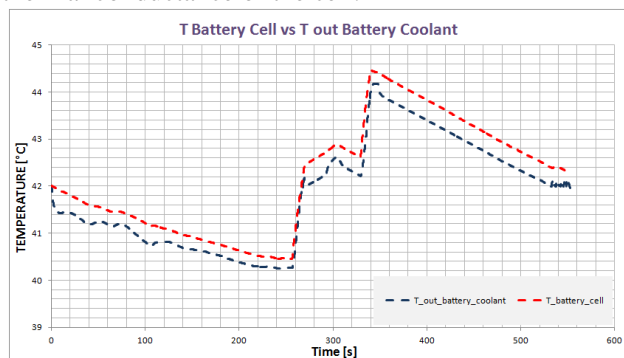


Figure 23. Dimensioning Cycle, T battery cell vs T out battery coolant, variable thermal system architecture

The thermal system configuration, managed by the control logic, changes during the simulation to best cope with the cycle requirements (Figure 24).

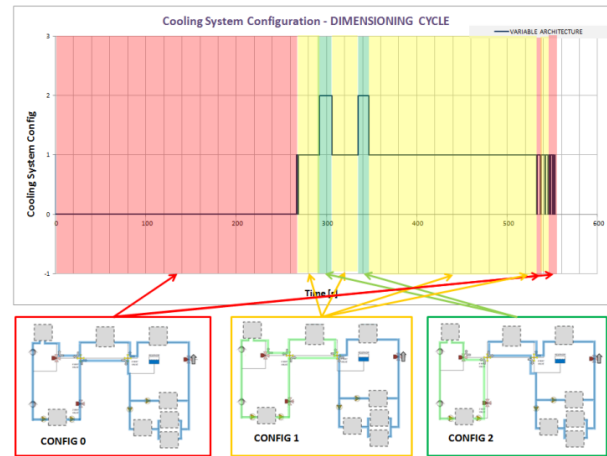


Figure 24. Dimensioning Cycle, thermal system configuration, variable thermal system architecture

The total mechanical power (front motor + rear motors) required to perform the Dimensioning Cycle is reported in Figure 25.

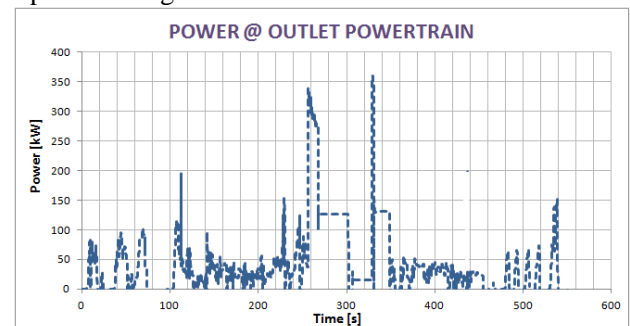


Figure 25. Dimensioning Cycle, total power (front + rear_1 + rear_2) @ outlet powertrain, variable thermal system architecture

Figure 26 shows that for low acceleration levels, the cooling demand from the powertrain is greater than the cooling demand from the battery; at high acceleration levels the opposite is true.

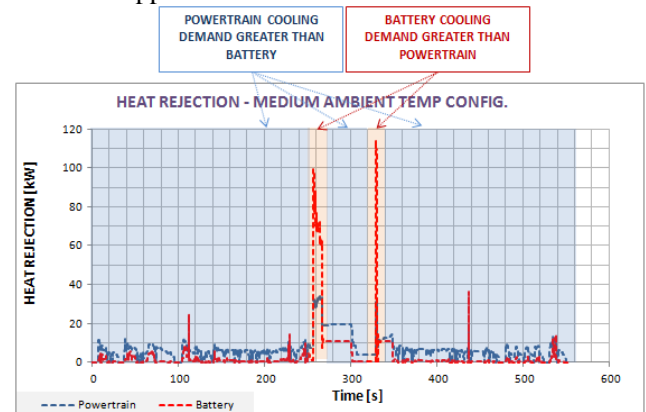


Figure 26. Dimensioning Cycle, powertrain heat rejection (front + rear_1 + rear_2) vs battery pack heat rejection, variable thermal system architecture

Further analyses were performed on the homologation cycles US06 and NEDC where power derating is not required. All the simulations are performed considering the same SOC start (0.95) and SOC end (0.15).

The homologation cycles are generally of low thermal demand, consequently the chiller is typically not necessary and a variable thermal system not required. In the real world, or considering more aggressive cycles, the advantages (energy saved) related to the introduction of a variable system architecture (with Chiller) could represent a noticeable increase of the range (up to 15 km), to further minimize the power derating as reported for the dimensioning cycle.

Figure 27 summarizes the main input and output for the US06 Cycle.

US06 CYCLE						
INPUT				OUTPUT		
CONFIG	Tamb [°C]	Twater start [°C]	Battery cell temperature start [°C]	Range with thermal loads [km]	Range without thermal loads [km]	TOP SPEED [kph]
variable thermal system	33	42	42	380	382	130

Figure 27. US06 Cycle, input and output, variable thermal system architecture

Figure 28 shows that the vehicle speed matches the US06 speed profile (input) without derating.

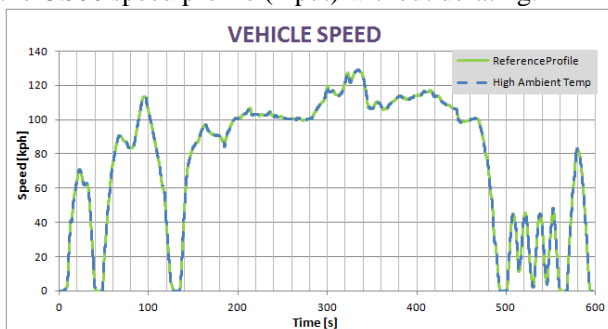


Figure 28. US06 Cycle, reference speed profile (input) vs vehicle speed, variable thermal system architecture

Figure 29 reports the motor outlet coolant temperature in the US06 cycle.

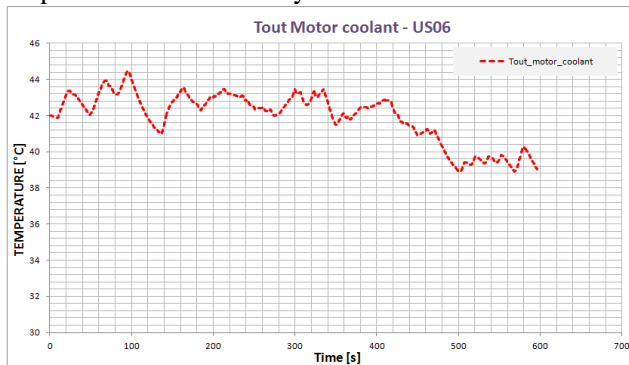


Figure 29. US06 Cycle, Tout motor coolant, variable thermal system architecture

Figure 30 reports the battery outlet coolant temperature and the battery cell temperature in the US06 cycle, the chiller is turned off.

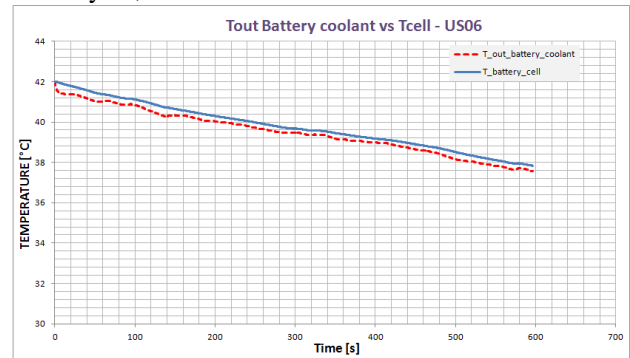


Figure 30. US06 Cycle, T battery cell vs T out battery coolant, variable thermal system architecture

Figure 31 summarizes the main input and output for the NEDC Cycle.

NEDC CYCLE						
INPUT				OUTPUT		
CONFIG	Tamb [°C]	Twater start [°C]	Battery cell temperature start [°C]	Range with thermal loads [km]	Range without thermal loads [km]	TOP SPEED [kph]
variable thermal system	33	42	42	531	534	120

Figure 31. NEDC Cycle, input and output, variable thermal system architecture

Figure 32 shows that the vehicle speed matches the NEDC speed profile (input) without derating.

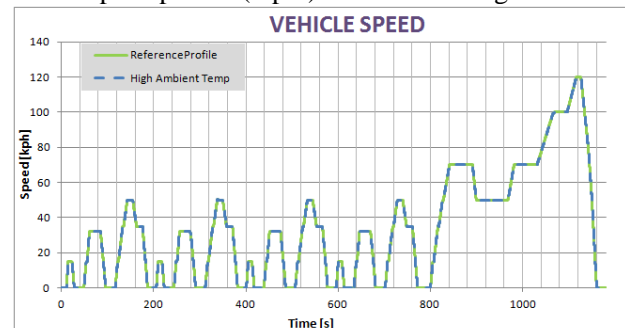


Figure 32. NEDC Cycle, reference speed profile (input) vs vehicle speed, variable thermal system architecture

Figure 33 reports the motor outlet coolant temperature in the NEDC cycle.

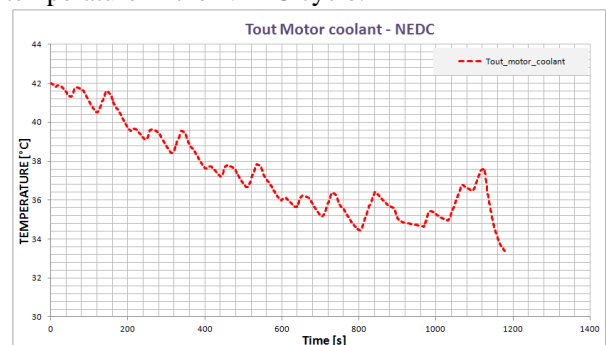


Figure 33. NEDC Cycle, Tout motor coolant, variable thermal system architecture

Figure 34 reports the battery outlet coolant temperature and the battery cell temperature in the NEDC cycle, the chiller is turned off.

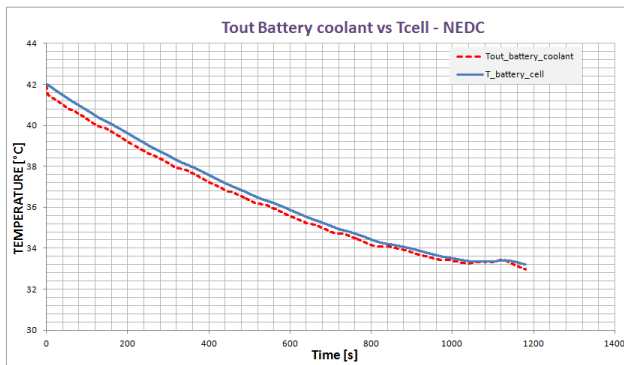


Figure 34. NEDC Cycle, T battery cell vs T out battery coolant, variable thermal system architecture

A weight sensitivity analysis has been performed on the NEDC cycle; an increase of 10 kg reduces the range by 2 km.

The homologation cycles were also helpful for a first validation of the system's model, because the car manufacturers typically declare the vehicle's range on these cycles. A systems model with the architecture and available data of a benchmark vehicle has been developed, achieving range results on US06 and NEDC cycles aligned with those declared by the benchmark vehicle constructor.

The last analysis reported (Figure 35) concerns the total electric power needed to cool the battery during the fast charge, considering a battery power supplies of 100 kW, 110 kW and 150 kW; these are representative of the typical and maximum values used in the real case for the fast charge of the electric vehicles.

In all three cases analyzed the pumps and fans are kept at max rpm, the chiller power is the power in surplus at the radiator needed to keep the coolant battery temperature below its limit.

FAST CHARGE ANALYSIS @ Tamb 35 °C, T_coolant_out_bat 45 °C

Battery Power Supply [kW]	Battery Heat Rejection [kW]	Radiator Heat Dissip [kW]	Coolant flow [l/min]	COP HVAC	Air Speed @ rad [m/s]	Pump Electric Power [kW]	Fan Electric Power [kW]	Chiller Heat Dissip [kW]	Compres Electric Power [kW]	Total Electric Power [kW]
100	3.8	3.8	15	1.3	5.5	0.10	0.32	0	0.0	0.4
110	4.6							0.8	0.6	1.0
150	8.9							5.1	3.9	4.3

Figure 35. Fast charge analysis, the total electric power required to cool the battery is reported in red color

6 Conclusions and Further Developments

The activity described in this paper was useful to evaluate the potential of the simulation model and to define the thermal system layout for a real case study.

The performance gains of a variable thermal system architecture with respect to a fixed architecture have been detailed.

The model continues to support and evolve with the case study and can be fully validated in the future with real vehicle tests, as well as being used as a starting point for future electric vehicle projects.

Ongoing work with this model to further support the case study includes the following:

- Powertrain and thermal system control logic optimization.
- Analysis of the battery heating required in low ambient temperature conditions, which constitutes another critical point in the design of Electric Vehicles (Bouvy *et al*, 2012).
- Analysis, supported by experimental test, of the fans and vehicle speed interaction for the air flow rate across the radiators.
- Battery model with electrochemical features development, which describes the battery physics in detail (Schmitke *et al*, 2015).
- Interface with the vehicle multi-body model for real time applications at Dallara Dynamic Driving Simulator.
- Air conditioning system development with effects on passenger human comfort.
- Active grill shutter model development (Batteh *et al*, 2014).

References

- J. Batteh, S. Chandrasekar and J. Gohl. Integrated Vehicle Thermal Management in Modelica: Overview and Applications. *Proceedings of 10th International Modelica Conference*, pp. 409-418, 2014
- C. Bouvy , P. Jeck, J. Gissing, T. Lichius, L. Ecksterin. Holistic Vehicle Simulation using Modelica – An Application on Thermal Management and Operation Strategy for Electrified Vehicles. *Proceedings of 9th International Modelica Conference*, pp. 263-270, 2012.
- C. Bouvy, P. Jeck, S. Ginsberg, P. Jeck, B. Hartmann, S. Baltzer and L. Eckstein. Holistic Battery Pack Design. *Aachen*, pp. 367-380, 2012.
- I. Krüger, A. Mehlhase and G. Schmitz. Energy Consumption of Battery Cooling In Hybrid Electric Vehicles. *Proceedings of 14th International Refrigeration and Air Conditioning Conference*, 2012.
- I. Krüger , A. Mehlhase and G. Schmitz. Variable Structure Modeling for Vehicle Refrigeration Applications. *Proceedings of 9th International Modelica Conference*, pp. 927-934, 2012.
- C. Schmitke and T. Son Dao. Developing Mathematical Models of Batteries in Modelica for Energy Storage Applications. *Proceedings of 11th International Modelica Conference*, pp. 469-477, 2015.
- Dassault Systèmes. Dymola 2017., 2016. www.Dymola.com

- Modelon. Heat Exchanger Library. Version 1.4.1, 2016.
www.modelon.com/products/modelicalibraries/heat-exchanger-library/
- Modelon. Liquid Cooling Library. Version 1.5, 2016.
www.modelon.com/products/modelicalibraries/liquid-cooling-library/
- Modelon. Vapor Cycle Library. Version 1.3, 2016.
www.modelon.com/products/modelica-libraries/vapor-cycle-library/
- Modelon. Vehicle Dynamics Library. Version 2.3, 2016.
www.modelon.com/products/modelica-libraries/vehicle-dynamics-library/

Investigating the Effect of a Sonic Restrictor in the Intake of an Engine

Maura Gallarotti Alessandro Picarelli Mike Dempsey

Claytex Services Ltd., Edmund House, Rugby Road, Leamington Spa, CV32 6EL, UK
{maura.gallarotti, alessandro.picarelli, mike.dempsey} @claytex.com

Abstract

The air induction system is one of the engine subsystems that most influences fuel efficiency and power generation, especially in restricted race engine applications.

In this paper, the quasi-1D model of a sonic restrictor is presented, together with its integration in an engine model, in order to investigate the behaviour of the engine power and torque when the choked condition is reached.

The study shows how power and torque curves are affected when a sonic restrictor is installed within the intake system and outlines the need of detailed simulations in a restricted engine development process, to avoid steep engine power reductions at high speeds.

Keywords: sonic restrictor, choked flow, engine, MVEM, intake manifold.

1 Introduction

The development of high-fidelity predictive models of vehicle engines is one of the main objectives of powertrain simulation engineers. Dymola is a convenient software for vehicle and engine modelling, since the underlying Modelica language is suited to complex multi-domain systems.

However, as Dymola is mainly limited to 0D-1D thermofluid systems, engineers can face some intricacies in modelling the more complex flows happening in engines. To get better results, CFD simulations can be performed, but often at the cost of losing the integration with the mechanical part of the model and losing any real-time simulation capability.

This paper shows that although Dymola is not a CFD code, it can handle the inherent non-linearities of the nozzle flow that arise in the transition from the subcritical to the choked state.

A sonic restrictor is a converging-diverging nozzle installed in the intake system in order to limit the maximum power output of the engine by limiting the mass flow of air flowing into the cylinders.

In situations where the engine would require a higher mass flow than the one allowed by the nozzle,

the constraint of sonic flow velocity at the throat limits the mass flow and a shock takes place in the divergent section of the nozzle, thus introducing strong pressure losses that ultimately limit the engine power.

Sonic restrictors are being used in several motorsport championships in order to equalize the maximum power of the engines. Among the racing series that make use of air restrictors are the Formula 3, the Formula SAE, the FIA GT1 World Championship, Le Mans Series and several others. Sometimes sonic restrictors are also used in road applications for de-rating purposes, mainly in motorbike engines.

In restricted engines, a reliable model of the air induction system is of paramount importance, as the flow in the sonic restrictor has direct effects on power generation and fuel efficiency.

The challenge when modelling a converging diverging nozzle lies in the asymmetric behaviour of the flow before and after the shock, with equations for the subsonic flow being very different from the ones used for supersonic conditions.

If not implemented in an efficient way, such a physical problem could trigger in Dymola several events and non-linear iterations, making the model computationally expensive.

2 The sonic restrictor model

The sonic restrictor is modelled in Dymola as a component where steady momentum, continuity and energy balances are performed. It uses the fluid connectors from Modelica.Fluid (Casella, F. et al., 2006) which make it fully compatible with the Modelica Standard Library.

As in pipes, the pressures at inlet and outlet determine the mass flow rate, the flow goes from the higher to the lower pressure and a greater pressure difference gives a higher mass flow.

For a pressure drop weaker than the critical one, the flow accelerates in the converging section and decelerates in the diverging one isentropically.

If the pressure at the throat equals the critical one, the nozzle becomes choked and the sonic condition is reached (see Equation 1, where p_c is the critical

pressure and p_i the inlet pressure). Being the speed of sound the speed of propagation of small disturbances, the Mach at throat cannot be higher than 1, as the flow upstream the throat does not receive any information about what is happening downstream.

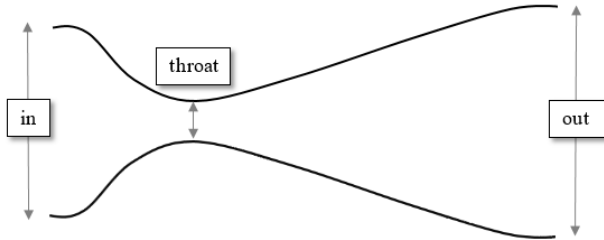


Figure 1. Sonic restrictor.

$$\frac{p_c}{p_i} = \left(\frac{2}{\gamma + 1} \right)^{\frac{\gamma}{\gamma - 1}} \quad (1)$$

If the pressure at the throat is further reduced, the mass flow at the throat is limited to the critical value, the flow becomes supersonic in the diverging section and a shock occurs. The critical mass flow rate can be found using Equation 2, where p^o is the total pressure and T^o the total temperature.

$$\dot{m}_c = \frac{A \cdot p^o}{\sqrt{T^o}} \sqrt{\frac{\gamma}{R}} \left(\frac{\gamma + 1}{2} \right)^{-\frac{\gamma + 1}{2(\gamma - 1)}} \quad (2)$$

For air, with $\gamma = 1.4$, the critical pressure ratio (given by Equation 1) is 0.528, meaning that in a nozzle, the sonic condition is reached when the pressure at the throat is lower or equal to 0.528 times the pressure at inlet.

In all the other air ducts of the engine, the Mach is much lower than 1 and the continuity equation for a subsonic flow states that a decrease in area causes an increase in velocity. However, if the flow becomes supersonic, the flow behaviour changes and Equation 3 tells us that an increase in velocity is associated with an increase in area. In fact, the flow accelerates in the diverging section of a choked nozzle.

$$\frac{dA}{A} = (M^2 - 1) \frac{du}{u} \quad (3)$$

In a choked nozzle, the flow accelerates isentropically from the inlet and its static pressure decreases maintaining a constant total pressure, until a shock occurs in the diverging section.

A shock is a discontinuity in the flow field across which the flow abruptly slows down from a supersonic to a subsonic speed while increasing the static pressure

with huge associated viscous losses that reduce the total pressure.

The Mach numbers upstream (u) and downstream (d) of the shock are related by Equation 4 (Anderson J., 2002)

$$M_d^2 = \frac{1 + \frac{\gamma - 1}{2} M_u^2}{\gamma M_u^2 - \frac{\gamma - 1}{2}} \quad (4)$$

This equation states that the further along the nozzle the shock occurs, the stronger it will be: as the Mach upstream increases above 1, the normal shock becomes stronger and M_d becomes progressively less than 1, decreasing the total pressure of the flow leaving the sonic restrictor and entering the cylinders.

From a practical point of view, this means that the section in which the shock will occur will influence the total pressure of the flow entering in the cylinders.

The total pressures upstream and downstream the shock are linked by Equation 5 (Anderson J., 2002):

$$\frac{p_d^o}{p_u^o} = \left[1 + \frac{2\gamma}{\gamma + 1} (M_u^2 - 1) \right] \left[\frac{1 + \frac{\gamma - 1}{2} M_d^2}{1 + \frac{\gamma - 1}{2} M_u^2} \right]^{\frac{\gamma}{\gamma - 1}} \quad (5)$$

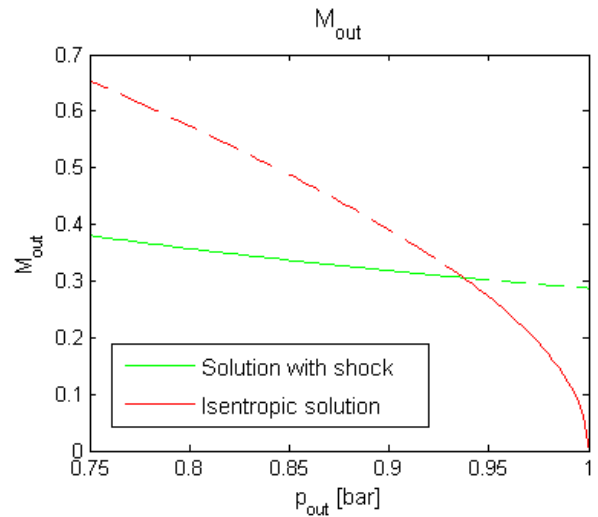


Figure 2. Discontinuity in the Mach number at the nozzle outlet going from a subsonic (red curve) to a supersonic (green curve) flow.

Figure 2 shows the Mach at the outlet of the sonic restrictor both when the sonic condition is not reached (red curve) and in case of shock (green curve). For a given inlet pressure, if the throat pressure is higher than the critical value, the flow remains isentropic, while if the throat pressure decreases further, the sonic condition is reached and the outlet Mach decreases drastically.

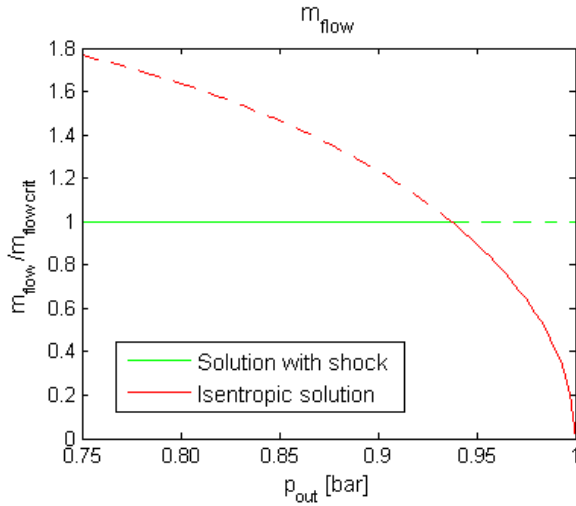


Figure 3. Discontinuity in the mass flow going from a subsonic (red curve) to a supersonic (green curve) flow.

Figure 3 shows the mass flow through the sonic restrictor for a subsonic flow (red curve) and for a supersonic flow (green curve).

The transition from the subsonic to the supersonic condition has been implemented in Dymola avoiding the use of *if* statements, as conditional expressions can trigger events. The mass flow rate through the sonic restrictor has been defined as the minimum value between the critical mass flow rate and the value from the isentropic solution using the operator *min*.

In case the sonic condition is not reached, the mass flow rate can be calculated using Equation 6:

$$\dot{m}_{is} = A_{out} \frac{p_{in}^{\circ}}{\sqrt{T^{\circ}}} \sqrt{\frac{\gamma}{R}} M_{out} \left(1 + \frac{\gamma-1}{2} M_{out}^2 \right)^{-\frac{\gamma+1}{2(\gamma-1)}} \quad (6)$$

For the choked condition, the mass flow rate can be calculated using Equation 2.

In the same way, the Mach at the outlet of the sonic restrictor has been defined as the minimum value between the one reached in case of shock and the one in case of an isentropic solution, avoiding the use of conditional expressions.

The Mach at the outlet in case the sonic condition is not reached can be calculated using the definition of total pressure (Equation 7), assuming that $p_{in}^{\circ} = p_{out}^{\circ}$.

$$M_{out,is} = \sqrt{\frac{2}{\gamma-1} \left(-1 + \left(\frac{p_{in}^{\circ}}{p_{out}} \right)^{\frac{\gamma-1}{\gamma}} \right)} \quad (7)$$

When the shock occurs, the Mach at the outlet can be calculated using Equation 8, that can be derived from Equations 2 and 6:

$$M_{out,choked} = \sqrt{c_1 + \sqrt{c_2 + c_3 \left(\frac{p_{in}^{\circ} \cdot A_t}{p_{out} \cdot A_{out}} \right)^2}} \quad (8)$$

$$c_1 = -\frac{1}{\gamma-1}$$

$$c_2 = \frac{1}{(\gamma-1)^2}$$

$$c_3 = \frac{2}{\gamma-1} \left(\frac{2}{\gamma+1} \right)^{\frac{\gamma+1}{\gamma-1}}$$

In this way, the solution will follow the continuous line of figures 2 and 3, discarding the dotted parts without using computationally expensive conditional expressions.

As far as the energy balance is concerned, the total temperature has been assumed to be constant between inlet and outlet.

3 The engine model

The sonic restrictor was integrated in an engine model developed using the Engines library (Picarelli, A. et al., 2009; Roberts, N. et al., 2013). A 0.6 L motorcycle-derived four-cylinder naturally aspirated spark ignition engine was used.

A MVEM (Mean Value Engine Model) was used in place of a more detailed CAREM (Crank Angle Resolved Engine Model) in order to be able to focus on the effects of the sonic restrictor on the average air mass flow rate rather than on an oscillating value.

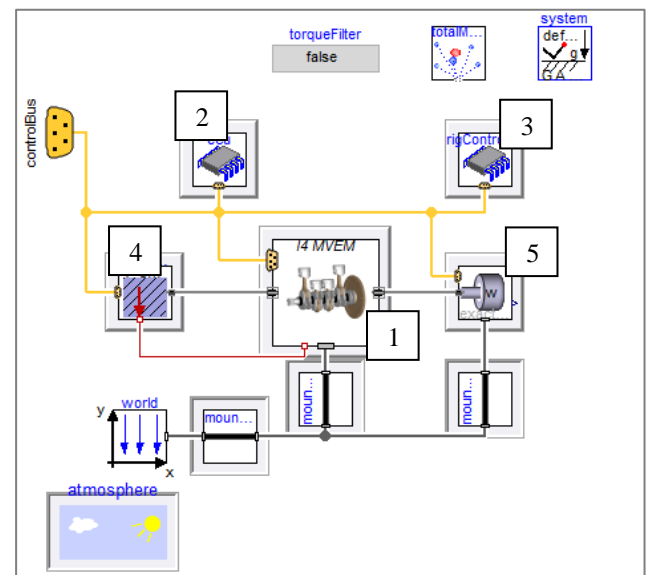


Figure 4. Engine test model: 1-Engine, 2-Engine Control Unit, 3-Rig Controller, 4-Engine coolant system, 5-Dyno.

Figure 4 shows the test rig, where the engine is connected to a dyno (5) and controlled by the ECU (2). The engine is run on a dynamometer and controlled to ramp up from around 5000 rpm to 11000 rpm with wide open throttle in order to generate the full load curve. To achieve this, the rig controller (3) specifies the throttle opening.

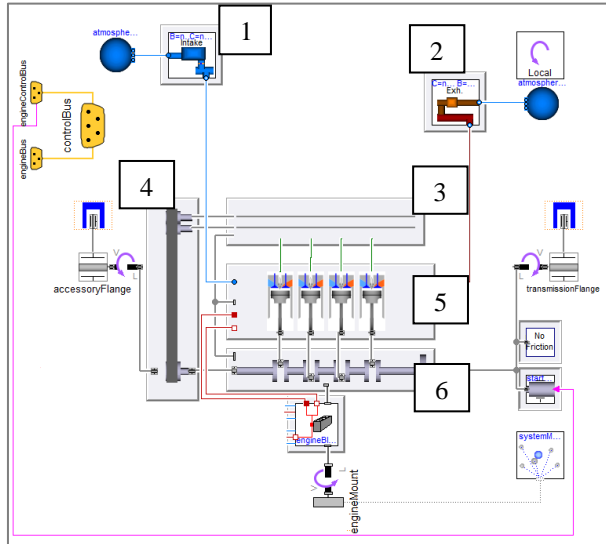


Figure 5. Engine model: 1-Intake, 2-Exhaust, 3-Camshaft 4-Timing belt, 5-Cylinder block, 6-Crankshaft.

Figure 5 shows the engine model, with the intake system (1), the exhaust system (2), the camshaft (3), the timing belt (4), the cylinder block (5) and the crankshaft (6).

Having used a MVEM in place of a CAREM, the camshaft and the timing belt models are empty, but they can be replaced with detailed models in case a CAREM engine is used.

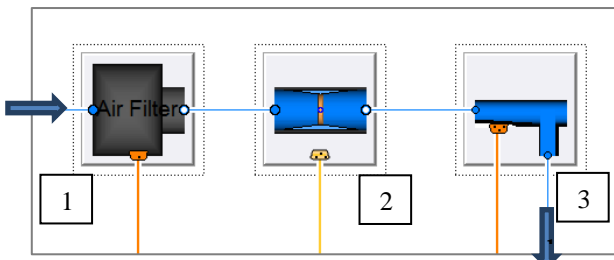


Figure 6. Intake model: 1-Air filter, 2-Throttle, 3-Intake manifold with sonic restrictor.

The intake system is shown in Figure 6 and consists of:

- The **air filter** (1) modelled with a pressure loss characteristic curve
- The **throttle** (2), modelled with an orifice also capable of modelling the choked condition. If the throttle is almost closed, the pressure ratio across it can be higher than the critical value (0.528) and the choked condition can be reached. Also in this

case, in the same way as for the sonic restrictor, the air flow rate at a given throttle position will be independent of manifold pressure and engine speed.

Having tested the engine at WOT (wide open throttle), there is no risk that the flow could become choked in the throttle, influencing the flow in the sonic restrictor.

- The **intake manifold** (3), containing the sonic restrictor and the plenum volume.

The sonic restrictor was placed after the throttle and before the plenum, as shown in Figure 7.

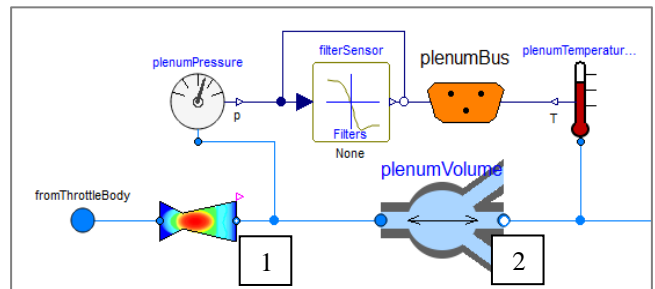


Figure 7. Intake manifold model: 1-Sonic restrictor, 2-Plenum.

To compute the cycle-averaged torque, the Mean Value combustion model uses IMEP maps where the output is a function of engine speed and plenum pressure, with corrections for air fuel ratio, spark timing and cam timing. It's clear that, by influencing the plenum pressure, the sonic restrictor can yield a different engine torque characteristic.

The mass flow rate through the engine is calculated using Equation 9, as a function of intake air temperature, engine speed and plenum pressure (Hendricks et al., 1996).

$$\dot{m}(n, p) = \frac{V_d}{120RT} (s_i \cdot p_i + y_i) \frac{n}{1000} \quad (9)$$

Where n is the engine speed [rpm], V_d the volumetric displacement of each cylinder [m^3], p_i the intake manifold pressure, R the specific air gas constant [J/Kg/K] and T the fluid temperature [K].

The coefficients s_i and y_i are function of speed and are provided in tabular format using experimental data.

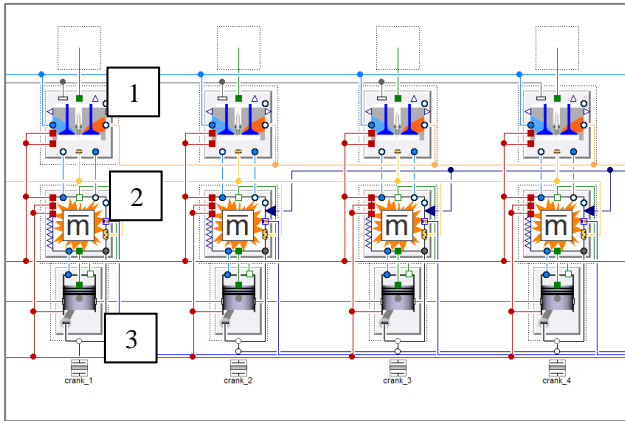


Figure 8. Cylinder block: 1- Piston head, 2-Combustion block, 3-Piston mechanism.

4 Results

The sonic restrictor model was integrated on a 0.6 L four-cylinder naturally aspirated spark ignition engine in order to analyse engine power and torque in case of choked flow.

Three different sonic restrictor throat areas were tested, with a throat diameter ranging from 20 mm to 25 mm, increasing the throat area in each test by 25%.

The results are shown in the following plots where At represents the case with a throat diameter of 22.4 mm, 0.75 At the case with a throat area 25% smaller (throat diameter: 20 mm) and 1.25 At with a throat area 25% larger (throat diameter: 25 mm).

The inlet area has always been assumed to be the same as the outlet one.

In all the three cases, the engine was run at WOT from 5500 to 10500 rpm.

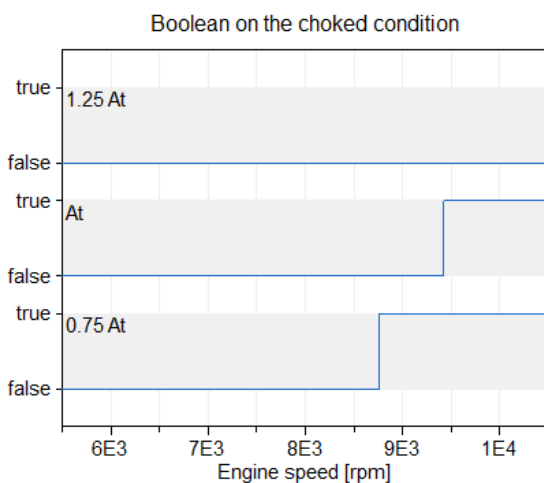


Figure 9. Boolean on the choked condition, true means that the restrictor is choked.

First of all, Figure 9 shows that in two of the 3 analysed cases the sonic condition was reached, while for the largest throat area (1.25 At) the flow remained always subsonic.

For a throat diameter of 20 mm (0.75 At), the choked condition was reached at 8760 rpm, while for a throat area 25 % larger the choked flow was reached at 9450 rpm (a speed around 8% higher).

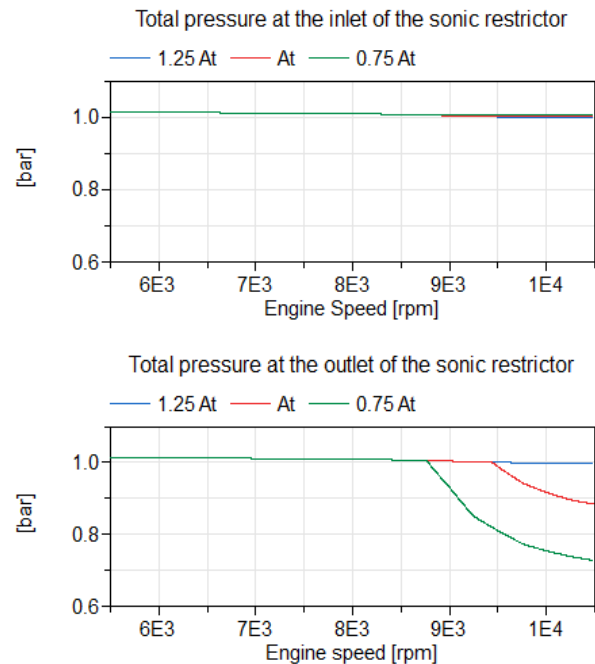


Figure 10. Total pressure at the inlet and the outlet of the sonic restrictor.

In the case of the largest throat area 1.25 At, the flow through the nozzle was isentropic and the total pressure across the sonic restrictor remained constant as shown in Figure 10, where the total pressures at the inlet and outlet of the sonic restrictor are plotted.

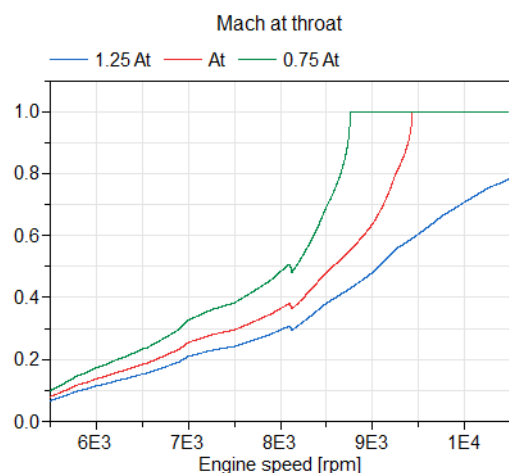


Figure 11. Mach at the sonic restrictor throat.

The Mach at throat reached 0.8 at 10500 rpm, the engine torque, the engine power and the engine air mass flow rate followed the same trends as in a non-restricted engine.

The engine torque reached its maximum at around 9000 rpm, the engine power reached its maximum at around 10000. At 10500 rpm the engine torque was already decreasing relatively steeply, while the engine power had just started to decline.

The mass flow rate increased following the engine speed ramp, as shown in Figure 14.

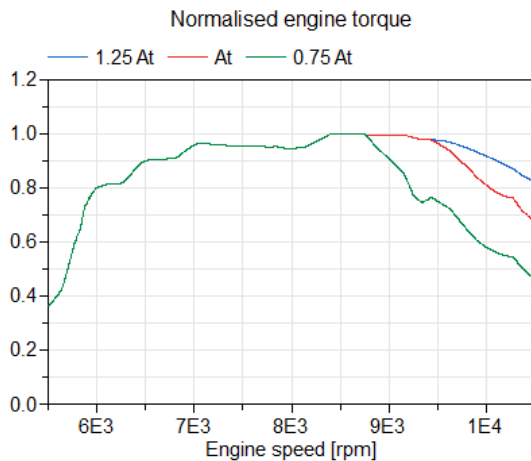


Figure 12. Normalised engine torque.

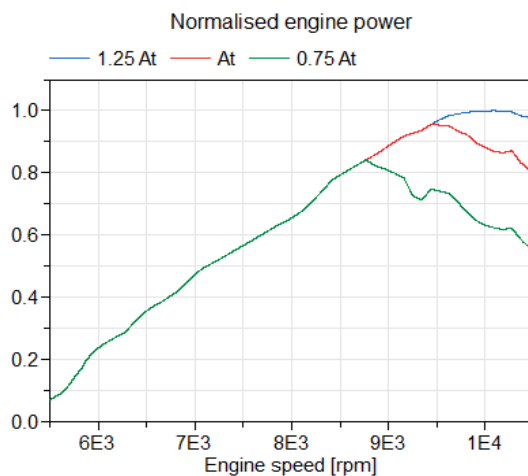


Figure 13. Normalised engine power.

For a throat area A_t , the sonic condition was reached at 9450 rpm, as shown in Figure 9.

Seemingly surprisingly, both the engine torque and the engine power decreased substantially at higher speeds (by around 18% with respect to the non-choked condition at 10500 rpm). This happens because of the sonic restrictor losses associated to the shock. As the static pressure required by the engine downstream the sonic restrictor decreases, the nozzle needs a stronger

shock to keep a constant mass flow, and the stronger shock corresponds to increased total pressure losses, as shown in the total pressure chart in Figure 10.

After the sonic speed was reached at throat, the mass flow rate was limited to the choked value and at 10500 rpm the mass flow rate was 13% lower than in the non-choked case.

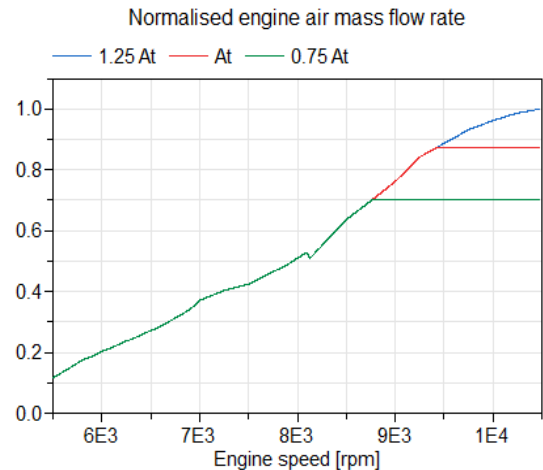


Figure 14. Normalised engine air mass flow rate.

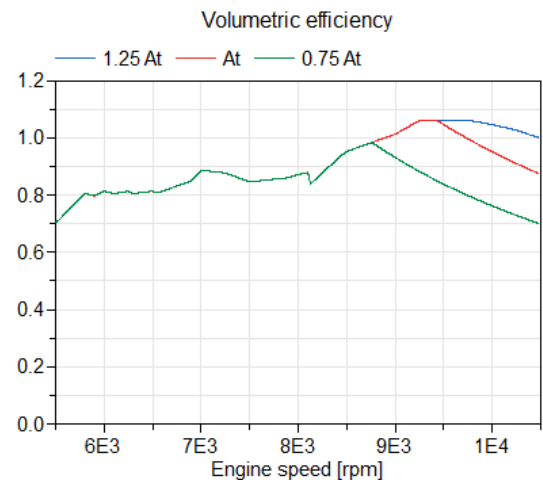


Figure 15. Volumetric efficiency.

By reducing the throat area a further 25%, the simulations showed that the shock was reached at a lower engine speed (8760 rpm).

An important result to outline is that the shock was stronger in case of a smaller throat area, as shown by the greater total pressure drop in Figure 10 (from 1 bar to 0.73 bar at 10500 rpm).

At 10500 rpm both the engine torque and the engine power were around 40% lower than in the case without shock.

For the same speed, the engine air mass flow rate was around 30% less than in the case without shock.

The drop in efficiency and power for the smallest throat area is clear also from Figure 15, where the volumetric efficiency is plotted.

The case with the smallest throat area shows clearly that the engine should not operate at speeds much higher than the one at which the sonic condition is reached. This suggests that the rev limiter should be set not far from the engine speed at which the air flow through the sonic restrictor becomes choked.

Furthermore, for a given displacement, engines with a higher torque at lower rotational speeds are likely to produce a higher power before the sonic restrictor becomes choked, thus bringing an advantage over ones optimised for higher regimes.

5 Conclusions

In this paper a sonic restrictor was integrated in an engine model to analyse the effect of choked flow through a nozzle on the engine mass flow rate, engine torque and engine power. Three cases with decreasing throat areas were analysed to assess the effect of the throat diameter on the shock intensity.

The full load curve showed that a considerable torque and power drop was reached after the choked condition, highlighting the need of limiting the maximum engine speed around the one at which the nozzle starts to be choked.

The study shows how Dymola can be used to analytically solve the fluid mechanics in engines. Of course, a quasi-1D code cannot solve phenomena such as flow separation and boundary layer development, but it can solve the shock and the compressible choked flow, making it a good starting point for testing and development of restricted engines.

References

- Casella F., Otter M., Proelss K., Richter C., Tummescheit H., The Modelica Fluid and Media library for modeling of incompressible and compressible thermo-fluid pipe networks, Proceedings of the 5th Int. Modelica Conference, Vienna, 2006.
- Anderson J., *Modern Compressible Flow: With Historical Perspective (Aeronautical & Aerospace Engineering)*, McGraw-Hill Education; 3rd edition, 1 Aug. 2002.
- Hendricks, E., Chevalier, A., Jensen, M., Sorenson, S. et al., Modelling of the Intake Manifold Filling Dynamics, SAE Technical Paper 960037, 1996, doi:10.4271/960037.
- Picarelli, A., Dempsey M., Investigating the multibody dynamics of the complete powertrain system, Proceedings 7th Modelica Conference, Como, Italy, 2009. doi: 10.3384/ecp09430085
- Roberts, N., Dempsey M., Picarelli A., Detailed Powertrain Dynamics Modelling in Dymola – Modelica, IFAC Proceedings Volumes, 2013, doi: 10.3182/20130904-4-JP-2042.00111

Engine thermal shock testing prediction through coolant and lubricant cycling in Dymola

Eduardo Galindo¹ Rodolfo Soler¹ Alessandro Picarelli² Victor Avila²

¹AVL IBERICA SA - VALLADOLID, Spain, {Eduardo.Galindo, Rodolfo.Soler}@avl.com

²Claytex Services Ltd. – Leamington Spa, UK, {alessandro.picarelli}@claytex.com

Abstract

In this work, an acausal multi-domain physical system model is used to study the interaction between an internal combustion engine operation and a range of cooling and lubrication system thermal cycling scenarios. Although the model can be used for modelling a wide range of scenarios, this paper concentrates on the application of engine thermal shock test dynamics prediction through coolant and lubricant cycling. An internal combustion engine is load-controlled on a dynamometer. Coolant and lubricant temperature transients are imposed on the engine system. Using freely available and commercial Modelica Libraries within the Dymola environment, the systems integration of the coolant rigs, lubricant rigs and engine is achieved. The rigs and the controllers are validated against test data to create predictive models of such systems for test virtualisation. This allows the user to develop and define control strategies for the tests from desktop, prior to engaging in laboratory tests.

Keywords: Engine testing, thermal-shock, control system development

1 Introduction

Engines need to work under a variety of temperature conditions. Some engine failure modes are caused by temperature cycling which in turn causes thermal expansion and contraction of the components. This phenomenon can induce mechanical stresses which in extreme cases can lead to component failure.

This paper builds on (Picarelli et al, 2014) and seeks to validate engine coolant and lubricant conditioning rigs for virtualisation of test scenarios in order to predict the system behaviour and to tune the control systems prior to the real testing taking place.

In addition to previous tests, where only the engine coolant was conditioned, in this paper we present thermal shock testing where the dynamics of the lubrication system are also included.

2 Thermal shock testing

Many manufacturers carry out thermal shock tests to understand and prevent component failure, as well as to accelerate durability testing of engines and engine components, including cylinder-head gaskets.

Thermo-mechanical fatigue is the term used to describe the type of fatigue in which temperature is varied throughout a cycle. The maximum tensile strain occurs at the same time as the maximum temperature.

Maximum compressive strain occurs at the minimum temperature. The main factor causing thermos-mechanical failure is a large number of temperature cycles. As in fatigue testing, it is possible to accelerate thermal cycling failure modes by increasing the frequency or amplitude of the thermal cycles.

These thermal tests are used to simulate critical conditions inside the engine by circulating a coolant flow with very large temperature gradients occurring over short periods of time (e.g. 30°C to 120 °C). This cycle is repeated a several times.

The main task performed in this study is simulation of repeated hot/cold thermal cycles. The engine is cycled between rated power and idle speed. The coolant and lubricant are also cycled between hot and cold temperatures by means of external conditioning units.

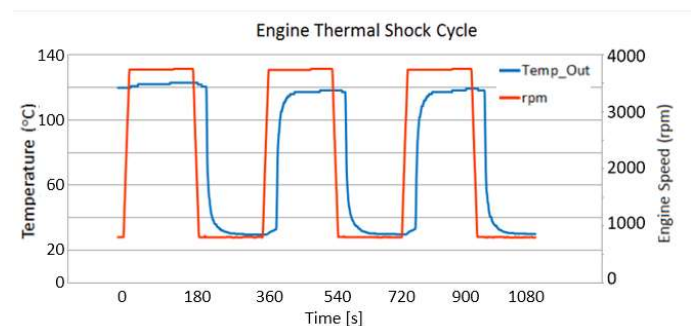


Figure 1. Example of an engine thermal shock cycle. Engine speed shown in red and coolant outlet temperature from engine shown in blue (°C).

The temperature gradient in the warm up and cooling down cycles is critical to generating the mechanical stresses applied to the engine due to the thermal shock. These kinds of tests allow manufacturers to reproduce the whole life of an engine in about 500 hours for a light duty passenger car and 2000 hours for a heavy-duty vehicle. Manufacturers expend great efforts in obtaining a good correlation between specific tests and the actual lifetime of an engine. Once the correlation is completed, the test must be performed as accurately as possible to preserve this correlation.



Figure 2. Cracks in the valve seat produced by thermal stress.

3 The need for a simulation model

An accurate and representative simulation model allows us to reduce engineering time for the prediction of new tests and design of new systems giving us the ability to predict the behaviour of a given system before manufacturing it.

This simulation ability also allows us to change the test or equipment parameters and foresee their impact on the results. This way, we can have a better view on how the system will behave, so that any specific issue or change can be adapted quickly and easily.

Furthermore, the simulation model has already predicted some unexpected and unwanted behaviours such as pressure spikes, giving the opportunity to make the necessary corrections early enough, thus avoiding additional engineering efforts and potential system failures.

4 Case Study

The thermal shock rig system in this study was intended to test engines from 60 kW to 120 kW, with an engine mass of 90kg to 120 kg. This power was limited by a maximum torque of 130-250Nm and a maximum speed of 6700rpm.

The actual engine used in the real test was a 4-cylinder gasoline engine with a maximum torque of 130 Nm and

a maximum speed of 5600 rpm, yielding a maximum power of 76.2kW.

The physical model is tested in two relatively different scenarios, which are as follows:

1. Thermal-shock test with low temperature gradients for heating and high temperature gradients for cooling, running between 110°C and -30°C. This test also includes the cooling of the engine's oil down to -20°C. In advance we'll refer to this test cycle as "Thermalshock 1":

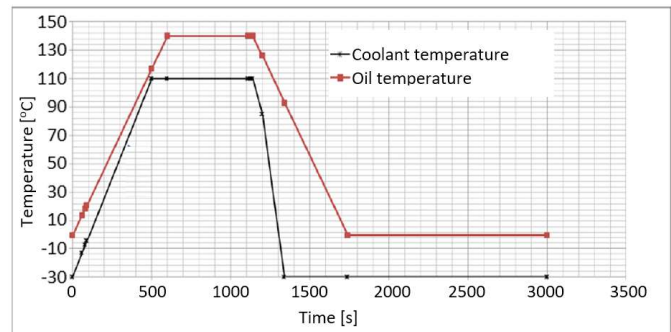


Figure 3. Temperature path for the engine coolant (black) and oil (red) for Thermalshock 1.

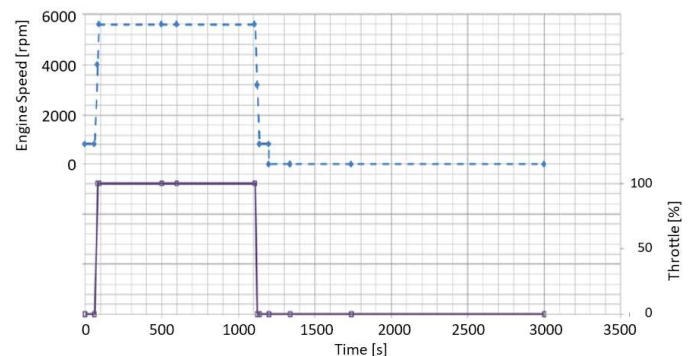


Figure 4. Throttle position of the engine for the Thermalshock 1 (solid line) and engine speed (dashed line).

2. Hot and cold test, with low temperature gradients for both cooling and heating, but with high frequency heat transients produced by quick variation on the engine throttle position. Hereon we will refer to this test cycle as "Thermalshock 2".

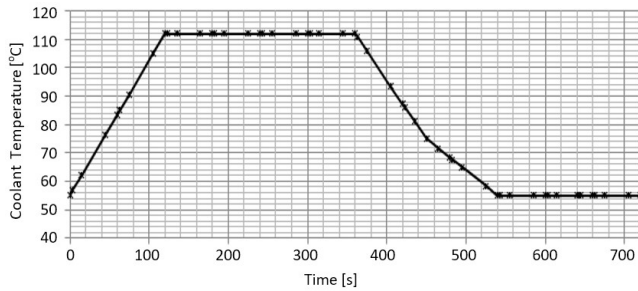


Figure 5. Temperature profile for the engine coolant and oil for the Thermalshock 2.

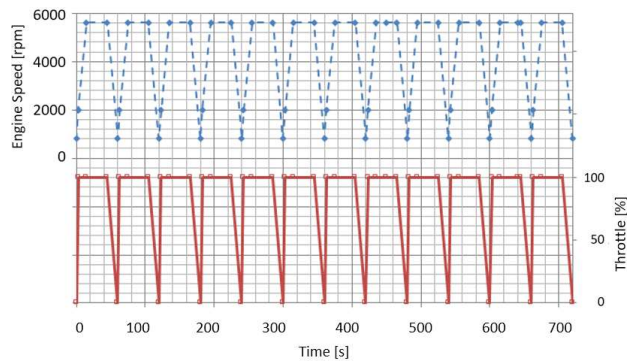


Figure 6. Throttle position of the engine for the Thermalshock 2 (solid line) and engine speed (dashed line).

These two tests are important in order to check the durability of their internal combustion engines, specially focused on the endurance of the head gaskets which are particularly affected by the thermal stress.

4.1 Thermal Shock Equipment Concept

The equipment consists of several fluid conditioning devices all connected to each other and/or the engine (Figure 7). Given that the thermalshock test itself has two well differentiated parts (hot part and cold part), there are two coolant conditioning devices and a further device that switches the connection of the engine between them.

Since the oil has to be conditioned too, the engine is connected to a heat exchanger on the gallery connections (the engine's oil pump is responsible for the flow), and to an oil cooling device on the sump.

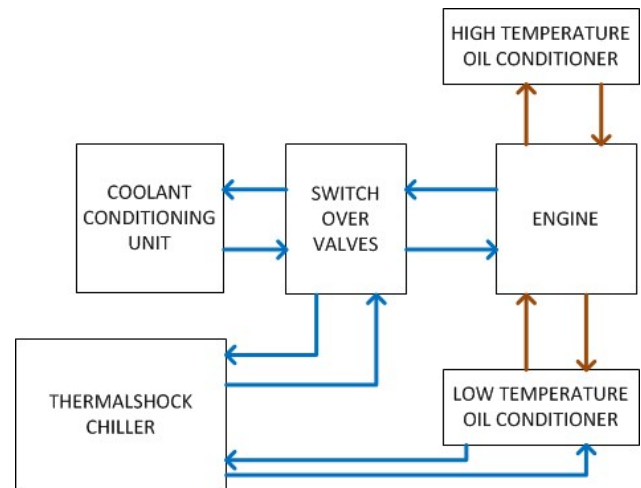


Figure 7. Complete system's simplified P&ID (Piping and Instrumentation Diagram).

4.2 Thermal Shock Testing Equipment

Coolant conditioning unit:

Composed of a pump and a 3-way valve that directs the coolant through a heat exchanger (for cooling) or a heating resistance (for heating).

Switch over valves:

A device composed of several 2 way pneumatic valves that allows the engine to be connected either to the coolant conditioning unit (Consyscool) or to the thermalshock chiller. This way the valves connect the engine to the coolant conditioning unit during the hot phase, and to the chiller during the cold phase.

Thermalshock chiller:

Composed of a water chiller specially designed and built for engine thermalshock testing. A pump flows the coolant from the inertia tank to the engine and to the oil cold heat exchanger.



Figure 8. Thermalshock chiller picture showing the large water tank on the left and the controllers and valves on the right hand side.

The design of this chiller is specially customized for engine testing, with special features like a on-standby design that allows the system to be ready for a thermalshock at any time (Figure 8).

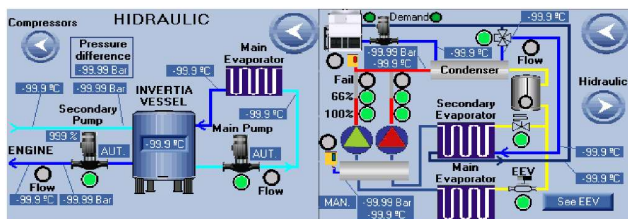


Figure 9. Thermalshock chiller controller screen.

The controller of the chiller is custom-designed for the engine testing process, with several programming parameters, interface with the testing facilities and remote control for operation and diagnosis (Figure 9).

High temperature oil conditioner (see Figure 12):

Consisting of a plate heat exchanger which is connected to a conditioning unit (similar to the engine coolant conditioning unit). The oil is cooled by means of a cold-water heat exchanger.

Low temperature oil conditioner (see Figure 12):

Consisting of an oil pump (variable speed), and a heat exchanger cooled by the same chiller used for cooling the engine coolant.

5 Model Development

5.1 Engine Model

The engine type used on the rig is a 1.8l turbo-petrol inline 4-cylinder engine.

The engine model in these tests is a thermal representation of the real engine which includes heat rejection from combustion to the coolant, lubricant and the engine's thermal mass.

The engine heat release to coolant and lubricant has been defined as a fraction of the crank power and varies depending on engine speed and load. The fraction value is determined from steady state tests by calculating the power required for the coolant and lubricant temperature changes between the inlets and outlets of the engine circuits. The fluid paths within the engine are represented by a single pipe having average diameter of the passageways and the measured total engine pathway volume and surface area. The pipe dimensions are adjusted to achieve the required flow velocities through the engine.

The engine thermal mass used in this study is a lumped thermal mass and is not split by subsystem. More detailed models are available within the Claytex Engines library for studies which require higher level of engine thermal mass discretisation. The engines library was used in a previous study (Dempsey *et al*, 2009; Dempsey *et al*, 2012; Dempsey *et al*, 2013; Picarelli *et al*, 2014).

The coolant pump of the engine is replaced by electric coolant pumps within the rig which can be controlled to deliver specific flows or flow profiles. The lubricant pumped by the engine lubricant pump itself when the engine is running. An electric lubricant pump within the rig is used when the engine is switched off.

The heat transfer from the engine to the coolant is calculated by means of a Nusselt Number correlation, calculated specifically for this engine. The Nusselt Number (Nu) correlation is then used within the pipe model which represents the coolant path within the engine. Due to the fact that the thermal mass of the engine is of lumped type, the volume model used to represent the mass of coolant within the engine has one thermal node. The same Nu correlation can be implemented with multiple node fluid pipes derived from the Modelica.Fluid library should a more detailed thermal discretisation be required and will be the subject of further work when engine CAD data becomes available. This will also increase the predictive capabilities of the model. The exact same Nusselt

Number heat transfer approach is used for the heat exchangers in the rig model described in section 5.2.

5.2 Rig Model

The thermal shock rig must be able to supply preconditioned coolant to two different flow conditioning units and by means of a switch valves device controls the engine fluid temperatures. The rig described in this paper uses a 2000 litre coolant tank kept at temperature with fixed set-points and an external source which supplies water permanently. The water tank is kept at constant ambient temperature and the coolant tank is kept at low temperature, around -30°C (Figure 10. Complete thermal shock rig with water tank (1), coolant tank (2), **Mean Value** engine model (3)). The tanks are required to also smooth out and absorb any temperature fluctuations in the rig, in addition to these two tanks, there are also some small expansion tanks included throughout the rig to absorb any possible pressure, temperature and volume fluctuations.

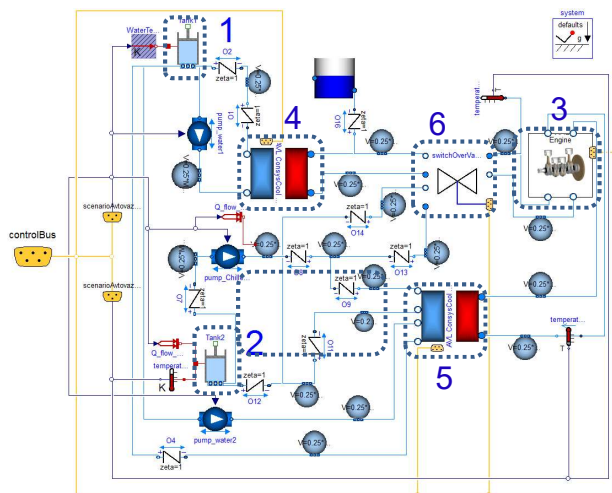


Figure 10. Complete thermal shock rig with water tank (1), coolant tank (2), Mean Value engine model (3), coolant conditioning (4), lubricant conditioning (5) and hot/cold switchover valve (6).

At particular points in the cycle, the switchover valves (Figure 11) model and the internal valves of the conditioning units before described, are controlled to channel either hot or cold coolant through the engine. These changes in coolant and lubricant temperatures yield the required thermal shock for the engine to experience and operate through.

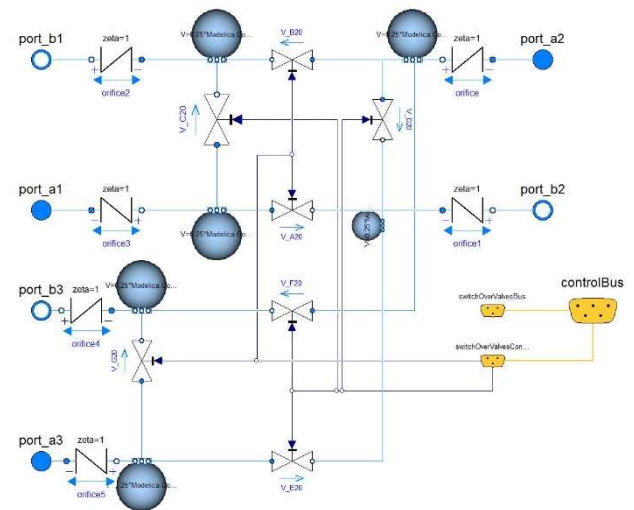


Figure 11. Switch over valve model used in the rig to lead the coolant from the different heat exchangers through the engine.

Both lubricant conditioning units (see **Error! Reference source not found.**) are included in the same model (Figure 12. Oil conditioning unit DiagramFigure 12).

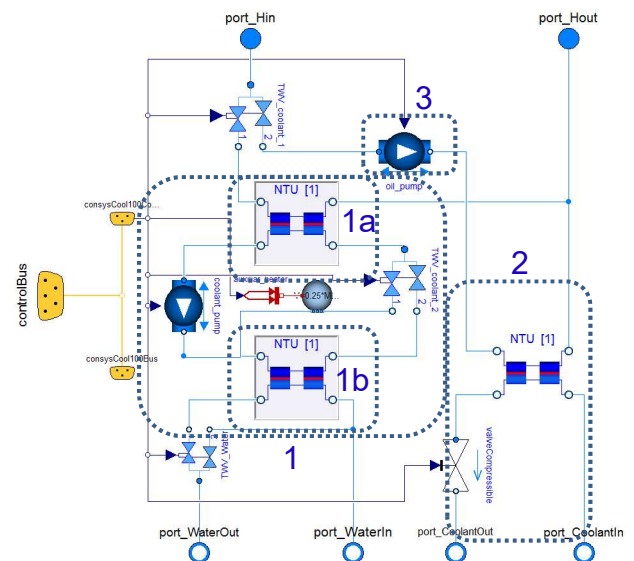


Figure 12. Oil conditioning unit Diagram: High temperature conditioner (1), plate heat exchanger (1a), cold water heat exchanger (1b), low temperature conditioner (2) and electric pump (3).

The rigs are modelled using the Modelica.Fluid and Modelica.Media libraries (Casella *et al*, 2006) with some customized components from the Claytex library which incorporates advanced functionality within the components both for visualization and enhanced model efficiency. The fluids used match that of the rig in terms of properties and are a mixture of 50% Ethylene Glycol and water with linear compressibility for the coolant

side and Oil with constant compressibility for the lubricant side.

The controller for the tank cooler is of on/off type and starts to cool with 40.4 kW of power when the tank fluid temperature has deviated from the set point by +1 °C.

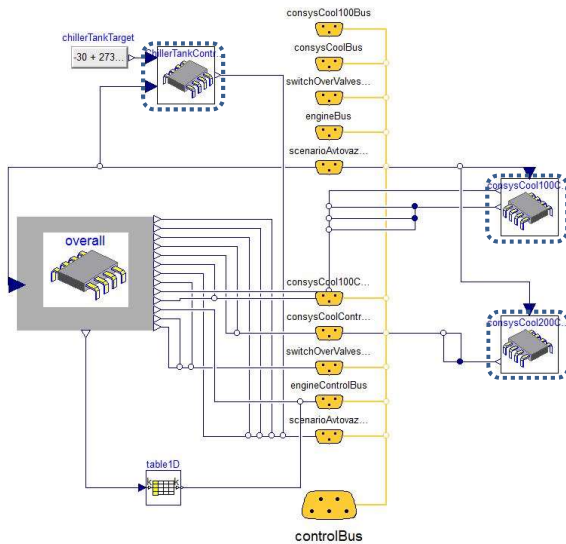


Figure 13. PID controllers for controlling the coolant and lubricant conditioning units and chiller tank to maintain the corresponding fluid temperatures close to the set points.

To control the 3-way valves, within the fluid conditioning devices, a Modelica.StateGraph model was used which is shown below (Figure 14). The valves are operated to route the coolant and the lubricant through the heat exchangers or bypassing them to restore desired temperature targets at particular points in the cycle.

The same type of StateGraph model controls the throttle pedal position which is cycled from 0-100% in a similar phase to the engine speed (Figure 15).

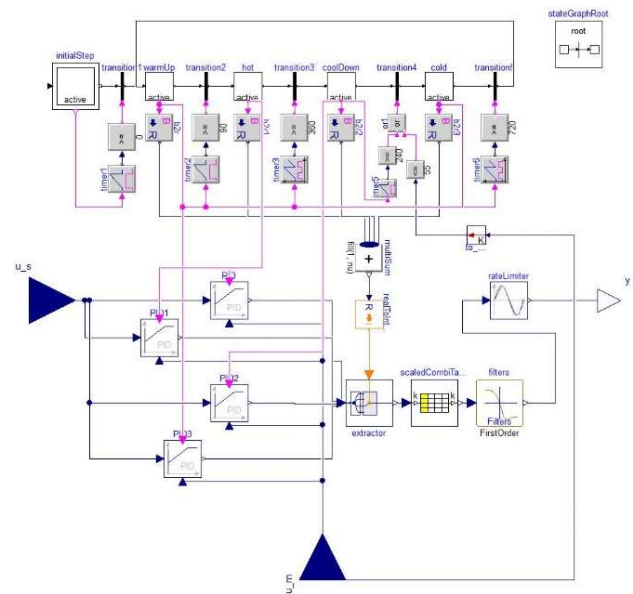


Figure 14. StateGraph controller for the internal coolant conditioning unit 3-way valve.

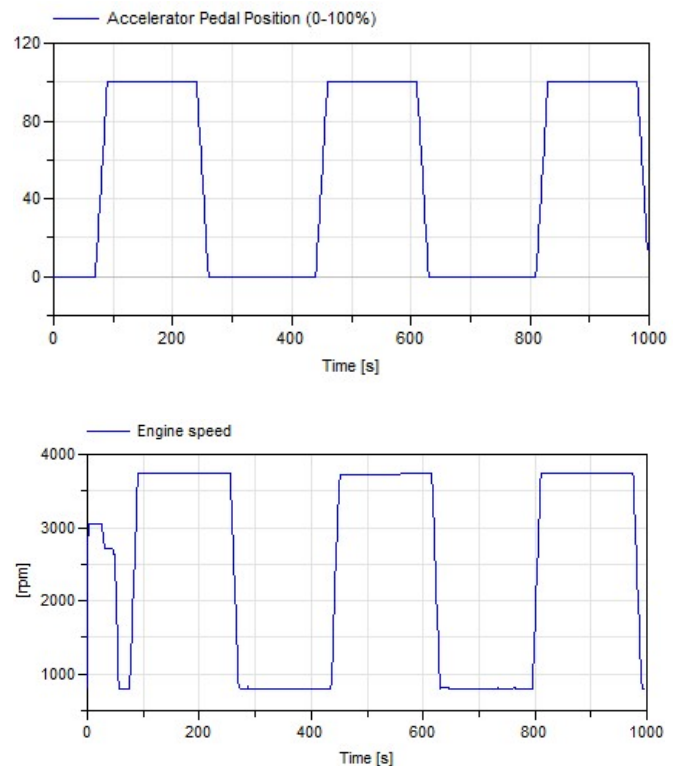


Figure 15. Resulting accelerator pedal position (top) and engine speed (bottom) for the thermal shock test.

6 Results

6.1 Initial Results from Dymola

Before the actual rig commissioning tests were undertaken, the Modelica system model was already finished, and showing the following expected results:

Thermalshock 2:

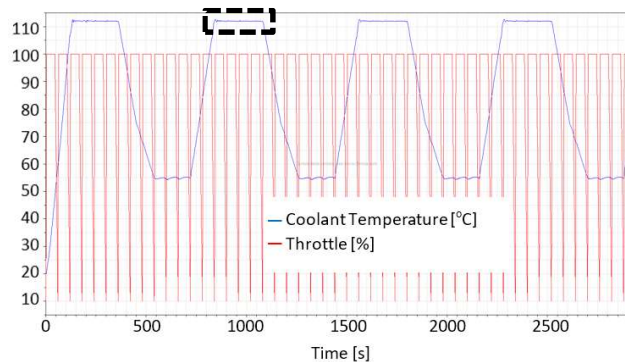


Figure 16. Dymola initial Thermalshock 2 stability estimated as $\pm 0.6^\circ\text{C}$ with only low frequency and amplitude oscillations.

Thermalshock 1:

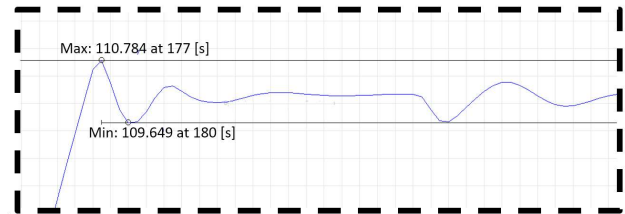
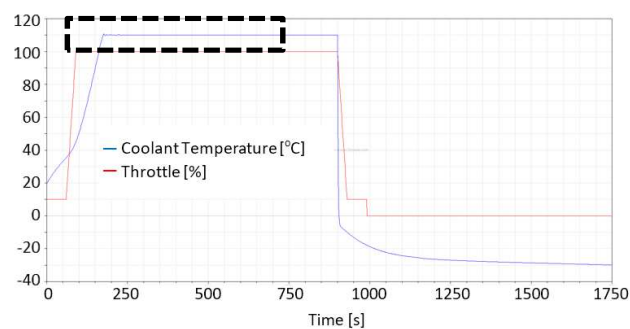


Figure 17. Dymola initial Thermalshock 1 stability estimated as $\pm 0.5^\circ\text{C}$ with only low frequency oscillations on the hot phase an asymptotic cooling up to -30°C on the cold phase.

6.2 Experimental Results

After the commissioning of the actual real life system, the following data was gathered:

Thermalshock 2 experimental results:

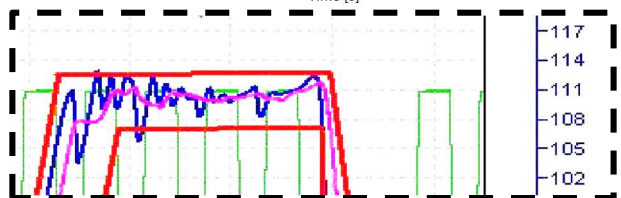
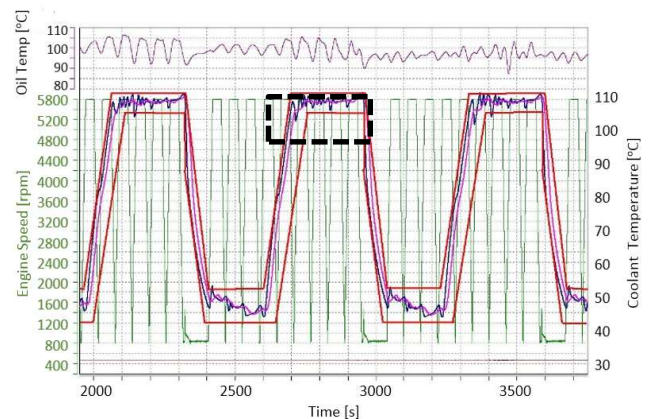


Figure 18. Real Thermalshock 2 results, with higher frequency oscillations and a maximum amplitude of $\pm 3^\circ\text{C}$.

At first sight it was seen that the expected accuracies and oscillation frequencies were underestimated.

6.3 Model Adjusting

Since the same system model was used for both tests, the model validation strategy was as follows:

For the thermalshock 2 tests, the real PID control parameters used for the test were recorded. Then, these PID parameters were introduced to the Dymola model. Next, the following parameters were adjusted in order

to have similar paths on the engine outlet coolant temperature:

- Engine thermal mass and heat transfer coefficient
- 3-way valve actuation speed

The following parameters/model properties did not require adjustment:

- Fluid properties
- Pipe/ducting geometries
- Bend losses
- Heat exchanger pressure drops
- Heat exchanger performance and thermal coefficients
- Pump flow characteristics
- Pump loss characteristics
- Valve losses
- Engine combustion heat release
- Engine combustion heat release
- Engine inertia

The 3 tuned parameters had different effects on the modelled outlet temperature of the system, and modifying them one by one the following results were achieved:

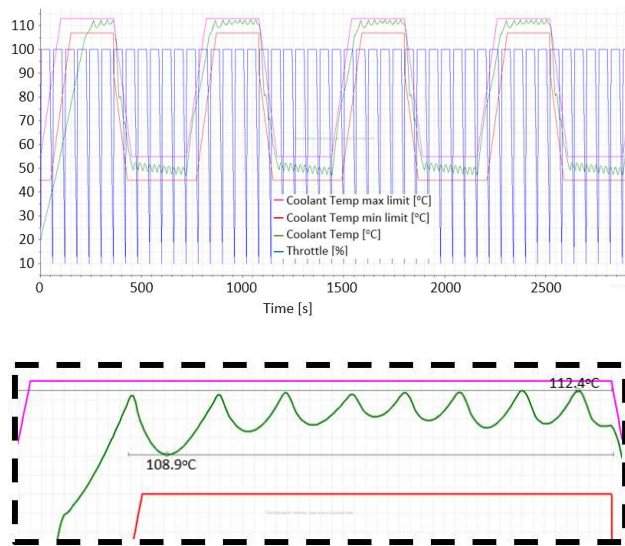


Figure 19. Adjusted Dymola model, running a Thermalshock 2 test, with higher frequency variations and maximum amplitudes of $\pm 1.75^{\circ}\text{C}$ on the coolant outlet temperature.

Although the maximum amplitudes measured in the experiment and model results differed (larger in the experiment results: $\pm 3^{\circ}\text{C}$ vs. $\pm 1.75^{\circ}\text{C}$, the average temperature and remaining oscillations we of similar value: $\pm 1^{\circ}\text{C}$ vs. $\pm 0.8^{\circ}\text{C}$).

6.4 Validation

After checking the model was running accurately simulations on the Hot-Cold tests, all the tuned parameters where frozen, thus obtaining a validated mathematical model.

This model was then used for the Thermalshock 1 test, with the following results:

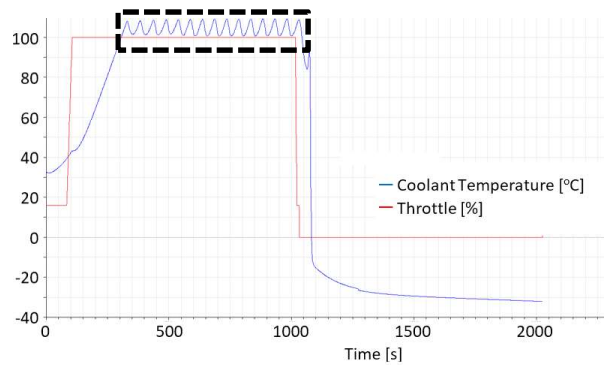


Figure 20. Validated Dymola model, running a Thermalshock 1 test, with higher frequency variations and amplitudes of $\pm 2.7^{\circ}\text{C}$ on the coolant outlet temperature on the hot phase, and asymptotic cooling on the cold part.

Where the Thermalshock 1 experimental results were:

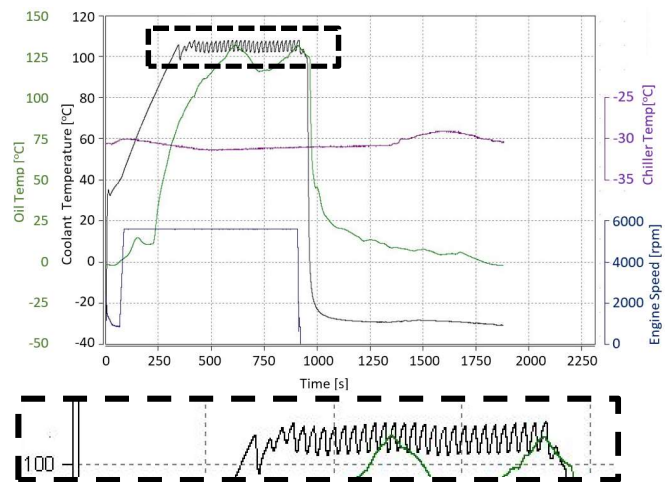


Figure 21. Real Thermalshock 1 results, high frequency oscillations and amplitude of $\pm 2.5^{\circ}\text{C}$.

After model validation and calibration using the hot-cold results, the results for the thermalshock tests were much more realistic, proving that the model can be used for any test made using the same system, regardless of the test conditions.

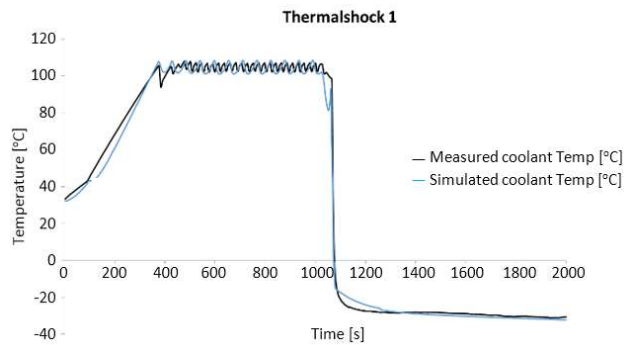


Figure 22. Comparison of the thermalshock 1 real results (black) and the validated Dymola model (blue).

Further investigation is required to confirm the reason for which the higher temperature oscillations in the model are half those measured in the experimental results. Further tuning of the valves pressure drops, hence flow speeds in this operation mode might improve the discrepancy.

6.5 Lubricant Dynamics Validation

Despite the rig model being prepared to also simulate lubricant conditioning and heating (

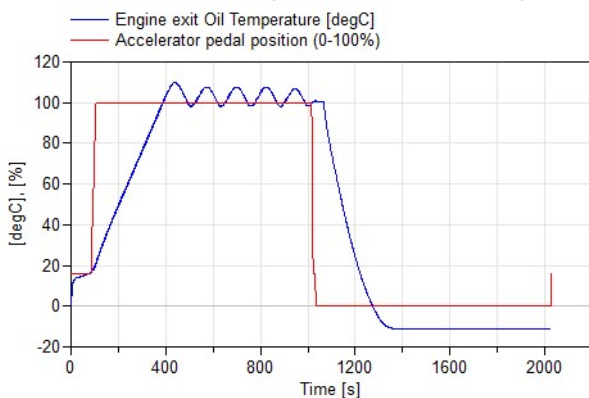


Figure 23), technical issues in the real rig suggested the lubricant measurement data could have been compromised, hence preventing detailed validation of lubricant conditioning (Figure 24).

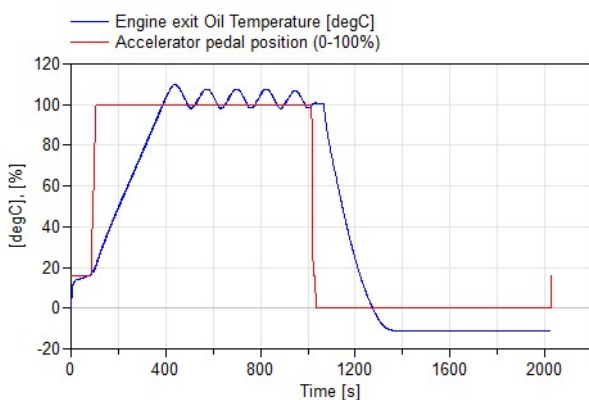


Figure 23. Example of the non-validated results for the lubricant temperature (blue line), running Thermalshock 1 test. Throttle position is also displayed (red line).

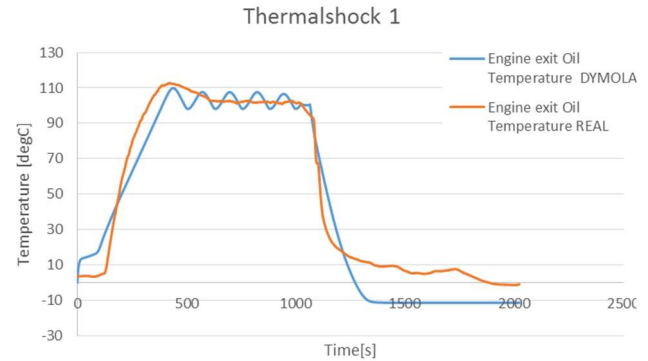


Figure 24. Comparison of the Thermalshock 1 real results (Orange) and the non-validated Dymola model (blue)

This validation will be the objective of a further investigation in the future.

7 Conclusions

After adjusting the Modelica model parameters, the simulation results were much more realistic. Even though the system's hysteresis/entropy is still a parameter that cannot be simulated and produces non-periodic oscillations, the more significant results such as temperature accuracies, heating and cooling times are correctly simulated and can predict an actual behaviour of a system under different test scenarios with an accuracy of $\pm 0.5^\circ\text{C}$.

The information about the engine's thermal mass and global heat transfer coefficient will be useful for future projects with similar engines. Even with different engines, these parameters are now a starting point for estimating these values otherwise impossible to know.

References

- Casella. F. et al. (2006) The Modelica Fluid and Media library for modeling of incompressible and compressible thermo-fluid pipe networks Modelica Conference, 2006
- Dempsey M., and Picarelli A. (2009). Investigating the multibody dynamics of the complete powertrain system. Como, Italy: Proceedings 7th Modelica Conference.
- Dempsey M., Picarelli A, Fish G. (2012). Using Modelica models for driver-in-the-loop simulators. Munich, Germany: Proceedings 9th Modelica Conference.
- Dempsey M., Roberts N., Picarelli A. (2013) Detailed Powertrain Dynamics Modelling in Dymola – Modelica. IFAC-AAC conference Tokyo, Japan.
- Picarelli A., Galindo E., Diaz G. (2014) Thermal shock testing for Engines in Dymola. Lund, Sweden. 10th Modelica Conference, 2014

Template based code generation of Modelica building energy simulation models

Christoph Nytsch-Geusen¹ Alexander Inderfurth¹ Werner Kaul¹
Katharina Mucha¹ Jörg Rädler¹ Matthis Thorade¹ Carles Ribas Tugores¹

¹Institut für Architektur und Städtebau, Berlin University of the Arts, Germany, nytsch@udk-berlin.de

Abstract

This contribution describes an approach for a template based code generation for different detailed Modelica models for building energy simulation (BES).

The information from several data sources, which describe the building geometry, the building construction, the building location and the building itself, is used to fill a building data model. This intermediate data structure is still independent of a certain building simulation tool.

A new developed tool for template based code generation (CoTeTo) uses the building data model and combines it with a set of different code generators, which are able to generate Modelica building models with a different level of detail: Strong simplified low-order building models for district energy simulation with a large population of buildings, more advanced multi-zone building models for building energy simulation and 3D space resolved room models for a detailed indoor climate analysis.

Three case studies for the mentioned building model types demonstrate the code generation approach.

Keywords: building energy simulation, adapted model level of detail, Modelica code generation

1 Introduction

The generation of machine-readable code usually combines static and dynamic data sources. The static part describes the keywords and syntactical requirements of a computer language and builds a static framework while the dynamic part injects real values and structures from the runtime environment of the code-generating application or from an external data source. In the simplest case the application uses some (potentially nested) print()-like statements. This approach has some limitations because even the smallest change in the output format requires access to the source code of the application, programming skills and potentially large compile cycles.

With the rise of dynamic web-sites a more flexible technology was widely used and much improved: the template engines. Such an engine is a program library

linked into an application, but the process of the code generation is controlled by external text files. These template files embed simple control structures and placeholders in normal text and can be easily edited. The concept is similar to the serial letter function in word processing applications.

The idea of code generation for Modelica BES libraries was first applied within the EnEff BIM project. In this project the structured data of an IFC files were used for the automatic generation of Modelica system models, consisting of a HVAC sub-model and a strong simplified building model (for more details see Thorade et al., 2015).

This contribution is focused on code generation for Modelica building energy models with different levels of detail. Important information for the code generation are the building geometries, the building topologies, the building constructions, the building locations and the behavior of the building occupants.

2 Template based code generation

A general approach for code generation of BES models has to consider the heterogeneous data formats (data sources) in the building industry sector and should be able to generate models with a different level of detail, which fits to the question of the simulation analysis.

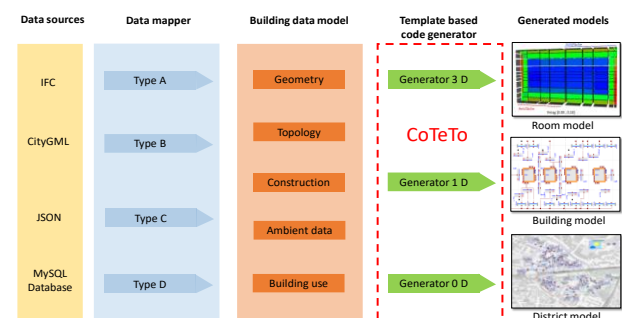


Figure 1. Template based code generation of BES models with a different level of detail

A set of data mappers transform the input data into a common building data model. Dependent on the present information within this data model one or more template based code generators can produce Modelica BES

models for room simulation, building simulation or district simulation (compare with Figure 1).

2.1 Data sources

In the building industry sector, there are different data sources and data formats available, which can satisfy the needs of the building energy simulation domain. On the scale of single buildings, the IFC-Format in the version IFC2x3 (IFC2x3, 2017) can be used and in near future also the version IFC4 (IFC4, 2017). This format represents the digital building model in a well-structured form (the entire building, several spaces, walls, windows etc.) in combination with a precise description of the building geometry. Most of the architecture CAD programs can export the IFC data format. It fits perfect to the structure of a multi-zone-building model (building model, thermal zones, building components) and the precise geometrical data also allows the parameterization of spatial resolved room models.

On the scale of city districts the CityGML format (CityGML, 2017) and the GeoJSON format (GeoJSON, 2017) can deliver the necessary building parameter for district energy models. Normally, GIS programs are able to export one or both of these data formats with simplified building geometries, which fits to the reduced parameter sets of the low-order building models on the district model scale. In this case, the challenge consists in the data acquisition of huge populations of buildings and not for a single building (Kaul et al., 2014).

In special cases, building parameter sets are also available in data base formats, e.g. MySQL (Inderfurth et al., 2017).

2.2 Data mapper

A data mapper is a specialized software module, which is able to map a certain data source file format to the format independent building data model (see paragraph 2.3). Two different data mappers were realized based on Python up to now: the first data mapper can be used for 1-dim. multi-zone-building simulation and 3-dim. room simulation and uses the IFC format as the data input. The Python bindings of the IfcOpenShell-library (IfcOpenShell, 2017) are used to read the IFC-files and Python bindings of the OpenCascade-library (pythonOCC, 2017) are used to transform in a second step the geometrical and the topology data in a manner, that they can be stored in the building data model. The second data mapper was implemented for district energy simulation and can read the GeoJSON-format. A third data mapper for information input from SQL data bases is under development.

2.3 Building data model

The building data model holds all the information, which is necessary for the Modelica code generation. This includes the data for the building geometry (full geometrical description or simplified geometry), the

building topology (substructure of a building in thermal zones), the used construction types (multi-layer definitions), the definition of the building ambient data (location, weather data) and the type of building use (e.g. air change rates, set temperatures for heating and cooling etc.). The building data model itself is independent of the type of the data sources (but it has functions for setting building parameters from data sources) and also on the type of the code generator (different code generators use the same function to get building parameters from the building data model).

2.4 CoTeTo

To automate some of the required steps for the generation and parametrization of Modelica code a software tool (**Code Templating Tool**) was developed in the context of the EnEff-BIM project (Thorade et al, 2015). CoTeTo (CoTeTo, 2017) comes with an open source license and can be download from GitHub (<https://github.com/UdK-VPT/CoTeTo>). It includes pluggable input, filter and output components that cover the process of data acquisition, preprocessing and output using a template system. CoTeTo is implemented in Python and can be used standalone or as a library imported in Python applications. A command line interface is provided for interactive usage or inclusion in shell scripts. A GUI based on PyQt4 (PyQt4, 2017) can be started as an application (see Figure 2) or included in PyQ4-based applications as a widget.

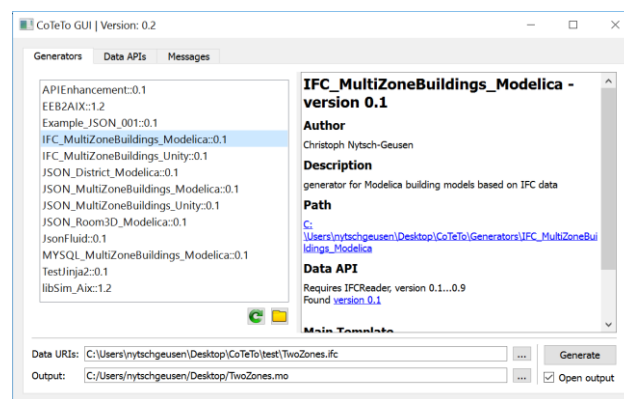


Figure 2. CoTeTo GUI for template based code generation

CoTeTo uses the Mako template engine (Mako, 2017) for the code generation step, but an experimental interface to the Jinja2 engine (Jinja2, 2017) is implemented as well.

2.5 Generators

CoTeTo documents (called generators) can be easily edited and shared without deep programming knowledge. A generator is stored in a folder structure or a zip file containing plain text files. The idea of a generator is to include all parts necessary to generate the code for a defined target (like a certain Modelica buildings library) from a defined source (like a special file format or database structure).

A generator depends on a so-called input API, which is defined in a Python module. Some standard input APIs are included in CoTeTo (CSV, JSON, XML, ...), but generators can define own input modules. Between the data input and the output templates filter functions can be called to preprocess the data structure. These Python functions are defined in the generator.

The CoTeTo framework handles this conversion process completely data-agnostic, the structure and format of the data objects is defined by the input APIs, generators and filter functions only.

2.6 Adaption to the BuildingSystems library

Based on the CoTeTo framework three code generators for the Modelica BuildingSystems library (<http://www.modelica-buildingsystems.de>) were implemented. This library is being developed for the dynamic simulation of the energetic behavior of single rooms, multi-zone buildings or entire city districts (Nytsch-Geusen et al., 2016). The simulation models of the library describe the dynamic energy balance of the building envelope under consideration of the building geometry, the thermal properties of the building construction, the ambient climate and the user behavior. As the Modelica library IDEAS, AIX Lib and Buildings, the BuildingSystems library uses as a core the same Annex 60 Library, which was developed as a common project from the authors of the four mentioned libraries in the Annex 60 project (Wetter et al., 2015).

The predefined components of the BuildingSystems library such as air volumes models, building construction models, wall and window models, zone models, low-order building models or ambient models (compare Figure 3, Figure 6 and Figure 10) are the base for the generated Modelica code. These model classes include the physical description (energy and mass balances, empirical equations etc.) and are instantiated and parameterized by the code generator using the information, which is stored in the building data model.

The following code shows as an example the Mako code, which generates the Modelica records for the definition of all multi-layered opaque constructions of a building model:

```
% for con in constructions:
record ${con.name}
  extends OpaqueThermalConstruction(
    nLayers=${con.nLayers},
    thickness={
% for value in con.thickness:
  ${value}${',' if not loop.last else ''}
% endfor
    },
    material={
% for value in con.material:
  ${value}()${',' if not loop.last else ''}
% endfor
    });
end ${con.name};
% endfor
```

Based on the stored information in the building data model the code generator generates for example the code for three different building constructions:

```
record ConstructionFacade
  extends OpaqueThermalConstruction(
    nLayers=4,
    thickness={0.015,0.2,0.15,0.02},
    material={
      HighGradePlaster(),
      Concrete(),
      ExpandedPolystyrene(),
      HighGradePlaster()});
end ConstructionFacade;

record ConstructionInnerWall
  extends OpaqueThermalConstruction(
    nLayers=3,
    thickness={0.015,0.12,0.015},
    material={
      HighGradePlaster(),
      Kalksandstein1800(),
      HighGradePlaster()});
end ConstructionInnerWall;

record ConstructionBottom
  extends OpaqueThermalConstruction(
    nLayers=3,
    thickness={0.02,0.06,0.2},
    material={
      Wood(),
      WoodFibreInsulation(),
      Concrete()});
end ConstructionBottom;
```

3 Case studies

The case studies shall demonstrate the general approach for template based Modelica code generation for building energy simulation. The examples address three different scales of building simulation: District modelling, multi-zone modelling and single room modelling.

3.1 City district

The first case study considers a city district in Berlin-Kreuzberg, which was designated by the Berlin city government as a redevelopment area (SenStadtWohn, 2016). In this context an analysis about the present energy efficiency of the building stock within this areal will be of interest. Because the whole district includes 144 buildings, the challenge for a district energy model, which could describe the present energy demand, consists in the data gathering of a huge parameter set (geometries, U-values etc.) for all buildings.

Data source: In the former research project Open eQuarter, a new layer-oriented geographic information system (GIS) based method was developed to obtain building sharp parameter data sets (Kaul et al., 2014). For this purpose, different city maps with information such as the building outlines, the number of stories, the building age in combination with a data base with U-values of the building elements were used, dependent on the building age (Loga et al., 2015). The open source GIS tool QGIS (QGOS, 2017) in combination with the

Open eQuarter plugin is able to export a GeoJSON file, which includes all the necessary building parameters (location, simplified building geometries, U-values) gained and calculated by the mentioned data sources. This GeoJSON file serves as the data input for the building data model in Figure 1.

Data mapper: In a first step the building data model takes the information through a data mapper from a GeoJSON file, which contains beside the mentioned building parameters also the building outlines for each building as polygon points. A python filter function calculates the centroids of these polygons to obtain the local placements of the building models within the district model. After this intermediate step all needed building data are stored in the building data model and can be used afterwards by the Modelica code generator.

Components: Two components of the BuildingSystems library are used for the code generation (see Figure 3). First, an ambient model, which describes the climate boundary condition of the city district, in particular the outside air temperature and the solar radiation on the building surfaces. Second, a low order building model (described in Nytsch-Geusen and Kaul, 2015), which is able to calculate the dynamic heating and cooling demand for an individual building with a small set of input parameters.

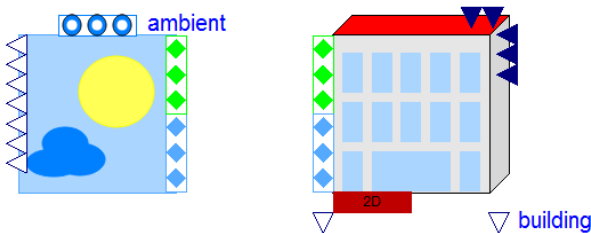


Figure 3. Components for district modelling.

Code generator: During the code generation the building centroids are used for component related annotations, which defines the graphical appearance of the individual building models on a realistic position. This is possible, because the positions of each individual building were gained from geo-referenced maps (compare with Figure 4). The excerpt of the generated code shows the instantiation and parameterization of the first two building models of the district, the ambient models and the connections between the ambient model and the two building models:

```
model DistrictModel
  extends Modelica.Icons.Example;
  Building1Zone0DDistrict building1(
    UValFac = 0.371,
    UValRoo = 0.269,
    UValGro = 0.4,
    UValWin = fill(1.575,4),
    fWin = 0.21,
    length = 8.127566,
    width = 5.318865,
    heightSto = 3.0,
    nSto = 4)
  annotation(Placement(transformation(
    extent={{0.0,0.0},{10.0,10.0}})));
```

```
Building1Zone0DDistrict building2(
  UValFac = 1.83,
  UValRoo = 1.23,
  UValGro = 1.2,
  UValWin = fill(3.1,4),
  fWin = 0.294,
  length = 48.020794,
  width = 7.903955,
  heightSto = 3.0,
  nSto = 4)
  annotation(Placement(transformation(
    extent={{29.574,1.040},{19.574,11.040}})));
...
Ambient ambient(
  nSurfaces = 720,
  weatherDataFile = WeatherDataFile_Berlin());
equation
  connect(ambient.toSurfacePorts[1:5],
    building1.toAmbientSurfacesPorts[1:5]);
  connect(ambient.toAirPorts[1:5],
    building1.toAmbientAirPorts[1:5]);
  connect(ambient.TAirRef, building1.TAirAmb);
  connect(ambient.xAir, building1.xAirAmb);
  connect(building1.airchange[1], airchange.y);
  connect(building1.T_setHeating[1], TSetHeating.y);
);
  connect(building1.T_setCooling[1], TSetCooling.y);
);
...
  connect(ambient.toSurfacePorts[6:10],
    building2.toAmbientSurfacesPorts[6:10]);
  connect(ambient.toAirPorts[6:10],
    building2.toAmbientAirPorts[6:10]);
...
end DistrictModel;
```

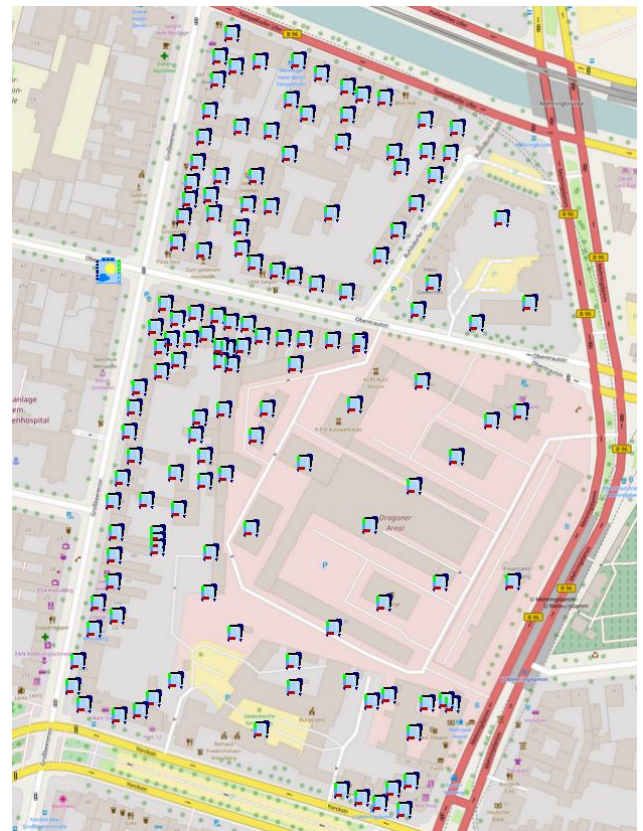


Figure 4. Generated Modelica district model with 144 low-order building models (the City map is taken from OpenStreetMap, <https://www.openstreetmap.org>)

3.2 Multi-zone building

The second case study demonstrates the code generation of a multi-zone building model (a storey of an office building) with thirteen thermal zones. It includes eight single office rooms, each of them with the same floor space, oriented to the North. Second, it has a bullpen with a large south oriented window and a smaller west oriented window. Beside the bullpen a conference room is attached, which has also a south oriented window. Further the storey includes two sanitary rooms without windows and a corridor, which divides the north oriented by the south oriented rooms as a thermal buffer zone.

Data source: The building was constructed in Archicad 19 (see Figure 5) and afterwards exported as an IFC2x3 file. This model includes a precise description of the building geometry, topology and also the information about the layered construction of the building (used materials and the thicknesses of each layer).

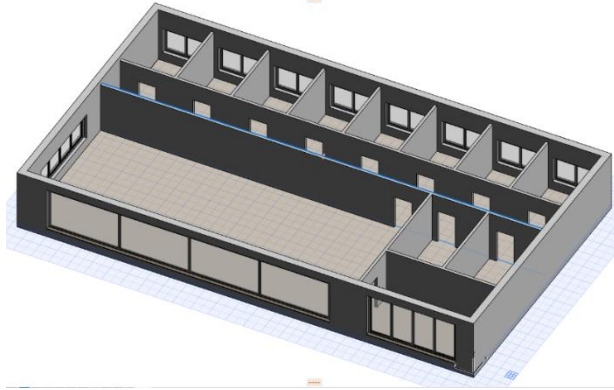


Figure 5. Building model, constructed in Archicad 19.

Data mapper: The data mapper reads the IFC file and analyses the building geometry and modifies if necessary the topology. For example, the south façade of the building is constructed in the CAD tool as one continuous element, but it has to be divided into two individual thermal wall models, because these models will have different thermal boundary conditions in a multi-zone building model. After this analysis the building data is stored in the building data model.

Components: Different models of the BuildingSystems library (opaque and transparent building element models, zone models, building template models etc. and again an ambient model) are used as the base for the code generation (see Figure 6).

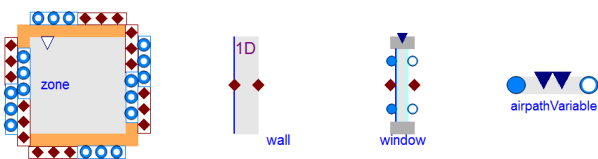


Figure 6. Components for multi-zone modelling.

Code generator: In this case study the stored information in the building data model is used twice:

First for the generation of the Modelica code of the thermal multi-zone building model (see Figure 7) and second for a corresponding C# script, which is able to visualize the simulation results within a 3-dimensional building model (see Figure 8), based on Unity 5 (Nytsch-Geusen et al., 2017).

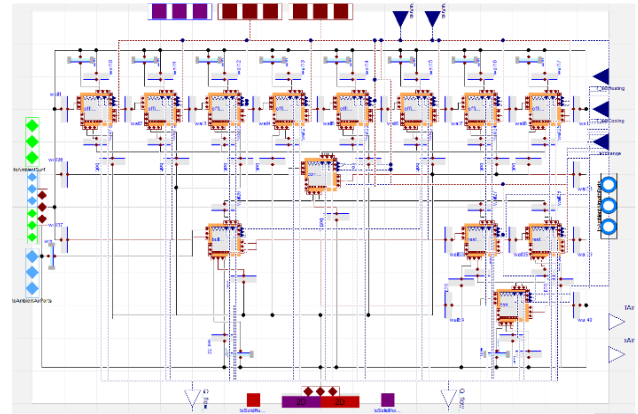


Figure 7. Generated Modelica multi-zone building model with 13 thermal zones.

The excerpt of the generated code shows the instantiation of the individual opaque and transparent building elements, thermal zones and their connections to a multi-zone building model (model Building). In the next step this “container class” is instantiated and connected on a higher level together with the ambient model to the Modelica system model (model MultiZoneBuilding):

```
model MultiZoneBuilding
  extends Modelica.Icons.Example;

  record ConstructionFacade
    extends OpaqueThermalConstruction(
      nLayers=4,
      thickness={0.015,0.2,0.15,0.02},
      ...
    );

  model Building
    extends BuildingTemplate(
      nZones = 13,
      surfacesToAmbient(nSurfaces = 43),
      nSurfacesSolid = 13, ...);

    // building zones
    ZoneTemplateAirvolumeMixed office1(
      V=36.0,height=3.0,
      nConstructions1=8,...);
    ...
    ZoneTemplateAirvolumeMixed bullpen(
      V=450.0,height=3.0,
      nConstructions1=11,...);

    // constructions elements
    WallThermal1DNodes wall11(
      redeclare ConstructionFacade
      constructionData,
      angleDegAzi = 180.0,angleDegTil = 90.0,
      nInnSur = 1, AInnSur = {window2.A},
      height = 3.0,width = 3.0);
    ...
    Window window2(
      angleDegAzi = 180.0,angleDegTil = 90.0,
      height = 1.5,width = 2.5, UVal = ...);
```



```

equation
// construction elements <-> zones
connect(wall111.toSurfacePort_1,
  office2.toConstructionPorts1[1]);
connect(window2.toSurfacePort_1,
  office2.toConstructionPorts1[5]);
...
// construction elements <-> ambient
connect(window2.toSurfacePort_2,
  surfacesToAmbient.toConstructionPorts[5]);
connect(wall111.toSurfacePort_2,
  surfacesToAmbient.toConstructionPorts[6]);
...
// construction elements <-> ground
connect(bottom1.toSurfacePort_2,
  surfacesToSolids.toConstructionPorts[1]);
...
end Building;

Building building(
  show_TSur = true,nSurfaces = 182,nZones = 13);
Ambient ambient(
  nSurfaces = building.nSurfacesAmbient,
  weatherDataFile = WeatherDataFile_Berlin());
equation
connect(ambient.toSurfacePorts,
  building.toAmbientSurfacesPorts);
connect(ambient.toAirPorts,
  building.toAmbientAirPorts);
connect(ambient.TAirRef, building.TAirAmb);
connect(ambient.xAir, building.xAirAmb);
...
end MultiZoneBuilding;

```

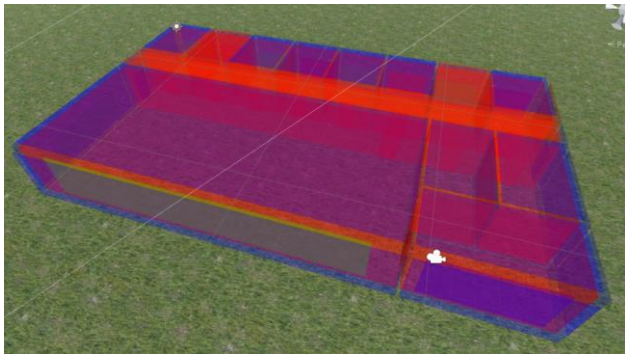


Figure 8. Generated multi-zone Unity building model for visualization of simulation results.

Figure 8 shows the visualization of the simulated surface temperatures of the multi-zone building model. The following code is an excerpt of the automatically generated C# script, which instantiates in Unity 5 this 3D visualization model:

```

using UnityEngine;
using System.Collections;

public class Surfaces : MonoBehaviour{
  public GameObject[] surfaces;
  private int nSur = 182;
  private Vector3 dirY = new Vector3(0,1,0);
  private Vector3 dy = new Vector3(0,0,0);
  private float[] rgb = new float[3];
  private float time = 0.0F;

  void Start(){
    sur = new GameObject[nSur];
    sur[0] = GameObject.CreatePrimitive(
      PrimitiveType.Cube);
    sur[0].name = "wall_sur1";
    sur[0].transform.localScale =
      new Vector3(4.0F,0.01F,3.0F);

```

```

    sur[0].GetComponent<Renderer>().material=
      new Material(Shader.Find("Transparent/Diffuse"));
    sur[0].GetComponent<Renderer>().material.
      color = new Color(1, 0, 0, 0.3F);
    sur[0].transform.rotation =
      Quaternion.Euler(90.0F,90.0F,0.0F);
    sur[0].transform.position =
      new Vector3(0.0F,1.5F,-2.0F);
    sur[0].GetComponent<Collider>().enabled = false;
    dy = sur[0].transform.TransformDirection(dirY);
    sur[1] = GameObject.CreatePrimitive(
      PrimitiveType.Cube);
    sur[1].name = "wall_sur2";
    ...
  }
  void Update(){
    time += 0.01F;
    float[] T_Surface = new float[]{}
    // C# code for reading the simulation results
    // from the Modelica simulation
    ...
  }
  for (int i = 0; i < nSurfaces; i++){
    rgb = RGBMapper (T_Surface[i],10.0F,30.0F);
    sur[i].GetComponent<Renderer>().
      material.color=
      new Color(rgb[0],rgb[1],rgb[2],0.3F);}
  }
}

```

3.3 Single room

The third case study for template based code generation was taken from the DFG Forschergruppe 1736 UCaHS (UCaHS, 2017). Within this project, the indoor climate of a patient room in a Berlin hospital (see Figure 9) was analyzed in detailed regarding the heat stress risk during hot summer weeks.

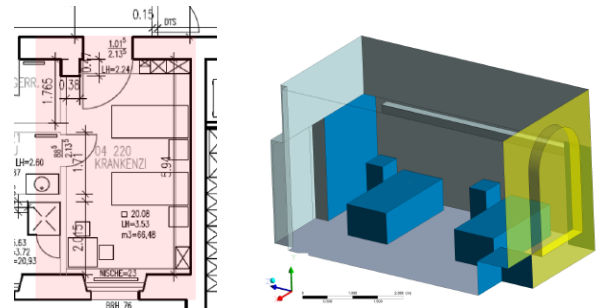


Figure 9. Floorplan and 3D model of the patient room.

For this purpose, a discretized room model in Modelica, a so called “zonal model”, which is based on a finite-volume-method and a simplified implementation of the Navier-Stokes equations was developed by Mucha (2017). A typical configuration of this room model includes between 300 to 500 air volume models, which are interconnected to each other by coupling models, which consider the friction between the air layers and the momentum transport. Caused by the high number of air volume elements and their necessary interconnections a manually failure free configuration of a room model, especially for non-box-shaped rooms would be nearly impossible.

Data source: At the moment the geometrical description of the 3-dim. room geometry inclusive its space discretization and also the physical parameter of

the building construction are stored in a structured JSON file.

DataMapper: The data mapper reads the building parameter from the JSON file and stores it in the building data model.

Components: Figure 10 shows the components of the BuildingSystems library, which are used for the configuration of the space-discretized room model: an air volume model (energy and mass balance), a flow element model (friction calculation within the air), a heat conduction model (heat conduction within the air) and an interface model for the boundary condition of the room model (surface, wall and window models).

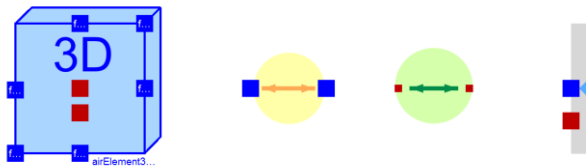


Figure 10. Components for room modelling.

Code generator: The code generator takes the information from the building data model and generates the Modelica code for the space discretized room model. This case study clearly demonstrates the advantage of the template based code generation approach. More than 500 air volume models have to be connected in three room coordinates with flow element models. In addition, different special cases have to be considered during the code generation process, for example the presence of furniture or the changing boundary condition models at the borders of the air space (e.g. a connection of a border air volume model with an adjacent wall or opening model).

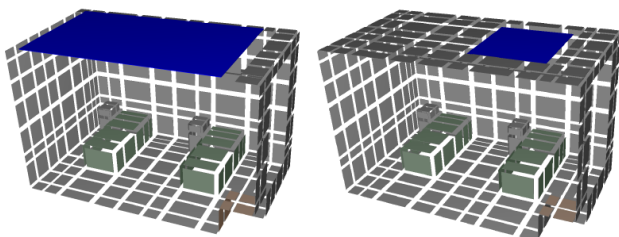


Figure 11. Generated discretized Modelica room model with 532 air volume models.

Figure 11 shows a variation of a generated room model of the patient room: one with a large cooling ceiling and one with a small cooling ceiling, which covers only the area of one of the patient beds. The correspondent adaptations in the building data model, before the code generation is repeated for the varied model are relative simple in comparison to manually changes in the generated code of the originally model.

The excerpt of the generated code exemplary shows the instantiation of two of the air volume elements, the flow and the heat conduction elements and the

interconnections of the components to the 3-dimensional air flow model:

```
model Room
...
FlowConnectionY floConY5710;

ZoneHeatConductionY heaConY5710;

AirElementThermal airEle6710(
  posX= vecX[10], posY= vecY[6], posZ= vecZ[7],
  T_start = T_inside,
  scalF = {scalX[10],scalY[6],scalZ[7]},
  enabled = false, BCwall_west = false,
  BCwall_east = true, BCwall_floor = false,
  BCwall_roof = false, BCwall_south = false,
  BCwall_north = true);
...
FlowConnectionY floConY6710;

ZoneHeatConductionY heaConY6710;

AirElementThermal airEle7710(
  posX= vecX[10], posY= vecY[7], posZ= vecZ[7],
  T_start = T_inside,
  scalF = {scalX[10],scalY[7],scalZ[7]},
  enabled = false,BCwall_west = false,
  BCwall_east = true, BCwall_floor = false,
  BCwall_roof = true, BCwall_south = false,
  BCwall_north = true);
...
equation
...
connect(floConX679.Port2, airEle6710.PortX1);
connect(airEle679.PortHeatIntern,
  heaConX679.Port1);
connect(heaConX679.Port2,
  airEle6710.PortHeatIntern);
connect(airEle679.PortY2, heaConY679.Port1);
connect(floConY679.Port2, airEle779.PortY1);
connect(airEle679.PortHeatIntern,
  heaConY679.Port1);
connect(heaConY679.Port2,
  airEle779.PortHeatIntern);
connect(airEle779.PortX2, airEle779.Port1);
...
end Room;
```

3.4 Analysis and discussion

The three case studies are compared to each other with the help of benchmark values, e.g. the line of codes, the number of components or the number of connections within the generated system model (see Table 1).

Table 1. Comparison of the three case studies.

	District	Building	Room
Lines of code	2,904	1,544	15,985
Number of components	150	173	3713
Number of connections	1,008	544	10,982
Number of equations	435,765	40,434	132,712
Continuous time states	1,872	305	2,481
Time-varying variables	34,285	3,139	30,194

It can be stated, that building energy simulation analysis in Modelica usually leads to large system models. System models with 3,000 up to 16,000 lines of Modelica code cannot be manually configured failure free. The number of the components reaches from 150 to 3,713 and the number of connections from 544 to 10,982.

The generated models of Table 1 can be compiled and simulated without any problems by Dymola 2017 FD01. A test with a generated district model with more than 500 buildings illustrated the present limitations of the Modelica simulation tools: Dymola 2017 FD01 was not able to compile this large model, neither with a 64 bit compiler.

In the case of the district model, information from the GIS system can be used to generate a Modelica model which is able to display the real location of the individual buildings in the city map.

In the case of the multi-zone building model the input data can be used to generate consistent program code for two different purposes (Modelica and Unity code).

In the case of the room model, the code generator enables configuration of 3 dimensional models, which cannot be really modeled within a 2-dimensional graphical editor of a Modelica simulation tool.

4 Summary and Outlook

The described new approach for a template based code generation for Modelica building models was successfully applied to three different case studies on different room scales: district simulation, multi-zone building simulation and room simulation. A building data model, which stores the information in a structured and compact manner in combination with a template based code generator (CoTeTo), can avoid failures of manually written large Modelica system models.

In the next development step, the described Modelica code generators will be extended for special modelling cases. For this purposes, Mako code for conditional code generation will be introduced, which allows variations of generated components and connections within the Modelica system model.

The import of complex building or district data based on IFC or CityGML can be potentially incomplete or error prone. For this purpose, a graphical viewer incl. a consistency check shall be developed in future to obtain a more reliable base for the following code generation process.

Modelica simulator developers should improve their tools regarding the compiler technologies and also their numerical efficiency and flexibility. Especially large city district models, which can be easily generated from the GIS data with the described method, can address a lot of computer memory and potentially need a huge amount of numerical resources. In this context, the application of parallel computing technologies could improve the situation.

Acknowledgements

The research described in this paper is conducted within research project “EnEff BIM: Planung, Auslegung und Betriebsoptimierung von energieeffizienten Neu- und Bestandsbauten durch Modellierung und Simulation auf Basis von Bauwerkinformationsmodellen” funded by the Federal Ministry for Economic Affairs and Energy in Germany (reference: 03ET1177D).

References

- CityGML. Exchange and storage of virtual 3D city models - <http://www.citygml.org> (last access on 2017 Jan 20).
- CoTeTo - Code Templating Tool - <https://github.com/UdK-VPT/CoTeTo> (last access on 2017 Jan 20).
- Alexander Inderfurth, Arda Karasu, Christoph Nytsch-Geusen, Claus Steffan. Architectural-Geometrical Simplification for Multi-Zone Building Models for Urban Refurbishment Projects. Accepted for Building Simulation 2017, 15th International Conference of IBPSA. San Francisco, August 2017.
- GeoJSON. A format for encoding a variety of geographic data structures - <http://geojson.org> (last access on 2017 Jan 20).
- PyQt4. Python bindings for the Qt application framework - <https://riverbankcomputing.com/software/pyqt> (last access on 2017 Jan 20)
- Werner Kaul, Christoph Nytsch-Geusen, Phillip Wehage, and Michael Färber. Teilautomatisierte Akquise energetischer Gebäudedaten für die Quartiersanalyse und - simulation durch den Einsatz von Geo-Informations-Systemen (GIS). BAUSIM 2014 IBPSA Germany. Conference Proceedings. Aachen, September 2014.
- Tobias Loga, Britta Stein, Nikolaus Diefenbach, and Rolf Born. Deutsche Wohngebäudetypologie. Beispielhafte Maßnahmen zur Verbesserung der Energieeffizienz von typischen Wohngebäuden, Institut Wohnen und Umwelt, Darmstadt / Germany, ISBN: 978-3-941140-47-9, 2015.
- IFC2x3. IFC2x Edition 3 specification - <http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html> (last access on 2017 Jan 20)
- IFC4. IFC4 specification - <http://www.buildingsmart-tech.org/ifc/IFC4/final/html/> (last access on 2017 Jan 20)
- IfcOpenShell. The open source IFC toolkit and geometry engine - <http://ifcopenshell.org/python.html> (last access on 2017 Jan 20)
- Mako. Mako templates for python - <http://www.makotemplates.org> (last access on 2017 Jan 20)
- Jinja2. Jinja2 (the python template engine) - <http://jinja.pocoo.org> (last access on 2017 Jan 20)
- Christoph Nytsch-Geusen, and Werner Kaul. Generation of dynamic energetic district models from statistical relationships. 14th IBPSA Building Simulation Conference, Hyderabad, Conference Proceedings, December 2015.
- Christoph Nytsch-Geusen, Christoph Banhardt, Alexander Inderfurth., Katharina Mucha, Jens Möckel, Jörg Rädler, Matthis Thorade, and Carles R. Tugores. BuildingSystems – Eine modular hierarchische Modell-Bibliothek zur energetischen Gebäude- und Anlagensimulation. BAUSIM

2016 IBPSA Germany, Conference Proceedings. Dresden, September 2016.

Christoph Nytsch-Geusen, Thaeba Ayubi, Jens Möckel, Jörg Rädler, Matthis Thorade. BuildingSystems_VR – A new approach for immersive and interactive building energy simulation. Accepted for Building Simulation 2017, 15th International Conference of IBPSA. San Francisco, August 2017.

Katharina Mucha. Ein Simulationsansatz zur Bewertung von Hitzestressrisiken in Innenräumen. Weiterentwicklung eines zonalen Modells in Modelica. Dissertation, Fakultät Gestaltung, Universität der Künste Berlin, 2017.

pythonOCC. pythonOCC – 3D CAD for python - <http://www.pythonocc.org> (last access on 2017 Jan 20).

QGIS. Ein freies Open-Source-Geographisches-Informationssystem - <http://www.qgis.org/de/site> (last access on 2017 Jan 20).

Senatsverwaltung für Stadtentwicklung und Wohnen: Sanierungsgebiet Friedrichshain-Kreuzberg – Rathausblock <http://www.stadtentwicklung.berlin.de/staedtebau/foerderprogramme/stadterneuerung/de/rathausblock/index.shtml> (last access on 2016 Dec 29).

Matthis Thorade, Jörg Rädler, Peter Remmen, Tobias Maile, Reinhard Wimmer, Jun Cao, Moritz Lauster, Christoph Nytsch-Geusen, Dirk Müller, and Christoph van Treeck. An open toolchain for generating Modelica code from Building Information Models. *11th International Modelica Conference*, p.383–391, Versailles, September 2015.

UCaHS. DFG Research Unit 1736 UCaHS - Urban Climate and Heat Stress in mid-latitude cities in view of climate change <http://www.ucahs.org> (last access on 2017 Jan 20).

Wetter Michael, Fuchs Marcus, Grozman Pavel, Helsen Lieve, Jorissen Filip, Lauster Moritz, Müller Dirk, Nytsch-Geusen Christoph, Picard Damien, Sahlin Per, and Thorade Matthis. IEA EBC Annex 60 Modelica Library - An international collaboration to develop a free open-source model library for buildings and community energy systems. 14th IBPSA Building Simulation Conference, Hyderabad, Conference Proceedings, December 2015.

Modelling and Simulation of Standardised Control Functions from Building Automation

Georg Ferdinand Schneider¹ Georg Ambrosius Peßler¹ Simone Steiger¹

¹Group Technical Building Systems, Fraunhofer Institute for Building Physics IBP, Nürnberg, Germany,
georg.schneider@ibp.fraunhofer.de, georg.pestler@ibp.fraunhofer.de,
simone.steiger@ibp.fraunhofer.de

Abstract

Despite the accepted fact that control logic deployed in future and existing buildings through building automation systems constitutes a key factor for increasing their energy efficiency, the support for modelling and simulation of these in current state-of-the-art simulation tools and libraries is rather limited. In particular a gap exists for modelling and simulation of standardised control functions. In this work we present an approach for modelling standardised control logic using Modelica. We evaluated the interoperability of the modelling approach by simulating a test case of an automation solution controlling the sunshade of a room and by reimplementing a state-based control for an air handling unit reusing models from two Annex60 compliant libraries.

Keywords: *Building Automation, Control Function, VDI 3813, VDI 3814, ISO 16484*

1 Introduction

The three pillars which influence the energy demand in buildings are a sophisticated façade, energy efficient technical equipment and the actual operation by means of control through a Building Automation System (BAS). The use of BAS is stipulated by relevant standards, e.g. EN 15232:2013. However, benefits in terms of reduced energy demand from the façade and/or technical equipment can easily be spoilt by operating a building using a poorly designed, misconfigured or malfunctioning BAS.

The operation of a building is a complex control task involving multiple sensors, actuators and control algorithms spanning different scales in terms of time and space; the sum of input and outputs can easily reach ten thousand. Hence, the design, (continuous-) commissioning and operation of such a complex system is a challenging, time- and cost-intensive task.

A possible solution for managing this complexity during BAS design and operation is the use of a Model-Based Design (MBD) methods, where all components of a building are modelled and simulated to design and test the control logic of a BAS prior to its deployment in a building. Also, comparison of the simulation model and the real-world implementation provides a helpful insight in detecting anomalies during operation (Venkatasubrama-

nian et al., 2003). The model and adjacent simulation infrastructure can further be used for Model-in-the-loop and Software-in-the-Loop (SIL) evaluation, e.g. for automotive applications (Chrisofakis et al., 2011) and later in Hardware-in-the-loop simulation, e.g. for circulating pump control (Schneider et al., 2015).

Models to describe the behaviour of control logic and algorithms are part of the Modelica Standard Library (MSL) since its very beginnings. A research effort from the International Energy Agency's Energy Buildings and Communities Programme (IEA EBC) develops as one outcome a core library for Building Performance Simulation (BPS) using Modelica. A set of four libraries, all suitable for MBD within the buildings domain, *Buildings* (Wetter et al., 2014), *AixLib* (Constantin et al., 2014) *BuildingSystems* (Nytsch-Geusen et al., 2013) and *IDEAS* (Baetens et al., 2012) now share one common core library *Annex60* (Wetter et al., 2015). A special library *NCLib* for the simulation of automation systems is presented by Liu (2013), however its focus is on modelling automation system devices rather than the control logic involved. A library specifically designed to model and simulate control functions from industrial automation in Modelica without specific control functions for standardised room or building automation is presented by Bonvini and Leva (2012).

Several national and international standards exist to support the design process of BAS, e.g. VDI 3813-2:2011; VDI 3814-6:2009; ISO 16484:2011 by providing a set of commonly utilised control functions which can be reused to compose an automation solution for room and building automation. The number of defined control functionalities and the detail of the descriptions varies between standards. Having these pieces of control logic readily available in a simulation environment to compose by drag-and-drop a control strategy for a room or a piece of equipment in a building can result in significant benefits in terms of time required for the design and quality of the resulting control logic. Automation engineers may design and test their control solution prior to the deployment in a real building by coupling its simulation to models of rooms, buildings and equipment. Also in later stages of the life-cycle the outcome of the simulation model can be compared with actual monitoring data for fault detection purposes.

However, to the best of our knowledge, no library exists for the modelling and simulation of standardised control functions (see comparison in Table 1). The contribution aspect of this work resides in presenting Modelica-based modelling approach for standardised control functions from BAS.

We present a model library `BuildingControlLib` which provides a basis to implement standardised control functions in a streamlined manner. For the implementation of block oriented control functions from standards we recommend representing the structure by class diagrams and the actual control behaviour for state-based control logic using activity diagrams, as defined by the Unified Modeling Language (UML) (Object Management Group, 2015). The design of the models and library is such that the compatibility to the MSL as well as libraries from Annex60 effort is ensured. The graphical visualisation included in the models is designed such that representations composed to the standards and control solutions may be compiled in a ease-to-use manner from interested BAS practitioners.

As a beginning we include models of control functions compliant to the standards VDI 3813-2:2011 and VDI 3814-6:2009. Due to partly ambiguous textual descriptions for control behaviour in standards the models compliant to VDI 3813-2:2011 are designed such that standardised interfaces and actual control functionality are separated. This offers the benefit that the actual functionality can be easily exchanged with own code or implementations, possibly using the Functional Mockup Interface (FMI) standard (Blochwitz et al., 2012). To support users when composing own control solutions by drag and dropping control functions from the library we include the notion of connector semantics (Dibowski et al., 2010) to help users composing only semantically correct automation solutions; for example it is not possible to connect an indoor air temperature output and an outdoor air temperature input. To represent state-based control we include models to simulate state-based control descriptions in BAS as defined in VDI 3814-6:2009

In the remainder of this work we describe the underlying design principles when modelling standardised control functions in section 2. We then demonstrate the usability of the approach by simulating two test cases where the automation models are coupled to physical room and equipment models from Annex60 compliant libraries in section 3.

2 Implementation

The models are included in a library termed `BuildingControlLib`. Its overall structure is depicted in Figure 1 from a screenshot in Dymola 2015 FD01 (Dymola, 2015) which we use for implementation. The design of the overall structure follows the best practices and conventions documented in the MSL, e.g. naming convention of models and classes, package

structure with a user's guide, ready-to-simulate examples, components, interfaces and types.

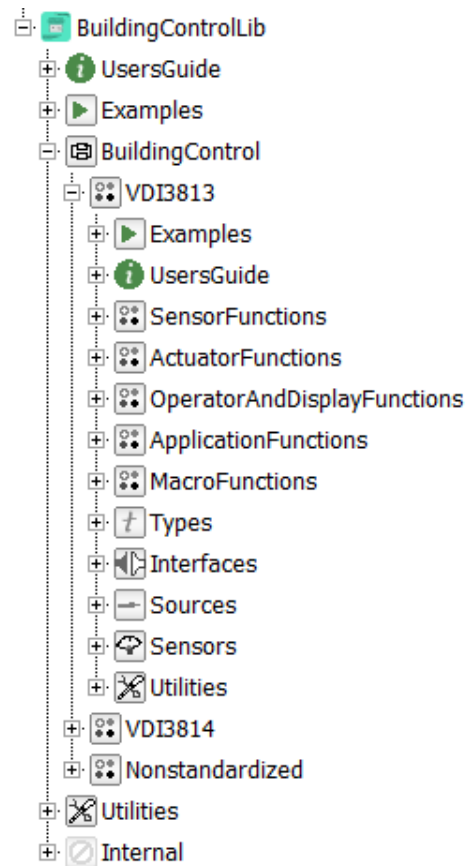


Figure 1. Overall structure of model library.

At the top level we include the mandatory packages for documentation and examples and three packages VDI3813, VDI3814 and Nonstandardized. The number of packages mentioned is not meant to be exhaustive. Instead the implementation undertaken so far may serve as a blue print for implementing control functions from further standards, e.g. ISO 16484:2011.

The first package contains models for the simulation of room automation functions from VDI 3813-2:2011; the second contains models for the simulation of state graphs as defined in VDI 3814-6:2009; and the third package gathers models for common non-standardised control functions. In some cases, e.g. a schedule, we reuse functionality already implemented for non-standardised applications.

Auxiliary models used in example models are kept in the `Utilities` package.

2.1 Room Automation According to VDI 3813

Beside the general best practices for Modelica libraries, e.g. package structure, the following requirements have been defined to enable seamless use of the library:

1. Modular design such that control functionality is encapsulated and may be exchanged as needed;

Table 1. Overview of the reviewed open-source libraries. X - criteria fulfilled and - not fulfilled. (1) - Annex60 (Wetter et al., 2015), (2) - Buildings (Wetter et al., 2014), (3) - AixLib (Constantin et al., 2014), (4) - BuildingSystems (Nytsch-Geusen et al., 2013), (5) - IDEAS (Baetens et al., 2012), (6) NCLib (Liu, 2013), (7) - IndustrialAutomationSystems (Bonvini and Leva, 2012) and (8) - this work.

Criteria	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Models for								
... control	X	X	X	X	X	-	X	X
... room automation	(X)	(X)	(X)	(X)	(X)	-	-	X
... building automation (BA)	(X)	(X)	(X)	(X)	(X)	-	-	X
... standardised BA	-	-	-	-	-	-	-	X
Semantic connectors	-	-	-	-	-	-	-	X
Based on Annex60	obsolete	X	X	X	X	-	-	-
Active development	X	X	X	X	X	-	X	X

2. Automated compatibility checking of control functions as proposed by Dibowski et al. (2010);
3. Graphical representation as defined within the standard.

The first requirement results from prevalent heterogeneity in actual implementations of control behaviour in standards. Existing standards do provide textual descriptions on how the actual behaviour should be, however this descriptions leave room for interpretation. Thus functionalities might comply to standards but have minor differences. To ensure the seamless exchange of functionality, possibly from non-Modelica implementations, we separate the definition of interfaces (function) from its functionality as described in detail in the next sections.

The second requirement is motivated by an approach reported by Dibowski et al. (2010). The methodology described allows to automatically derive interoperable automation solutions for room automation. The approach relies on the formal specification of input and output variables of control functions, by annotating exchange variables with information on e.g. its unit, quantity, etc. The approach is exemplified for control functions for room automation (Dibowski, 2013). As Modelica provides mechanisms for consistency checking on exchange variables the requirement should be fulfilled by implementation to support library users when implementing own solutions.

Finally to allow easy composition of control solutions also by interested practitioners the visual appearance of the modelled control functions should align with the definitions in the standards. Thus allowing the easy reuse and composition of automation solutions.

Structure of VDI3813 Package

The top-level structure of VDI3813 package is illustrated in Figure 1. It follows the taxonomy established within the standard which classifies available control functions into:

- `SensorFunctions` - which convert physical signals into automation signals;

- `ActuatorFunctions` - which receive a setpoint to generate a physical control command for a motor;
- `OperatorAndDisplayFunctions` - which exchange status information to occupants and give them the ability to send manual commands;
- `ApplicationFunctions` - which provide the actual automation functionality by processing sensor or operator functions and transmit new setpoints and commands to actuators;
- `Macrofunctions` - which provide an interface to compose reusable macro-functions from low-level control functions.

The standard also defines supervisory control functions such as data storage and external messaging which we found to be out of scope for dynamic simulation of control behaviour and coupling to models of physical processes. The category *Common I/O functions* with two control functions for interfacing the automation system to external applications is not explicitly modelled.

To ensure computationally efficiency we implement the control functions as Modelica `block` as suggested by Liu (2013). Whenever possible we reuse models from MSL.

Encapsulating Functionality

The actual functionality in the standard is defined using textual descriptions. This leaves a wide range for interpreting and implementing this descriptions; hence, implementations of control functions vary between different manufacturers of devices for room automation.

To represent and model this heterogeneity as defined in the first requirement (see section 2.1) the following design principle is applied within this library. We introduce a block `Function` as a base class for defining the interface to other functions. Each `Function` has a associated block `Functionality` (see Figure 2) which serves as a template to implement the respective intended functionality. This allows for easy maintainability and quick exchange of functionality within the library, e.g.

a manufacturer would like to place his own functionality within the model, potentially using the FMI standard (Blochwitz et al., 2012). By leaving the respective block `Function` unaltered its interoperability with other function blocks is ensured and changes are only applied within `Functionality`.

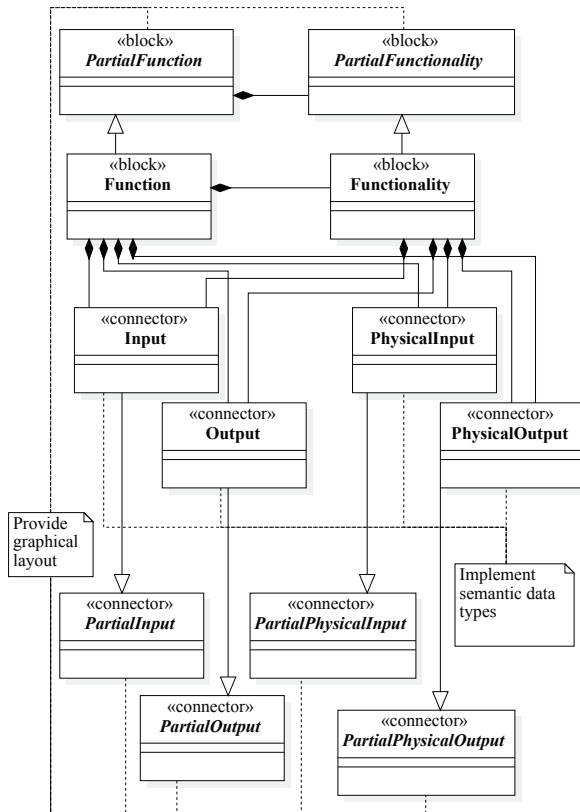


Figure 2. Class diagram in UML describing the modelling principle of encapsulating functionality.

To ensure uniform and complaint graphical layout of the control functions it is once defined in the partial block `PartialFunction` using Modelica annotations.

To implement a control function a block is created which inherits from `PartialFunction`. Parameters and connectors must be added. In the corresponding functionality block the actual control logic is implemented in what ever way is preferred, e.g. reusing models from other libraries.

For all implemented control functions we provide a default functionality which may be adapted to the users needs or exchanged if required. However, regarding the mentioned space for interpretation arising from textual descriptions in the standard these are not meant to be normative. To ensure understandability we provide UML activity diagrams for describing the respective functionality as implemented and include it into the documentation of each model. As an example we present the UML activity diagram of `AutomaticThermalControl` in Figure 3.

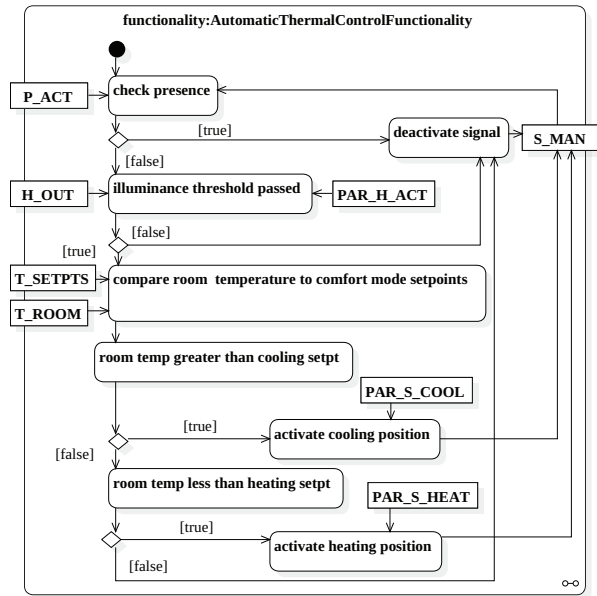


Figure 3. UML activity diagram of a functionality `AutomaticThermalControl` as described in VDI 3813-2:2011.

Semantic Connectors

To solve the task of automating the design process of room automation systems recently the semantically unambiguous specification of function profiles for room automation is introduced (Dibowski et al., 2010; Dibowski, 2013). This approach allows for the automated generation of room automation profiles, i.e. a set of control functions from a standard, automatically from initially defined requirements. The approach requires to formally specify automation devices including their functional profiles, i.e. the control functions implemented on a devices. Moreover to automatically bind variables of different control functions detailed semantics of the input and output variables are specified. The approach has been successfully demonstrate by modelling functions and devices complying to the VDI 3813-2:2011 standard.

To support library users in the design of a room automation solution using the library we integrate the notion of semantics in the design of connector classes. We define for every variable type in the standard a separate connector class. We specify the unit, quantity, basic type (`Real`, `Boolean`, `Integer`, ...) and direction of information flow (input/output) in the connector definition and a corresponding type definition using the known Modelica language elements for this. As emphasised by Dibowski et al. (2010); Dibowski (2013) these specifications are not enough to differentiate among input and output variables. As Modelica does not provide additional ways to specify variable semantics we introduce a naming convention for exchange variables specified in the connector classes. The naming convention has two parts: (1) Each variable starts with one of the strings *value*, *status*, *command* and *set-point*, a classification recommended by Dibowski (2013);

(2) in the second part we added additional strings specifying additional semantics e.g. to differentiate between an indoor and an outdoor air temperature.

For example for the exchange variable describing the outdoor illuminance abbreviated in the standard as *H_OUT* we specify a connector *ValueIlluminanceOutdoor*:

Listing 1. Source code of connector *ValueIlluminanceOutdoorInput*.

```
connector ValueIlluminanceOutdoorInput
  extends Partial.PartialInput;
input
  BuildingControlLib.[ ... ].
    ValueIlluminanceOutdoor
  valueIlluminanceOutdoor; // Specified
    variable name
end ValueIlluminanceOutdoorInput;
```

The semantic correctness when composing automation solutions from drag and dropping control function blocks in e.g. a macro is ensured by the ability of a Modelica simulation environment to check connector compatibility in terms of unit, quantity, basic type, input/output and the name of the variable. Hence, a user can only connect different control functions if input and output connectors match. To ensure compatibility with models which implement connectors using the MSL interfaces, converter models are provided in the *Sources* and *Sensors* packages.

2.2 State Graph According to VDI 3814

Control logic for the control of Heating Ventilation and Air-Conditioning (HVAC) often follows a state-centric behaviour. Multiple descriptions exist for modelling this behaviour originally described by Harel (1987). The standard VDI 3814-6:2009 defines the concept of a *State Graph* to provide a graphical representation of state-centric control behaviour in BAS.

In the package *VDI3814* we provide models to compose by drag-and-drop a State Graph. The graphical layout of the models fits the definitions in the standard (see Figure 4). For implementing we use models from *Modelica.StateGraph* package from MSL.

An example of the modelling capabilities of the package is displayed in Figure 4 where the control of a generic air handling unit is modelled with control states off, cooling, heating or frost protection. A specific characteristic of VDI 3814 State Graphs is that the explicit concept of a transition does not exist. A new state is active when a Boolean expression is evaluated as true. However transition conditions are included into a state. This definition is modelled by including an array of transitions and a state into one model. This allows to display the designed state graphs as defined in the standard while reusing the models from *Modelica.StateGraph* package. Also in the standard it is possible to put *return* objects symbolised by a circle around a number of the targeted state. This can also be modelled and when removing the graphical anno-

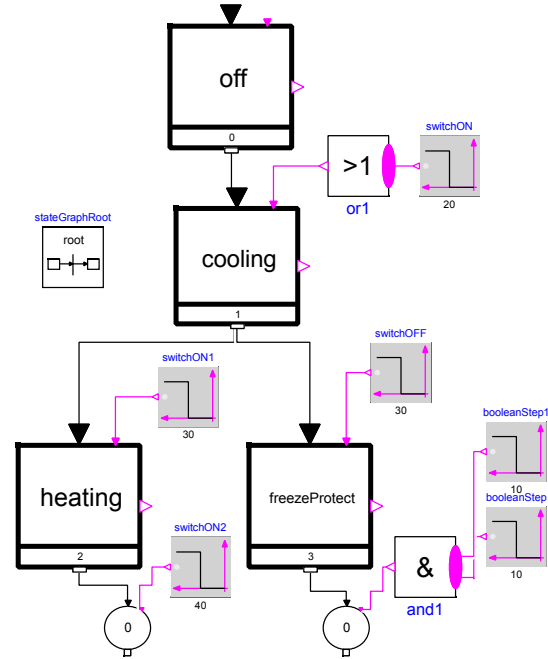


Figure 4. Ready to simulate model of a state graph from package *VDI3814*.

tations of the connect statements it is possible to derive a graphical layout which is very similar to the one used in the standard.

Additionally we reused blocks from MSL to model logical conjunction and logical disjunction as required in the standard.

3 Results from Test Cases

To evaluate the performance of the implemented control library and to examine its interoperability to existing Modelica libraries in the domain, we present results from two test cases implemented for demonstration purposes. In the first test case we implement an automation solution of a sunshade in a room using control function models from the package *VDI3813* where we reuse a room model included in the *AixLib*-library (Constantin et al., 2014). In the second example we re-implement the state-based control of an Air Handling Unit from *Buildings*-library (Wetter et al., 2014) using the models provided in the *VDI3814* package.

3.1 Room automation according to VDI 3813

Most room automation control functions focus on maintaining acceptable indoor comfort conditions for occupants while minimising the energy demand required to provide these comfort conditions to the occupant.

In Figure 5 the scheme of an automation solution for a room with a sunshade is illustrated. It is composed of control functions from VDI 3813-2:2011 which are modelled within the library presented in this work. As outlined above, the standard distinguishes between: sensor functions; actuator functions; operator and display func-

tions and application functions. Disconnected inputs in the simulation of this test case have been fixed to reasonable constants for keeping results easy to understand.

The overall functionality intended to be realised here automatically limits or increases the amount of solar heat gains of a room either by deploying or elevating a sunshade, respectively. It is implemented as a macro function (VDI 3813-2:2011) using the application functions `OccupancyEvaluation`, `PriorityControl`, `AutomaticThermalControl` and `SetpointCalculation`. The connection to the physical inputs required by these functions is realised using the control functions `PresenceDetection` (Check with a sensor if room is occupied), `WindowMonitoring` (Sensor function to check if window is open), `BrightnessMeasurementOutdoor` (Determine outdoor brightness), `AirTemperatureMeasurementRoom` (Determine room air temperature) and `AirTemperatureMeasurementOutdoor` (Determine outdoor air temperature). A user may adjust the current temperature setpoint via the `AdjustTemperatureSetpoint` function. The outcome of the automation solution affects the actual (real or virtual) sunshade by using the function `sunshadeActuator`.

`AutomaticThermalControl` is only active if the room is unoccupied and the outdoor illuminance levels are higher than a threshold. Then, dependent on the comparison of current setpoint of the room and the measured room temperature, the sunshade is either deployed or elevated.

Between the control output of `AutomaticThermalControl` and its actual deployment via the `ActuateSunshade` control function finally, i.e. `PriorityControl` checks if no higher prioritised signal is found or the window is open. If the signal of `AutomaticThermalControl` is of current highest priority, it is then forwarded to the `SunshadeActuator` control function and deployed on the actual sunshade.

We implemented the described room automation solution using models from the previously described library. We couple it to a model which captures the physical behaviour of a room which we adapted from the model `ASHRAE140.Case900FF` from Constantin et al. (2014). The boundary conditions (weather, internal gains, etc.) remained unchanged. We adapted the ventilation schedule to 30 min once every day and introduced a constant heat flow rate of 500 W to heat the room in order to limit the outcome of these disturbances to the sunshade control on the automation solution. Also we modified the model of the south facing wall to contain a window with a sunshade which may be deployed from outside via a Boolean connector. We calculate the value of the outdoor illuminance assuming a constant factor of 2.49 between the value of irradiation on a horizontal surface provided by the model `radOnTiltedSurf_Perez[5]`.

We simulated the coupled room and automation model for the first day in the weather file. We chose boundary conditions such that no presence is detected and no au-

tomatic or manual set point changes are applied. Hence the automatic control is active only when illuminance levels are sufficient. Results of the simulation are presented in Figure 6. Presented therein are the input signals affecting the `AutomaticThermalControl` control function, i.e. in the upper third the outdoor illuminance H which is compared within the control function with a threshold P_H . If the illuminance is too low, no control action happens. In the mid part the Boolean expressions indicating if `AutomaticThermalControl` is operating in heating y_{hea} or cooling mode y_{coo} and a signal telling the sunshade to be deployed or not are plotted (u_{sun}). In the lowest subplot in Figure 6 the outdoor T_{oa} and indoor air temperature T_{ra} are given and their respective heating ($T_{hea,s}$) and cooling ($T_{coo,s}$) set point. Also the air exchange rate AER is given, representing the ventilation scheme applied.

Given the described boundary conditions the control functionality allows to keep the room temperature within the bounds set by the heating and cooling set points, $T_{hea,s}$ and $T_{coo,s}$ respectively. Before ventilating (time < 43200 s) the sunshade is deployed several times when the control switches to cooling mode, triggered by the actual room temperature reaching the upper temperature set point at 24 degrees Celsius. The sharp decline during ventilation results in a period where the heating mode is active and the sunshade is elevated.

When the temperature recovers after ventilation with outdoor air the cooling mode is active until the automatic control enters the inactive mode when illuminance levels fall below the threshold specified (P_H).

3.2 Control of an Air Handling Unit via VDI 3814 compliant State Graph

To evaluate the models implemented in the package VDI3814 we re-implement the model `ModeSelector` from the `Buildings-library` (Wetter et al., 2014). The model is used in the example `VAVReheat.ClosedLoop` to control a model of an Air Handling Unit which supplies conditioned air to a thermal zone. It encompasses six states representing the behaviour of the system (initial, unoccupied off, morning pre-cooling, morning warm-up, unoccupied night set back and occupied).

We simulate `VAVReheat.ClosedLoop` with the provided model and with the same control logic implemented using models from VDI3814-package. We use a Dymola 2015 FD01, 64bit with the following simulation settings:

- start time: 0 s, end time: 172800 s;
- solver: Esdirk23a - order 3 stiff;
- interval length: 600 s; Tolerance: 1e-6.

We compare the results of the two simulations by calculating the R squared value and the Mean Absolute Percentage Error (MAPE) between the respective results. The

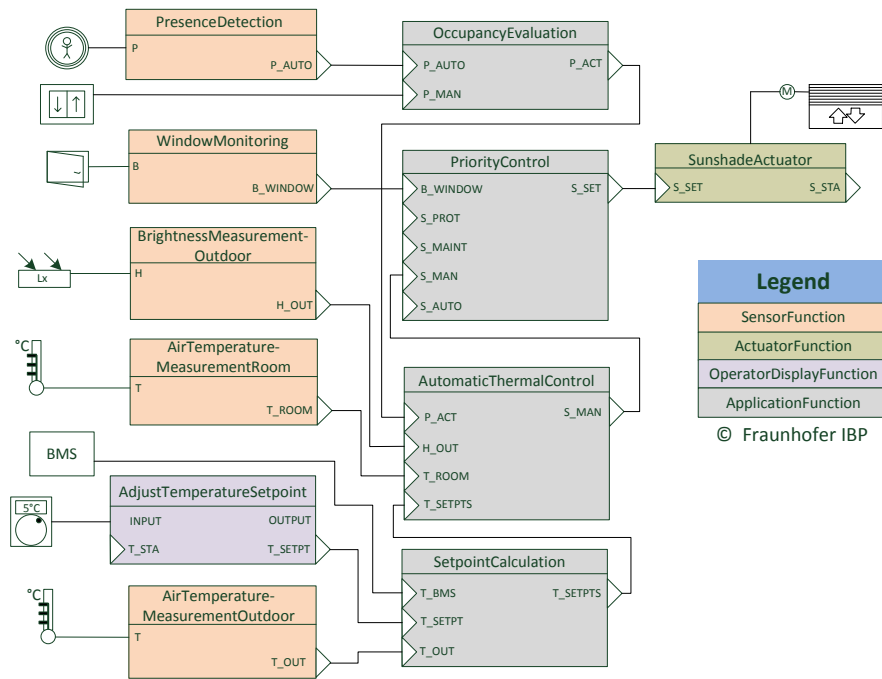


Figure 5. Scheme of the control functions utilised in the simulated room automation test case.

results are summarised in Table 2. The results show a high agreement of the two simulations reflecting the similar behaviour. Some discrepancies can be observed on the compared simulated data which can be explained from errors resulting from interpolating these values. An error of 0.192% for the control mode exists as a negligible deviation between the control mode signals of both simulation was identified; this originates from the waiting times which at some point in the state graph network need to be introduced. For the VDI3814 case with integrating transitions and states in one model, a fixed delay is introduced to avoid the termination of the simulation when initialising.

Table 2. Results from comparing the simulation of VAVReheat.ClosedLoop with the original model from Buildings library (Wetter et al., 2014) and this work. MAPE - Mean Absolute Percentage Error.

Variable	R ²	MAPE in %
TOut	.999	6.816e-6
controlMode	.970	0.192
occupied	1.00	0.000
TRooMin	0.999	6.475e-4
TRooAve	0.999	6.157e-4
TRooSetCoo	0.997	20.24e-4
TRooSetHea	0.997	19.48e-4

4 Discussion

From the results presented in this work the Modelica modelling language is found to be suitable for the proposed

modelling and simulation of standardised control functions.

However, some limitations have been identified when implementing the notion of semantic connectors. We found the ability of the Modelica language to support checking of the consistency of connector variables according to units, quantity and the name of the variable to be a helpful feature. However, when attempting to include detailed semantics of connector variables the only possibility is to introduce a naming convention for the exchange variables which is cumbersome to maintain and prone to errors. It may be of interest to extend the Modelica language in future releases in this direction to evaluate the types of a connector variable (not only the basic type, e.g. Real, Boolean, Integer, etc.) allowing to define a taxonomy of types instead of a naming convention. An approach known from the simulation of cyber-physical systems embedded in the Ptolemy II framework offers a possibility to include consistency checking (Lung et al., 2009) of units based on ontology. The concept of expandable connector is not found to be suitable in this context as a variable name might occur several times in one automation solution but must not be connected to each of its occurrences.

Standards are evaluated and revised on a regular basis, thus a regular revision and maintenance of the library is required. We are confident that the presented underlying design principles of the modelling approach remain relevant and applicable to future and upcoming versions of standards.

Most standards provide textual descriptions of the functionality of a control function which often is ambiguous

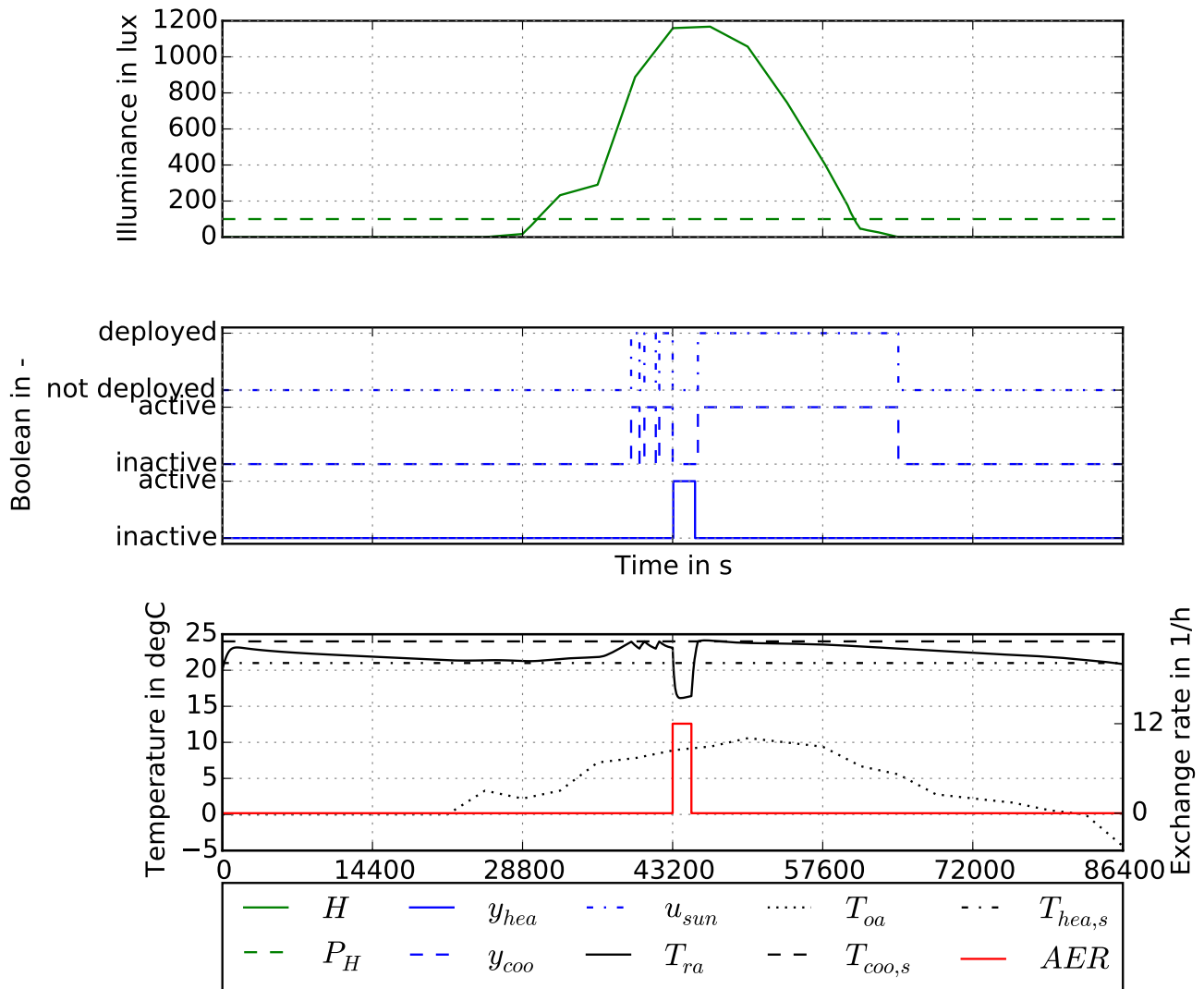


Figure 6. Simulation results of room automation example for one day. H - outdoor illuminance, P_H - threshold for illuminance, $y_{heating}$ - heating mode, $y_{cooling}$ - cooling mode, u_{sun} - sunshade control signal, $T_{hea,s}$ - temperature set point heating, $T_{coo,s}$ - temperature setpoint cooling, T_{ra} - room air temperature, T_{oa} - outdoor air temperature, AER - air exchange rate.

and is prone for different interpretation. When implementing the models we use established documentation measures, namely UML class and activity diagrams to document our implementation. Having well-documented, commonly-agreed on simulation models of the control functions available as a reference implementation may help the wide-spread and further adoption of BAS in the buildings domain.

When modelling buildings, technical equipment and components and control logic of BAS the resulting hybrid system involves continuous and discrete event dynamics (Fritzson, 2014). In particular modelled discrete behaviour in control logic, e.g. a transition from one state to another if the condition $t_{Room} > 22^{\circ}C$ is evaluated to true, triggers events involving state variables which need to be handled by the numerical solver. A large number of these events leads to a significant slow down of simulation speed when simulating the mentioned systems.

The Modelica modelling language provides built-in functionalities to efficiently handle events and should be applied when ever possible. Discrete behaviour with respect to time, e.g. sampling, can be efficiently handled using discrete variables or clocks introduced in Modelica 3.3 language specification (Otter et al., 2012).

The generation of state events can be prevented from using `noEvent(expression)` in case it is known that the respective expression is continuous and `smooth(p,expression)` if not known.

The use of clocked variables and expressions seems to be a promising path for efficient implementation of control behaviour in Modelica. In particular the ability to transfer clocked control systems from its Modelica implementation to clocked control hardware is a huge benefit. However, its effect on computational efficiency when simulating needs to be investigated as in the buildings domain distinguishing models with discrete and continuous dynamics is sometimes difficult; For example the discrete behaviour of a user opening a window when some temperature threshold is crossed may be modelled within the buildings model, thus discrete and continuous models are mixed.

5 Conclusion

In this work we present a modelling approach to model and simulate standardised control functions from building automation in Modelica. We exemplify this by modelling block like control functions from VDI 3813-2:2011 and state-centric control from VDI 3814-6:2009. In particular this includes models for sensor, actuator, operator and display, application control functions and a template to model macro functions from VDI 3813-2:2011 and set of models to compose state graphs as specified in VDI 3814-6:2009 built on top of *StateGraph* package from Modelica Standard Library.

The usability of the models is demonstrated in two example applications linking a room automation solution

to room models from *AixLib*-library (Constantin et al., 2014) and a state graph to control an air handling unit model from *Buildings*-library (Wetter et al., 2014).

The models presented, along with the models existing for building elements and equipment, allow to investigate the interaction and influences of an automation solution on the buildings behaviour in an integrated manner. Through the respective feedback from user models and also the interaction of user and automation solution as it is implemented in a real BAS is possible.

The total number of models and standards described included here is still limited. In future we plan to include more standardised, e.g. from ISO 16484:2011, and non-standardised control functions, e.g. for HVAC control.

We introduce the notion of a *semantic connector* which allows the library user to only connect control functions which are supposed to be connected, following the idea presented by Dibowski et al. (2010). The approach relies on a naming convention, despite the ability of consistency checking of Modelica modelling language for quantities and units. Future research may expand on this allowing to define variable semantics more freely as previously discussed and implemented by Leung et al. (2009).

In future we intend to investigate the potential benefits of using clocked variables in the definition of BAS control behaviour and standardised control functions for efficient simulation of the resulting hybrid systems.

Our intention is to stream line this effort with developments connected to the Annex60 effort and stipulate collaboration and reuse by open-sourcing the described models. For this purpose we intend to integrate the models within an open library. Through doing this we hope that this effort acts as a catalyst for implementing and providing control logic from BAS for building control in a comprehensive, well-documented and efficiently implemented way.

Acknowledgements

This research was performed as part of the Energie Campus Nürnberg and supported by funding through the 'Aufbruch Bayern (Bavaria on the move)' initiative of the state of Bavaria.

References

- R. Baetens, R. De Coninck, J. Van Roy, B. Verbruggen, J. Driesen, L. Helsen, and D. Saelens. Assessing electrical bottlenecks at feeder level for residential net zero-energy buildings by integrated system simulation. *Applied Energy*, 96:74–83, 2012.
- T. Blochwitz, M. Otter, J. Akesson, M. Arnold, C. Clauß, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauß, D. Neumerkel, H. Olsson, and A. Viel. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the International Modelica Conference*, pages 173–184, Munich, Germany, 2012.

- M. Bonvini and A. Leva. A modelica library for industrial control systems. In *Proceedings of the International Modelica Conference*, pages 477–484, Munich, Germany, 2012.
- E. Chrisofakis, A. Junghanns, C. Kehrer, and A. Rink. Simulation-based development of automotive control software with Modelica. In *Proceedings of the International Modelica Conference*, pages 1–7, Dresden, Germany, 2011.
- A. Constantin, R. Streblow, and D. Müller. The Modelica *HouseModels* Library: Presentation and Evaluation of a Room Model with the ASHRAE Standard 140. In *Proceedings of the International Modelica Conference*, pages 293–299, Lund, Sweden, 2014.
- H. Dibowski. *Semantischer Gerätebeschreibungsansatz für einen automatisierten Entwurf von Raumautomationssystemen*. PhD thesis, Department of Computer Science, TU Dresden, Dresden, Germany, 2013.
- H. Dibowski, J. Ploennigs, and K. Kabitzsch. Automated Design of Building Automation Systems. *IEEE Transactions on Industrial Electronics*, 57(11):3606–3613, 2010.
- Dymola, 2015. URL <http://www.3ds.com/products-services/catia/products/dymola>. Dassault Systemes AB, Lund, Sweden, [Accessed: 31-12-2016].
- EN 15232:2013. Energy performance of buildings - Impact of Building Automation, Controls and Building Management, 2013.
- P. Fritzson. *Principles of object-oriented modeling and simulation with Modelica 3.3: A cyber-physical approach*. John Wiley & Sons, 2014.
- D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.
- ISO 16484:2011. Building automation and control systems (BACS), 2011.
- J. M.-K. Leung, T. Mandl, E. A. Lee, B. Osyk, C. Shelton, S. Tripakakis, and B. Lickly. Scalable semantic annotation using lattice-based ontologies. In *Proceedings of MDELS*, pages 393–407, Denver, USA, 2009.
- L. Liu. *Object-oriented Modeling and Efficient Simulation of C3-Systems*. PhD thesis, University of Saarland, Saarbrücken, Germany, 2013.
- C. Nytsch-Geusen, J. Huber, M. Ljubijankic, and J. Rädler. Modelica BuildingSystems- eine Modellbibliothek zur Simulation komplexer energietechnischer Gebäudesysteme. *Bauphysik*, 35(1):21–29, 2013.
- Object Management Group. OMG Unified Modeling Language, 2015.
- M. Otter, B. Thiele, and H. Elmquist. A Library for Synchronous Control Systems in Modelica. In *Proceedings of International Modelica Conference*, 2012.
- G. F. Schneider, J. Oppermann, A. Constantin, R. Streblow, and D. Müller. Hardware-in-the-Loop-Simulation of a Building Energy and Control System to Investigate Circulating Pump Control Using Modelica. In *Proceedings of the International Modelica Conference*, pages 225–233, Versailles, France, 2015.
- VDI 3813-2:2011. Building automation and control systems (BACS) Room control functions (RA functions), 2011.
- VDI 3814-6:2009. Building automation and control systems (BACS) Graphical description of logic control tasks, 2009.
- V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. *Computers & Chemical Engineering*, 27(3):293 – 311, 2003.
- M. Wetter, W. Zuo, T. S. Noudui, and X. Pang. Modelica Buildings Library. *Journal of Building Performance Simulation*, 7(4):253–270, 2014.
- M. Wetter, M. Fuchs, P. Grozman, L. Helsen, F. Jorissen, M. Lauster, D. Müller, C. Nytsch-Geusen, D. Picard, P. Sahlin, and M. Thorade. IEA EBC Annex 60 Modelica Library - An International Collaboration to Develop a Free Open-Source Model Library for Buildings and Community Energy Systems. In *BuiSim 2015*, Hyderabad, India, 2015.

Modelling of Heat Pumps with Calibrated Parameters Based on Manufacturer Data

Massimo Cimmino¹ Michael Wetter²

¹Department of Mechanical Engineering, Polytechnique Montreal, Montreal QC, Canada,
massimo.cimmino@polymtl.ca

²Lawrence Berkeley National Laboratory, Energy Technologies Area, Building Technology and Urban Systems Division,
Simulation Research Group, Berkeley CA, USA, mwetter@lbl.gov

Abstract

A Modelica model for the simulation of heat pumps is presented. The model uses a simplified vapor compression cycle with only five refrigerant states. Parameters to the model are evaluated using an optimization procedure to minimize the differences between the model predicted heating capacities and power input and those provided in the manufacturer technical data. The optimization process is done from a Python implementation of the heat pump model.

The model is first tested by verifying that calibration from performance data generated by the heat pump model results in the same parameters as the ones used in the generation of the performance data. In the presented example, calibrated parameters were found close to the original parameters used to generate the data, except for the evaporator heat transfer coefficient for which the model was found not to be very sensitive. In a second example, the model is calibrated against manufacturer data. The heating capacities and power input calculated from the calibrated model are within 2.7% and 4.7% of the manufacturer data, respectively. Finally, the computational performance of the model is tested in a system simulation of a hydronic heating system. The simulation using the presented heat pump model was executed in 48 seconds, compared to 17 seconds for the same system using a simple boiler model.

Keywords: Heat Pump, Vapor Compression Cycle, Model Calibration

1 Introduction

Heat pump systems offer great potential for the reduction of energy use for heating, cooling and heat recovery, and are attractive heat delivery systems in applications involving low temperature thermal networks (Lund et al., 2014). To optimize the design and evaluate the energy performance of such systems, efficient simulation tools are required to model the annual behavior of the system components.

Heat pump models can be divided into two major categories: empirical models and refrigerant cycle models. Empirical models are obtained by mapping the

heat pump performance in terms of capacity, power input and coefficient of performance to the operating conditions, i.e. the water mass flow rates and temperatures on the load and source side of the heat pump. The performance map can then be interpolated during numerical simulations, or used to produce an equation-fit of the heat pump performance. On the other hand, refrigerant cycle models are obtained from first principles, with varying degree of details in the definition of each heat pump component.

Empirical models have been shown to provide good approximations of the heat pump performance as shown, for instance, by Swider (2003), Lee and Lu (2010) and Carbonnell et al. (2012). However, researchers have pointed out that these models might not be suitable for extrapolation of the heat pump performance outside of the operating conditions used to formulate the model (Jin, 2002; Scarpa et al., 2012). Unfortunately, such extrapolation is often required as manufacturers generally provide performance data for a narrow operating range. Models based on first principles offer better potential to accurately predict the heat pump performance over a wider range of operating conditions.

Refrigerant cycle models are often more demanding in terms of computational time when compared to empirical models, and may require parameters not provided by manufacturers. Simplified vapor compression cycles may be used to reduce the computational time (Domanski and McLinden, 1992; Jin, 2002; Lemort and Bertagnolio, 2010; Scarpa et al., 2012). These simplified cycles divide the vapor compression cycle into a limited number of steps and refrigerant states, thereby reducing the number of – usually computationally expensive – refrigerant thermodynamic properties to evaluate. Parameters to these models may then be obtained through calibration, using an optimization procedure to minimize the model predicted heat pump performance and the performance data from the manufacturer.

A calibrated water to water heat pump model with a scroll compressor is presented in this paper, based on the work of Jin (2002). The model relies on a simplified vapor compression cycle with 5 refrigerant states, where

only 3 of the states need to have refrigerant thermodynamic properties evaluated. The model is implemented into the Modelica `Buildings` library (Wetter et al., 2014). An external implementation of the heat pump model into Python is used to obtain the calibrated model parameters based on tabulated manufacturer data. The computational efficiency of the Modelica model is tested in the simulation of a hydronic heating system.

2 Heat Pump Model

A heat pump model has been built from components from – and components added to – the `Buildings` library (Wetter et al., 2014). The model presented in this paper is for a water to water heat pump with a scroll compressor using refrigerant R410A and is shown in Figure 1. The heat pump model incorporates two new component models, `Buildings.Fluid.HeatExchangers.EvaporatorCondenser` for the evaporator and condenser and `Buildings.Fluid.HeatPumps.Compressors.ScrollCompressor` for the scroll compressor, as well as a refrigerant package for the thermodynamic properties of R410A `Buildings.Media.Refrigerants.R410A`.

The heat pump model is based on the work of Jin (2002), and has been extended to allow for single- and variable-speed compressors and dynamic heat storage on the water side. The model relies on a simplified vapor compression cycle, which removes the need to explicitly model the expansion device. The model is meant to use parameters for the sub-components, obtained from calibration of the model to manufacturer data.

The vapor compression cycle and the refrigerant, evaporator, condenser and compressor models and their implementation in Modelica are presented in this section.

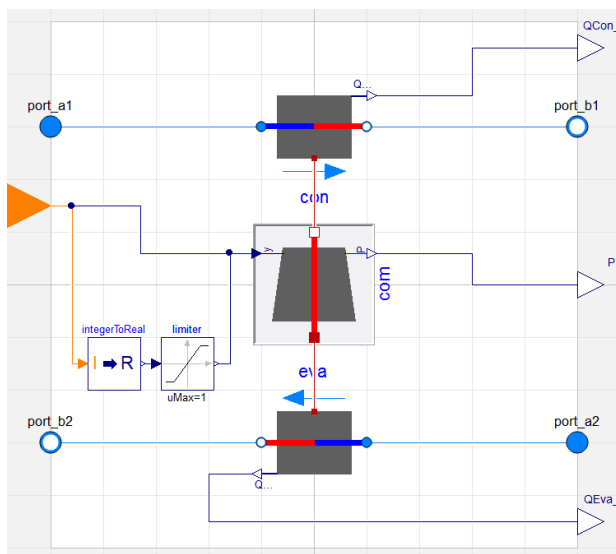


Figure 1. Model of a water to water heat pump with a scroll compressor.

2.1 Simplified Vapor Compression Cycle

A simplified vapor compression heat pump cycle, as proposed by Jin (2002), is presented in Figure 2. The simplified cycle serves two purposes: (1) to reduce the number of parameters in the heat pump model and thereby facilitate the calibration process, and (2) to reduce the number of evaluations of thermodynamic properties of the refrigerant and thereby reduce computing time.

The simplified vapor compression cycle relies on the following assumptions:

1. The refrigerant leaves the condenser in the saturated liquid state, *i.e.* there is no subcooling of the refrigerant.
2. The refrigerant leaves the evaporator in the superheated vapor state, with a constant degree of superheating ΔT_{sup} . The enthalpy increase from superheating has been magnified in Figure 2 and is usually small compared to the latent heat of evaporation.
3. The theoretical compressor work is the result of isentropic compression at the built-in volume ratio followed by isochoric compression or expansion to the condensing pressure.
4. Sensible heat transfer to the refrigerant is neglected in the evaporator.
5. The expansion process is isenthalpic.

From this set of assumptions, only a limited number of refrigerant thermodynamic properties need to be evaluated to solve the complete vapor compression cycle: the temperatures, pressures and specific enthalpies of the saturated vapor and saturated liquid refrigerant (*i.e.* points A and B), and the specific volume and isentropic exponent of the superheated vapor refrigerant (*i.e.* point C).

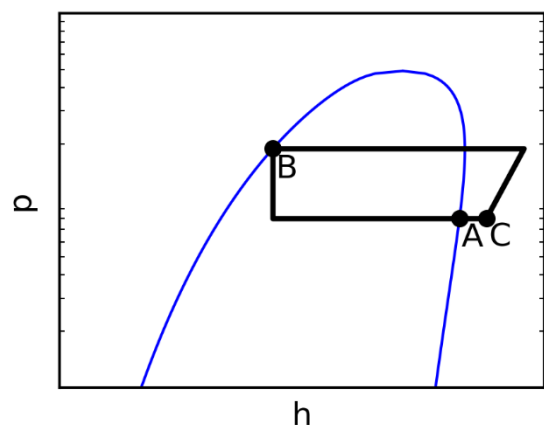


Figure 2. Simplified vapor compression cycle.

2.2 Refrigerant Properties

The necessary routines for the evaluation of the thermodynamic properties of refrigerant R410A were implemented in a media package. Except for the

enthalpy of saturated refrigerant vapor, coefficients for the equations presented in this section were obtained from commercial supplier data (du Pont, 2004). Coefficients for the enthalpy of saturated refrigerant vapor were produced from tabulated properties in the supplier data.

The implemented refrigerant routines and their associated inputs are presented in Table 1. Specific enthalpies and pressures of the saturated liquid and saturated vapor refrigerant are calculated from degree 5 polynomial correlations. Thermodynamic properties of the superheated refrigerant vapor are calculated using the 11-term Martin-Hou equation of state (Martin and Hou, 1955). Note that temperatures in all equations related to thermodynamic properties are in Kelvin.

Table 1. Refrigerant routines.

Output	Input(s)
Specific enthalpy (Saturated liquid), h	T
Pressure (Saturated liquid), p	T
Specific enthalpy (Saturated vapor), h	T
Pressure (Saturated vapor), p	T
Isentropic exponent (Vapor), γ	v, T
Specific isobaric heat capacity (Vapor), c_p	v, T
Specific isochoric heat capacity (Vapor), c_v	v, T
Specific volume (Vapor), v	p, T

The specific enthalpy and pressure of saturated liquid and saturated vapor refrigerant are calculated from degree 5 polynomial correlations of the following form:

$$h = \sum_{i=1}^6 a_i X_h^{i-1} \quad (1)$$

$$\ln(p/p_{cri}) = \sum_{i=1}^6 a_i X_p^{i-1} \quad (2)$$

where p_{cri} is the critical pressure ($= 4926.1$ kPa for R410A), $X_h = (1 - T/T_{cri})^{1/3} - X_0$, T_{cri} is the critical temperature ($= 72.13^\circ\text{C}$ for R410A), $X_p = (1 - T/T_{cri}) - X_0$, and a_i and X_0 are correlation coefficients that differ for Eqs. 1 and 2 and for saturated liquid and saturated vapor.

The specific volume of the superheated vapor refrigerant is evaluated from the Martin-Hou equation of state:

$$p = \frac{RT}{v - b_0} + \sum_{i=1}^4 \frac{a_i + b_i T + c_i \exp\left(-k \frac{T}{T_{cri}}\right)}{(v - b_0)^{i+1}} \quad (3)$$

where R is the gas constant ($= 0.11455$ kJ/(kg $\cdot^\circ\text{C}$) for R410A), and a_i , b_i , c_i and k are coefficients to the equation of state.

During the development of the heat pump model, it was found that the numerical solver needs to solve Eq. 3 for v . In many cases, the numerical solver could not converge since it could not choose a proper guess value for v . A refrigerant routine was then implemented to evaluate the specific volume based on pressure and temperature by successive evaluation of p and $\frac{\partial p}{\partial v}$, starting from a guess value $v_{guess} = RT/p + b_0$. This leads to an efficient implementation of the inverse of Eq. 3, as the guess value v_{guess} is relatively close to the final value.

The isentropic exponent is calculated from the derivatives and integrals of the equation of state (de Monte, 2002):

$$\gamma = c_p/c_v \quad (4)$$

$$c_p = c_v - T \left(\frac{\partial p}{\partial T} \right)_{v,T}^2 / \left(\frac{\partial p}{\partial v} \right)_{v,T} \quad (5)$$

$$c_v = c_{p,id} - R - T \int_{v',T}^v \frac{\partial^2 p}{\partial T^2} dv' \quad (6)$$

The derivatives and integrals in Eqs. 5 and 6 are calculated directly by implementations of the corresponding derivatives and integrals of the equation of state in Eq. 3. The specific isobaric heat capacity of ideal gas, $c_{p,id}$, is evaluated from a degree 3 polynomial correlation based on temperature, in the form:

$$c_{p,id} = \sum_{i=1}^4 a_i T^{i-1} \quad (7)$$

where a_i are correlation coefficients.

2.3 Compressor

The compressor model solves the complete vapor compression cycle presented in Section 2.1. The model interfaces with the evaporator and condenser models through `HeatPorts`. The temperature and heat transfer rates at the ports correspond to the refrigerant temperature and heat transfer rates in the evaporator and condenser.

The scroll compressor model proposed by Jin (2002) was implemented and extended to consider variable-speed compressors. As outlined in Section 2.1, the theoretical compressor work is the result of isentropic compression at the built-in volume ratio followed by isochoric compression or expansion to the condensing pressure. The volume ratio between discharge and suction of the scroll compressor is fixed and the compressor work must be adjusted if the pressure ratio does not match the pressure ratio obtained from isentropic compression at the fixed volume ratio. The theoretical compressor work is then:

$$\dot{W}_t = \frac{\gamma}{\gamma-1} p_{eva} \dot{V}_{nominal} \left(\frac{\gamma-1}{\gamma} \frac{p_{con}}{p_{eva} V_r} + \frac{1}{\gamma} \frac{p_r^{\frac{\gamma-1}{\gamma}}}{p_r} - 1 \right) \quad (8)$$

where \dot{W}_t is the theoretical compressor work, p_{eva} and p_{con} are the evaporating and condensing pressure, γ is the normalized speed of the compressor, with $\gamma = 1$ the value at the nominal speed, $\dot{V}_{nominal}$ is the nominal refrigerant volume flow rate, V_r is the “built-in” volume ratio between discharge and suction of the compressor and $p_r = V_r^\gamma$ is the “built-in” pressure ratio.

The theoretical compressor work is adjusted for the electro-mechanical efficiency of the compressor to obtain the power input into the compressor. A constant electro-mechanical efficiency is assumed:

$$\dot{W} = \frac{\dot{W}_t}{\eta} + \dot{W}_{loss} \quad (9)$$

where \dot{W} is the power input into the compressor, η is the electro-mechanical efficiency of the compressor and \dot{W}_{loss} is the constant part of the compressor power losses.

Since sensible heat transfer is neglected in the evaporator and expansion is considered isenthalpic, the evaporator heat transfer rate is obtained from the enthalpy difference between the enthalpy of saturated vapor at the evaporating pressure (point A in Figure 2) and the enthalpy of saturated liquid at the condensing pressure (point B in Figure 2):

$$\dot{Q}_{eva} = y \left(\frac{\dot{V}_{nominal}}{v_{suc}} - \dot{m}_{leak} \right) (h_A - h_B) \quad (10)$$

where \dot{Q}_{eva} is the evaporator heat transfer rate, v_{suc} is the specific volume at the suction of the compressor (point C in Figure 2) and $\dot{m}_{leak} = C \frac{p_{con}}{p_{eva}}$ is the leakage mass flow rate in the compressor, with C being the leakage coefficient.

The condenser heat transfer rate is then evaluated from an energy balance:

$$\dot{Q}_{con} = -(\dot{Q}_{eva} + \dot{W}) \quad (11)$$

where \dot{Q}_{con} is the condenser heat transfer rate.

The specific enthalpies h_A and h_B are evaluated from the implemented refrigerant routines presented in Table 1 for the saturated liquid at $T = T_{con}$ the condensing temperature and the saturated vapor at $T = T_{eva}$ the evaporating temperature. The specific volume at suction is evaluated for $p = p_{eva}$ and $T = T_{eva} + \Delta T_{sup}$, where T is the temperature of the superheated vapor at the compressor suction. The evaporating and condensing pressure used in Eq. 8 are evaluated from the refrigerant routines for the pressure of saturated vapor evaluated at $T = T_{eva}$ and $T = T_{con}$. The isentropic exponent used in Eq. 8 is evaluated at $v = v_{suc}$ and $T = T_{eva} + \Delta T_{sup}$. Superheating of the

refrigerant is included in the compressor model and not in the evaporator model. Only the effects of superheating on the suction specific volume and isentropic exponent are considered. The superheating enthalpy increase is neglected.

2.4 Evaporator and Condenser

The evaporator and condenser model is shown in Figure 3. It extends from the already implemented `TwoPortHeatMassExchanger` of the `Buildings` library. It interfaces with the compressor model through a `HeatPort`. The refrigerant in both the evaporator and condenser is assumed to exchange heat with the fluid stream at a constant temperature. The effective heat transfer coefficient UA_{eff} between the refrigerant and the fluid is calculated by the $\varepsilon - NTU$ method:

$$NTU = UA / \dot{m}_f c_{p,f} \quad (12)$$

$$\varepsilon = 1 - \exp(-NTU) \quad (13)$$

where NTU is the number of transfer units, ε is the heat exchanger effectiveness, UA is the heat transfer coefficient of the evaporator or condenser, \dot{m}_f is the fluid mass flow rate and $c_{p,f}$ is the fluid specific isobaric heat capacity.

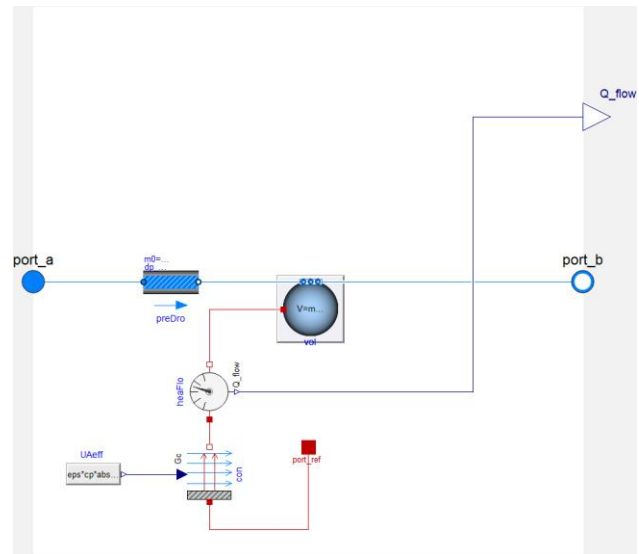


Figure 3. Model used for both the evaporator and condenser of the heat pump.

The effective heat transfer coefficient is then evaluated based on the outlet fluid temperature, since the `HeatPort` of the `MixingVolume` returns the outlet fluid temperature. The heat transfer rate is given by:

$$\{\dot{Q}_{eva}, \dot{Q}_{con}\} = UA_{eff} (\{T_{eva}, T_{con}\} - T_{f,out,\{eva,con\}}) \quad (14)$$

$$UA_{eff} = \varepsilon c_{p,f} \dot{m}_f / (1 - \varepsilon) \quad (15)$$

where $T_{f,out,\{eva,con\}}$ is the outlet fluid temperature in the evaporator or condenser.

3 Model Calibration

The parameters required by the heat pump sub-component models are typically not provided by the heat pump manufacturers. These parameters therefore need to be determined by calibrating the model to the manufacturer data. There are 8 parameters that need to be evaluated: the nominal refrigerant flow rate $\dot{V}_{nominal}$, the volume ratio V_r , the leakage coefficient C , the degree of superheating ΔT_{sup} , the electro-mechanical efficiency η , the constant part of the power losses \dot{W}_{loss} and the heat transfer coefficients UA_{eva} and UA_{con} of the evaporator and condenser.

Manufacturers usually provide technical data in the form of tabulated values of heat pump capacities and power input at different operating conditions in terms of inlet water temperatures and mass flow rates into the evaporator and condenser. Jin (2002) proposed the use of optimization methods to identify the set of parameters that minimize the sum of normalized square errors of the heat pump capacities and power inputs. The cost function to minimize is:

$$Cost = \sum_i \left[\left(\frac{\dot{Q}_{con}^{(i)} - \dot{Q}_{con,data}^{(i)}}{\dot{Q}_{con,data}^{(i)}} \right)^2 + \left(\frac{\dot{W}^{(i)} - \dot{W}_{data}^{(i)}}{\dot{W}_{data}^{(i)}} \right)^2 \right] \quad (16)$$

An optimization routine was set-up in Python using the SciPy (Jones et al., 2001) package. Analogous models for the refrigerant properties, the compressor, the evaporator and the condenser were implemented in Python. The set of parameters that minimizes the cost function are evaluated from the Python model using a sequential least square programming method. Once the parameters are evaluated, the Python implementation of the heat pump model is verified against the Modelica model.

The time required to calibrate the model increases with the number of manufacturer data points that are used. Jin (2002) showed that using the combinations of maximum and minimum entering water temperature and mass flow rates on the evaporator and condenser sides, for a total of 16 data points, decreases the calibration time significantly with minimal effect on the accuracy of the calibrated model. The Python optimization routine thus only uses a subset of 16 data points from the manufacturer data, and compares the model with the complete manufacturer data set once the calibration is complete.

Not all combinations of parameters yield a valid heat pump model. For example, certain sets of parameters may result in refrigerant temperatures in the condenser to be greater than the critical temperature. In these cases, it is not possible for the model to evaluate the capacity and power input of the heat pump, since property

routines for saturated refrigerant (Eqs. 1 and 2) are only valid for temperatures below the critical temperature.

It is then important to choose proper guess values for the parameters when calibrating the model. Guess values of the electro-mechanical efficiency and the degree of superheating are simply chosen to be $\eta = 0.95$ and $\Delta T_{sup} = 4^\circ\text{C}$. The rest of the parameters are evaluated from the nominal values of the heat pump capacity $\dot{Q}_{con,nominal}$, power input $\dot{W}_{nominal}$ and corresponding entering water temperatures $T_{f,in,eva,nominal}$ and $T_{f,in,con,nominal}$, assuming a 5°C temperature difference between the inlet fluid temperatures and the refrigerant temperatures and a 1% leakage mass flow rate. The guess values of the parameters are evaluated following this sequence:

1. Evaluate the refrigerant temperatures:

$$T_{eva} = T_{f,in,eva,nominal} - 5^\circ\text{C} \quad (17)$$

$$T_{con} = T_{f,in,con,nominal} + 5^\circ\text{C} \quad (18)$$

2. Evaluate the evaporator heat transfer rate at nominal conditions:

$$\dot{Q}_{eva,nominal} = \dot{W}_{nominal} - \dot{Q}_{con,nominal} \quad (19)$$

3. Evaluate the evaporating pressure p_{eva} and condensing pressure p_{con} at the corresponding refrigerant temperature from the refrigerant routines.

4. With the suction temperature $T = T_{eva} + \Delta T_{sup}$ and evaporating pressure p_{eva} , evaluate the specific volume and isentropic exponent from the refrigerant routines.

5. Evaluate the volume ratio:

$$V_r = (p_{con}/p_{eva})^{1/\gamma} \quad (20)$$

6. Evaluate the nominal refrigerant volume flow rate:

$$\dot{V}_{nominal} = (\dot{m}_{ref} + \dot{m}_{leak})v_{suc} \quad (21)$$

$$\dot{m}_{ref} = -\frac{\dot{Q}_{eva,nominal}}{(h_A - h_B)} \quad (22)$$

$$\dot{m}_{leak} = 0.01\dot{m}_{ref} \quad (23)$$

7. Evaluate the leakage coefficient:

$$C = \dot{m}_{leak}/(p_{con}/p_{eva}) \quad (24)$$

8. With the theoretical power evaluated from Eq. 8 and the previously evaluated parameters, evaluate the constant part of the power losses:

$$\dot{W}_{loss} = \max(0, \eta\dot{W}_{nominal} - \dot{W}_t) \quad (24)$$

9. Evaluate the condenser and evaporator heat transfer coefficients:

$$\{UA_{con}, UA_{eva}\} = \dot{Q}_{con,nominal}/5^\circ\text{C} \quad (25)$$

This sequence has been implemented in Python and is used to choose starting values for the parameters. It was found to produce valid parameters in all cases considered.

4 Examples

4.1 Calibration from Model Produced Data

The calibration method for the heat pump model is first verified using data produced by the model. Heat pump capacities and power input were calculated using the Python model for water mass flow rates of 0.6, 0.9 and 1.2 kg/s at both the evaporator and condenser, inlet water temperatures of 0, 5, 10, 15, 20 and 25°C at the evaporator and inlet water temperatures of 15, 25, 35 and 45°C at the condenser, for a total of 216 data points. The calibration is done using only the 16 points corresponding to the combinations of minimum and maximum values of the inlet water temperatures and flow rates. The set of parameters used to evaluate the heat pump capacities and power input, the guess values for each parameter and the set of parameters resulting from the calibration are shown in Table 2. A comparison of the heat pump capacities and input power at all 216 points for the model values and calibrated values is shown in Figure 4.

Table 2. Heat pump parameters for calibration using model produced data.

Parameter	Original value	Guess value	Calibrated value
V_r (-)	2.365	1.668	2.362
$\dot{V}_{nominal}$ (m ³ /s)	0.00288	0.00193	0.00287
C (kg/s)	0.0041	0.00049	0.0041
ΔT_{sup} (°C)	6.84	4.00	6.49
η (-)	0.924	0.950	0.922
\dot{W}_{loss} (W)	396.1	2206	398.7
UA_{con} (W/°C)	7007.7	5044.9	7014.5
UA_{eva} (W/°C)	29991	5044.9	49136

The calibration process yielded parameters within 0.7% of the model value, except for the degree of superheating (5.1%) and the heat transfer coefficient of the evaporator (64%). The model appears not be very sensitive to the heat transfer coefficient of the evaporator. For instance, the sum of normalized square errors (Eq. 16) is 8.94×10^{-6} using the calibrated values and 1.344×10^{-5} when replacing only the heat transfer rate of the evaporator with the model value. The computing time for the calibration of the model was 80.5 sec.

4.2 Calibration from Manufacturer Data

The calibration method is also verified against commercial heat pump data. Technical data for a commercial water to water heat pump with 19.3 kW nominal capacity and 4.5 nominal coefficient of performance was used to calibrate the heat pump model. The technical data includes values of the capacity and

power input for mass flow rates of 0.47, 0.71 and 0.94 kg/s at both the evaporator and condenser, inlet water temperatures of -1.2, 4.5, 10.1, 15.6, 21.2 and 26.7°C at the evaporator and inlet water temperatures of 15.6, 26.7, 37.8 and 48.9°C at the condenser, for a total of 216 data points. Once again, the calibration is done using only the 16 points corresponding to the combinations of minimum and maximum values of the inlet water temperatures and flow rates. The guess values for each parameter and the set of parameters resulting from the calibration are shown in Table 3. A comparison of the heat pump capacities and input power at all 216 points for the model values and calibrated values is shown on Figure 5.

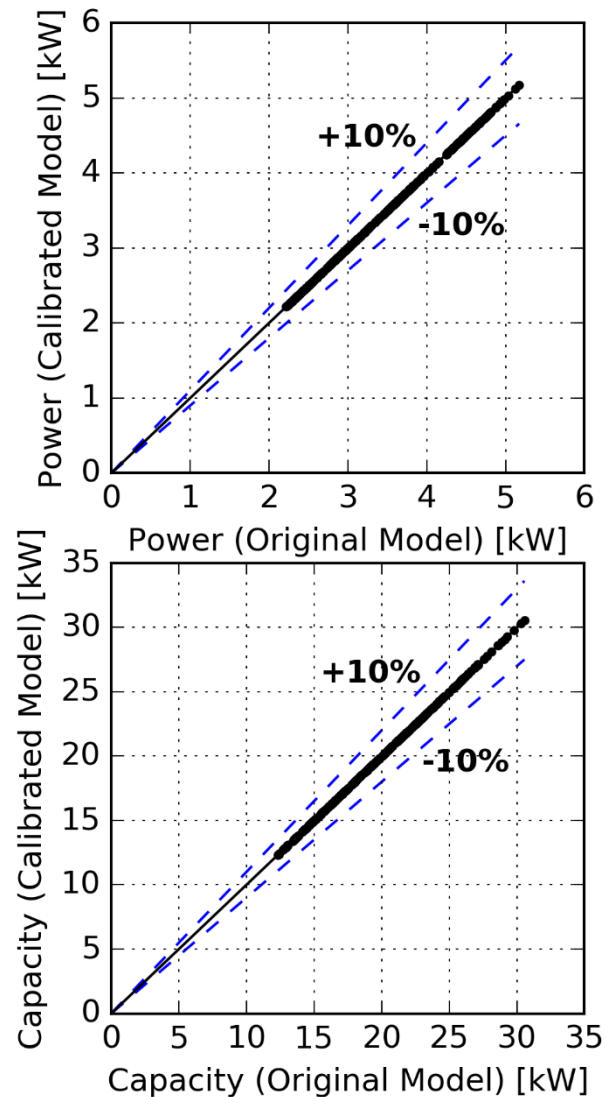


Figure 4. Comparison of model produced and calibrated model heat pump capacities and power input.

Overall, the calibrated model is in good agreement with the manufacturer data. The sum of the normalized square errors is 0.00507 and the maximum differences between calculated heat pump capacities and power input from the model and the manufacturer data are

2.7% and 4.7%, respectively. Similar results have been obtained for different technical data from different manufacturers. The computing time for the calibration of the model was 72.3 sec. A database of sets of parameters, provided via Records, for various heat pumps from different manufacturers will be included with the heat pump model.

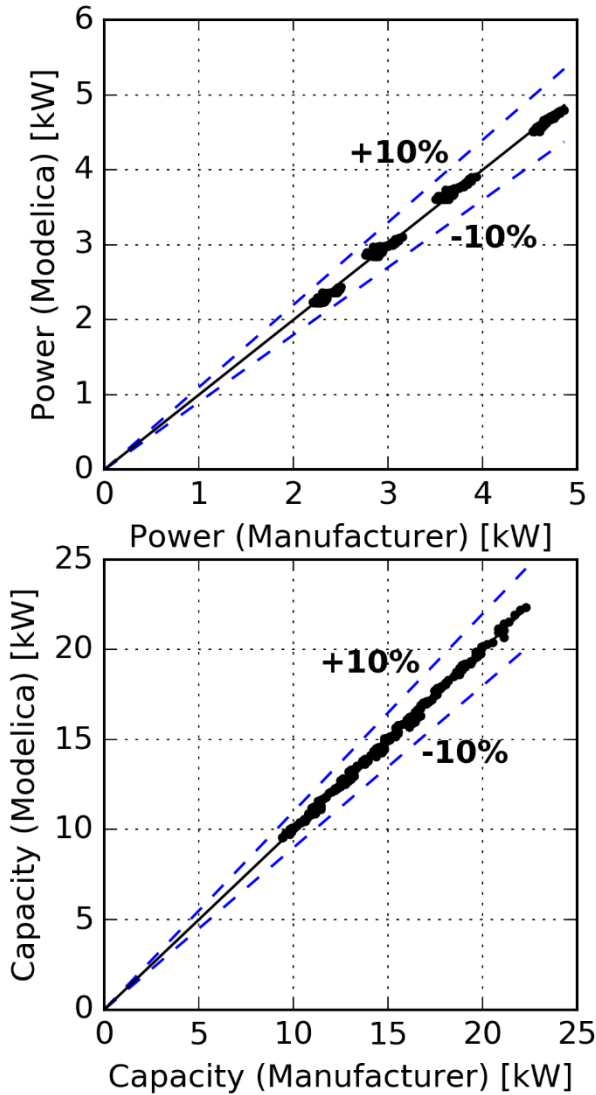


Figure 5. Comparison of manufacturer and calibrated model heat pump capacities and power input.

4.3 Hydronic Heating System

The heat pump model is integrated into a simulation model of a hydronic heating system. The system model is equivalent to the `Buildings.Examples.HydronicHeating.TwoRoomsWithStorage` system model from the `Buildings` library, with the boiler replaced by a water to water heat pump with a constant source temperature of 8°C.

The hydronic heating system consists of two rooms equipped with radiators. Hot water is produced by the heat pump, stored into a storage tank and fed to the

radiators when required. The radiators are turned on when the room temperature falls below the temperature set points of 21°C during the day and 16°C at night. The heat pump is turned on if the supply water temperature from the radiators falls below the current set point and turned off when the temperature at the bottom of the storage tank rises above 55°C. Cooling is provided by outside air if the room temperatures rise above 22°C.

Table 3. Heat pump parameters for calibration using manufacturer data.

Parameter	Guess value	Calibrated value
V_r (-)	1.436	1.975
$\dot{V}_{nominal}$ (m ³ /s)	0.001484	0.001984
C (kg/s)	0.0004947	0.002566
ΔT_{sup} (°C)	4.0	5.703
η (-)	0.95	0.8192
\dot{W}_{loss} (W)	2134	856.9
UA_{con} (W/°C)	6633.2	2840.4
UA_{eva} (W/°C)	6633.2	21523

The nominal heating power of the boiler in the original system is 2.2 kW. Therefore, the parameters to the heat pump model were the same as those presented in Table 3, with parameters $\dot{V}_{nominal}$, C , \dot{W}_{loss} , UA_{con} and UA_{eva} scaled by a factor 0.125 to obtain approximately the same heating capacity.

The simulation time for the simulation model using the heat pump is compared to the simulation time for the model using the boiler. Both simulations are done using the Radau solver, a tolerance of 1×10^{-6} and a simulation stop time of 1 week. The simulation time using the heat pump model was 48 seconds while the simulation time using the boiler was 17 seconds.

5 Conclusions

A model for a water to water heat pump with a scroll compressor is presented. To keep the computational time small and to reduce the number of evaluations of refrigerant thermodynamic properties, the model is based on a simplified vapor compression cycle with only five refrigerant states. Components for the compressor, the evaporator and condenser, as well as routines for the evaluation of thermodynamic properties of refrigerant R410A were implemented in Modelica. Parameters to the model are evaluated from manufacturer data by solving the optimization problem that minimizes the differences between the model predicted heat pump capacities and power input and those found in the manufacturer technical data.

The heat pump model was also implemented in Python to facilitate the calibration process. While it

would be possible to call the Modelica model during the optimization, a Python implementation was judged more convenient in terms of ease of use. However, it duplicates the implementation of the heat pump model, which would make it difficult to apply the same methodology to more complex systems. Support for the preprocessing of parameters using Modelica models within the Modelica framework would facilitate the use of calibrated Modelica models.

The calibrated model presented in this paper has been shown to generate heat pump capacities and power input very close to the manufacturer data, and to be able to be integrated into simulation models with minimal impact on the simulation time. Future work will be devoted to the extension of the methodology to more complex cycles, such as multi-stage cycles, and to the modeling of chillers.

Acknowledgements

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231.

References

- Carbonell, S. D., Cadafalch, R. J., Pärlich, P., and Consul, S. R. (2012). Numerical analysis of heat pumps models: comparative study between equation-fit and refrigerant cycle based models. in *Proc. Int. Conf. on Solar Heating, Cooling and Buildings*, EuroSun 2012 (Rijeka, HR).
- De Monte, F. (2002). Calculation of thermodynamic properties of R407C and R410A by the Martin-Hou equation of state — part I: theoretical development. *International Journal of Refrigeration*, 25(3): 306-313.
- Domanski, P. A., and McLinden, M. O. (1992). A simplified cycle simulation model for the performance rating of refrigerants and refrigerant mixtures. *International Journal of Refrigeration*, 15(2): 81-88.
- E. I. du Pont de Nemours and Company (2004). Thermodynamic properties of du Pont Suva 410A refrigerant. URL https://www.chemours.com/Refrigerants/en_US/assets/downloads/h64423_Suva410A_thermo_prop_si.pdf.
- Jin, H. (2002). Parameter estimation based models of water source heat pumps. Ph.D. Thesis. Oklahoma State University, Stillwater, OK, USA.
- Jones, E., Oliphant, T., and Peterson, P. (2001). Open source scientific tools for Python. URL <http://www.scipy.org>, 73, 86.
- Lee, T. S., and Lu, W. C. (2010). An evaluation of empirically-based models for predicting energy performance of vapor-compression water chillers. *Applied Energy*, 87(11): 3486-3493.
- Lemort, V., and Bertagnolio, S. (2010). A Generalized Simulation Model of Chillers and Heat Pumps to be Calibrated on Published Manufacturer's Data. In *Proceedings of the International Symposium on Refrigeration Technology 2010*, Zhuhai, China.
- Lund, H., Werner, S., Wiltshire, R., Svendsen, S., Thorsen, J. E., Hvelplund, F., and Mathiesen, B. V. (2014). 4th Generation District Heating (4GDH): Integrating smart thermal grids into future sustainable energy systems. *Energy*, 68: 1-11.
- Martin, J. J., and Hou, Y. C. (1955). Development of an equation of state for gases. *AIChE Journal*, 1(2): 142-151.
- Scarpa, M., Emmi, G., and De Carli, M. (2012). Validation of a numerical model aimed at the estimation of performance of vapor compression based heat pumps. *Energy and Buildings*, 47: 411-420.
- Swider, D. J. (2003). A comparison of empirically based steady-state models for vapor-compression liquid chillers. *Applied Thermal Engineering*, 23(5): 539-556.
- Wetter, M., Zuo, W., Nouidui, T. S., and Pang, X. (2014). Modelica Buildings library. *Journal of Building Performance Simulation*, 7(4): 253-270.

Simulation of Large Grids in OpenModelica: reflections and perspectives

Francesco Casella¹ Alberto Leva¹ Andrea Bartolini²

¹Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy,

{francesco.casella, alberto.leva}@polimi.it

²Dynamica s.r.l., Italy, andrea.bartolini@dynamica-it.com

Abstract

This paper belongs to a long-term research activity on modelling and simulation of large-size power grids in Modelica, using the OpenModelica Compiler. We describe the present state of the research, its evolution over the last year, the conclusions we could reach in this period in comparison with the initial hypotheses, and some results. Finally, we outline the future of the presented activity.

Keywords: Grid Modelling and Simulation, Large-Scale Systems, Efficient Simulation.

1 Introduction

The modelling and simulation of large power grids is an emerging domain of interest for the Modelica language, as the encountered problems basically consist of large networked systems with decentralized control, where multiple producers and consumers cooperate to the goals of stable network behaviour, satisfaction of all the load requests, and system optimality.

Although control strategies for such large-scale systems are usually designed as hierarchical systems, abstracting low-level behaviours within higher levels, it is sometimes necessary to simulate the entire system. This can be the case when a full verification of the designed strategy, including the interactions among its parts, is in order—and this is an issue shared by any large-scale system.

In the case of electric grids, there is another problem to address. For management reasons at the nation- or continent-wide scale, it is required to periodically assemble a model of the entire system and use it to run numerous simulations, to verify that the stress expected in the next time period can be sustained without incurring in stability problems, to test critical manoeuvres when required, and possibly to take decisions in a view to optimise the operation. This particular use of simulation makes a fast code generation vital.

Over the last two years, we have been working on this subject, with the goal of providing an entirely Modelica-based solution using the open-source OpenModelica Compiler (OMC) for code generation. The problem at hand is one very interesting case of an emerging class of large-scale models, see (Casella, 2015) for an

overall discussion on this topic. Preliminary results were presented in (Casella et al., 2016), which was mainly addressed to the power system community. This paper incorporates the results of additional work carried out since then, and presents the current state of the research from the perspective of the Modelica community.

2 Previous research

In this section we summarise the research context and the results from which we started, referring the interested reader to (Casella et al., 2016) for further details.

National grids in Europe are rapidly evolving (ENTSO-E, 2015, 2014). The penetration of intermittent sources like wind and solar enhances the need for continent-level integration for countries to help one another. Transmission networks are moving from the traditional structure dominated by large synchronous generators and AC links, toward an increasing share of HVDC links and of medium- and small-scale generators interfaced to the grid via AC/DC/AC links. As a consequence, the management of transmission grids by national Transmission System Operators (TSOs) increasingly requires knowledge of the dynamic behaviour of the system outside the country boundaries.

Traditionally, well-established domain-specific tools are used such as PowerFactory, PSS/E, and Eurostag. These tools come with extensive component libraries, but the exact formulation of the said models is difficult to access, since they are written in low-level languages like FORTRAN. With commercial tools, the models' source code might even be unavailable to the end user. This hinders the required interoperability, as models of the same object in different tools may behave differently. Indeed, full interoperability would ideally require all European TSOs to use the same simulation tool.

Modelica has been already used for the modelling of electrical power systems, including detailed machine models (Franke and Wiesmann, 2014; Kral and Haumer, 2005), and more recently it has been considered also to model electro-mechanical transients in high-voltage generation and transmission system. In this context, an activity worth mentioning is the iTesla European FP7 research project (Vanfretti et al., 2013, 2014; Zhang et al., 2015), although the results of the project refer to small- or

medium-sized power systems, with at most a few dozens generators and transmission lines.

At the beginning of this activity, we formulated the following research question: "Are Modelica and Modelica tools adequate to support the simulation of electro-mechanical models of national- and continental-size power grids?". From that moment till now, we have been building a prototype model library, using which many test cases have been created and analysed to answer the research question stated above. The library contains representative models for the main components used in the addressed systems, i.e., generators, governors, transformers, transmission lines, and loads. Note that the goal of this library is not the accurate modelling of any real system, but rather to build realistic models of large-scale power systems in order to test the ability of Modelica tools to handle them. The simulation code is generated with the open-source OpenModelica Compiler (OMC).

The results obtained so far are encouraging, but at the same time the activity has revealed several shortcomings of the OpenModelica environment, in particular referring to the efficiency of both the code generation and the simulation phase. A development activity was therefore carried out – and is still ongoing – within the OpenModelica Consortium to address the evidenced problems, and verify the effects of the introduced improvements with respect to some representative benchmark cases. The result of the activities just sketched is presented in the following.

3 Current research activity

For the purpose of this study, a prototype library has been built, providing models of synchronous generators, transformers, transmission lines with breakers and over-current protections, electrical loads, and governors. All the high-level modelling features of Modelica, like the support for complex numbers, were extensively used.

```
operator record ComplexVoltage = Complex(
  redeclare SI.Voltage re, redeclare SI.Voltage im);
operator record ComplexCurrent = Complex(
  redeclare SI.Current re, redeclare SI.Current im);
connector Pin
  Types.ComplexVoltage V "Line-to neutral voltage";
  flow Types.ComplexCurrent I "Line current";
end Pin;
```

Figure 1. Connector definition.

Figure 1 shows the types for complex current and voltage, used to define the electrical connector. It is assumed that the three-phase voltages and currents are always balanced and described by phasors referred to a common reference frame rotating with a reference speed/frequency, usually that of a strong generator in the network.

Under these assumptions, a three-phase voltage and current system can be described by just one voltage and one current phasor, provided the appropriate factors of 3 or $\sqrt{3}$ are taken into account when computing the actual power flows. Most large-scales grid studies are made under this assumption; extensions to unbalanced three-phase

systems are feasible, but are far more computationally demanding, and outside the scope of our study.

It is also assumed that the network frequency stays close enough to its reference value, so that the impedances can be computed with that value, and considered constant.

There are some similarities between the design of this library and that of the Modelica.Electrical.QuasiStationary library. However, the specific modelling framework which is required for large power grid studies, i.e., three-phase balanced systems represented by one equivalent phase only, is not directly available there.

```
algorithm
  // Detection of high current - side a
  when I_a_mod > I_lmax_mod then
    TimerOn_a := true;
    TimerStartValue_a := time;
  end when;
  when I_a_mod < I_lmax_mod and pre(TimerOn_a) then
    TimerOn_a := false;
  end when;
algorithm
  // Handles the actual status of the breaker - side a
  when pre(TimerOn_a) and
    time > pre(TimerStartValue_a)+I_lmax_delay then
    BreakerStatus_a := 0;
  end when;
equation
  Yl_act = Yl * Complex(BreakerStatus_a * BreakerStatus_b);
  Ysa_act = Ys * Complex(BreakerStatus_a);
  Ysb_act = Ys * Complex(BreakerStatus_b);
  Ia = Il + Isa;
  Ib = Isb;
  Isa = Ysa_act * Va;
  Isb = Ysb_act * Vb;
  Il = Yl_act * Vl;
  Va = Vl + Vb;
```

Figure 2. Model of a transmission line (excerpt).

Figure 2 shows an excerpt of transmission line model, including breakers for current protection. The two algorithms compute the state of the breaker on the one side of the line (the other is omitted for brevity), while the equations describe admittances, currents and voltages.

```
equation
  // ideal transformer:
  VTl = n * Vb;
  ITl = -Ib / n;
  // actual admittances
  Yl_act = Yl * Complex(LineBreakerClosed);
  Ys_act = Ys * Complex(LineBreakerClosed);
  // pi-model
  Ia = Il + Isa;
  Ib = ITl + IsTl;
  Isa = Ys_act * Va;
  IsTl = Ys_act * VTl;
  Il = Yl_act * Vl;
  Va = Vl + VTl;
```

Figure 3. Model of a transformer (excerpt).

The equations for the transformer model are analogous—see Figure 3, where again just an excerpt is reported for brevity.

The model of a synchronous generation unit is built hierarchically, by connecting those of synchronous machine, governor, and exciter controller, see Figure 4. The synchronous generator is described by the simplest possible 4-state model, taking the mechanical power input P_{m_req} from the governor, and the normalised excitation voltage v_f from the voltage controller.

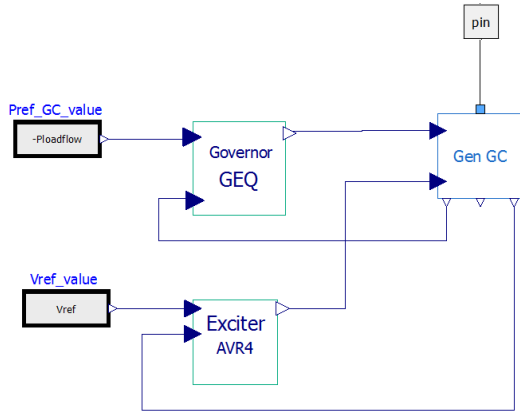


Figure 4. Generator model.

```

equation
// mechanical equations
Snom_GC_mod*Ta*der(omega)/omega = Pm_req - Pe;
der(delta) = omega - omega_ref;
// lead-lag vd
Tqo*der(ed) + ed = (Xq - X)*iq;
ed = vd - X * iq;
//lead-lag + lag vq
Tdo*der(eq) + eq = vf - (Xd - X)*id;
eq = vq + X * id;
//normalization
Vd = vd*V_ll_nom_mod;
Vq = vq*V_ll_nom_mod;
Id = id*Snom_GC_mod/V_ll_nom_mod;
Iq = iq*Snom_GC_mod/V_ll_nom_mod;
// conversion from Park ref. to pin ref.
Vpr_ll = Vd*sin(delta) + Vq*cos(delta);
Vpi_ll = -Vd*cos(delta) + Vq*sin(delta);
Ipr = Id*sin(delta) + Iq*cos(delta);
Ipi = -Id*cos(delta) + Iq*sin(delta);
// power calculation
Pe = 3 * (Vpr*Ipr + Vpi*Ipi);
Qe = 3 * (Ipr*Vpi - Vpr*Ipi);

```

Figure 5. Synchronous generator equations (excerpt).

This model is interfaced to the rest of the system through a `Pin` connector (see Figure 1). The core equations are shown in the excerpt of Figure 5.

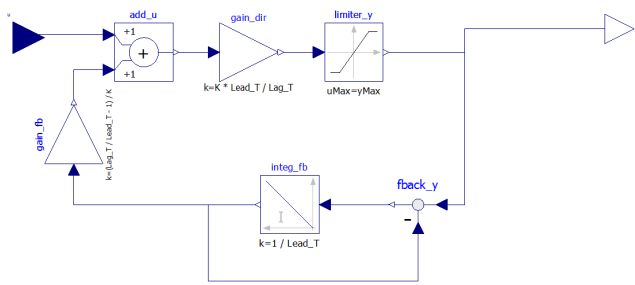


Figure 6. Excitation system model, according to IEEE Std 421.5-2005.

The governor and exciter models (see for example Figure 6) are simple block diagrams, in accordance to the IEEE standards. A graphical representation is here preferred to a text-based one, as it is immediately familiar to any practitioner in the field.

Coming to loads, both linear and nonlinear models are provided. The basic linear load model is described

by the equation $V = ZI$, where Z is a constant complex impedance. PQ models can be easily obtained by writing equations that prescribe the real and imaginary parts of the complex power flow through the `Pin` connector of the load. However, doing so makes the (large) implicit system of equations describing the network nonlinear.

Since reliable sparse nonlinear solvers were not available in the Modelica tool used for this study at the beginning of the work, a linearized PQ model was also implemented, in which the relationships between complex voltage, current and power were linearized around the nominal operating point, which is supplied by an external power flow computation. Later on, as full nonlinear sparse solvers became available both for initialization and simulation, the regular PQ load models were used.

In order to simulate network protection strategies, it is necessary to be able to simulate the dynamic formation of more than one electrical islands from a single synchronous network, due to the opening of strategically placed circuit breakers. The newly formed islands need separate frequency references and may drift apart from each other. In this case, three factors shall be considered:

1. *topological factor*, i.e., detecting the formation of sub-islands in the network, starting from the actual status of circuit breakers,
2. *functional factor*, i.e., assessing the ability of each island to survive in terms of voltage and frequency regulation,
3. *modelling factor*, i.e., finding a model structure which allows the models to properly work after the islanding event in each of the possible functional condition, avoiding singularities or other numerical problems that would cause the simulation to abort.

Up to the authors' understanding, this is a major departure from the modelling assumption and the structure of all the existing Modelica libraries for multi-phase power system modelling, which assume a fixed connection topology throughout the simulation, and exploit this property to use the over-constrained connector features originally introduced in Modelica 3.0, propagating the phase reference through the connectors. Unfortunately, this feature cannot be used for the grid models considered in this paper, unless it is extended to handle dynamically changing connection graphs; this in turn would require a change of the Modelica language, and major changes to how this feature is handled in the Modelica tool.

In this study a prototype framework to manage this aspect was implemented using Modelica 3.3. In fact, for the purposes of the testing activities carried out so far, the topological analysis was not handled with a general-purpose algorithm (that could be implemented as an external C function), but rather hard-coded in simple Modelica functions that returned the results of the analysis, which were known a-priori for those tests.

In a nutshell, the prototype framework is based on a Network Supervisor model, which is unique for the entire grid model. The supervisor:

- receives the status of the network breakers via input/output connections and monitors their changes;
- performs the topological analysis and detects the formation of islands in the grid each time the breaker status changes;
- sends to each load via input/output connections the new activation status (active/not active) and to each generator the new frequency reference (or reference generator), when the breaker opening actually leads to island formation.

Generators and loads change their active equations (using conditional equations) and frequency reference, according to the information received from the Network Supervisor, in order to avoid singularities that may prevent the simulation from continuing. For example, all PQ load models are turned into open circuits when they find themselves in a not active island, i.e., an island without generators, because otherwise the system of equations of the sub-island would have no solution, aborting the simulation.

Network	Nodes	Gens	Lines	Trafos	Equations
GRID_C	751	74	369	583	56386
GRID_E	1817	267	1458	1202	157022
GRID_D	8376	2317	1946	2489	579470
GRID_G	8113	407	6833	2824	593886

Table 1. Features of the exemplary grids.

Coming to the test cases, four exemplary grids of different sizes were considered, named in the following $\text{GRID}_{\{C, E, D, G\}}$. Table 1 summarizes the main features of the models, which describe the Irish power system, the 400 kV Italian power system, the 400 kV pan-european transmission system, and the detailed 400-220-150-132 kV transmission system, respectively. The models were supplied by CESI in the context of the study reported in (Casella et al., 2016). Note that the number of nodes, reported for convenience, is not always a reliable complexity indicator, because a node can have a very variable number of attached entities, each in turn of different complexity; for this reason, we also report the number of equations. The results obtained by simulating these models are summarised and discussed in the next section.

4 Simulation results

During the first round of activity, that took place between November 2015 and July 2016, the only fully reliable large-scale sparse solver made available by the OpenModelica tool was the KLU linear solver, which is geared

specifically towards the efficient solution of electrical circuit equations. This restricted the choice of system models to those in which the very large strong component of the causalized system equations is linear. This sub-set of equations comprises the transformer and transmission lines components (which are linear) and the load models, which can then be either constant impedances or PQ load models linearized around the nominal operating point.

The only viable integration strategy given this limitation was then to causalize the system of differential-algebraic equations, bringing it into state-space form, and then integrating it with an explicit ODE solver. At each time step, the calculation of the derivatives requires the solution of the very large strong component of the system, which is performed by the KLU sparse solver.

Steady-state initialization was also feasible by prescribing the currents at the boundaries of the synchronous generators to the values obtained by the external power-flow computations, which allows to split the initialization problem into one very large linear system (transformers + transmission lines + loads) and many small nonlinear problems (each individual synchronous generator). The availability of an external power-flow computation is also essential to set proper initial guess values on the nonlinear problems.

The models were simulated for 20 seconds, which is the typical length of transients for stability studies, using Heun's algorithm (2nd order Runge-Kutta) and a fixed time step of 20 ms. The transmission lines currents are monitored on both sides, but no breaker ever tripped.

Code generation and simulations reported here were carried out on an Intel Xeon CPU E5-2650 server with 20 virtual cores at 2.30GHz, 72 GB of RAM installed, running Linux Ubuntu 16.04 LTR 64 bit and using OMC 1.11.0-dev-59. Each simulation was carried out as a single thread, which is reasonable as multi-core systems can be exploited by running several simulation *scenarios* in parallel. The parts of the code generation process that can run independently are instead parallelised in OMC, as well as the compilation of the C code.

Network	Flattening	C gen.	Compilation	Simulation
GRID_C	24	24	13	12
GRID_E	73	67	35	44
GRID_D	334	315	123	111
GRID_G	318	303	144	186

Table 2. Performance results (times in seconds).

Performance results are summarised in Tab. 2. Notice for clarity that the third column includes both the time for structural analysis and optimisation, and that for C code generation. The fourth column is the time used by the C compiler and by the linker, while the fifth shows the total simulation time. The simulation time is almost twice as fast as real time for the smallest grid, and about 10 times slower for the largest one.

The time spent for flattening, structural analysis, C-code generation and compilation currently dominates, taking up to about 13 minutes for the largest case. This is already a feasible situation for off-line applications, in particular if one generates the simulation code once and then runs many simulations with it, by only changing the parameters in the initialization files, which can include for example the tripping times for circuit breakers, the load values, and so forth. However, such a code generation and compilation time is still definitely too long for real-time applications, with the typical turnover time of TSO operations, which is around 15 minutes. The peak recorded memory allocation was about 20 GB of RAM, which does not pose any problem on reasonably sized systems.

As to event handling, the event detection logic currently implemented in OMC uses a simple bisection algorithm to determine the exact point in time when thresholds are crossed. If a great precision is not necessary, only a few iterations would be required, whose cost will be comparable to that of carrying out a two-stage time integration step. Otherwise, it could be possible to implement a more sophisticated event detection, for example using a Newton-based algorithm.

Later on, as the sparse nonlinear solver Kinsol and the sparse DAE solver IDA became available in the OpenModelica tool, it was possible to use the full nonlinear PQ load models, as well as to employ a variable step size sparse implicit DAE solver, which turns out to be more efficient than explicit solvers as the underlying system is somewhat stiff. Note that in this case the system is not causalized and brought to state-space form; after alias reduction (and possibly index reduction, which however is not required for these specific models), the resulting DAEs are passed directly to the solver.

An example is shown in Figures 7 and 8, where three solvers are compared:

- Runge-Kutta/KLU on the grid model with linearized PQ loads,
- IDA/Kinsol/KLU on the grid model with linearized PQ loads,
- IDA/Kinsol/KLU on the grid model with nonlinear PQ loads.

The simulated transient is a 30% step reduction of the active power of one of the PQ loads (node N_152) in the smaller GRID_C model. The transients obtained with KLU and IDA/Kinsol on the linear network model match within the relative tolerance of the variable-step integrator, i.e., 10^{-6} .

Figures 7 and 8 show the frequency transient in node N_152 (load) and node N_144 (generator). The frequency peak at node N_152 is about 50.1 Hz. The blue and green (overlapped) traces refer to the PQ linearised model, integrated using the KLU and the IDA solvers respectively,

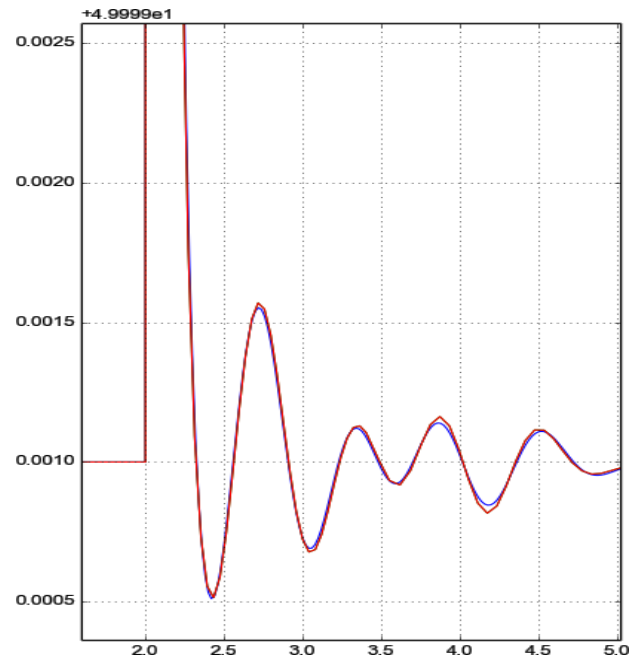


Figure 7. KLU and IDA/Kinsol test – frequency transient at load N_152.

while the red one refers to the PQ non-linear model, integrated using the IDA solver. It is apparent how the linearized model is perfectly adequate to solve this kind of transients, although it could end up being badly off in other more severe transients.

Performance results obtained with the IDA solver are reported in Table 3, using the same hardware of earlier experiments and OMC 1.12.0-dev-731. The simulation time shown is net of the time for set-up, initialisation and writing results to mass storage. Comparing these results with those of Table 2, it is apparent how this solution strategy is much more efficient, despite the additional computational complexity brought in by the nonlinear load models.

The advantage of using the variable step-size DAE solver are even more evident if longer simulation intervals are taken, as is for example the case when addressing voltage stability studies. The ability of the implicit DAE solver to take steps with a length of many seconds when the system is close to steady-state, allow to massively outperform the explicit ODE solver, whose step length is unconditionally limited to a few tens of milliseconds owing to numerical stability problems.

The times for code generation and compilation are not reported here, as these phases have not yet been optimized for this kind of solver, so that the results are not indicative of the performance that could be achieved once all current performance bottlenecks have been resolved.

Finally, the Network Supervisor prototype was tested on the same GRID_C model, using PQ linearized load models, and changing some over-current protection thresholds in such a way to result in the opening of four lines after 1 s from the simulation start. These lines opening generate three sub-islands.

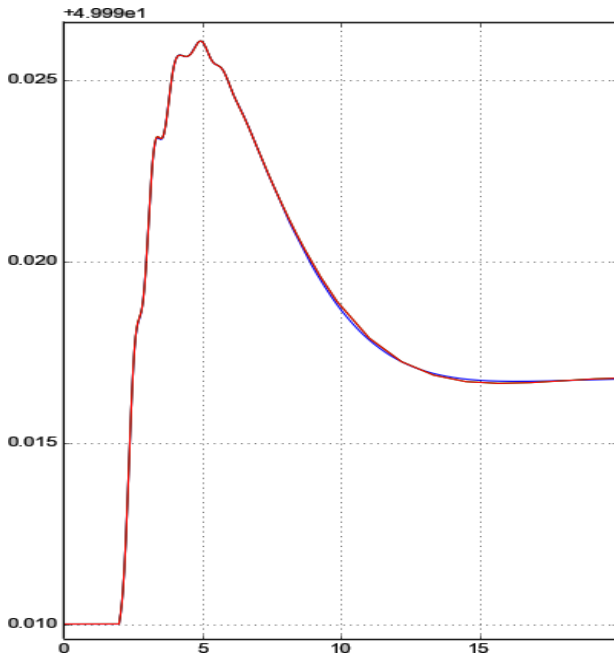


Figure 8. KLU and IDA/Kinsol test – frequency transient at generator N_144.

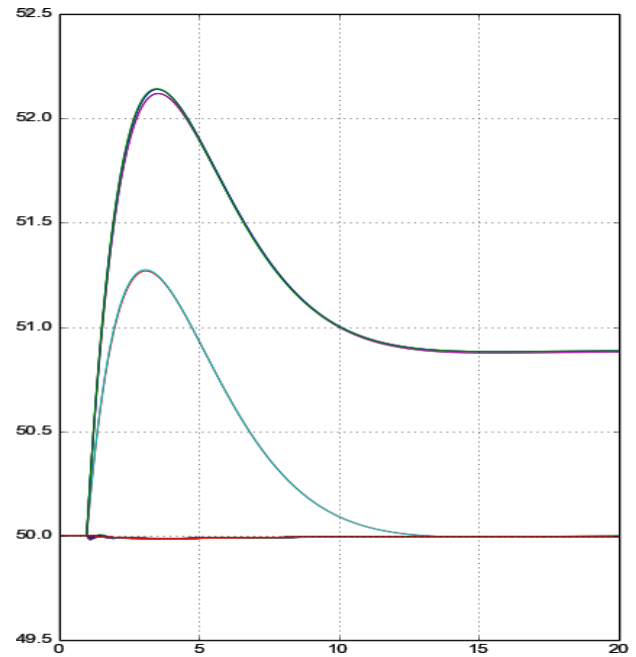


Figure 9. Network generator frequencies.

Table 3. Simulation performance with the IDA sparse DAE solver

Network	Rel. tol.	No. of steps	Sim. time [s]
GRID_C	10^{-4}	39	0.96
GRID_C	10^{-6}	146	3.18
GRID_E	10^{-4}	140	8.80
GRID_E	10^{-6}	364	15.22
GRID_G	10^{-4}	221	59.95
GRID_G	10^{-6}	615	123.19

- *Sub-island 1*, which contains only three generators, two of these in frequency regulation. The generators will be shut down, bringing their power output to zero rapidly.
- *Sub-island 2*, a small sub-island with six generators. All generators are kept in regulation and a new reference generator will be assigned (N_517).
- *Sub-island 3*, a big sub-island, which contains the rest of the network. All generators are kept in regulation and the reference generator does not change.

Figure 9 shows the new frequency rearrangement after the protection opening. Starting from the top, the first trace refers to the sub-island 2, which reaches a new steady-state with a frequency deviation of about 0.9 Hz; the second trace refers to the sub-island 1, which is shut down, and shows a transient with a frequency peak deviation of about 1.2 Hz; the last trace refers to the sub-island 3, which is the most stable due to its large dimension.

Figure 10 shows the shut-down transient in the sub-island 1 for the two generators in frequency regulation.

One can see a small power oscillation (lower than 20 kW), taking place symmetrically between the two machines.

5 Conclusions and future work

At the beginning of the activity to which this paper belongs, the research question was whether or not is it feasible to use the Modelica language and Modelica simulation tools to handle nation- and continental-wide electro-mechanical power system models. Over the last year, we reached an affirmative conclusion, though there is clearly work to be done to speed up the code generation phase, which is still too long for many application contexts.

More in detail, we could prove the feasibility of using 100% Modelica models for the simulation of transients in systems of national and continental size, albeit currently with very simple generator and controller models. The only exception is topological analysis, which will arguably be better handled by external C code, possibly re-using legacy code that performs the same task.

The simulation times we observed, particularly when using the variable step-size sparse DAE solver IDA, are acceptable, and are certainly amenable to further improvements as the implementation of that solver in OpenModelica is streamlined and optimized. On the other hand, there is still much work to do in order to reduce the time for code generation by at least one order of magnitude. Development activities are under way on the OpenModelica compiler to achieve this goal, most notably a new, much faster front-end, as well as code generation algorithms that are optimized for the sparse DAE solver. We also evidenced the need for further improvements as for the model initialisation and the event handling.

It is worth noticing that we could carry out all the re-

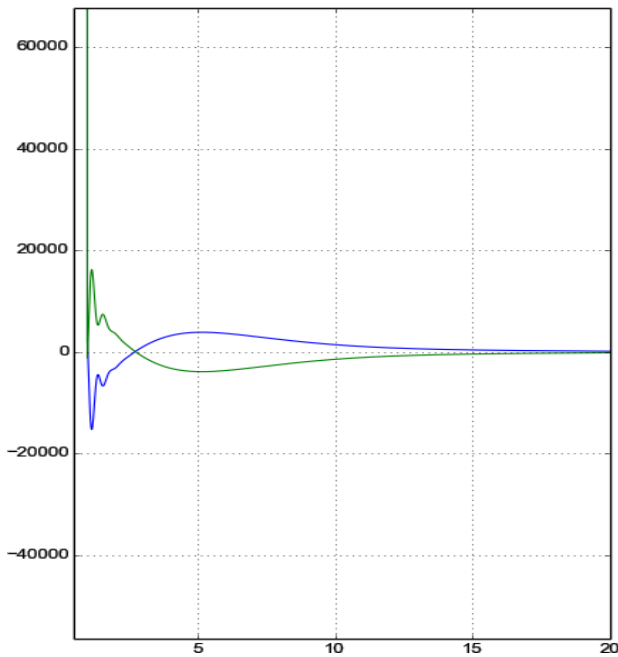


Figure 10. Active power of generators in sub-island 1.

search activity entirely within the OpenModelica framework, particularly after several improvements were made to the compiler front-end, back-end, and simulation run-time.

Given the positive outcome of this first one and a half year of ground-breaking work, the authors believe that some more specific investment in the development of the OpenModelica tool for this type of applications could lead to much better performance than what is reported in this paper. Even if the performance of domain-specific tools may not be fully reached, the added value brought in terms of flexibility and openness by the use of the Modelica object-oriented modelling framework, as well as by the use of open-source tools like OpenModelica, makes this research activity worth to be further pursued.

6 Acknowledgements

The authors gratefully acknowledge the financial support of CESI S.p.A. and RTE for supporting this research work. They also want to thank the entire development team of OpenModelica for their support and hard work, which made these developments possible.

References

F. Casella. Simulation of large-scale models in Modelica: state of the art and future perspectives. In *Proc. 11th International Modelica Conference*, pages 459–468, Versailles, France, 2015.

F. Casella, A. Bartolini, S. Pasquini, and L. Bonuglia. Object-oriented modelling and simulation of large-scale electrical power systems using Modelica: a first feasibility study. In *Proc. 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 6298–6304, Florence, Italy, 2016.

ENTSO-E. ENTSO-E policy paper: Future TSO coordination for Europe. Technical report, ENTSO-E, 2014.

ENTSO-E. ENTSO-E work programme 2015 through 2016. Technical report, ENTSO-E, 2015.

R. Franke and H. Wiesmann. Flexible modeling of electrical power systems – the Modelica PowerSystem library. In *Proc. of the 10th International Modelica Conference*, pages 515–522, Lund, Sweden, 2014.

C. Kral and A. Haumer. Modelica libraries for DC machines, three phase and polyphase machines. In *Proc. 4th International Modelica Conference*, pages 549–558, Hamburg, Germany, 2005.

L. Vanfretti, M. LI, T. Bogodorova, and P. Panciatici. Unambiguous power system modeling and simulation using Modelica tools. In *2013 IEEE Power & Energy Society General Meeting*, 2013.

L. Vanfretti, T. Bogodorova, and M. Baudette. A Modelica power system component library for model validation and parameter identification. In *Proc. 10th International Modelica Conference*, pages 1195–1203, Lund, Sweden, 2014.

M. Zhang, M. Baudette, J. Lavenius, S. Løvlund, and L. Vanfretti. Modelica implementation and software-to-software validation of power system component models commonly used by nordic TSOs for dynamic simulations. In *Proc. 56th SIMS Conference on Simulation and Modelling*, pages 105–112, Lund, Sweden, 2015.

A Tool to ease Modelica-based Dynamic Power System Simulations

Raul Viruez¹ Silvia Machado¹ Luis María Zamarreño¹ Gladys León¹ François Beade²
Sébastien Petitrenaud² Jean-Baptiste Heyberger²

¹Aplicaciones en Informática Avanzada S.L., Sant Cugat del Vallès, Spain.

{viruezr,machados,zamarrenolm,leonge}@aia.es

²Réseau de Transport d'Électricité, Paris, France.

{francois.beade,sebastien.petitrenaud,jean-baptiste.heyberger}@rte-france.com

Abstract

Developments made during the EU FP7-funded project *iTesla* towards automatic ways of transforming power systems from proprietary format to Modelica, served as a proof of concept for the adoption of Modelica as a common and standardized language for power system modelling and simulation. This work is a continuation of the progress made during the *iTesla* project. This paper presents a tool developed with the main purpose of providing users with an easy way to generate power system networks in Modelica and perform time-domain simulations. The tool is validated by generating Modelica systems for IEEE cases and comparing simulation outputs with a reference commercial tool (Eurostag).

Keywords: *Modelica, open source software, power system modelling, power system dynamics, CIM.*

1 Introduction

The work presented here is a continuation of the developments made during the *iTesla*¹ project towards the automatic transformation of power system networks for performing phasor time-domain simulations using Modelica. The specification for the automatic transformation was presented in (Vanfretti et al., 2016) and this work presents its full implementation in a user-friendly and open source tool.

One of the most challenging objectives of the *iTesla* project was to conduct accurate pan-European security assessments taking into account system dynamics in addition to static analysis. This task requires the execution of time-domain simulations considering a large number of contingencies. But one of the main obstacles for running such simulations is that each European TSO relies on its own (proprietary) data format in order to describe dynamic models.

To address this issue, *iTesla* partners agreed to use Modelica² as the standard language for power system dynamic modelling. The use of Modelica as a common language for power system dynamic simulations began with

a previous European project named PEGASE³, where simple systems were studied. During the *iTesla* project several software modules were developed to automatically transform power system models from different proprietary formats, used by TSOs, to Modelica. Also, a Modelica library containing electrical and logical models was implemented (Bogodorova et al., 2013) and validated against domain-specific simulation tools (PSS/E and Eurostag). This library called iPSL (*iTesla* Power System Library), is open source and can be found at the project repository⁴.

The developments made during the *iTesla* project allowed the successful conversion of several European power systems ranging from a dozen to hundreds of buses, and a similar number of generator machines, from PSS/E or Eurostag format to Modelica. Time-domain simulations, including contingencies, were performed on the automatically converted systems, thus achieving one of the main objectives of the project. The *iTesla* project served as a *proof of concept* for the usage of Modelica as a standard language for power system modelling, but many manual tasks were still required for each study conducted.

At the end of the project, the authors of the present work decided to further expand the *iTesla* software developments exploiting some of the current results, the expertise gained in Modelica, and the team working strength, for the development of a simulation tool based on Modelica. This user-friendly tool, from now on referred to as Power Systems on Modelica (PSM) tool, automatically generates and simulates power systems in Modelica. The tool is intended to be fully compatible with the iPSL library (iPSL has been enlarged and improved for this purpose), but the user will not be limited to the library models, since any Modelica model may also be freely added.

One of the main goals pursued with the development of such a tool, which will be released open source, is to enable users to easily convert CIM⁵ electrical networks to Modelica, in order to ease the transition to an open equation-based language for power system modelling. This paper presents the automatic model generation

¹*iTesla*: Innovative Tools for Electrical System Security within Large Areas. <http://www.itesla-project.eu/>

²Modelica® and the Modelica Association. <https://modelica.org/>

³PEGASE: Pan European Grid Advanced Simulation and State Estimation. <http://www.fp7-pegase.com/>

⁴iPSL repository. <https://github.com/itesla/ipsl>

⁵ENTSO-E Common Information Model (CIM) for grid model exchange. <https://www.entsoe.eu>

developed for the tool and shows some preliminary results.

The paper is organized as follows. Section 2 presents the PSM tool and how the automatic model generation is achieved. In section 3 specific cases are used to validate the tool. And finally, section 4 presents the conclusions and future work.

2 PSM tool

The PSM tool⁶ is intended to generate power system networks in Modelica (*.mo files) from CIM files containing static system models and state variables from a given snapshot, plus dynamic data given in XML files. A power flow computation is run over the network to obtain the steady-state of the system. A Dynamic Data Repository (DDR) is used for defining which dynamic models, either from iPSL library or user-defined, should be used to map network equipment, and which parameters must be used to instantiate those dynamic models. The users also have the possibility to perform time-domain simulations using a Modelica solver engine (either OpenModelica or Dymola).

The tool has been designed to ensure modularity, allowing users to either run processes individually or as a full workflow. The software is built as a modular Java application. Modules can be used directly from the command line or from a simple JavaFX user interface. Various power flow engines and Modelica simulation engines can be used, and further user-defined interfaces may be added later on.

The general architecture of the tool is shown in Figure 1. The different modules involved in the conversion process are: 1) CIM importer, 2) Power flow computation, 3) Modelica file generation, 4) Dynamic simulation definition, and 5) Time-domain simulation.

The tool first converts CIM data files into the iTesla Internal Data Model⁷ (IIDM), on which a power flow is computed. Then another module generates the Modelica file connecting to the Dynamic Data repository to retrieve dynamic data. The user has the possibility of introduce events to be studied in the dynamic simulation, and finally run time-domain simulations with a Modelica engine (currently OpenModelica or Dymola) selecting the desired simulation parameters.

The individual modules are described below.

2.1 CIM importer

The main goal of this module is to import a network system model file in CIM format and convert it to IIDM. The module only supports CIM-compliant files (at the time this work was prepared: ENTSO-E CIM Profile 1). The user

is free to manually write/update CIM files, as long as the generated file is CIM-compliant⁸.

2.2 Power flow computation

This module is in charge of computing the power flow over the IIDM network obtained in the previous step. The tool has been designed to work with two different alternatives for power flow computation:

- HELMTM-Flow power flow engine⁹.
- HADES power flow engine¹⁰.

The tool is intended to allow easy switching between these two engines. The values obtained from the power flow computation are re-injected into the IIDM network before moving on to the next module, i.e. the Modelica file generation. The user also has the possibility of deactivating the power flow computation and generating the Modelica file using input values from CIM.

2.3 Modelica file generation

This module is responsible for the generation of the Modelica (.mo) file. The tool enables the user to specify dynamic models (relying on the iPSL library and/or user-defined libraries) and parameters for each static item in the given network setup, as well as to provide default parameters for each dynamic model. Default dynamic models can also be defined for each static object type.

As depicted in the general architecture of the tool (Figure 1), this module connects to the DDR populated with the system dynamic data and all necessary models.

The DDR is based on a set of XML files that store:

- Which mappings from static network elements to dynamic models to use in the dynamic system model building (and its connections).
- The definitions required for building full model initialization simulations of complex dynamic models.
- The parameter sets used for dynamic model instantiations. Global declarations and system wide equations.

Full model initializations (see the blue box at the bottom of the *Modelica file generation* module in Figure 1) refer to the derivation of relevant and coherent initial values for all model variables based on power flow outputs and external parameters. This is done by performing very short-time simulations of the specific models using either OpenModelica or Dymola. This initialization is performed

⁶PSM tool will be released open source in a specific repository starting June 2017. The exact link will be published in the iPSL repository <https://github.com/itesla/ipsl>.

⁷The IIDM allows to import, export and edit power system models in the iTesla platform. Specifications on IIDM format can be found at the GitHub repository for iTesla Power System Tools. <https://github.com/itesla/ipst>

⁸Although CIM format can be used for dynamic models exchange, it is currently hardly used for this purpose in the power system community. Therefore the DDR option was considered more appropriate for PSM.

⁹For more information on HELMTM-Flow, see <http://elequant.com/>. This product is based on the Holomorphic Embedding Load Flow Method (Trias, 2012).

¹⁰RTE official power flow engine.

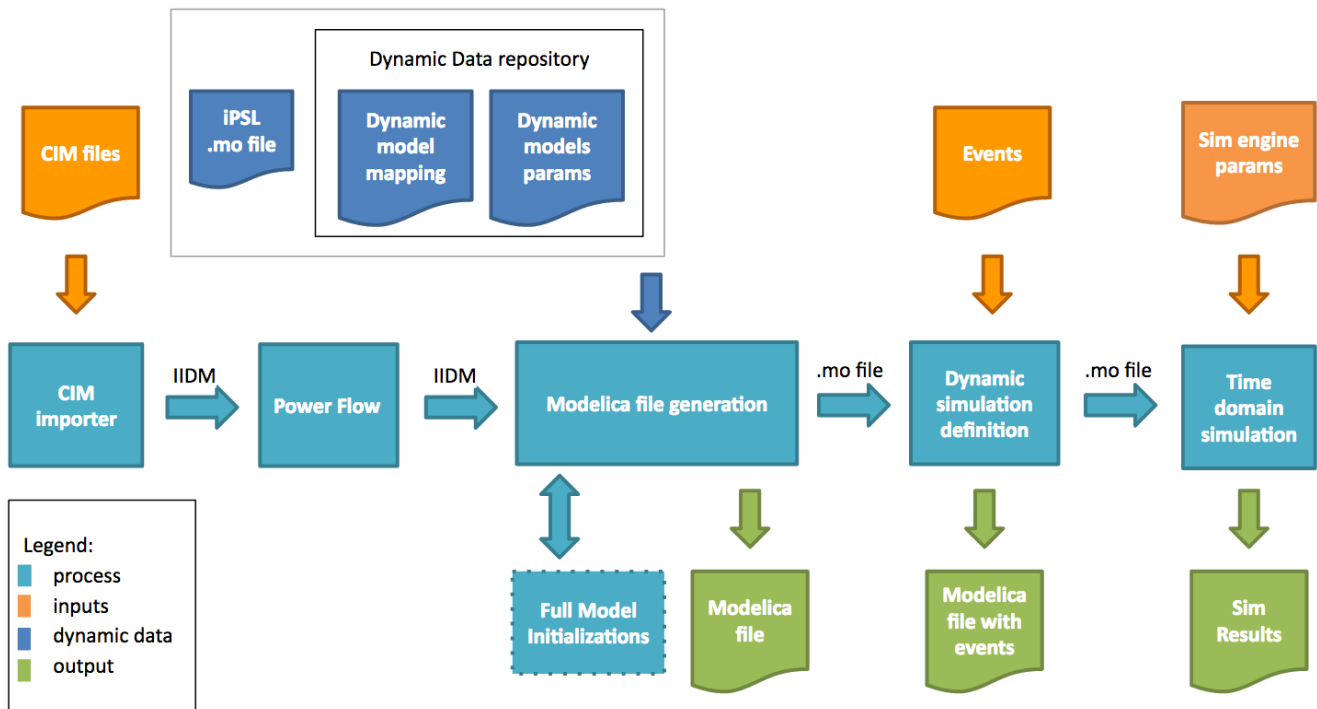


Figure 1. General Architecture of the PSM tool

only for component models requiring explicit initialization, which is the case of some generator machines and may be the case for specific control models. The main advantage of using *full model initialization* instead of *initial equation* inside Modelica models (the common initialization strategy in Modelica) is that the use of external (local) initializations can significantly increase simulation speed since it reduces the size of the initial system to be solved (especially for large systems) and it is designed to be very scalable. The use of *initial equation* may cause boundary problems and increase the number of equations, resulting in a decreased performance. See (Vanfretti et al., 2016) for more details on initializations schemes.

2.4 Dynamic simulation definition

This module is intended to define the simulation scenario. The user can define events to be triggered at specific times during the simulation. This module also allows defining load variations, capacitor changes, etc. As a result, one or several Modelica files are generated in addition to the Modelica base case generated in the previous step.

2.5 Time-domain simulation

This module is in charge of running time-domain simulations on the Modelica files generated with the tool, allowing the user to choose between two Modelica simulation engines: OpenModelica or Dymola. PSM fully relies on these engines to run simulations, and thus one of them must be previously installed before the user starts running the PSM tool.

OpenModelica is an open source Modelica environ-

ment, freely downloadable¹¹. Instead, Dymola is a commercial environment¹². The user may also directly open the generated Modelica file in any preferred Modelica environment for edition and simulation.

In PSM, the user has access to the definition of standard simulation setup (simulation interval, output interval, integration method, integration tolerance, fixed integrator step, etc.) given by the available simulation solver: OpenModelica or Dymola. The *Time-domain simulation* module will generate output files in the standard Modelica output format (MATLAB binary format *.mat), which is provided by both supported simulation engines, as well as in *.csv format. This will allow the user to import simulation results into the preferred Modelica environment or other software for plotting and analysis of results.

2.6 User interface

The tool also includes a basic graphical user interface with functionalities allowing the user to select which processes to run, on which data sets, and displaying progress and logs, without having to manually run command line actions.

3 Tool's validation

Different power system networks are being used to test and validate PSM. Some of these systems were already used for testing developments made during the iTesla project and the results are presented in (Vanfretti et al., 2016) and (León et al., 2015). In this work, validation was

¹¹OpenModelica. <https://openmodelica.org/>

¹²Dymola by Dassault Systèmes. <http://www.3ds.com>

also performed using IEEE¹³ cases (IEEE14, IEEE30, IEEE57, and IEEE118), which have not yet been used before.

All IEEE cases have been generated in Modelica format using PSM, and for this all necessary dynamic models from Eurostag were included in iPSL and DDR files were generated with the corresponding dynamic data. Eurostag equivalent models were created in Modelica using the exact same equations used by this proprietary software. This section presents the results obtained with the IEEE57 and IEEE118 cases.

The results obtained with the different generated Modelica systems are compared against the exact same networks built in Eurostag, taking this software as a reference. Validations are done both graphically and numerically. The numerical assessment is carried out using the Root Mean Square Error (RMSE) as done in previous works (Vanfretti et al., 2016). The assessment metric chosen to accept the results as valid is that the absolute RMSE is $\leq 10^{-03}$ for all compared values (voltage magnitudes in p.u. and angles in radians in all buses).

3.1 IEEE57 case

The IEEE57 test case is composed of 57 buses, 7 generators, 63 lines, 17 transformers, 42 loads, and 3 compensation banks. The generators are modelled in Modelica as synchronous machines defined by external parameters. This Eurostag's model, together with the machine described using internal parameters, were developed and included in iPSL during the iTesla project. The control systems present in this IEEE case are a voltage controller ExcSEXS, an stabilizer PSS13E2B, and a governor GovSteam0¹⁴. These models were developed in Modelica and Eurostag specifically for PSM, and have been included in iPSL¹⁵. Two types of events were introduced to validate the results of time-domain simulation performed with the Modelica model, using Dymola, against Eurostag.

Figure 2 shows the voltage magnitude at a given specific bus when a bus fault event is introduced (at this bus) at time $t=1$ seconds, lasting 0.2 seconds. As can be seen in the figure, the curves obtained with the Modelica system generated with PSM and simulated using Dymola (blue dashed line), and the Eurostag system (red solid line) match very well. The numerical assessment test was carried out obtaining a RMSE for all voltage magnitudes and angles below the defined threshold (10^{-03}).

Figure 3 shows the response of the IEEE57 test case to a line fault lasting 0.1 seconds occurring at time $t=1$ seconds, where again, the results obtained with Modelica system using Dymola (blue dashed line) and Eurostag (red solid line) are very similar. The RMSE obtained is below

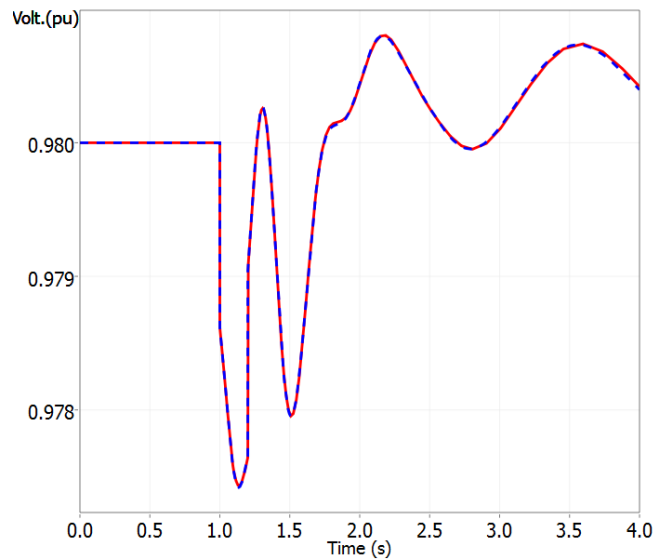


Figure 2. Simulation of the IEEE57 case with a bus fault of 0.2 seconds occurring at $t=1$ s. The blue dashed line corresponds to Dymola simulation results and the red solid line to Eurostag results.

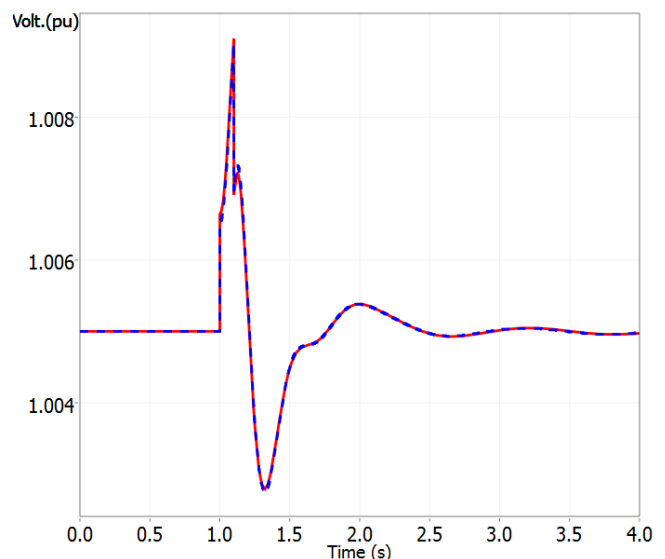


Figure 3. General Simulation of the IEEE57 case with an open line of 0.1 seconds occurring at $t=1$ s. The blue dashed line corresponds to Dymola simulation results and the red solid line to Eurostag results.

the threshold.

All simulations in Dymola were performed using the DASSL integration method with a tolerance of 10^{-06} . Typical times for running the type of simulations shown for the IEEE57 case are ~ 300 seconds to simulate 4 actual seconds. These simulations were run in a machine with the following characteristics: Intel® Xeon® CPU E5-2690 2.60 GHz (2 processors), 48 GB of installed memory (RAM) and 64-bit Operating System. Eurostag simulation for the same system takes only a few seconds to complete.

¹³Power Systems Test Case Archive from the University of Washington. <https://www2.ee.washington.edu/research/pstca/>

¹⁴Control systems were built following standard IEC 61970-302.

¹⁵All these Modelica models can be found at the iPSL repository. <https://github.com/itesla/ips1>

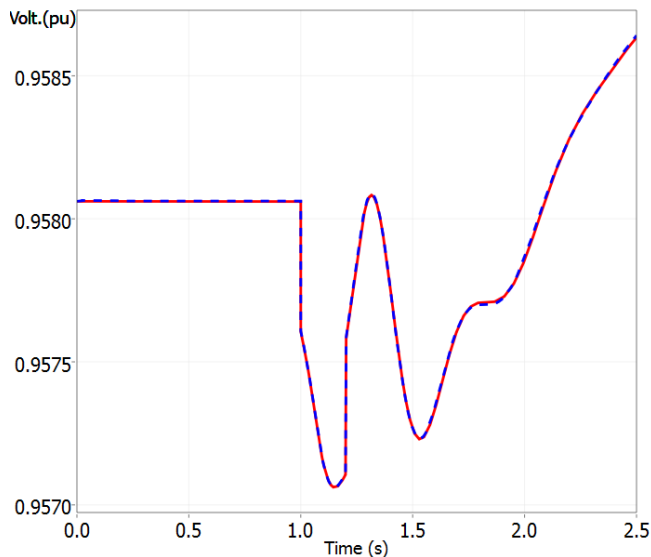


Figure 4. General Simulation of the IEEE118 case with a bus fault of 0.2 seconds occurring at $t=1$ s. The blue dashed line corresponds to Dymola simulation results and the red solid line to Eurostag results.

3.2 IEEE118 case

The IEEE118 case was also used to test PSM. This system is composed of 118 buses, 177 lines, 9 transformers, 54 generators, 91 loads, and 14 compensation banks. The dynamic models are the same as those used in the IEEE57 with CMCONST governor added.

Figure 4 shows the comparison between the Modelica generated file and Eurostag for the voltage magnitude at a specific bus, when a bus fault is introduced at time $t=1$ sec, lasting 0.2 seconds.

Due to the size of the IEEE118 network, the number of equations in Modelica ($\sim 14,000$ equations) increases considerably compared to the IEEE57 case ($\sim 2,000$ equations), and this greatly increases the simulation time in any Modelica engine. For the IEEE118 results shown, it takes Dymola approximately 6 hours to run 2.5 actual seconds (on the same machine used for the IEEE57 case). This is clearly a limitation for achieving scalability to real networks, which are of the order $\sim 10,000$ nodes. Further work is needed in this direction¹⁶.

4 Conclusions and future work

This paper presents the implementation of a user-friendly and open source tool that allows users to automatically generate and simulate power system networks in Modelica. This work is the result of years of development made during the iTesla project and the extension of those developments in a post-iTesla collaboration work. This collaboration motivated by the authors' commitment

to open simulation tools to improve cooperation and exchange for performing dynamic power system simulations.

Modelica allows enhanced transparency and results in power system modelling and simulation, with all the necessary ingredients to become a standard language for power system modelling. However, there is a clear need for easier ways to transform power systems networks into the Modelica language. The proposed tool is designed to provide such possibility and bridge the gap between Modelica and power system simulations, in order to ease widespread adoption among the power system community.

For the development of this tool, the authors have proposed a simple architecture that, given the correct inputs, allows the user to generate a Modelica file and simulate it using a Modelica solver engine. The tool has been validated with a variety of power system networks, comparing simulation results with a reference commercial software (Eurostag). In particular, results obtained with two IEEE cases are presented here, showing very accurate results.

Nevertheless, further work on Modelica simulation engines is needed in order to achieve scalability. The networks studied are below or around a hundred nodes, but simulation time increases excessively with systems size and rapidly becomes unmanageable above 100 nodes.

PSM will be released as open source in a specific repository starting June 2017. The exact link will be published in the iPSL repository: <https://github.com/itesla/ipsl>.

References

- T. Bogodorova, M. Sabate, G. León, L. Vanfretti, M. Halat, J-B. Heyberger, and P. Panciatici. A Modelica power system library for phasor time-domain simulation. *Smart Grid Technologies Europe (ISGT EUROPE)*, 2013 4th IEEE/PES, pages 1–5, 2013. doi:10.1109/ISGTEurope.2013.6695422.
- G. León, M. Halat, M. Sabate, J-B. Heyberger, F.J. Gomez, and L. Vanfretti. Aspects of power system modeling, initialization and simulation using the Modelica language. *IEEE PES Innovative Smart Grid Technologies Europe*, pages 1–6, 2015. doi:10.1109/PTC.2015.7232504.
- A. Trias. The Holomorphic Embedding Load Flow method. *Power and Energy Society General Meeting, 2012 IEEE*, pages 1–8, 2012. doi:10.1109/PESGM.2012.6344759.
- L. Vanfretti, A. Adib Murad, F. Gómez, G. León, S. Machado, J-B. Heyberger, and S. Petitrenaud. Towards Automated Power System Model Transformation for Multi-TSO Phasor Time Domain Simulations using Modelica. *PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 2016 IEEE, pages 1–7, 2016. doi:10.1109/ISGTEurope.2016.7856341.

¹⁶OpenModelica developers are working on the integration of the Sundials IDA solver in order to significantly improve simulation times for large DAE systems. For more information see <https://www.openmodelica.org/>.

A Modelica VSC-HVDC Average Value Model Implementation and its Software-to-Software Validation using an EMT Power System Domain Specific Simulator

Mohammed Ahsan Adib Murad¹ Luigi Vanfretti²

¹School of Electrical Engineering, University College Dublin, Ireland, mohammed.murad@ucdconnect.ie

²School of Electrical Engineering, KTH Royal Institute of Technology, Sweden, luigiv@kth.se

Abstract

This paper reports the implementation of a three-phase VSC-HVDC model using the Modelica language. The model is suitable for power system simulation where the power electronic circuitry can be represented using equivalent voltage and current sources to model the high frequency switching process. Differently from the authors previous work, this model is built using as much components as possible from the MSL (Modelica Standard Library) to represent the three-phase electrical circuit, while implementing the *de facto* control system models used within typical power system simulation tools. To show the applicability of Modelica for modeling a VSC-HVDC, a software-to-software validation is performed using the EMTP-RV power system simulator.

Keywords: VSC, HVDC, power systems, software-to-software validation, power electronics, electro-magnetic transients, DC grids, power systems

1 Introduction

1.1 Motivation

High Voltage Direct Current (HVDC) transmission systems have received renewed attention in the last decade due to their applications for long distance power transmission, particularly to enable interconnections between distant wind farms and the main electrical grid (Bahrman, 2006). There are two main converter technologies used in HVDCs: Line-Commutated Converter (LCC) and Voltage Source Converter (VSC), which are used for different applications in power systems (Abildgaard and Molinas, 2012). VSC-based HVDC systems provide certain advantages w.r.t. those based on LCC, including (Reed et al., 2003; Flourentzou et al., 2009), including independent control of active and reactive power, energy supply to weak and passive grids, etc.

An overview of different VSC topologies are reported in (Andersen et al., 2002) and include conventional two-level, multi-level diode-clamped, floating capacitor multi-level converters, etc.

Recently, the Modular Multilevel Converter (MMC) technology has been adopted because of its advantages compared to other multilevel converter topologies for HVDC

applications. With the adoption of MMC-based VSC technologies, modeling and simulation is becoming of crucial importance for different network studies; where modeling and simulation tools are needed in all facets related to their utilization: from design, through implementation, and in their operation.

1.2 Related Works

Power system electro-mechanical dynamic modeling and simulation is used for the analysis of dynamic behavior of large power networks, and the use of Modelica is now becoming attractive because of several advantages offered by the Modelica language as compared to existing power system simulation tools (Vanfretti et al., 2016; Casella et al., 2016). Another modeling and simulation approach that is important for power system analysis is the Electro-Magnetic Transient (EMT) methodology, and previous work has shown the advantages and limitations of the use of Modelica (Bachmann and Wiesmann, 2000) in adopting the EMT approach.

EMT analysis tools, such as EMTP-RV (see <http://emtp-software.com/>), are typically used for the analysis of VSC-HVDC systems (Peralta et al., 2012), which allow to analyze their performance for different levels of modeling granularity of the power electronics of these systems (including average value models). To the knowledge of the authors, there only exists two previous implementations of VSC-HVDC models using the Modelica language in the literature (Majumder et al., 2013; Olenmark et al., 2015), however, these have not been implemented-in nor validated-against EMT (Electro-Magnetic Transient)-type power system simulation tools (e.g. EMTP-RV), and more importantly, they are not publicly available.

1.3 Paper Contributions

This paper reports the implementation of a three-phase VSC-HVDC average value model, and a power system test model that is compared against EMTP-RV. The aim is to show the potential use and challenges of applying the Modelica language for EMT-type analysis of VSC-HVDC networks when detailed switching circuits do not need to be represented (e.g. system-level control design purposes).

The remainder of this paper is organized as follows. Section 2 gives a brief description of the VSC-HVDC model. In Section 3, the model implementation in Modelica is explained, while in Section 4, software-to-software validation results are summarized. Finally, in Section 5, conclusions are drawn and future work is outlined.

2 VSC-HVDC Model

In EMTP-RV two types of VSC-based MMC station models are available, which are based on the results by (Peralta et al., 2012): Monopole, and Bipole configuration with ground return. The MMC stations are represented using four kinds of models: (a) Full detailed model, (b) Detailed equivalent model, (c) Switching function of arm model, and (d) Average-value model (AVM). The three-phase configuration of the MMC topology assumed by these models is shown in Figure 1.

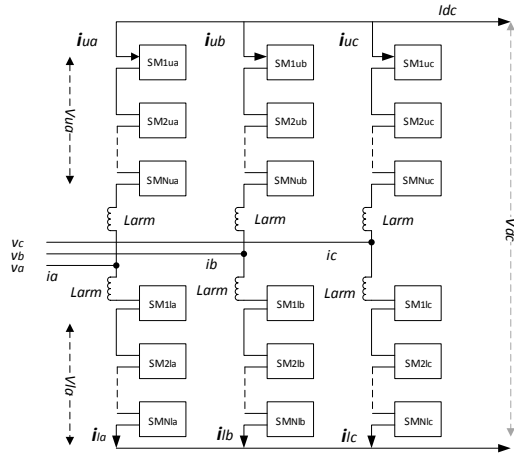


Figure 1. MMC topology.

In this work the AVM model with an high level control system is implemented. The full description of the model is documented in (Peralta et al., 2012). In this Section, the most relevant components of the model available in EMTP-RV are reviewed, as they are replicated in the Modelica implementation, in Section 3.

2.1 Average-Value Model (AVM)

In an AVM, the power electronic switches (IGBTs) and diodes are not modeled in detail, instead the MMC behavior is represented using controlled voltage and current sources. Thus, an ideal behavior of the internal variables of the MMC is assumed. For each phase $j = a, b, c$; the voltage of the converter is derived from Figure 1, from where,

$$v_{convj} = \frac{L_{arm}}{2} \frac{di_j}{dt} - v_j. \quad (1)$$

Assuming the total number of sub-modules in each phase is constant,

$$v_{uj} + v_{lj} = V_{dc} \quad (2)$$

where, v_{uj} and v_{lj} are upper and lower arm voltages. Using (1) and (2) the MMC is represented as a classical VSC. The controlled voltage source in the AC side is determined by:

$$v_{convj} = v_{refj} \frac{V_{dc}}{2} \quad (3)$$

where, v_{refj} are the reference voltages generated from the inner controller of the high level control system (i.e. they are dimensionless quantities in per unit). Based on the principle of power balance, the DC side model equations are derived assuming no energy is stored inside the MMC converter, as follows

$$V_{dc} I_{dc} = \sum_{j=a,b,c} v_{convj} i_j \quad (4)$$

where the DC side current is given by,

$$I_{dc} = \frac{1}{2} \sum_{j=a,b,c} v_{refj} i_j. \quad (5)$$

Using these principles, the AVM model implementation in EMTP-RV allows to build up an entire VSC-HVDC model. Figures 2 and 3 show the schematic of the implementation of the two basic components as available in EMTP-RV for this purpose.

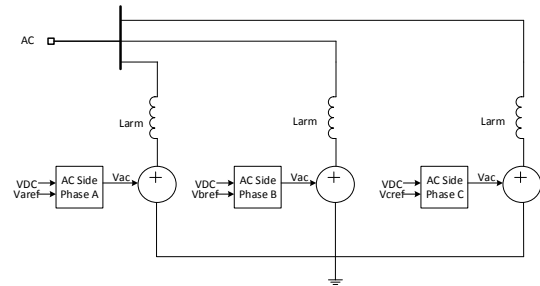


Figure 2. AC side of AVM model.

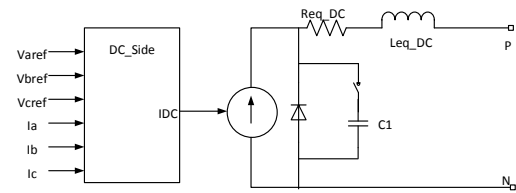


Figure 3. DC side of AVM Model.

In Figure 2 the AC side voltage for each phase is calculated using (3) and L_{arm} is the arm inductance. In Figure 3 the DC side current IDC is calculated using (5) and equivalent inductance, total conduction loss and the equivalent capacitor are given by, $L_{eq_DC} = (2/3)L_{arm}$, $R_{eq_DC} = (2/3)NR_{ON}$ and $C1 = 6C/N$; where N is the number of sub-modules per arm, C is calculated using the energy conservation principle, and R_{ON} represents the conduction loss of each IGBT.

2.2 Control System

The VSC type MMC topology uses an “upper level” control system, which includes an outer and inner control. The structure of the overall “upper level” control system is shown in Figure 4. The “upper level” control system serves two main purposes: (i) to regulate “system variables”, i.e. the active and/or reactive power or voltages (labeled “outer control” in Figure 4), and (ii) to generate reference voltages (v_{d_ref} and v_{q_ref}), which are used as input to the AVM.

2.2.1 Upper Level Control

The VSC-MMC model uses the classical vector control strategy. The inputs to the upper level control are three-phase per unit (p.u.) variables, using the matrix in (8), these variables are converted to direct-and-quadrature-axis components rotating at synchronous speed ($\frac{d\theta}{dt}$). The phase angle θ is calculated using a Phase-Locked Loop (PLL). The blocks for Clarke transformation, P/Q/VAC calculations and d-q transformation are used to compute the variables required for the outer and inner controllers. The d-q transformed voltage and currents are calculated using the transformation matrix, T , as follows:

$$i_{dq} = T i_{abc} \quad (6)$$

$$v_{dq} = T v_{abc_grid} \quad (7)$$

where

$$T = \frac{2}{3} \begin{bmatrix} \cos(\omega t) & \cos(\omega t - \frac{2\pi}{3}) & \cos(\omega t + \frac{2\pi}{3}) \\ -\sin(\omega t) & -\sin(\omega t - \frac{2\pi}{3}) & -\sin(\omega t + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}. \quad (8)$$

The AC grid voltage, active and reactive power are calculated from the d-q reference,

$$P = v_d i_d + v_q i_q \quad (9)$$

$$Q = -v_d i_q + v_q i_d \quad (10)$$

$$v_{grid} = \sqrt{v_d^2 + v_q^2} \quad (11)$$

The signals are converted to per unit (p.u.) quantities before entering to the upper level control system. The outer and inner control block is used to control active power, reactive power, DC and AC voltage. All these controllers are realized using proportional and integral (PI) control loop. The input to these PI controller loops are the difference between the reference (set by the user) and the controlled variable. The references to the outer control loop are usually fixed set points, that in practice are varied by a remote dispatcher. In this model the references to the outer control loop are fixed and can be varied by the user. The final block (d-q to abc) is used to convert the d-q reference to three-phase voltage references.

3 VSC Model Implementation in Modelica

All the components included in the VSC model available in EMTP-RV are implemented in Modelica and described

next. In addition to the VSC model, an equivalent generator model and a two winding transformer three-phase models were also implemented for software validation purposes.

3.1 AVM Model in Modelica

The AVM model was implemented using component models from the MSL (Modelica Standard Library). The connector models used in the AC side are the three-phase `plug`, and in DC side is the single phase positive and negative `pin`. The AVM model in Modelica is shown in Figure 5.

3.2 Upper Level Control in Modelica

Next, all the blocks of the upper level control system shown in Figure 4 were implemented in Modelica. As all the controllers in the upper level control system use the same PI controller implementation, first a PI controller using components from the MSL was implemented. Next the Modelica implementation was compared to the one implemented in EMTP-RV. After validating this component against EMTP-RV, the same PI controller was used in the remaining P, Q, VDC, VAC, inner control and PLL blocks.

3.3 PLL in Modelica

The phase locked loop (PLL) implemented in Modelica is shown in Figure 6. The main function of the PLL loop is to synchronize with the phase angle and frequency of the AC grid voltage. The implementation of the PLL used similar components in Modelica, as those available in the specific power system tool's documentation/description (i.e. EMTP-RV). Given the fact that the reference documentation has a copyright, so the detail description is not given here.

4 Software-to-Software Validation

4.1 Sub-system Model Validation

The AVM and each block of the upper level control system were implemented as individual models within one package, then all the blocks were assembled to realize complete the implementation of the VSC model.

Next, software-to-software validation was carried out against the EMTP-RV model. For example, consider the PLL block shown in Figure 6, it has two inputs and two outputs. After the implementation of the entire PLL block in Modelica, this model was validated by simulating it using the same input signals in both Modelica and EMTP-RV. The results of the PLL block simulations are shown in Figures 7 and 8.

The same procedure is followed for the three-phase equivalent generator and three-phase two winding transformer models shown in Figure 9. After implementing needed components, a test power system model described next is used to validate the VSC-HVDC model.

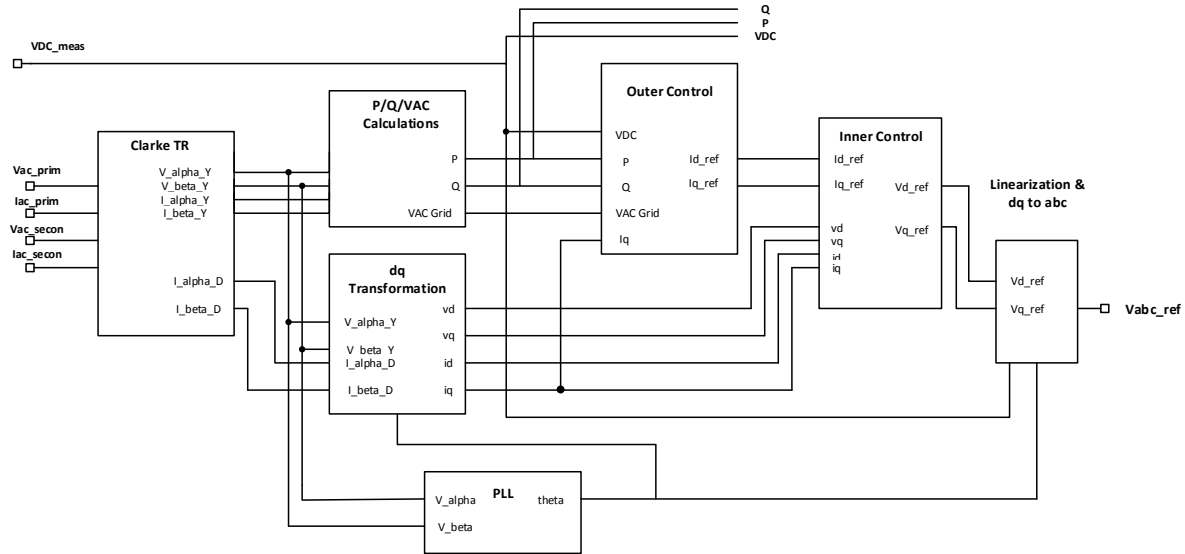


Figure 4. Block diagram of the control system of the VSC-HVDC.

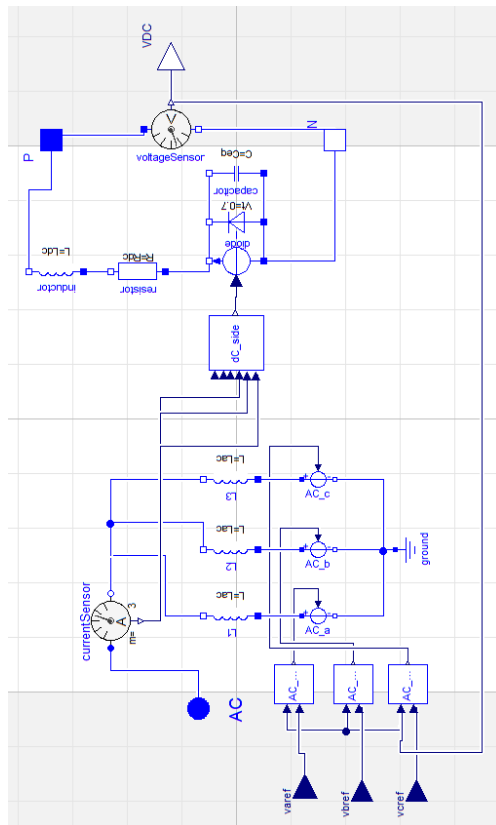


Figure 5. AVM Model in Modelica.

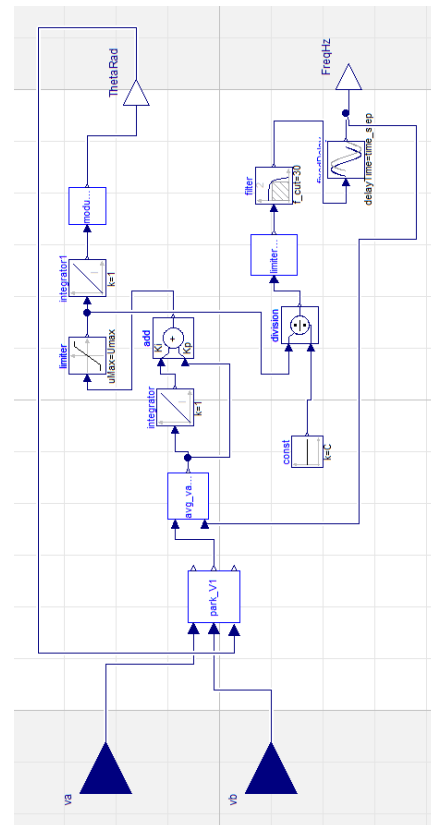


Figure 6. PLL in Modelica.

4.2 Power System Test model

The VSC-HVDC test power system model implemented in Modelica and EMT-PV is shown in Figure 10. A DC cable model is yet to be implemented in Modelica, and thus, resistive line model ($R = 1.022\Omega$) is used instead. Converter 1 (VSC1) controls the active power and Con-

verter 2 (VSC2) controls the DC voltage, while 1000 MW active power is transferred from VSC1 to VSC2. The user can select which controller should be active in each VSC. The model parameters used (i.e. transformer resistance and reactance, MMC arm inductance, etc), are exactly the same in both software tools, and are summarized in

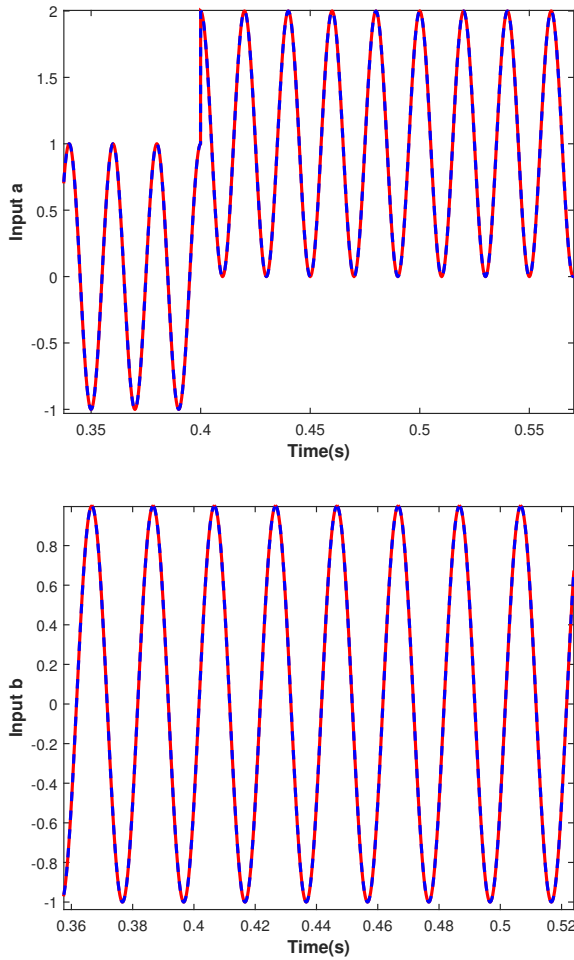


Figure 7. Inputs to the PLL block. (Red traces: EMTP-RV and blue traces: Modelica)

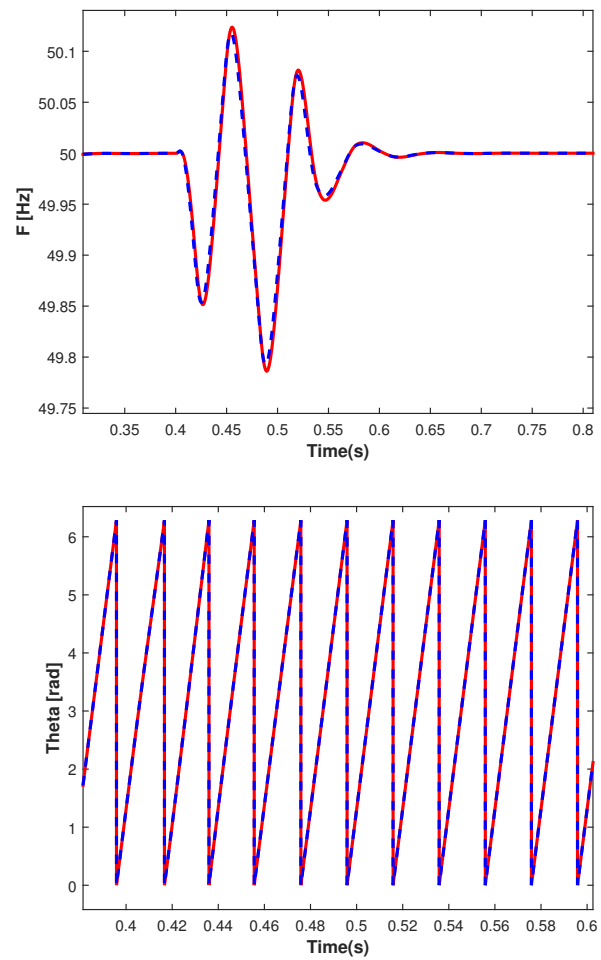


Figure 8. Outputs of the PLL block. (Red traces: EMTP-RV and blue traces: Modelica)

the Appendix. In EMTP-RV the integral and proportional gains of each PI controller (in upper level control system) are automatically calculated by specifying the desired settling time (with 5% error). The computation method used by EMTP-RV is proprietary, and thus, for the sake of consistency, the values computed by this tool are used in the Modelica model.

4.3 Steady State

EMTP-RV initializes the model variables using a three-phase power flow solver, which is not available outside of this tool. To validate the sub-system models (i.e. equivalent generator, controllers, PLL, etc.), no initialization values were provided to the Modelica models (starting values of the voltage and currents were set = 0). At the beginning of the simulation, the Modelica and EMTP-RV results do not match, however, after the Modelica trajectories reach in their steady-state, the simulation results show an adequate match. To illustrate, consider the test system shown in Figure 9, where an equivalent generator and three-phase two winding transformer are included. No initialization values for the inductor currents (i.e. the states) were pro-

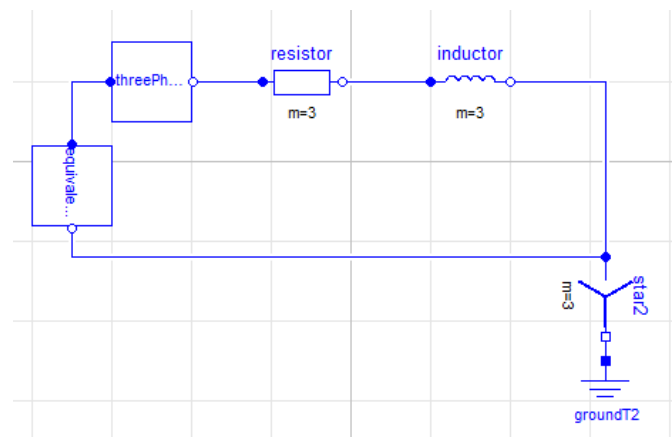


Figure 9. Test system of the equivalent generator and the two winding transformer.

vided.

The simulation was carried out using the solver `Dassl` with interval length equal to $1e-5$. The same interval length is used in EMTP-RV, however note that EMTP-RV uses a Trapezoidal solver.

The simulation results shown in Figure 11 show that at

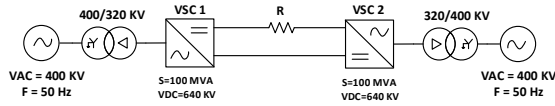


Figure 10. VSC-HVDC Test system.

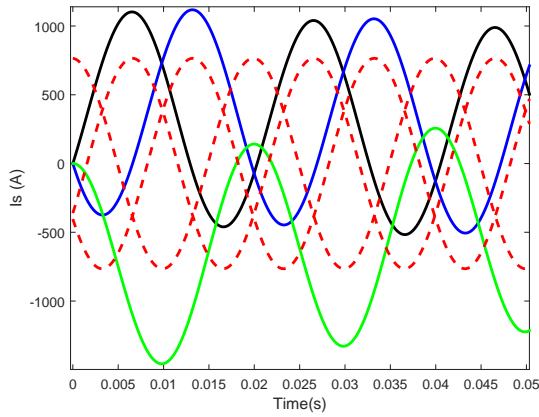


Figure 11. Secondary current of the transformer before the steady-state is reached (Red: EMTP-RV, other: Modelica).

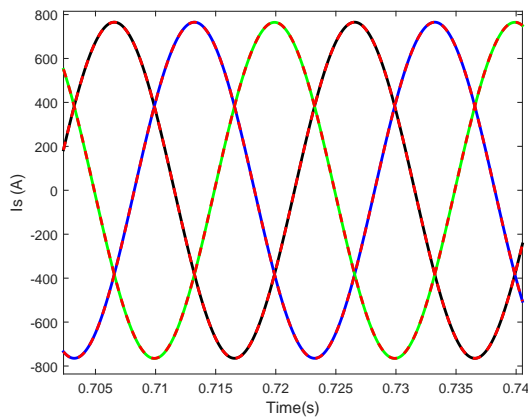


Figure 12. Secondary current of the transformer when the steady-state is reached .

the beginning of the simulation the traces do not match for the two different implementations. The traces in red are from EMTP-RV, while other traces in different colors are from the Modelica tool used. The simulation output of the Modelica model matches the EMTP-RV results after the steady-state is reached (shown in Figure 12).

Observe that when a larger test system model is to be simulated (see Section 4.2), there are more states that need to be initialized. The authors found that some of the Modelica-tools (OpenModelica and Dymola), the solvers are not able to solve the initialization problem and/or to execute the simulation successfully. For the test system shown in Figure 10 the Rkfix4 solver with interval length $1e-5$ and tolerance of 0.01 were used.

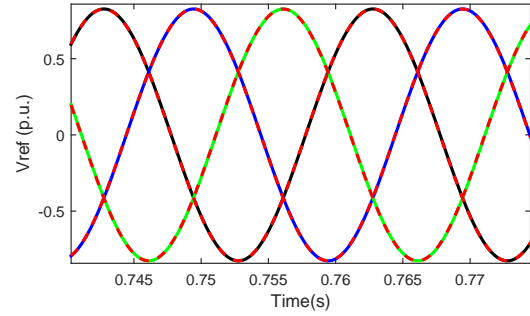


Figure 13. V_{abc_ref} of VSC1 i.e. output of upper level control (Red: EMTP-RV, other: Modelica).

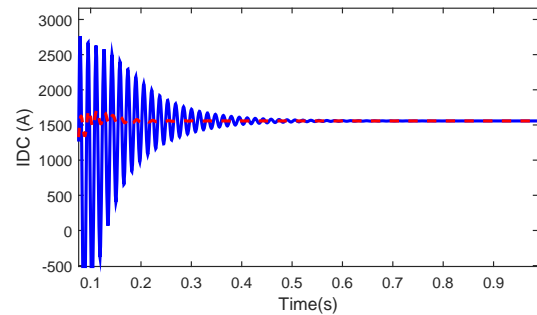


Figure 14. DC current of VSC1 (Red: EMTP-RV, Blue: Modelica).

4.4 Software-to-Software Validation

Software-to-Software validation of the VSC-HVDC model (see Figure 10) is carried out in two steps. First, simulations are carried out without applying any perturbations to the model in order to check whether the steady-state trajectories match or not. In addition, no initial values were provided to the controller's integrators in the Modelica model. As a result, the simulation is allowed to reach the steady state value before disturbances are applied and comparisons are made. Figures 13 and 14 show the simulation results for V_{abc_ref} and IDC of VSC1, showing the close match between the two different implementations.

Next, a step change in the active power reference from 1 to 0.5 (1000 MW to 500 MW) at 0.8 second is applied. The step change and response of the controller are shown in Figure 15, while other trajectories are shown in Figures 16-19. It is noted that a close match is achieved between both implementations.

5 Conclusion

This paper showed the potential use of the Modelica language to model EMT-type models of VSC-HVDC systems when the high-frequency switching process can be represented using equivalent voltage and current sources. Differently from the authors previous work (Vanfretti et al., 2017), this model is built using as much components as possible from the MSL to represent the three-phase elec-

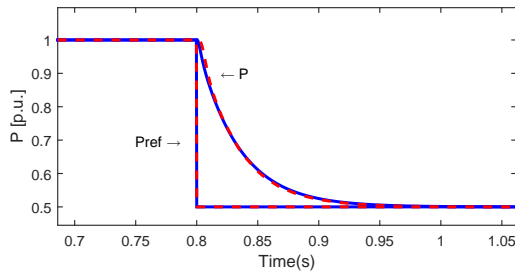


Figure 15. Active power response of VSC1 (Red: EMTP-RV, Blue: Modelica).

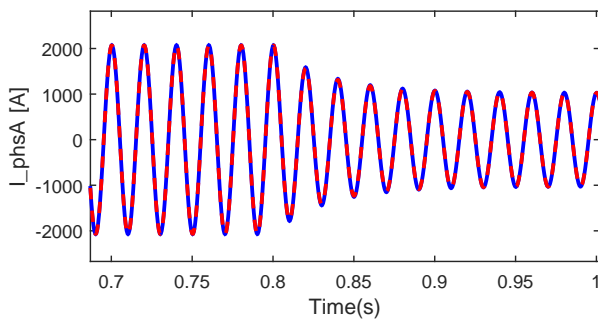


Figure 16. Primary current (Phase A) of VSC1 (Red: EMTP-RV, Blue: Modelica).

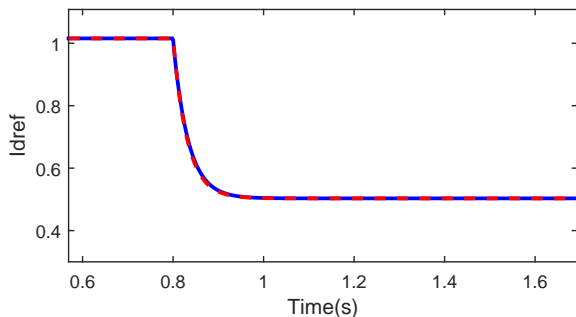


Figure 17. Current reference of upper level control of VSC1 (Red: EMTP-RV, Blue: Modelica).

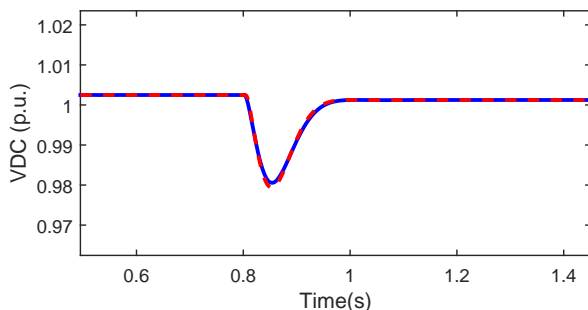


Figure 18. DC voltage on the VSC1 side (Red: EMTP-RV, Blue: Modelica).

trical circuit, while implementing the *de facto* control system models used within typical power system simulation tools.

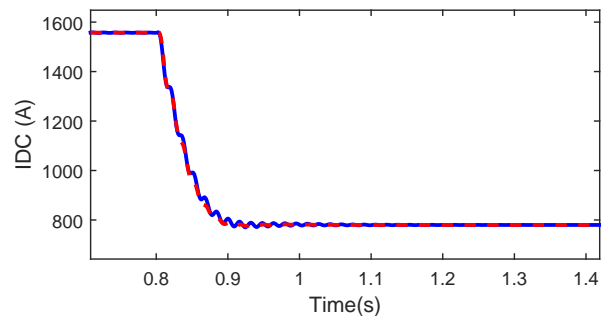


Figure 19. DC current on the VSC1 side (Red: EMTP-RV, Blue: Modelica).

The Modelica implementation was compared to the EMTP-RV software, a *de facto* power system modeling and simulation tool used for VSC-HVDC analyses, yielding surprisingly similar results (even identical when a desired disturbance is applied after the steady-state is reached). The major benefit of the work reported herein is that the control system implemented can now be exchanged with different tools that support the FMI standard, including Simulink and EMTP-RV, which makes it possible to keep and maintain a single version of the control system model implemented (i.e. the one in Modelica).

The results from this work show that there is great potential for the use of Modelica for EMT-type modeling and simulation of electrical power systems, and particularly of power electronic components. However, further work must be carried out with respect to the provision of adequate starting guess values for the initialization problem, and more importantly, to efficiently simulate switching processes without substantially affecting simulation time.

The Modelica files of the model presented in this paper are available under the GPLv3 license in the following GitHub repository: https://github.com/Smarts-Lab/2017_ModelicaConf_VSC-HVDC_AVM_Model

6 Acknowledgment

This work was supported in part by the FP7 iTesla project, the ITEA3 openCPS project, and the STandUP Collaboration Initiative.

Mohammed Ahsan Adib Murad is supported by Science Foundation Ireland under Grant No. SFI/15/IA/3074.

The authors would like to thank Professor Federico Milano for supporting the first author during the preparation of this paper.

7 Appendix

The two node test power system model parameter data are provided in the Tables 1 and 2.

References

E. N. Abildgaard and M. Molinas. Modelling and control of the Modular Multilevel Converter (MMC). *En-*

Table 1. Equivalent generator data.

Parameter	Value
Line to line RMS voltage (KV)	400
Generator short circuit capacity (MVA)	10000
R/L ratio (p.u.)	10
Frequency (Hz)	50

Table 2. VSC and two winding transformer data.

Parameter	Value
Rated power (MVA)	1000
Transformer primary voltage [r.m.s. LL] (KV)	400
Transformer secondary voltage [r.m.s. LL] (KV)	320
Frequency (Hz)	50
Transformer resistance (p.u.)	0.001
Transformer reactance (p.u.)	0.18
MMC arm inductance (p.u.)	0.15
Capacitor energy in each sub-module (KJ/MVA)	40
Number of sub-module per arm, N	400
Conduction loss of each IGBT (p.u.)	.001

A. Olenmark, J. Sloth, A. Johnsson, C. Wilhelmsson, and J. Svensson. Control development and modeling for flexible DC grids in Modelica. In *2015 The 11th International Modelica Conference*, September 2015.

J. Peralta, H. Saad, S. Denetiere, J. Mahseredjian, and S. Nguefeu. Detailed and averaged models for a 401-level MMC-HVDC system. *IEEE Transactions on Power Delivery*, 27(3):1501–1508, July 2012. ISSN 0885-8977. doi:10.1109/TPWRD.2012.2188911.

G. Reed, R. Pape, and M. Takeda. Advantages of voltage sourced converter (VSC) based design concepts for FACTS and HVDC-link applications. In *2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No.03CH37491)*, volume 3, page 1821 Vol. 3, July 2003. doi:10.1109/PES.2003.1267437.

L. Vanfretti, T. Rabuzin, M. Baudette, and M. Murad. itesla power systems library (iPSL): A Modelica library for phasor time-domain simulations. *SoftwareX*, 5:84 – 88, 2016. ISSN 2352-7110. doi:http://dx.doi.org/10.1016/j.softx.2016.05.001.

L. Vanfretti, M.A.A. Murad, and F.J.Gómez. Calibrating a VSC-HVDC model for dynamic simulations using RaPID and EMTD simulation data. In *2017 IEEE Power Energy Society General Meeting*, pages 1–5, July 2017.

ergy Procedia, 20:227 – 236, 2012. ISSN 1876-6102. doi:http://dx.doi.org/10.1016/j.egypro.2012.03.023.

B. R. Andersen, L. Xu, P. J. Horton, and P. Cartwright. Topologies for VSC transmission. *Power Engineering Journal*, 16(3):142–150, June 2002. ISSN 0950-3366. doi:10.1049/pe:20020307.

B. Bachmann and H. Wiesmann. Advanced modeling of electromagnetic transients in power systems. In *Modelica Workshop 2000*, Oct 2000.

M. P. Bahrman. Overview of HVDC transmission. In *2006 IEEE PES Power Systems Conference and Exposition*, pages 18–23, Oct 2006. doi:10.1109/PSCE.2006.296221.

F. Casella, A. Bartolini, S. Pasquini, and L. Bonuglia. Object-oriented modelling and simulation of large-scale electrical power systems using Modelica: A first feasibility study. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 6298–6304, Oct 2016. doi:10.1109/IECON.2016.7793558.

N. Flourentzou, V. G. Agelidis, and G. D. Demetriadis. VSC based HVDC power transmission systems: An overview. *IEEE Transactions on Power Electronics*, 24(3):592–602, March 2009. ISSN 0885-8993. doi:10.1109/TPEL.2008.2008441.

R. Majumder, B. Berggren, and M. Larsson. Development and comparison of DC grid model in Powerfactory and Dymola for controller design. In *2013 IEEE Power Energy Society General Meeting*, pages 1–5, July 2013. doi:10.1109/PESMG.2013.6672328.

From system model to optimal control - A tool chain for the efficient solution of optimal control problems

Manuel Gräber¹ Jörg Fritzsche² Wilhelm Tegethoff³

¹TLK Energy GmbH, Germany, manuel.graeber@tlk-energy.de

²Volkswagen AG, Germany

³Institut für Thermodynamik, TU Braunschweig, Germany

Abstract

Based on a specific application example - the thermal management system of an internal combustion engine - a toolchain is presented for formulating and solving of nonlinear optimal control problems. Starting from graphical modeling of the thermal management system with the Modelica library TIL, the model is exported to the standardized model exchange format Functional Mock-up Interface (FMI). Furthermore, it is imported to the optimal control software package MUSCOD-II. Python is used as scripting language for the problem formulation, the numerical solution and the processing of results. By using FMI as an interface, models from any simulation and modeling tools can be used if there is an FMI model export and the models fulfill certain mathematical requirements (smoothness).

Keywords: *Optimal control, Functional mock-up interface, thermal management, cooling system*

1 Introduction

When developing control concepts or superior operating strategies, frequently the question arises, what is the theoretically best possible control of a system. Questions of this kind can be mathematically expressed as *Optimal Control Problems* (OCP) describe. What is special about this class of optimization problems is the dynamics of the controlled system. Contrary to static optimization problems, not a finite number of parameters are free for optimization, but trajectories of system inputs. Therefore, an OCP is an infinite-dimensional optimization problem, which usually cannot be solved directly. However, different mathematical methods exist to determine approximated numerical solution. Detailed introductions in the theory of optimal control can be found in (Bryson and Ho 1979) and (Betts 2001). The *Direct Multiple Shooting Method* according to (H. G. Bock and Plitt 1984) used in this article is explained in Section 3.

Although these and other specialized OCP algorithms have existed for a long time, they have not yet made it into the broad industrial application. An exception to

this is the aerospace industry, in which OCPs have been solved for optimal trajectory planning for decades. The largest (in our opinion) obstacle to a widespread industrial use of optimal control is the necessary time and knowledge-intensive effort. Successful work with existing software requires a high degree of expert knowledge. According to our experience, the by far largest time effort in optimization projects cannot be seen in performing the actual optimization calculations, but rather in the modeling of the system under consideration. On the one hand, derivative-based optimization algorithms require a certain numerical model quality (differentiability) that go beyond the requirements of pure simulation algorithms. On the other hand, it is important to depict the correct positive and negative effects, the superposition of which determines the optimum. The modeling process is almost always iterative. Reliable results can only be produced by repeatedly interpreting optimization results and changing modelling details.

Based on the described experiences and observations, we suggest a tool chain in this article to use optimal control efficiently. The thermal management system of a combustion engine serves as a continuous example. Starting with the modeling of the controlled system in Section 2, the model is exported as FMU (Blochwitz et al. 2012) and imported into to the specialized optimization package MUSCOD-II (H. G. Bock and Plitt 1984; Diehl 2001; Leineweber et al. 2003). For the problem formulation, the numerical solution and the processing of results Python is used as scripting language.

Other Modelica-related optimal control projects are described in (Åkesson et al. 2010), direct collocation (Imslund et al. 2010), single shooting, and (Franke 2002), multiple shooting.

2 Modelling of the controlled system

The most important part in the successful work with optimal control is the dynamic model of the system under consideration. Mathematically, the system model

is described as a system of ordinary differential equations:

$$\frac{dx}{dt} = f(x, u, p, t) \quad (1)$$

Here x denotes the vector differential states, u the vector of inputs, p the vector of parameters and t is time. It is also possible system models with additional algebraic implicit equations. Since this is not supported by the current FMI 2.0 standard and due to better readability, we limit ourselves to explicit ordinary differential equations (ODE) of the form shown above.

Equation systems of this type are the mathematical basis of various industrially used system simulators such as Simulink or Dymola. In order to be able to exchange models between different simulators, the open standard Functional Mock-up Interface (FMI) was developed. We use FMI to link system models to optimization algorithms. Thus, it is possible to develop the system model of an optimum control problem in the modeling tool of choice. The only requirement is the availability of FMI exports.

Models that are suitable for simulation need not yet be suitable for the use of derivative-based optimizers. Discontinuities in the model equations can lead to poor or even failing convergence behavior. In practice, however, the theoretical mathematical requirements for models must not be completely satisfied. Even if the continuous differentiability of all model equations is not fulfilled, for example by the linear interpolation of characteristic fields, reasonable results can be achieved with derivation-based optimizers.

Thermal systems such as the thermal management system considered here, can be graphically modelled with the Modelica library TIL (Schulze 2013; Gräber et al. 2010; Richter 2008). Large parts of TIL are directly suitable for use in optimizers. This includes circuits with compressible liquids and ideal gas mixtures. Two-phase fluid circuits modeled with TIL are not yet suitable for optimization. However, current research deals with this topic.

Figure 1 shows the thermal management system as a TIL model. The coolant (blue) is pumped through the engine block by an electrically driven pump. There, waste heat from the combustion engine is added as time-dependent heat flow. The upper circuit through the heating heat exchanger is constantly flowed through. The lower circuit for heat dissipation to the environment can be connected via an electrical valve as required.

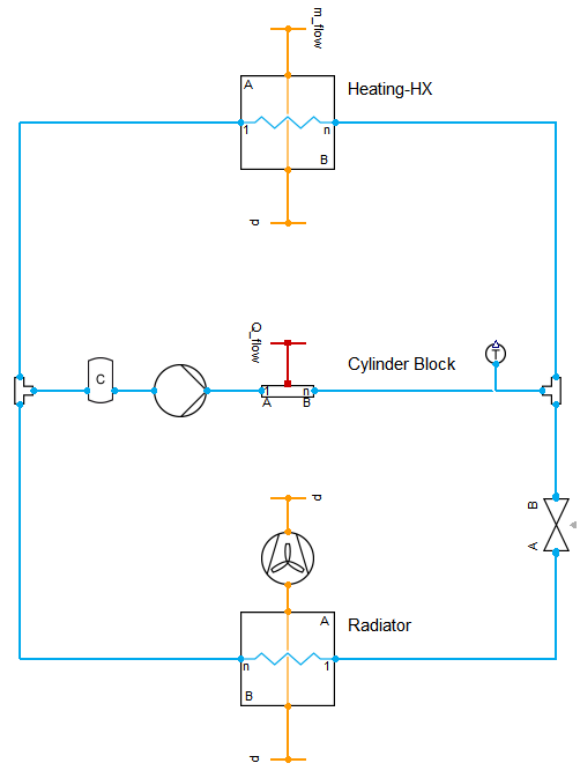


Figure 1. Sketch of the thermal management system. Screenshot of Modelica / TIL system models.

The manipulated variables of this system are:

- Water pump speed
- Cooling fan speed
- Opening degree of the valve

Both heat-exchangers are modelled according to the finite volume method with 5 discrete volumes. The system model has 36 differential states in total.

The primary control task is temperature control of the engine, which is achieved by demanding a setpoint of 90°C for the fluid temperature at the engine block outlet. Three manipulated variables and only one control variable leave two degrees of freedom. An open question is how to deal with these degrees of freedom. An obvious idea is to introduce the additional demand for the lowest possible energy consumption. Thus, the cost function to be minimized follows as:

$$C = \int_0^{t_f} P_{\text{pump}} + P_{\text{fan}} + c(T - T_{\text{set}})^2 dt \quad (2)$$

The electrical power consumption of pump and fan as well as a squared penalty term for setpoint deviations are integrated over a given period of time. The two control objectives can be weighted with the factor c . High values result in higher energy consumption but temperatures closer to the setpoint.

The input trajectories within the period under consideration are free for optimization. However, upper and lower bounds for all manipulated variables are taken into account:

$$\mathbf{u}_{lb} \leq \mathbf{u}(t) \leq \mathbf{u}_{ub} \quad \forall t \in [0, t_f] \quad (3)$$

In addition, the given initial state of the system model enters as equality constraint:

$$\mathbf{x}(0) = \mathbf{x}_0 \quad (4)$$

The complete optimal control problem follows as:

$$\begin{aligned} \min_{\mathbf{x}(\cdot), \mathbf{u}(\cdot)} \quad & C(\mathbf{x}(\cdot), \mathbf{u}(\cdot), \mathbf{p}) \\ \text{s.t.} \quad & \frac{d\mathbf{x}}{dt}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t) \quad \forall t \in [0, t_f] \\ & \mathbf{u}_{lb} \leq \mathbf{u}(t) \leq \mathbf{u}_{ub} \quad \forall t \in [0, t_f] \\ & \mathbf{x}(0) = \mathbf{x}_0 \end{aligned} \quad (5)$$

3 Numerical solution of optimal control problems

This section is based on (Gräber 2013) and attempts to explain the basic mathematical ideas behind the used numerical methods. For a deeper and more mathematical representation, references to further literature are given in several places.

Optimal control problems of the above-described form are not directly solvable by numerical methods. Considering the continuous trajectories sought as a set of infinitely many individual points, it becomes clear that an OCP is an infinite-dimensional optimization problem. Deriving analytical solutions is only possible for very simple subclasses. For most real problems, only an approximate numerical solution is possible.

Within the last decades, various methods have been developed to numerically solve optimal control problems. These can be divided into two large groups: indirect and direct methods. Indirect methods are based on *Pontryagin's Maximum Principle*. With the help of this necessary optimum condition, the OCP is analytically transformed into a boundary value problem with the original differential equations and additional adjoint equations. This boundary value problem can then be solved with various numerical methods. A frequently mentioned disadvantage of these methods is the difficult consideration of restrictions. Since it is necessary in many technical applications to limit state variables and controls to certain areas, this disadvantage is not insignificant.

Recent work deals almost exclusively with *direct* methods. The term comes from the fact that not a transformed problem, but directly the original OCP is

used. By discretizing the trajectories, the infinite-dimensional OCP is approximated with a finite-dimensional *Nonlinear Program* (NLP). This NLP can then be solved with common numerical methods such as *Sequential Quadratic Programming* (SQP) or *Interior Point Method* (IP). Within direct methods, a distinction is made between sequential and simultaneous methods.

In direct sequential procedures, the control trajectories are described by piecewise defined functions – mostly polynomials. In the simplest case, the polynomials are of the order of zero, and the controls are piecewise constant functions over time. The coefficients of these polynomials are the free optimization variables. Using an ODE or DAE solver, the cost function can now be evaluated for given control trajectories and initial values. Coupled to an optimization algorithm, the OCP can be iteratively solved by repeatedly solving an initial value problem with different control trajectories. It has been shown, that particularly for ill-conditioned problems convergence and stability properties of such methods are not very good (Hans Georg Bock 1987; Albersmeyer and Diehl 2010).

Direct simultaneous methods go one step further and discretize not only the control but also the state trajectories. In the case of direct collocation, the trajectories of all state variables and controls are again described by piecewise defined functions. The continuous ODE is converted into a system of difference equations using a suitable scheme. This equation system is included as an equality constraint in the optimization problem. Leading to a very large but finite-dimensional NLP, which can be solved with conventional methods. In order to reduce the computation time, the special structure of the equation systems can be exploited. (Biegler 2007) provides an overview of current simultaneous methods.

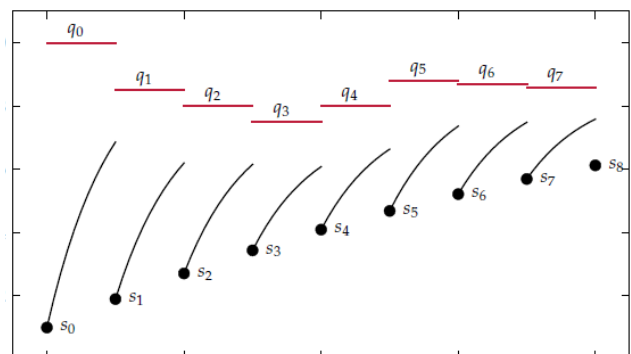


Figure 2. Multiple Shooting Method. Control trajectories (red) are discretized with piecewise constant functions and state trajectories (black) are discretized by solving independent initial value problems.

The direct multiple shooting method used in this work is usually seen as simultaneous method, but could also be interpreted as a mixed form between the sequential and simultaneous methods. Control trajectories are discretized analogously to the methods described so far. The state trajectories are also divided into individual sections. However, the path within these sections is not described by polynomials. Rather, initial values for the states are introduced at the nodes of the multiple shooting grid. At node i , these additional variables are designated as s_i . Based on these initial values and the original ODE, the trajectories of the states are determined by solving several independent initial value problems. For an arbitrary choice of the initial values, the resultant total trajectories of the states have jumps, see Figure 2. Therefore, closing conditions are included in the OCP as additional equality constraints. The value of a state variable at the end of a section must be equal to the initial value of the next section.

If an identical discretization grid with n intervals is chosen for controls and states and the controls are parameterized piecewise constant with the values q_i , the following NLP results from OCP (5):

$$\begin{aligned} \min_{\substack{s_0, \dots, s_n \\ q_0, \dots, q_{n-1}}} & \sum_{i=0}^n k_i(s_i, q_i, p) \\ \text{s.t.} \quad & s_{i+1} = x_i(t_{i+1}; t_i, s_i, q_i, p) \quad \forall i \in \{0, \dots, n-1\} \\ & u_{lb} \leq q_i \leq u_{ub} \quad \forall i \in \{0, \dots, n\} \\ & s_0 = x_0 \end{aligned} \quad (6)$$

It should be noted that the solution of an initial value problem is behind the evaluation of the cost functions $k_i(s_i, q_i, p)$ and the determination of the states at the end of an interval $x_i(t_{i+1}; t_i, s_i, q_i, p)$. In the solution of this NLP with derivative-based methods, it is of great importance to determine the derivatives of these functions with respect to the free optimization variables accurately and efficiently. This is a non-trivial task when using variable step size integrators. An extensive discussion of this topic can be found in (Bauer 1999) and (Albersmeyer 2010).

To illustrate the multiple shooting method, the discretization for a simple example is shown in Figure 2. On each shooting interval i , an independent initial value problem is solved with the initial value s_i and the constant control q_i . The figure shows the result of an optimization iteration, that has not yet converged. The violation of the matching conditions for the state variables is clearly visible.

4 Optimal Control of a Thermal Management System

This section describes optimization results for the thermal management system described in section 2.

The system model is graphically generated and parameterized in Dymola using the library TIL. With the Dymola FMI export functionality, the model is exported as FMU for Model Exchange 2.0. Control inputs must be declared as top level inputs in Modelica and variables, which are used in the cost function, as top level outputs.

The coupling of the FMU to the optimizer MUSCOD-II, and the complete configuration of the calculations is done in the Python language. The Python code used here is shown in Figure 3.

```
import fmi_muscod as fm

# create an OCP object
ocp = fm.OptimalControlProblem()

# Load FMU as system model
ocp.loadFMU('ThermomanagementSystem.fmu')

# Configure time: 600s divided into 60 intervals
ocp.setTimeHorizon(60, 600.0)

# Estimate start values, bounds and scales
# based on a forward simulation.
# Inputs are set constant to given values.
ocp.estimateSettings([50.0, 50.0, 0.3])

# Overwrite bounds for inputs
ocp.setLowerBounds(uBounds=[10.0, 1.0, 0.01])
ocp.setUpperBounds(uBounds=[100.0, 100, 1.0])

# Scale cost function
ocp.setScaleValues(cScale = 1e5)

# Define cost function (Lagrange type)
lval = '(x33 - 90.0)*(x33 - 90.0) + 0.001*y0'
ocp.setCostFunction(lval)

# Start optimization
ocp.solve()

# Plot results for some states
# directly in Python
ocp.plotResult([1, 33, 34, 35, 31])

# Export result to Modelica
# for detailed post-processing
ocp.exportResultToModelica()
```

Figure 3. Python code for the numerical solution of the optimal control problem.

The scenario considered is a 10-minute drive up a mountain pass after a cold start at 20°C. This means that relatively much engine waste heat is introduced into the cooling circuit, which in turn has to be dissipated to the ambient air. Due to the comparatively low uphill speed the cooling fan has to be used more extensively. The

optimum control trajectories are calculated as piecewise constant curves with an interval length of 10s. A simulation of the thermal management system with a simple control concept is used as a basis for comparison. For this purpose, the pump, such as a mechanical water pump, is operated at a rotational speed coupled to the motor speed. The valve is controlled by a P-controller and a setpoint of 85°C for the coolant temperature. While the fan controls the same temperature with a PI controller to the desired setpoint of 90°C.

Figure 4 shows the optimum and simulated (with PI controllers) controls. Obvious differences are:

- Maximum pump speed in the first minute of the optimal solution
- The fan becomes active in the optimal solution earlier.

The first difference is only to be explained by the fact that the electrical power of the pump is used to heat the coolant. At the beginning all temperatures are at 20°C. In order to reach the setpoint of 90°C as quickly as possible, it is worth (in the sense of the cost function) to use the electrical power of the pump to heat up the system.

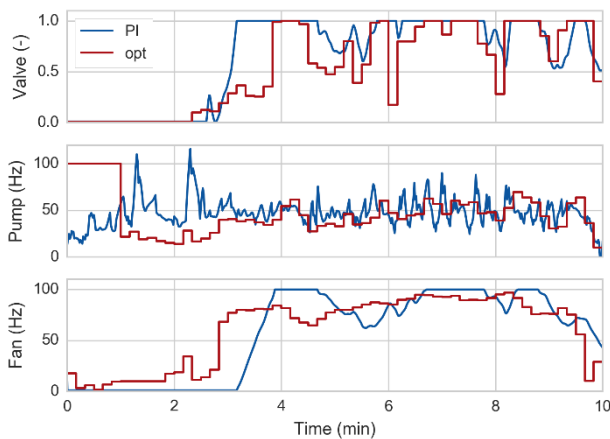


Figure 4. Controls from optimization and simulation. The optimum control uses the fan earlier and completely opens the valve later.

The second difference can be explained by looking at figure 5. While the PI control of the temperature does not become active until the setpoint is exceeded, the optimum control reacts earlier. With increased fan speed and valve opening, cooling is started before 90°C is reached. An exact approach of the setpoint can thus be achieved, without overshooting. In addition, it is avoided that the fan is operated with maximum speed and disproportionately high energy demand.

This second positive aspect is clearly visible in figure 6. The cumulative electrical energy consumption of both variants is shown, divided into fan and pump energy. The fan energy clearly dominates in both cases. In the case of PI control, the fan runs at a much higher speed

compared to optimal control, especially between minutes 4 and 5. Within this time span, the PI controller reacts to the overshooting temperature. This has the consequence that the cumulative energy consumption increases sharply. Whereas the more uniform optimal control of the fan speed leads to total energy consumption reduced by 19%. The individual numerical values are listed in Table 1.

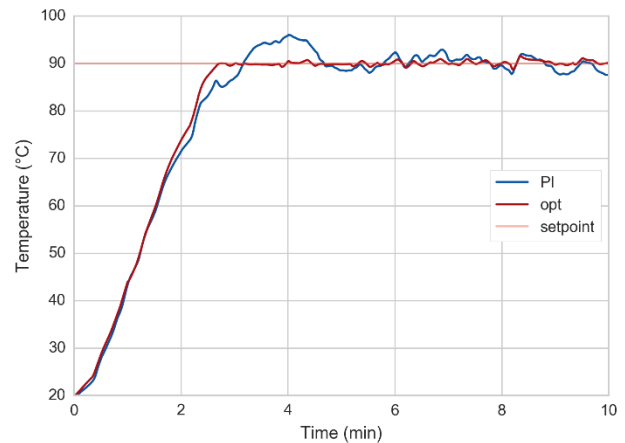


Figure 5. Results from optimization and simulation. The primary control objective of keeping the coolant temperature at 90 °C is achieved in both cases. The optimum control achieves the setpoint value somewhat earlier, without overshooting.

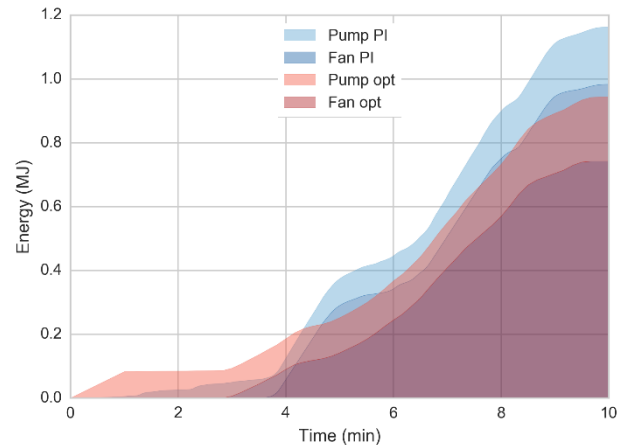


Figure 6. Cumulative electrical energy consumption from optimization and simulation. The optimum control achieves a 19% reduction in energy consumption.

For the described cold-start high-load scenario, the optimum control shows a significant reduction in energy consumption while at the same time better compliance with the setpoint for the coolant temperature. With the presented tool chain, such investigations can also be carried out for other systems and scenarios. Such optimal control results can be used for various purposes:

- as reference for control concepts
- finding heuristic (almost optimal) control laws
- for online optimization (NMPC)

Table 1. Total consumed electrical energy for both variants.

	<i>Pump</i>	<i>Fan</i>	<i>Total</i>
<i>PI control</i>	<i>0.18 MJ</i>	<i>0.98 MJ</i>	<i>1.16 MJ</i>
<i>Optimal control</i>	<i>0.20 MJ</i>	<i>0.74 MJ</i>	<i>0.94 MJ</i>
<i>Difference</i>	<i>+12%</i>	<i>-25%</i>	<i>-19%</i>

5 Summary

With the presented tool chain Modelica/TIL → FMI → MUSCOD-II thermal system can be modeled conveniently and optimal control trajectories can be calculated rapidly. For the example application, thermal management of an internal combustion engine, electrical energy savings of 19% (fan and pump) compared to PI control are achieved. By comparing the optimization and simulation results, the causes for energy savings can be explained.

Optimization calculations of this type can serve as a reference for control concepts to be developed. The interpretation of the optimal trajectories can also be used in finding heuristic (almost perfect) control laws. In principle, optimum control calculations are also suitable for online use in vehicles or other technical systems. Which is known as nonlinear model predictive control (NMPC). For prototypical NMPC applications on a Windows laptop, the presented tool chain can be used directly. However, it is not yet suitable for implementation on embedded control units.

References

- Åkesson, Johan, Karl-Erik Årzén, Magnus Gäfvert, Tove Bergdahl, and Hubertus Tummescheit. 2010. "Modeling and Optimization with Optimica and JModelica.org--Languages and Tools for Solving Large-Scale Dynamic Optimization Problems." *Computers & Chemical Engineering* 34 (11): 1737–49. doi:10.1016/j.compchemeng.2009.11.011.
- Albersmeyer, Jan. 2010. "Adjoint Based Algorithms and Numerical Methods for Sensitivity Generation and Optimization of Large Scale Dynamic Systems." Ruprecht-Karls-Universität Heidelberg.
- Albersmeyer, Jan, and Moritz Diehl. 2010. "The Lifted Newton Method and Its Application in Optimization." *SIAM Journal on Optimization* 20 (3): 1655–84. <http://epubs.siam.org/doi/abs/10.1137/080724885>.
- Bauer, Irene. 1999. "Numerische Verfahren Zur Lösung von Anfangswertaufgaben Und Zur Generierung von Ersten Und Zweiten Ableitungen Mit Anwendungen Bei Optimierungsaufgaben in Chemie Und Verfahrenstechnik." Universität Heidelberg. doi:10.1159/000328458.
- Betts, John T. 2001. *Practical Methods for Optimal Control Using Nonlinear Programming*. Society for Industrial and Applied Mathematics.
- Biegler, L. 2007. "An Overview of Simultaneous Strategies for Dynamic Optimization." *Chemical Engineering and Processing: Process Intensification* 46 (11): 1043–53.
- Blochwitz, T., M. Otter, J. Åkesson, M. Arnold, C. Clauß, H. Elmqvist, M. Friedrich, et al. 2012. "Functional Mockup Interface 2.0: The Standard for Tool Independent Exchange of Simulation Models." In *9th International Modelica Conference*.
- Bock, H. G., and K. J. Plitt. 1984. "A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems." In *Proc. of the 9th IFAC World Congress Budapest*, 243–47. Pergamon Press.
- Bock, Hans Georg. 1987. *Randwertproblemmethoden Zur Parameteridentifizierung in Systemen Nichtlinearer Differentialgleichungen*. Universität Bonn.
- Bryson, Arthur E., and Yu-Chi Ho. 1979. *Applied Optimal Control: Optimization, Estimation, and Control*. John Wiley & Sons Inc.
- Diehl, Moritz. 2001. "Real-Time Optimization for Large Scale Nonlinear Processes." Universität Heidelberg.
- Franke, Rüdiger. 2002. "Formulation of Dynamic Optimization Problems Using Modelica and Their Efficient Solution." In *2nd International Modelica Conference*, 315–23. Oberpfaffenhofen.
- Gräber, Manuel. 2013. "Energieoptimale Regelung von Kälteprozessen." TU Braunschweig.
- Gräber, Manuel, Kai Kosowski, Christoph Richter, and Wilhelm Tegethoff. 2010. "Modelling of Heat Pumps with an Object-Oriented Model Library for Thermodynamic Systems." *Mathematical and Computer Modelling of Dynamical Systems* 16 (3): 195–209. doi:10.1080/13873954.2010.506799.
- Imsland, L., P. Kittilsen, and T. S. Schei. 2010. "Model-Based Optimizing Control and Estimation Using Modelica Models." *Modeling, Identification and Control* 31 (3): 107–21. doi:10.4173/mic.2010.3.3.
- Leineweber, D B, I Bauer, A A S Schäfer, H G Bock, and J P Schlöder. 2003. "An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization (Parts I and II)." *Computers and Chemical Engineering* 27: 157–74.
- Richter, Christoph. 2008. "Proposal of New Object-Oriented Equation-Based Model Libraries for Thermodynamic Systems." Technische Universität Braunschweig.
- Schulze, C. 2013. "A Contribution to Numerically Efficient Modelling of Thermodynamic Systems." Technische Universität Braunschweig.

Nonlinear Model Predictive Control of a Thermal Management System for Electrified Vehicles using FMI

Torben Fischer¹ Tom Kraus² Christian Kirches² Frank Gauterin³

¹Fraunhofer Institute for Chemical Technology (ICT), Project Group New Drive Systems, Germany, torben.fischer@ict.fraunhofer.de

²Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Germany, {tom.kraus,christian.kirches}@iwr.uni-heidelberg.de

³Institute of Vehicle System Technology, Karlsruhe Institute of Technology (KIT), Germany, frank.gauterin@kit.edu

Abstract

Energy-efficient thermal management systems for E-mobility help to decrease energy consumption and increase range. Due to transient external conditions and the increasing system complexity, optimization-based control approaches are required in order to harness the full potential of such systems. In (Fischer et al., 11th Int. Modelica Conf, 2015), we have presented a model-based development cycle for a thermal management system in E-mobility to this end. In this article, we build upon this work to describe the use of this model within a nonlinear model predictive control (NMPC) approach. The main benefits of using an advanced optimization-based control system in this application are a) the ability to control the battery temperature and the cabin temperature simultaneously, b) the increased energy efficiency achieved by exploiting the predictive character of the optimization-based control approach, c) the possibility to include operational limits as constraints in the optimization problems and d) the fast reaction to disturbances or model parameter changes. We evaluate the merit of the proposed advanced control system by way of a comparison to conventional PID controller.

Keywords: thermal management system, nonlinear model predictive control, Functional Mock-up Interface

1 Introduction

E-mobility is widely considered to be a key concept to achieve ambitious goals set forth in contemporary climate and environmental protection plans. Due to higher costs and lower ranges compared to combustion engine driven vehicles, a breakthrough in the mass market has yet to take place. In this article we propose an optimization-based control for energy-efficient operation of a thermal management system. In (Fischer et al., 2015) we observed a decrease of the energy consumption of up to 30%, depending on ambient conditions. To improve the system further, a nonlinear model predictive control (NMPC) approach is proposed with the aim to harness the full potential of the multiple-input multiple-output system (MIMO).

This article constitutes a follow-up of (Fischer et al.,

2015), where the concept of the thermal management system is presented, including simulation results. The remainder of this article is structured as follows: §1 introduces the subject and describes the related state-of-the-art. §2 recalls the process model of the thermal management system. A short discussion of the NMPC approach resides in §3 covering the formulation as a mathematical optimization problem, the multiple-shooting discretization, a real-time solution algorithm and the employed software interface. In §4 process model modifications are described which were necessary in order to employ derivative-based optimization techniques. §5 contains an "offline" case study to compare different approaches of jacobian matrix generation on the basis of the Karush-Kuhn-Tucker (KKT) violation and an "online" case study to compare NMPC to conventional PI control. Finally, we provide conclusions and an outlook on future topics in §6.

1.1 State of the Art

NMPC is widely used in, e.g., process control and chemical engineering, often with rather slow sampling rates. In the past years, the automotive industry has also shown an increased interest in model predictive control. Applications like adaptive cruise control (Kirches, 2011; Kirches et al., 2013), lateral dynamic stabilization, etc., can be typically controlled by a MPC-controller. Further examples may be found in, e.g., (del Re et al., 2010). In the area of heating, ventilation, and air conditioning (HVAC), conventional control methods like PID-controllers and bang-bang-controllers are still state-of-the-art, mostly due to simplicity of design and implementation. There are, as well, investigations on advanced control systems. For example, (Esen et al., 2014) and (Karnik et al., 2016) use MPC-controllers in the application field of thermal management systems, and (Afram and Janabi-Sharifi, 2014) gives an general overview for HVAC systems. To reduce the computational effort, these approaches however often do not rely on first-principles models, but rather on data models or linearized state-space representations. The first publication of an NMPC-controller based on first-principle models using Modelica is (Franke, 2002). In (Gräber et al., 2012), a functional mock-up unit (FMU)

of a first-principle Modelica model was used for the first time within an fully nonlinear MPC setting. This example is closely related to the present article, as it treats a compression-vapor cycle.

2 Thermal Management System

In this section, we review the layout of a thermal management system as introduced in (Fischer et al., 2015). The thermal management system of a vehicle has multiple tasks. Primarily, for passenger comfort heating or cooling of the passenger cabin is required. Moreover, there are a number of legal requirements to be met including windshield defrosting and defogging. Finally, powertrain components have to be kept within their thermal operational range. In the particular case of an electrified vehicle, increasing demands due to thermally even more sensitive components and significant less amount of waste heat lead to a development of new thermal management systems.

2.1 Concept and Model

The main feature of the system, depicted in Fig. 1, is a reversible heat pump, called thermal module (1), which provides heating and cooling power to keep the components within a thermal operating range. Waste heat emitted by electric components (2) is used to increase the temperature level of the heat source, thereby contributing to a higher efficiency of the system. By way of a flexible interconnection, independent thermal conditioning of the cabin (3) and the battery (4) may be achieved. By using waste heat from an optional energy converter like a fuel cell or an internal combustion engine (5) in a hybrid electric vehicle, the thermal module can be switched off.

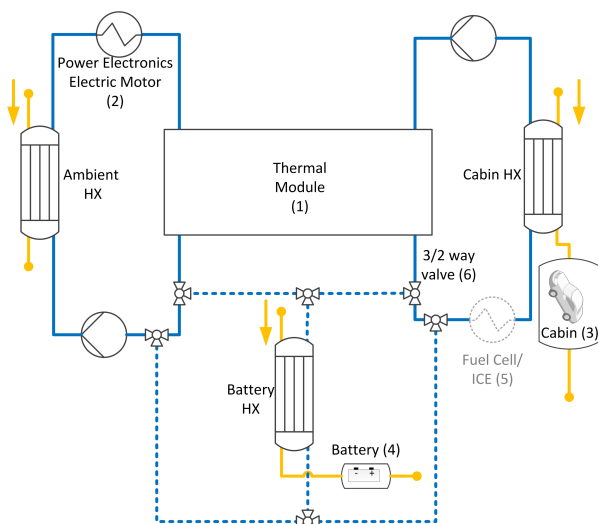


Figure 1. Scheme of the thermal management system

The heat exchangers in the refrigerant cycle are modeled in distinct ways. The condenser is modeled using a moving boundary approach while the evaporator is realized using a finite volume method. Details can be found in §4.2. The heat exchangers modeling the heat flow between

coolant and air are also realized using a finite volume method. The refrigerant accumulator, the coolant reservoirs, and the passenger cabin are modeled using lumped volumes. The fluid data is taken from the TILMedia library, which provides bi-cubic spline interpolants for the used refrigerant. The coolant is modeled as pure water and the ambient air as dry air.

2.2 Controlled Variables

The controlled variables in the system shown in Fig. 1 are the cabin temperature T_{cabin} , the superheating temperature of the heat pump T_{sh} and also the battery temperature T_{battery} . The desired cabin temperature T_{cabin} is assumed to be at 22°C according to passenger preference. To ensure a safe and efficient operating mode, the superheating temperature of the heat pump T_{sh} is desirable to be at 5 K. The admissible thermal operating interval of the battery is from 20°C to 40°C. A temperature above this range can lead to increased aging effects and eventually to degradation and a thermal runaway. The operating temperatures of the electric motor and the power electronics are usually observed but not tracked to a preset value.

2.3 Manipulated Variables

One manipulated variable in the system is the rate of change u_{compr} of the compressor frequency of the heat pump, which in a PID framework would be controlled according to the cabin temperature setpoint. The second input is the rate of change u_{valve} of the expansion valve which in a PID logic would be assigned to the superheating temperature setpoint. As the system contains two thermally conditioned components, the output heat rate is split up by a 3/2-way-valve ((6) in Fig. 1) in the coolant by dividing the mass flow. The valve is a linear control valve and accepts continuous values between 0 (path to battery closed) and 1 (path to battery open).

2.4 Control Approach

In (Fischer et al., 2015), we have described a first control approach, namely a decoupling of the MIMO-system into multiple SISO-systems, which can then be controlled individually and by separate PID controllers. This approach turned out to be problematic, as the battery temperature and thus the cabin comfort was affected by a starting thermal conditioning of the battery. The influences of the PID controllers on each other also led to inefficient overshooting behavior and oscillations. As the main control goal is energy-efficiency in order to allow for maximum electric range of the vehicle, we propose an NMPC approach to control the system.

In a first step, a reduced model of the thermal management system without battery and attached vehicle model is used to compare the control schemes in the computational studies in §5. This is appropriate since it is still convenient to control this reduced system by PI-controllers which serves as good reference for the NMPC-controller.

3 Nonlinear Model Predictive Control

In this section, we briefly review Nonlinear Model Predictive Control (NMPC) as an optimization-based scheme for advanced closed-loop feedback control of dynamic processes.

3.1 NMPC Problem Formulation

A suitable mathematical model description, capable to predict the future behavior of the dynamic process under consideration, must be available for the realization of any NMPC controller. For the system at hand, a nonlinear differential-algebraic system of equations (DAE) is suitable. The objective function to be minimized is typically assumed to be of “tracking type”, i.e. set-points are provided for the controlled variables. In addition, NMPC permits to include nonlinear constraints on process quantities. The optimization problem to be solved in order to find optimal controls may then be formulated as a DAE-constrained optimal control problem and reads

$$\begin{aligned} \min_{x,z,u} & \left\| \int_0^T \ell(x(\tau), z(\tau), u(\tau), p) d\tau \right\|_{2,Q}^2 + \|e(x(T), p)\|_{2,W}^2 \\ \text{s.t.} & \quad \dot{x}(\tau) = f(x(\tau), z(\tau), u(\tau), p) \quad \tau \in [0, T] \quad (1a) \\ & \quad 0 = g(x(\tau), z(\tau), u(\tau), p) \quad \tau \in [0, T] \quad (1b) \\ & \quad x(0) = \hat{x}_0(t) \quad \tau \in [0, T] \quad (1c) \\ & \quad 0 \leq c(x(\tau), z(\tau), u(\tau), p) \quad \tau \in [0, T] \quad (1d) \\ & \quad 0 \leq r_i(x(\tau_i), z(\tau_i), p) \quad \{\tau_i\} \subset [0, T] \quad (1e) \end{aligned}$$

Herein, an objective function of least-squares type on the prediction horizon $[0, T]$ is composed of an integral term ℓ with weight matrix Q and an end-point term e with weight matrix W , and tracks set-points provided for certain controlled variables. The problem is constrained by the DAE model equations (1a, 1b), by inequality path constraints on dynamic states and controls (1d), and by point constraints on a grid $\{\tau_i\} \subset [0, T]$ that may cover, for example, boundary or periodicity conditions.

By way of constraint (1c), the current process state $\hat{x}_0(t)$ at physical time t is embedded into the dynamic optimization problem, and must hence be available as a measurement or be provided by an observer. If the process model is sufficiently accurate and the formulation of the optimization problem is suitable, its solution yields optimal process inputs $u^*(\tau)$ on $\tau \in [0, T]$ where $\tau = 0$ coincides with the physical time t at which the observation was taken.

In practice, however, model predictions must be assumed inaccurate because of inevitable measurement or actuation errors as well as due to systematic model errors. Naturally, this effect becomes more apparent over longer time horizons T . For this reason, u^* , computed from the initial state $\hat{x}_0(t)$, is applied to the process for a “short” time only. The natural choice for the length of this “short” time interval is the system’s sampling time.

The optimization procedure is continuously repeated, each time a new measurement is available. This makes

NMPC a true closed-loop control scheme. Fig. 2 visualizes this control concept.

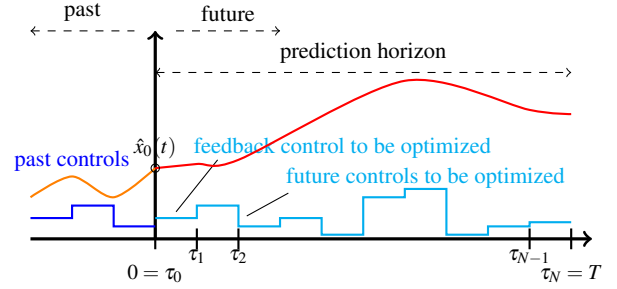


Figure 2. The Nonlinear Model Predictive Control paradigm for a piecewise constant control subject to optimization on the prediction horizon.

3.2 Direct Multiple Shooting

In order to computationally solve problem (1) efficiently, a parameterization of the control u and a discretization of the states x and z in time is necessary in a direct approach to optimal control. With the direct multiple shooting method (Bock and Plitt, 1984), we employ a simultaneous approach.

To this end, the control u is parameterized by piecewise constant control parameters q on a discretization grid $0 = \tau_0 < \tau_1 < \dots < \tau_{N-1} < \tau_N = T$,

$$u(\tau) := q_i \in \mathbb{R}^{n_u}, \quad \tau \in (\tau_i, \tau_{i+1}), \quad 0 \leq i \leq N-1.$$

On each control interval $[\tau_i, \tau_{i+1}]$, a separate DAE initial-value problem (IVP)

$$\dot{x}(\tau) = f(x(\tau), z(\tau), q_i, p), \quad \tau \in [\tau_i, \tau_{i+1}] \quad (2a)$$

$$0 = g(x(\tau), z(\tau), q_i, p) - \theta_i(\tau)g(s_i, z_i, q_i, p) \quad (2b)$$

$$x(\tau_i) = s_i, \quad z(\tau_i) = z_i \quad (2c)$$

is solved, given the initial value $s_i \in \mathbb{R}^{n_x}$.

The DAE condition is relaxed using a function $\theta_i(\tau)$ that is monotonically strictly decreasing on $[\tau_i, \tau_{i+1}]$ and that satisfies $\theta(\tau_i) = 1$ and $\theta(\tau_{i+1}) = 0$. This relieves us of having to find consistent initial values $z_i \in \mathbb{R}^{n_z}$ for solving the IVP. Consistency in the optimal solution will be ensured by requiring

$$0 = g(s_i, z_i, q_i, p), \quad 0 \leq i \leq N.$$

Continuity of the IVP solutions thusly obtained is enforced by additional matching conditions,

$$0 = x(\tau_{i+1}; \tau_i, s_i, z_i, q_i, p) - s_{i+1}, \quad 0 \leq i \leq N-1,$$

wherein $x(\tau_{i+1}; \tau_i, s_i, z_i, q_i, p)$ denotes the solution of the i -th IVP on $[\tau_i, \tau_{i+1}]$ and for initial values s_i and z_i . Finally, path and point constraints (1d, 1e) are enforced in the time grid points τ_i only. The integral least-squares objective function is evaluated along the solution of (2).

The nonlinear programming problem (NLP) resulting from this discretization and parameterization reads

$$\min_{s,z,q} \Phi := \sum_{i=0}^{N-1} \|L_i(s_i, z_i, q_i)\|_{2,Q}^2 + \|e(s_N, z_N)\|_{2,W}^2 \quad (3a)$$

$$\text{s.t. } 0 = x(\tau_{i+1}; s_i, s_i, z_i, q_i) - s_{i+1}, \quad 0 \leq i \leq N-1 \quad (3b)$$

$$0 = g(s_i, z_i, q_i) \quad 0 \leq i \leq N \quad (3c)$$

$$0 = s_0 - \hat{x}_0(t) \quad (3d)$$

$$0 \leq r_i(s_i, z_i, q_i) \quad 0 \leq i \leq N. \quad (3e)$$

Herein, L_i is an appropriate quadrature rule for ℓ on $[\tau_i, \tau_{i+1}]$, r_i summarizes the path and point constraints (1d,1e) in τ_i , $q_N := q_{N-1}$, and the dependencies on p have been omitted. For future reference, we denote by $c(\cdot)$ the set of equality constraints (3b, 3c, 3d) and $d(\cdot)$ refers to (3e).

This highly structured nonlinear programming problem is solved by a tailored sequential quadratic programming (SQP) method as described in (Bock and Plitt, 1984; Leineweber et al., 2003). For NMPC, a single iteration k of this method may be divided into three distinct phases according to the real-time iteration scheme first proposed in (Diehl, 2001), as follows:

1. **Prepare:** In the k -th SQP iterate $w^{(k)} = (s^{(k)}, z^{(k)}, q^{(k)})$, compute the gradient $b^{(k)}$ of the objective and the Jacobian $J^{(k)}$ of the least-squares objective residuals of (3), evaluate the (in-)equality constraint residuals $c^{(k)}$, $d^{(k)}$, and compute linearizations $C^{(k)}$, $D^{(k)}$ of the (in-)equality constraints.

2. **Feedback:** Obtain a state measurement or estimate $\hat{x}_0(t)$. Solve the quadratic programming problem

$$\begin{aligned} \min_{\Delta w} \quad & \frac{1}{2} \Delta w^T (J^{(k)T} J^{(k)}) \Delta w + b^{(k)T} \Delta w \\ \text{s.t.} \quad & 0 = C^{(k)} \Delta w + c^{(k)} \\ & 0 \leq D^{(k)} \Delta w + d^{(k)} \end{aligned} \quad (\text{QP})$$

to find $\Delta w = (\Delta s, \Delta z, \Delta q)$ and return $u_0^{(k)} + \Delta u_0^{(k)}$ as the new feedback control.

3. **Transition:** Determine a step length $\alpha^{(k)} \in (0, 1]$ by way of a globalization approach, and let $w^{(k+1)} \leftarrow w^{(k)} + \alpha^{(k)} \Delta w$, $k \leftarrow k + 1$.

For online optimal control (NMPC), the three phases are continuously repeated as fast as CPU resources permit and state estimates become available. For offline optimal control, the three phases constitute one iteration of an SQP method for nonlinear programming, cf. (Nocedal and Wright, 2006). These are carried out until the termination criterion

$$\|\nabla \mathcal{L}(w^{(k)})\| + \sum_i \lambda_i |c_i(w^{(k)})| + \sum_j \mu_j [d_j(w^{(k)})]^-,$$

referred to as the KKT violation, falls below a preset threshold. Herein, \mathcal{L} denotes the Lagrangian of (3) and

λ , μ denote the most recent Lagrange multipliers of the equality and inequality constraints of (QP), respectively. In the offline case, the embedding of $\hat{x}_0(t)$ is replaced by a fixed initial value.

3.3 Software Interfaces

A state of the art software package that implements the numerical algorithm just presented is MUSCOD, see (Bock and Plitt, 1984; Leineweber et al., 2003). The DAE initial value problems (2) are solved by DAESOL, cf. (Bauer et al., 1999; Albersmeyer, 2010).

The developed Modelica model has to be interfaced with the DAE solver of MUSCOD. To this end, the Functional Mockup Interface (FMI) (Blochwitz et al., 2011) is one convenient way to do this. Advantages are easy handling, simulation speed (as the model is provided as a dynamic link library), and the small effort required to export existing Modelica models as Functional Mockup Units (FMU). The interfacing between MUSCOD and the FMU is carried out in C++, which has already been described in detail in (Gräber et al., 2012).

Due to limitations in the current version 2.0 of the FMI standard, only an ODE interface can be exposed to MUSCOD. Hence, in place of the DAE IVPs (2), the ODE IVPs

$$\begin{aligned} \dot{x}(\tau) &= f(x(\tau), g^{-1}(x(\tau), q_i, p), q_i, p), \tau \in [\tau_i, \tau_{i+1}], \\ x(\tau_i) &= s_i \end{aligned}$$

are solved and the local inversion of the algebraic constraint g for the unknown $z(t)$ is internally taken care of by the FMU by way of an iterative nonlinear root-finding method. This situation is unfortunate from the point of view of an all-at-once method for dynamic optimization, as these inner iterations could be carried out much cheaper as part of the solution procedure for the nonlinear programming problem (3). Moreover, the possibility of different outcomes of adaptive choices during finite difference approximation of Jacobians of f may introduce unnecessary approximation errors here. Nonetheless, we have not observed numerical instabilities that could be traced back to this issue.

Offline Optimization. "Offline" optimization is a dynamic optimization for a given initial point, time horizon and number of intervals which does not incorporate any feedback from the real-world plant. The output data comprises the state vector and the optimal control vector at each interval, which can be provided in the simulation within a time table in a further step. A stable and robust offline optimization is essential for the online optimization. Fig. 3 shows the scheme of the offline optimization.

For the "online" optimization an integration of the real-world plant model is needed. The optimized manipulated variables, provided by MUSCOD, have to be applied online to the real-world plant model. This requires that an up-to-date measurement of the real-world plant state is available, since it serves as the initial state for predictions

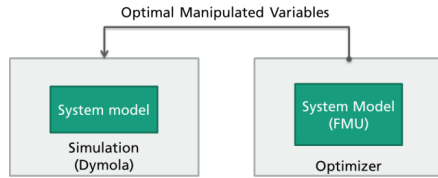


Figure 3. Scheme of the offline optimization

in the control model. Fig. 4 shows the scheme of the on-line optimization. Basically, there are two paths we follow to interface the model:

Online Optimization with Simulation in Dymola. A Python script is the central control script, which starts the simulation in Dymola, sets the optimal manipulated variable and gets the updated measurement data via the Direct Data Exchange (DDE) Interface. It is challenging though to filter the needed variables, as there is no function in Dymola to get the state vector of a model. The DDE Server of Dymola has to be executed before and the simulation speed must be reduced to real-time.

Online Optimization with Simulation in MUSCOD. In this approach no simulation environment is needed anymore. A Python script starts the optimization, and starts a sequential simulation after each optimization interval. At the end of each simulation part, the states can be extracted out of the FMU and passed to MUSCOD as new initial point. This leads to a very fast result since the simulation is a lot faster than real time. Furthermore, an ideal NMPC can be simulated, where no time is needed to calculate the optimal manipulated variables.

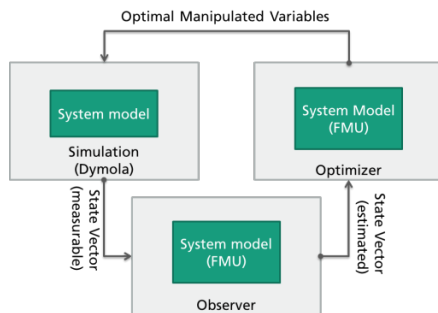


Figure 4. Scheme of the online optimization.

4 Model Adaptations for Optimization

In this section, we report on lessons learned while developing and carrying out adaptation procedures necessary in order to make the existing thermodynamical model fit for optimization based control using gradient-based methods.

4.1 Continuous Differentiable Model

This choice of numerical optimization algorithm and initial value problem solver implies certain smoothness and

regularity assumptions for (1). In particular, ℓ , e , f , g , c , and r_i in (1) need to be twice continuously differentiable w.r.t. all arguments. Furthermore, the algebraic constraint function g needs to be invertible w.r.t. the algebraic state $z(t)$, i.e., the Jacobian $\partial g(x, z, q, p) / \partial z \in \mathbb{R}^{n_z \times n_z}$ has full rank for all applicable values of its arguments.

A first requirement hence is to adapt the existing model to a model that conforms to the requirements set forth. The model must not contain any discontinuities; conditional statements, $\min()$, $\max()$, $\text{abs}()$ -functions and limiters have to be avoided; the use of the `actualStream()`-operator is no longer possible. Occurrences must be replaced by continuous and twice differentiable statements. Fig. 5 shows a discontinuous function, typically used in hybrid simulations, which in this case is replaced by the logistic function with $k = 10$ and $x_0 = 3$,

$$f(x) = (1 + e^{-k(x-x_0)})^{-1}.$$

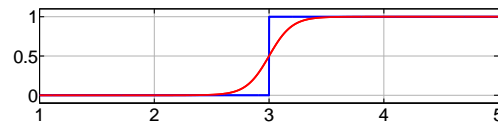


Figure 5. Discontinuous transitions have to be replaced by numerical smooth functions.

4.2 Phase Change in Condenser

The rise of the density when crossing the boiling point curve of the refrigerant in the condenser is discontinuous and leads to problems in the optimizing process. As stated in section 2.1, the heat exchangers are modeled by a finite-volume-method. This means, that the flow path is discretized into N cells. Each cell consists of mass flow and energy conservation. Depending on the operating point or in transient conditions, the crossing of the boiling point curve can occur in different cells, and also within a cell. The discontinuous rise of the density affects the state variable enthalpy, which eventually leads to problems for the optimizer finding a proper gradient.

To solve this problem, the modeling approach of a moving boundary heat exchanger, cf. (Jensen and Tummescheit, 2002) is used. In this case, the flow path is not discretized into N cells, but always into three cells according to the fluid phases: subcooled, two-phase and superheated. Thus, the rise of the density within a cell can be avoided, as one cell is always considered as a homogeneous phase.

However, using a moving boundary heat exchanger leads to another major problem. The model is only valid, if the condenser still contains the three zones superheating, two-phase and subcooled. It is therefore essential to keep the heat pump in an operating state, where all three zones exist. This is achieved by introducing soft constraints punishing operating points of the heat pump which should be avoided. The limits are defined as follows: the length of

the subcooled condenser cell l_{sc} should be greater than 5% of the total length, and the superheating temperature T_{sh} should be greater than 4 K. In case of violating these soft constraints, a high weighted penalty is added to the objective variable, indicating the optimizer to avoid this state. There are some approaches to model a switching moving boundary model, cf. (Bonilla et al., 2015). Since the switching algorithm always relies on boolean logic, it cannot be used with a gradient-based SQP algorithm.

4.3 FMU Status Report

The status of the FMU is returned by each function after calling to indicate the success of the function call. In the case of `fmi2SetContinuousStates` the status can be `fmi2Discard`, indicating that it is recommended to discard the last solution and to evaluate the model equations again with a smaller time step. This information has to be directed to the ODE solver. Ignoring this information can obviously lead to much higher computation time or even to non-convergence.

5 Computational Results

In this section, we report on computational findings for the adapted thermodynamical model both for offline optimal control and in the NMPC context. The results presented in this section focus on the thermal management system without battery and attached vehicle model.

5.1 Problem Setup

The manipulated variables u_{compr} and u_{valve} of the reduced model are the rates of change of the compressor frequency and the area of the expansion valve. The controlled variables are the cabin temperature T_{cabin} and the superheating value of the heat pump T_{sh} . All results are generated with the same basic system model, counting a total number of 43 differential states, and with a time horizon of $h = 20$ s discretized by $N = 20$ shooting intervals.

The sole difference between the model used in the offline and the NMPC study is the incorporation of the ambient temperature as an additional pseudo-dynamic state in the NMPC controller's model ($\dot{T}_{amb} = 0$) while in the offline controller's model the temperature is merely a constant. An incorporation of this easily measureable quantity obviously is the more suited choice since it yields better predictions. However, since required derivative functions for the ambient temperature were not available this was omitted in the offline study for sake of comparability (see also §5.2).

The objective function, defined in Eq. (4), penalizes the differences between controlled variables and their set-points, as well as input changes. Moreover, a soft constraint formulation is chosen to avoid that the superheating temperature T_{sh} drops below the lower operational bound of $T_{sh, LB} = 4$ and that l_{sc} drops below $l_{sc, LB} = 0.05$:

$$\begin{aligned} \max(0, T_{sh, LB} - T_{sh}) &= 0.5 \cdot (T_{sh, LB} - T_{sh} + |T_{sh, LB} - T_{sh}|) \\ \max(0, l_{sc, LB} - l_{sc}) &= 0.5 \cdot (l_{sc, LB} - l_{sc} + |l_{sc, LB} - l_{sc}|) \end{aligned}$$

The final objective function including the two soft constraints reads as follows:

$$\ell(x(t), z(t), u(t), p) = \left(w_i^{-\frac{1}{2}} \Theta_i(\xi_i(t) - \bar{\xi}_i) \right)_{i=1, \dots, 6} \quad (4)$$

wherein $\xi^T = (T_{cabin}, T_{sh}, u_{compr}, u_{valve}, -T_{sh}, -l_{sc})$ and with weights

$$w^T = \left(\frac{10^4}{292}, \frac{10}{5}, \frac{10^{-5}}{72.5}, \frac{10^{-5}}{4.5 \cdot 10^{-7}}, \frac{10^4}{8}, \frac{10^4}{0.1} \right)^T$$

chosen such that denominators normalize quantities to 1 and numerators indicate relative weights. The set-point is

$$\bar{\xi}^T = (295.15, 5, 0, 0, -4, -0.05)^T.$$

The functions Θ_i are $\Theta_i(x) = \text{Id}$ (identity) for $i = 1, \dots, 4$ and $\Theta_i(x) = x \cdot H(x)$ (Heaviside integral) for $i = 5, 6$. To guarantee smoothness assumptions, the Heaviside integral function $x \cdot H(x)$ is exponentially smoothed in computational practice. The end-point term is

$$e(x(T), z(T), p) = \left(w_i^{-\frac{1}{2}} (\xi_i(T) - \bar{\xi}_i) \right)_{i=1, \dots, 2}$$

for $\xi^T = (T_{cabin}, T_{sh})^T$, identical set point, and weights

$$w^T = \left(\frac{10^6}{292}, \frac{50}{5} \right)^T.$$

5.2 Performance Comparison of Different Jacobian Methods

The performance of the online optimizing controller critically depends on the choice of the method to generate Jacobians. Fundamentally, derivatives can be computed by automatic differentiation (AD) or numerically by a finite difference scheme (ND). The FMUs generated by Dymola use numerical Jacobians by default. By setting the flag *Advanced.GenerateAnalyticJacobian*, Dymola can be configured to generate analytic Jacobians and include them in the FMU. For this to be effective, it is necessary that every function used in the Modelica model also declares a corresponding derivative function. For the given system, this required the use of a tailored version of the fluid database TILMedia supplying derivative functions for a wide range of material-dependent functions. MUSCOD can be configured to use Jacobians, which are provided from "outside", e.g. the FMU (numerical or analytical), or to approximate Jacobian matrices numerically by its built-in finite difference scheme.

This section's numerical study compares these three generation methods for Jacobians on the basis of respective offline optimization runs. In the scenario, an instantaneous step change of the ambient temperature of 5 K is applied to a stationary system state (compare also the "online" study at $t = 600$ s). To investigate effects on precision the integration tolerance was set to 10^{-9} .

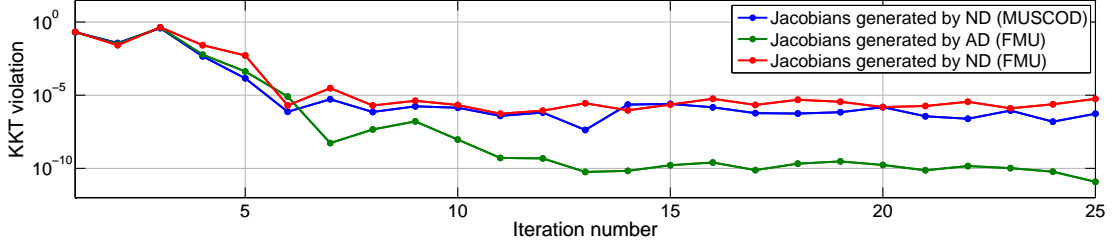


Figure 6. After 25 SQP iterations, the remaining KKT violation during offline optimization is smallest (best) when using AD Jacobians. ND Jacobians provided by MUSCOD are runner-up. ND Jacobians provided by the Modelica FMU perform worst.

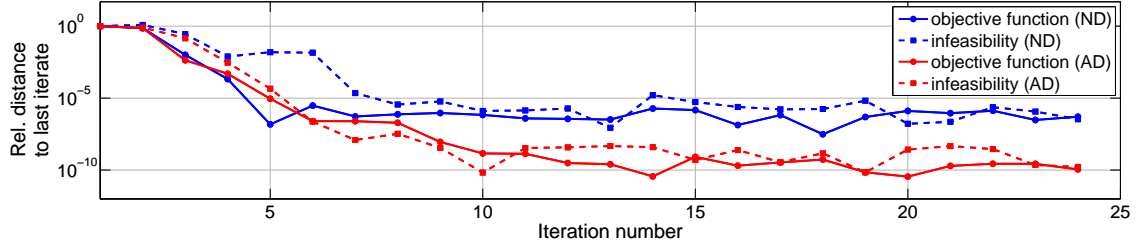


Figure 7. A self-convergence plot of objective function values and infeasibility measures reveals convergence after 10 SQP iterations regardless of the choice of method for generating Jacobians.

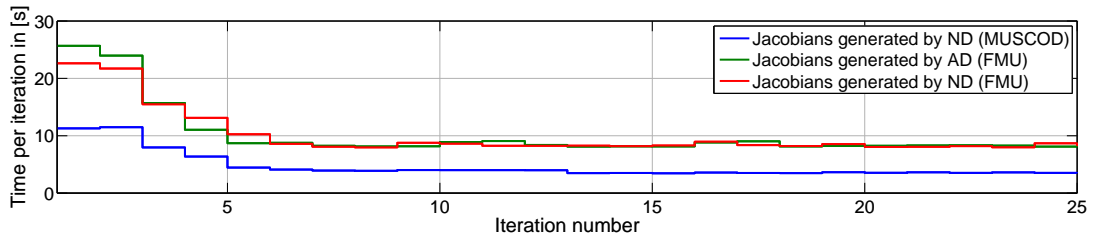


Figure 8. AD Jacobians from the Modelica FMU are computationally more expensive than MUSCOD ND Jacobians. Measured time refers to an integration tolerance of 10^{-9} .

Fig. 6 assesses the impact of the Jacobian generation method on the convergence behavior of MUSCOD in terms of the remaining KKT violation. As expected, AD yields a solution with the highest precision. Fig. 7 shows a self-convergence plot for the objective function value and the infeasibility measure, i.e., we show for iterations $k = 0, \dots, N-1$ the fractions

$$|\chi_m^{(k)} - \chi_m^{(N)}| / (\chi_m^{(0)} - \chi_m^{(N)})$$

of objective function ($m = 0$) and infeasibility ($m = 1$):

$$\chi_0^{(k)} = \Phi(s^{(k)}, q^{(k)})$$

$$\chi_1^{(k)} = \sum_i [c_i(s^{(k)}, q^{(k)})] + \sum_j [d_j(s^{(k)}, q^{(k)})]^{-}$$

As can be seen, after six SQP iterations convergence has essentially been achieved regardless of the chosen method. As Fig. 8 reveals the MUSCOD internal finite difference scheme is, to our surprise, faster than both of the FMU Jacobian generation methods. Thus, all following numerical results on NMPC were obtained using this Jacobian generation scheme.

5.3 Comparison of PI Control and NMPC

In this section two tuned PI-controllers are compared to the developed NMPC controller. The parameters for the PI-controller were determined using a step response of the system and were manually tuned to a normal and more aggressive behavior. The scenario used for this purpose consists of a transient heat-up, starting from a steady state, at an ambient temperature of 5°C and of a following abrupt ambient temperature change of $+5^\circ\text{C}$, which is applied at $t = 600\text{ s}$ after reaching steady state again, see Fig. 9.

The resulting controlled variables are shown in Fig. 10 (cabin temperature) and in Fig. 12 (superheating value). The corresponding manipulated variables are plotted in Fig. 11 (compressor frequency) and in Fig. 13 (expansion valve area). In contrast to §5.2 the integration tolerance is now set to 10^{-5} . Based on experience this is a sufficient value for this application.

Transient Heat-Up of Passenger Cabin We observe, that the NMPC curve rises and settles significantly faster than both PI-controllers and without any temperature overshoot indicating a very efficient control system (cf.

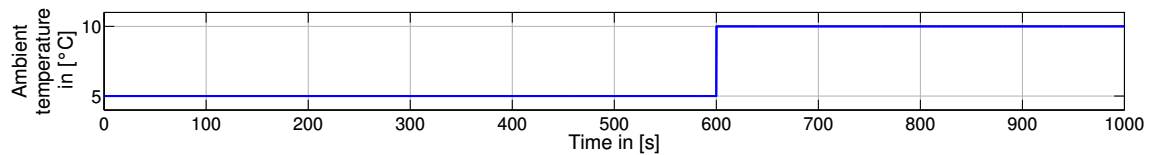


Figure 9. Ambient temperature step of +5K.

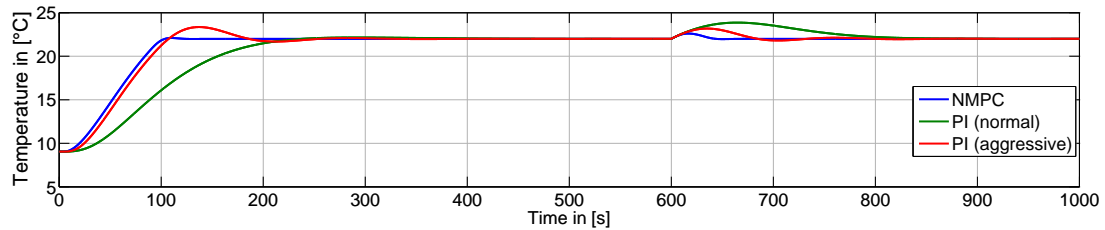


Figure 10. Passenger cabin temperature during scenario (transient heat-up and temperature step) with different controllers.

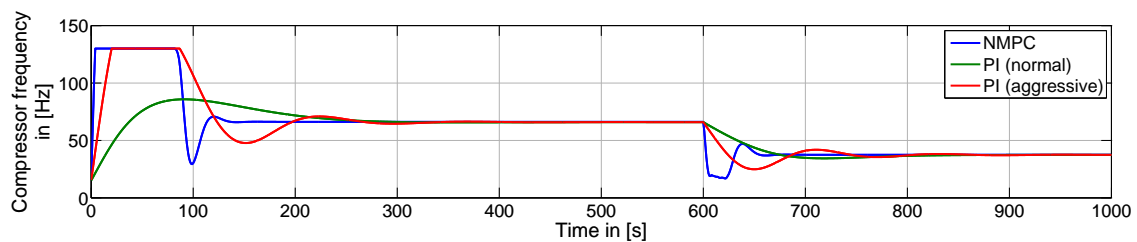


Figure 11. Frequency of the compressor during scenario (transient heat-up and temperature step) with different controllers.

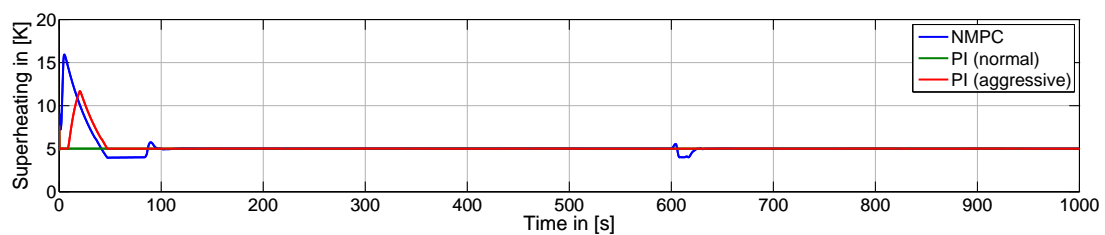


Figure 12. Superheating value during scenario (transient heat-up and temperature step) with different controllers.

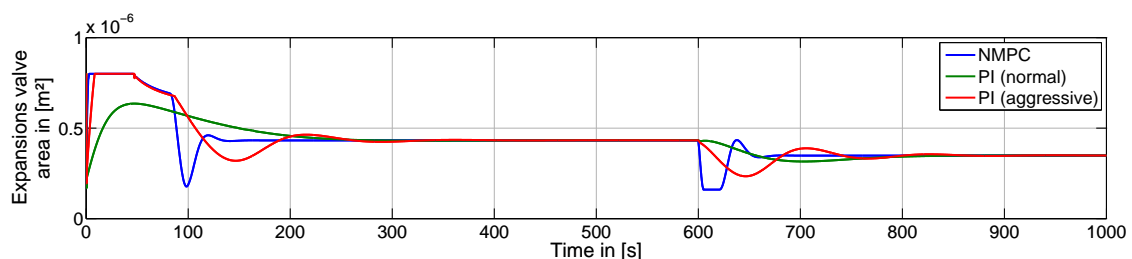


Figure 13. Expansion valve area during scenario (transient heat-up and temperature step) with different controllers.

Fig. 10). The compressor frequency and expansion valve reach their upper bounds faster and, subsequently, stabilize the system to a steady-state far more rapidly with NMPC (cf. Fig. 11 and Fig. 13). The predictive character of NMPC taking into account the system's thermal inertia can be identified on the basis of the compressor frequency already being decreased at a temperature well below 20°C. Fig. 12 depicts the effect of the rather weak "tracking" weighting for the superheating temperature in the NMPC

objective function and the corresponding soft-constraint to ensure a lower operational bound of 4 K (cf. §5.1). Superheating value fluctuations are acceptable unlike violations of the operational limits, which are prevented here by the NMPC approach. On the basis of Tab. 1, showing criteria for controller performance as overshoot, settling time and rising time, we can eventually state, that NMPC combines the fastest and most efficient way to heat up the cabin temperature to its set value.

	NMPC	PI (norm.)	PI (aggr.)
Rising time [s]	78.2	162.7	80.4
Settling time [s]	95	194.6	173.1
Overshoot [%]	0.4	1.25	10.50

Table 1. The NMPC controller's characteristics during transient heat-up significantly outperform PI control.

Reaction to Disturbances At $t = 600$ s the controllers can be compared when dealing with a disturbance, here, an ambient temperature step of $+5^\circ\text{C}$. On the basis of Fig. 10, we conclude that the NMPC shows the best behavior, with a very small amplitude and a very short time interval before the steady state is reached again. This is not due to the predictive character of NMPC, as the temperature change is not known in advance. The NMPC controller gets the information about a temperature change along with the measured state vector at $t = 600$ s. Again, the weak weighting of the superheating value can be observed in Fig. 12, as the superheating value is affected by the disturbance in the case of the NMPC-controller.

CPU Time The computations were performed on a workstation using a single core of an Intel Xeon CPU at 3.5 GHz. To guarantee real-time feasibility for future application in a vehicle, it is necessary that the duration of feedback and prepare phase is shorter than the chosen sampling time (1 s here). If this can be ensured, the feedback phase duration is the time delay between the measurement of the system state and the availability of new values for the manipulated variables. Naturally, a short duration is essential to make sure the applied feedback relies on up-to-date system state information.

Fig. 14 shows a graph of the CPU time consumed by both phases. The real-time feasibility limit is indicated by a dotted red line and mostly not exceeded in the scenario. However, three CPU time peaks within the transient heat-up phase and the abrupt temperature change phase still violate the limit. The peak at the beginning is due to a cold start of the NMPC controller. Since we start stationarily this peak could be avoided by the execution of sufficiently many SQP iterations before a respective warm-start of the NMPC controller. An investigation of the remaining peaking behavior must yet be carried out, i.e. whether it originates from the particular model implementation or is a general property of the system. In future work, the latter could be addressed algorithmically by introducing adaptive relinearization into the NMPC schemes using, e.g., multi-level schemes (Bock et al., 2005; Kirches et al., 2012) or mixed-level schemes (Frasch et al., 2012). Fig. 15 shows a graph of the CPU time consumed by the feedback phase only. It is in the order of 0.5 – 2 milliseconds, which is a near instantaneous response to the measured system state relative to the system dynamics time scale. The feedback phase CPU time rises only very mildly during transient phases, and remains satisfyingly low throughout.

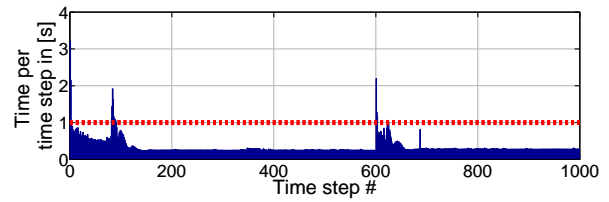


Figure 14. CPU time in seconds per NMPC iteration, consumed by all three phases and including FMU evaluation calls during the preparation phase. Measured time refers to an integration tolerance of 10^{-5} .

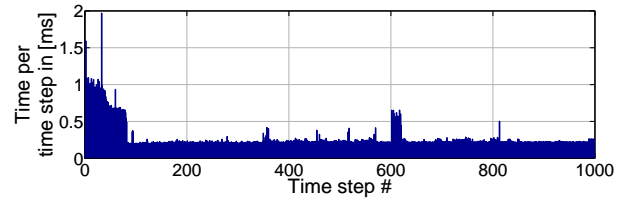


Figure 15. CPU Time in milliseconds per NMPC iteration consumed by the feedback phase only.

6 Conclusion & Outlook

The article discusses the development of an NMPC setup for a thermal management system of electrified vehicles. Compared to conventional PI control, several advantages concerning the transient heat-up and in reaction to disturbances were noted. The NMPC reaches the set-point value and settles considerably faster, nearly without any overshoot. This indicates an overall high degree of energy efficiency. Also, NMPC reacted faster on external disturbances that were not known in advance. A further benefit is the safe operating mode, as each state variable of the system can be constrained and constraints remained satisfied throughout all experiments. In the context of the heat pump application, the superheating value could be kept at a safe distance from the dew line in every operating point. The only observed drawback was the comparably high development effort that was necessary for developing the model and deploying NMPC for the system at hand.

The Functional Mockup Interface turned out to be a convenient way to export a previously developed Modelica model and to use it within the optimizer. To use the developed Modelica simulation model also for optimal control, though, several adaptations concerning smoothness assumptions were necessary. Although the optimizer could not be given access to the whole differential-algebraic equation (DAE) system due to intrinsic limits of the current version 2.0 of the FMI standard, the derivative-based optimization was found to work satisfactorily for the model at hand. A direct implementation of the DAE system in MUSCOD still promises a significant future increase in performance and numerical stability. Finally, we expected a higher impact of using analytic Jacobians provided to the optimizer by the FMU. Without insight into the auto-generated source code from which the FMU was

compiled, we could not rigorously answer the question of why the analytic Jacobian provided by the FMU is about twice as slow as the comparably simple one-sided finite difference approximation method we used in conjunction with MUSCOD.

Outlook. This article focused on controlling the cabin temperature through the use of NMPC. In a further step, the temperatures of cabin and battery will be tracked in parallel and the temperature of the electric components of the powertrain will be restricted to a realistic thermal range. The evaluation of the complete system can then be carried out on the basis of driving cycles.

Right now the NMPC controller still receives the entire measured state vector from the simulation. This is not realistic after deploying the controller to the final hardware application, as only a subset of the system state can be measured in reality. Thus, an observer will be employed to estimate the states that are not physically measurable.

In a second step, the hardware application will be targeted, where the whole developed thermal management system in an electric vehicle is controlled by the NMPC. The optimization must prove real-time feasibility to guarantee a solution within the defined time interval under all circumstances. The direct implementation of the DAE system in MUSCOD might prove to be essential to this end.

Acknowledgements. The authors acknowledge support by DFG Graduate School 220 and the Institutional Strategy of Heidelberg University funded by the German Excellence Initiative, and by the German Federal Ministry of Education and Research program “Mathematics for Innovations in Industry and Service 2013–2016”, grant n° 05M2013-GOSSIP. This publication was also written in the framework of the Profilregion Mobilitätssysteme Karlsruhe, which is funded by the Ministry of Science, Research and the Arts and the Ministry of Economic Affairs, Labour and Housing in Baden-Württemberg and as a national High Performance Center by the Fraunhofer-Gesellschaft. The authors are grateful to the TILMedia team at TLK-Thermo GmbH in Braunschweig for kindly providing access to a specialized version that supplies derivatives.

References

- A. Afram and F. Janabi-Sharifi. Theory and applications of HVAC control systems: A review of model predictive control (MPC). *Building and Environment*, 72:343–355, 2014.
- J. Albersmeyer. *Adjoint based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems*. PhD thesis, Heidelberg University, 2010.
- I. Bauer, H.G. Bock, and J.P. Schlöder. DAESOL – a BDF-code for the numerical solution of differential algebraic equations. Internal report, IWR, SFB 359, Heidelberg University, 1999.
- T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauss, H. Elmquist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidholdt, D. Neumerkel, H. Olsson, J.-V. Peetz, and S. Wolf. The functional mockup interface for tool independent exchange of simulation models. *8th Int. Modelica Conf.*, 2011.
- H.G. Bock and K.J. Plitt. A Multiple Shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9th IFAC World Congress*, pages 242–247, Budapest, 1984. Pergamon Press.
- H.G. Bock, M. Diehl, P. Kühn, E. Kostina, J.P. Schlöder, and L. Wirsching. Numerical Methods for Efficient and Fast Nonlinear Model Predictive Control. In R. Findeisen, F. Allgöwer, and L. T. Biegler, editors, *Assessment and future directions of Nonlinear Model Predictive Control*, volume 358 of *LNCIS*, pages 163–179. Springer, 2005.
- J. Bonilla, S. Dormido, and F. E. Cellier. Switching moving boundary models for two-phase flow evaporators and condensers. *Communications in Nonlinear Science and Numerical Simulation*, 20:743–768, 2015.
- L. del Re, F. Allgöwer, L. Glielmo, C. Guardiola, and I. Kolmanovsky. *Automotive Model Predictive Control*. Springer, 2010.
- M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, Universität Heidelberg, 2001.
- H. Esen, T. Tashiro, D. Bernardini, and A. Bemporad. Cabin heat thermal management in hybrid vehicles using model predictive control. *22nd Med. Conf. Contr. Autom. (MED)*, 2014.
- T. Fischer, F. Götz, L. Berg, H.-P. Kollmeier, and F. Gauterin. Model-based development of a holistic thermal management system for an electric car with a high temperature fuel cell range extender. *11th Int. Modelica Conference*, 2015.
- R. Franke. Formulation of dynamic optimization problems using modelica and their efficient solution. *2nd International Modelica Conference*, pages 315–323, 2002.
- J.V. Frasch, L. Wirsching, S. Sager, and H.G. Bock. Mixed-Level Iteration Schemes for Nonlinear Model Predictive Control. In *Proc. IFAC Conf. on NMPC*, 2012.
- M. Gräber, C. Kirches, D. Scharff, and W. Tegethoff. Using functional mock-up units for nonlinear model predictive control. *9th International Modelica Conference*, 2012.
- J.M. Jensen and H. Tummescheit. Moving boundary models for dynamic simulations of two-phase flows. *2nd International Modelica Conference*, pages 235–244, 2002.
- A.Y. Karnik, A. Fuxman, P. Bonkoski, M. Jankovic, and J. Pekar. Vehicle powertrain thermal management system using model predictive control. *SAE International*, 2016.
- C. Kirches. Fast Numerical Methods for Mixed-Integer Nonlinear Model-Predictive Control. In H.G. Bock, W. Hackbusch, M. Luskun, and R. Rannacher, editors, *Advances in Numerical Mathematics*. Springer Vieweg, Wiesbaden, July 2011.
- C. Kirches, L. Wirsching, H.G. Bock, and J.P. Schlöder. Efficient Direct Multiple Shooting for Nonlinear Model Predictive Control on Long Horizons. *J. Proc. Contr.*, 22(3):540–550, 2012.
- C. Kirches, H.G. Bock, J.P. Schlöder, and S. Sager. Mixed-integer NMPC for predictive cruise control of heavy-duty trucks. In *European Control Conference*, pages 4118–4123, Zurich, Switzerland, July 17–19 2013.
- D.B. Leineweber, I. Bauer, A.A.S. Schäfer, H.G. Bock, and J.P. Schlöder. An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization (Parts I and II). *Comp. Chem. Eng.*, 27:157–174, 2003.
- J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Verlag, Berlin Heidelberg New York, second edition, 2006.

Defining and Solving Hybrid Optimal Control Problems with Higher Index DAEs

Radosław Pytlak¹ Damian Suski² Tomasz Tarnawski³ Tomasz Zawadzki⁴

¹Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland,

`r.pytlak@mini.pw.edu.pl`

²Institute of Automatic Control and Robotics, Warsaw University of Technology, Poland,

`{d.suski,t.zawadzki}@mchtr.pw.edu.pl`

³Department of Quantitative Methods and Information Technology, Kozminski University, Poland,

`ttarnawski@kozminski.edu.pl`

⁴Research and Academic Computer Network (NASK), Poland,

`tomasz.zawadzki@nask.pl`

Abstract

The paper deals with optimal control problems defined for hybrid systems described by higher index DAEs. We present a prototype solution that supports the whole process from defining such problem to solving it and presenting results. Problem's definition is done with Dynamic Optimization Modeling Language (DOML) which is based directly on Modelica. The proposed numerical procedure for solving the problems of interest has the following features: 1) it is based on the appropriately defined adjoint equations formulated for the discretized equations being the result of the numerical integration of system equations by an implicit Runge–Kutta method; 2) initialization for higher index DAEs is performed with the help of Pantelides' algorithm; 3) it does not require the system to be transformed to ODEs (through differentiation of some algebraic equations).

The paper presents numerical examples related to hybrid systems described by index three DAEs, showing the validity of the proposed approach. All software components needed to carry out the computations, i.e. the code editor, compiler, numerical libraries and GUI for presenting results are prepared as parts of a combined platform: Interactive Dynamic Optimization Server (IDOS).

Keywords: hybrid systems, optimal control problems, higher index DAEs

1 Introduction

The paper presents recent development of solver functionality implemented within DOML (Dynamic Optimization Modeling Language) environment and deployed as part of the IDOS (Interactive Dynamic Optimization Server, described in (Pytlak et al., 2014), see also (Pytlak et al., 2013)) infrastructure. It uses a numerical procedure based on control vector parameterization and RADAU5 together with event (discrete state transition) handling and is capable of solving optimal control problems for hybrid, high-index DAEs.

The DOML language (introduced in (Pytlak et al., 2014)) was devised as an extension of Modelica towards defining optimal control problems for systems described with Modelica language – quite analogically to Optimica (proposed earlier, in (Åkesson, 2007), see also (Åkesson, 2008)). In fact, the DOML compiler environment is heavily based on the open source Modelica (and Optimica) compiler environment JModelica.org (see e.g. (Åkesson et al., 2009)). During the efforts of adapting Optimica for the purpose of deploying it within IDOS environment a conclusion was reached to redesign some of its optimization-related constructs, as its original design brought in some troublesome limitations (details can be found e.g. in (Pytlak et al., 2013) and (Tarnawski and Pytlak, 2014)). To avoid confusion with Optimica, we then chose to refer to the language as DOML. Although the framework is (eventually) intended to be fully compatible with Modelica, the current development efforts are focused strictly on building prototype, proof-of-concept implementations of advanced optimization algorithms. Therefore, ensuring DOML's wide and flawless compatibility with Modelica syntax (and MSL models in particular) has to wait for its turn (still, being based on JModelica.org environment, DOML is in the position to enjoy a fair deal of compatibility inherited 'in the package'). Up to this point several different optimization algorithms and solver libraries have already been implemented (see Table 1 in (Pytlak et al., 2014)): e.g. solvers for optimal control problems with ODEs based on *a priori* discretization of system equations (HQP package), solvers that use adjoint equations and do not require *a priori* discretization of equations, solvers based on multiple shooting methods, solvers for control problems with integer valued controls which use BONMIN package, solvers that use integration procedures from SUNDIALS package and IPOPT as the optimization engine.

The new algorithm implementation presented here is designed to solve optimal control problems defined by hybrid systems, i.e. systems with mixed discrete-continuous

dynamics (van der Schaft and Schumacher, 2000), whose dynamics are described with high-index DAEs. It must be noted, that even separately each of these categories poses nontrivial difficulties for optimization and even for simulation alone. In particular:

- higher index DAE systems need the procedure for the automatic consistent initialization, the integration procedure for higher index DAEs and the optimization solver equipped with the procedure for evaluating gradients of functionals defining optimization problems
- hybrid systems require the procedure for the accurate location of discrete transition times, the integration procedure with an automatic handling of discrete dynamics and the optimization solver equipped with the procedure for evaluating gradients of optimization problem functionals, which is dedicated for a hybrid system dynamics.

The authors are not aware of any implementations (other than the one introduced in (Pytlak, 2011)) of a dynamic optimization algorithm capable of reliably solving high-index DAE problems without an application of the index reduction procedure. Similarly, the authors are not aware of an established and widely-used implementation of a dynamic optimization algorithm applicable to hybrid systems. In particular, the optimization algorithms implemented in environments featuring Optimica (JModelica.org, OpenModelica) cover the problems described by ODEs (or DAEs with the help of the index reduction) but not hybrid systems.

The paper is organized as follows. In section 2 we outline the features of DOML and in particular the language constructs used to express the problems of interest. They are then used in section 3 to define two simple exemplary problems used to test and illustrate capabilities of the implemented solver. Trajectories obtained by solving these problems are also presented. The following two sections are devoted to formal description and analysis of the algorithm. First, in section 4, a formal definition of hybrid optimal control problem is laid out (and used to formally re-define the example problems). Then, based on that, section 5 discusses the most important mathematical and implementational details of the proposed optimization procedure. The paper is wrapped up with short concluding remarks.

2 Hybrid optimal control problem definition in DOML

The provisions of DOML that differentiate it against Optimica were already described in earlier works (e.g.: (Pytlak et al., 2014), (Tarnawski and Pytlak, 2014) or (Pytlak et al., 2013)). To avoid excessive repetition, here they are only listed very briefly:

- new keywords, `minimize` and `maximize` were introduced making it possible to have meaningfully named optimized parameters with the direction of optimization chosen by the user. It also became possible to specify multicriteria optimization problems.
- `annotation(solver)` was used to allow the user to specify the algorithm to be used to solve the given problem, along with its runtime settings. A preliminary procedure of choosing the most appropriate solver, based on elements detected in the problems's definition, was also implemented.
- a mechanism for labeling equations and constraints was introduced, by means of which the adjoint variables (for equations) and Lagrange multipliers (for constraints) could be referred to. Implementation of some solvers required that functionality.
- decision variable's `InitialGuess` attribute was re-defined with *continuous* variability, so that it became possible for initial guess to be defined as a signal changing over time. This was developed particularly with 'chaining of solvers' in mind: an approximate solver could be run first, then the solution obtained could be used to warm-start another solver – a more exact one, yet also more fussy with respect to the starting point.

For the most part, these earlier provisions were well geared for formally defining the optimal control problems of interest. In the case of hybrid systems, however, we found it necessary to devise specific syntactic constructs to fully express the system's dynamics. The resulting 'canonical form' description of a hybrid system is structured around:

- singular `enumeration` type variable whose values range over possible discrete states of the system;
- compound `if elseif` statement containing equations defining the dynamics specific for each state;
- embedded `when` clauses specifying *transition guards*—conditions for exiting the current state and the new state reached with the transition;
- optional `reinit` operator used to define any potential discontinuous behavior upon a change-of-state event.

In result, the description of a hybrid system may take the form along the lines drawn on Listing 1. Please note, that the presented code is meant as a mere example and does not necessarily reflect sensible dynamics of any actual system.

In a general case the state-transition conditions (guards) between a pair of states do not have to be the same for both directions. One good example, when this is certainly not the case, is hysteresis (in the example shown: between

Listing 1. Example of DOML code defining a hybrid system

```

optimization Hybrid_ex(startTime = 0.0,
                        finalTime = 1.0)
  minimize Real objective = x3(finalTime);
  ... // decl's of vars (x) and inputs (u)
  type Q = enumeration(A, B, C);
  discrete Q q(start = Q.A); // the state
equation
  ... // equations common to all states
  if q == Q.A then
    der(x1) = -x1 + x2 + 2*u;
    when x1 < -1 then //transition A->B
      q = Q.B;
    end when;
  elseif q == Q.B then
    der(x1) = -2*x1 + x2 + u;
    when x1 > 1 then //transition B->A
      q = Q.A;
    end when;
  else // q == Q.C
    der(x1) = -x2 + u;
    when x2 < 0 then //bounce-back C->C
      q = Q.C;
      reinit(x3, -x3);
    elseif x2 > 10 then //transition C->A
      q = Q.A;
    end when;
  end if;
end Hybrid_ex;

```

states A and B). In addition, with the proposed construct, it is possible to define transition from a state onto itself (in the example: happens in C) which is applicable for instance in the case of a ball bouncing off a wall, back into the realm of the same dynamics, but with it's velocity (abruptly) altered (hence the `reinit` placed in the example).

3 Examples

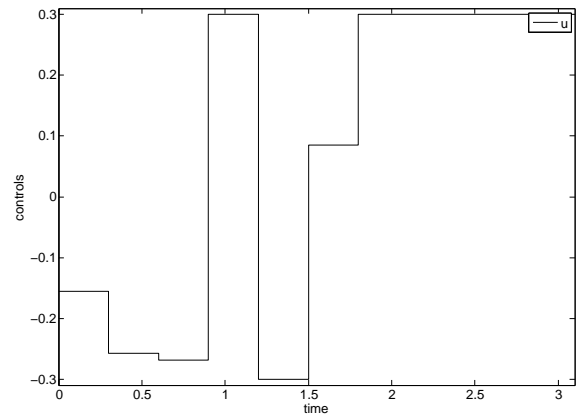
Below, we present the results of applying the presented algorithm to solving two optimal control problems based on hybrid systems. They have been previously discussed in (Pytlak and Suski, 2017), but here we show results for different versions of these problems. Please note, that the purpose of these examples is purely illustrative within the discussion of capabilities of the implemented solver and they do not necessarily represent sensible problems of real engineering importance or application. Also, as noted earlier, the algorithm's implementation is at the stage of proof-of-concept and has only been tested on cases with limited complexity, where the whole problem is defined in one stand-alone input file. Defining optimization problems through models constructed with MSL components was not tested (however, the compiler is build on top of the JModelica.org compiler, which itself provides a substantial MSL support).

Below, we define the exemplary optimal control problems solely by means of DOML code. In the following

section, after introducing the necessary formalism, we restate them in a mathematically formal way.

Example 1. It is an optimal control problem of a non-standard pendulum described by DAEs with index three (van der Schaft and Schumacher, 2000), in Cartesian coordinates x_1 and x_2 . On the vertical axis there is a fixed pin interfering with pendulum's string and effectively halving its length when $x_1 \geq 0$. For $x_1 \leq 0$ the pendulum swings with its original length. There is no jump in differential state variables during state transition. The control variable u represents force applied horizontally to the pendulum's end (constrained above by 0.3, in either direction). The objective of the control is to cause the system to reach such final state in which the pendulum's position $x_1(t_f)$ is as close as possible to the neutral point, while its velocity component $v_2(t_f)$ is equal to 0.06. The DOML definition of the problem is presented in Listing 2.

After 10 iterations optimality conditions were satisfied with accuracy 10^{-6} while equality constraint $v_2(t_f) = 0.06$ with accuracy 10^{-7} . The obtained optimal trajectories are given in Figure 2 while the optimal control in Figure 1 – as it turns out, it exhibits a relatively complex switching structure.

**Figure 1.** The pendulum example—optimal control.

Example 2. The second example is the popular bouncing ball problem discussed in several papers e.g. (van der Schaft and Schumacher, 2000). The ball bounces off the fixed surface level at $x_1 = 0$ with the opposite velocity (assuming no energy loss, i.e. the coefficient of restitution is 1). The system has only one discrete state. The control variable u is a force applied vertically to the ball (with the constraint that it cannot exceed the value of 2.5, either upwards or downwards). The objective is to end up (at $t_f = 1$) with the ball being at the height of 0.5 and having minimum velocity (in terms of its absolute value). The DOML script of the problem is presented in Listing 3.

Applying the SQP code resulted, after 23 iterations, in an approximate optimal solution given in Figure 3. The

```

package Pendulum

  optimization Pendulum_opt(startTime = 0,
    finalTime = 3.0)

    minimize Real obj = x1(finalTime)^2;

    type states =
      enumeration(short, normal);
    discrete states State;
    parameter Real p = 0.5;
    ... // decl's of vars (x1, x2, ... L)
    input Real u1(min=-0.3, max=0.3,
      initialGuess=0.0);

  initial equation
    if x1 >= 0.0 then
      State = states.normal;
    else
      State = states.short;
    end if;

  equation
    der(x1) = v1;
    der(x2) = v2;
    der(v1) = w1;
    der(v2) = w2;

    if State == states.normal then
      0 = w1+x1*L-u1;
      0 = w2+1.0+x2*L;
      0 = x1*x1+x2*x2-1.0;
      when x1 < 0.0 then
        State = states.short;
      end when;

    else
      0 = w1+x1*L/p-u1;
      0 = w2+1.0+(x2+1.0-p)*L/p;
      0 = x1*x1+(x2+1.0-p)*(x2+1.0-p)-p*p;
      when x1 > 0.0 then
        State = states.normal;
      end when;

    end if;

  constraint
    c1: v2(finalTime) = 0.06;

end Pendulum_opt;

end Pendulum;
    
```

Listing 2. The DOML file of the pendulum problem.

```

package Bouncing
  optimization Bouncing_opt (startTime =
    0.0, finalTime = 1.0)

    minimize Real obj = x2(finalTime)^2;

    type states = enumeration(normal);
    discrete states State;

    parameter Real c = 5.0;
    Real x1(start = 1.0);
    Real x2(start = 0.0);

    input Real u(min=-2.5,max=2.5,
      initialGuess=0.001);

  initial equation
    State = states.normal;

  equation
    der(x1) = x2;
    der(x2) = -c + u;

    if State == states.normal then
      when x1 < 0.0 then
        State = states.normal;
        reinit(x2, -pre(x2));
      end when;
    else
      State = states.normal;
    end if;

  constraint
    c1: x1(finalTime) = 0.5;

end Bouncing_opt;

end Bouncing;
    
```

Listing 3. The DOML file of the bouncing ball problem.

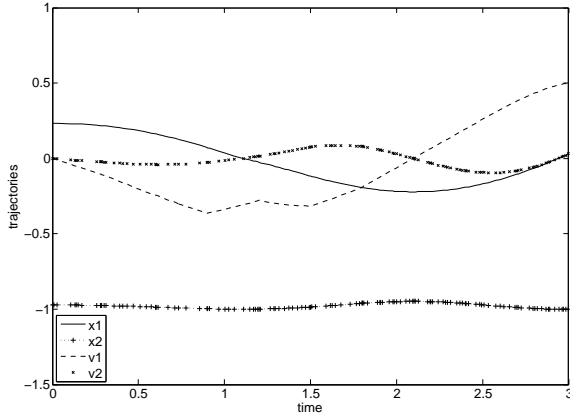


Figure 2. The pendulum example—optimal trajectories.

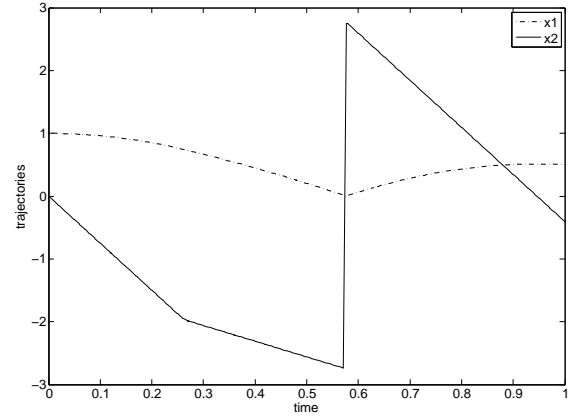


Figure 4. The bouncing ball example—optimal trajectories.

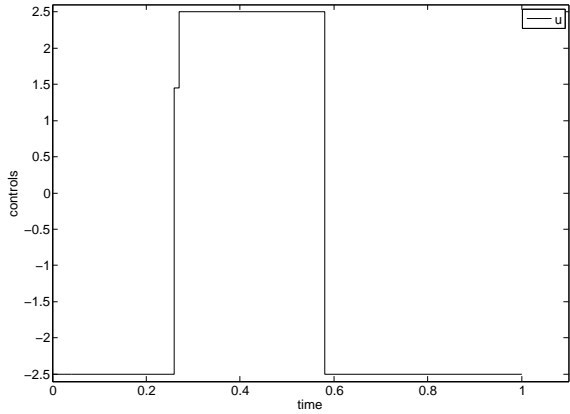


Figure 3. The bouncing ball example—optimal control.

optimality conditions and equality constraint $x_1(t_f) = 0.5$ were satisfied with accuracy 10^{-6} . Some of the final trajectories are illustrated in Figure 4. At final time the ball is at the required height, and has a relatively small velocity, $|x_2(t_f)| \approx 0.3$.

4 Formal discussion of hybrid systems

Hybrid systems are systems with mixed discrete-continuous dynamics (van der Schaft and Schumacher, 2000). In this work we use the formal definition of a hybrid system based on the one given in (Suski and Pytlak, 2016), which is similar to many other definitions given in the literature e.g. (Lygeros et al., 1999), (van der Schaft and Schumacher, 2000), (Shaikh, 2004). We restrict our analysis to systems with autonomous transitions.

A hybrid system \mathcal{H} is a tuple

$$\mathcal{H} = (\mathcal{Q}, \mathcal{U}, \mathcal{I}, \mathcal{F}, \mathcal{T}, \mathcal{G}, \mathcal{J}) \quad (1)$$

where

- \mathcal{Q} is a finite set of discrete states. Its elements are denoted by q .
- \mathcal{U} is a set of admissible controls. The elements of \mathcal{U} are measurable functions $u : I \rightarrow U$, where I can be any closed interval of \mathbb{R} and U is a fixed subset of \mathbb{R}^n .
- \mathcal{I} is a function which assigns to every discrete state q a set

$$\mathcal{I}(q) = \{x \in \mathbb{R}^n : \psi_q(x) \leq 0\}, \quad \psi_q : \mathbb{R}^n \rightarrow \mathbb{R}^{n_{\psi_q}} \quad (2)$$

such that as long as a hybrid systems is in a discrete state q the continuous state trajectory x stays in $\mathcal{I}(q)$. We therefore say that $\mathcal{I}(q)$ is an *invariant set* for a discrete state q .

- \mathcal{F} is a function which assigns to every discrete state q a function $F_q : \mathbb{R}^n \times \mathcal{I}(q) \times U \rightarrow \mathbb{R}^n$ such that in a discrete state q the continuous state evolves according to a differential-algebraic equation

$$F_q(\dot{x}, x, u) = 0. \quad (3)$$

- \mathcal{T} is a subset of $\mathcal{Q} \times \mathcal{Q}$, which collects all pairs of discrete states (q, q') such that the transition from a state q to a state q' is possible.
- \mathcal{G} assigns to each pair $(q, q') \in \mathcal{T}$ a subset of $\mathcal{I}(q)$ boundary such that when a continuous state trajectory is about to leave $\mathcal{I}(q)$ through its boundary $\partial \mathcal{I}(q)$ at a point $x_t \in \mathcal{G}(q, q') \subset \partial \mathcal{I}(q)$ a discrete state changes from q to q' . We call such an event a *transition* and \mathcal{G} plays a role of a *transition guard*.
- \mathcal{J} assigns to each pair $(q, q') \in \mathcal{T}$ a function $\chi_{qq'} : \mathcal{G}(q, q') \rightarrow \mathcal{I}(q')$ such that when a discrete state

changes from q to q' at a transition time instant t_i then a continuous state undergoes a jump according to

$$x(t_i^+) = \chi_{qq'}(x(t_i^-)). \quad (4)$$

Having the definition of hybrid systems we can formally state the optimal control problem with hybrid system as:

$$\min_u \phi(x(t_f)) \quad (5)$$

subject to the constraints:

$$h_i^1(x(t_f)) = 0, i \in E \quad (6)$$

$$h_j^2(x(t_f)) \leq 0, j \in I \quad (7)$$

$$u(t) \in \Omega \text{ a.e. } t \in T. \quad (8)$$

where the continuous dynamics of a hybrid systems is described by a system of higher index differential–algebraic equations (DAEs)

$$F_q(\dot{x}, x, u, t) = 0 \text{ a.e. } t \in T = [0, t_f] \quad (9)$$

which depends on the actual discrete state q .

We assume that $u(t) \in \mathcal{R}^m$, $x(t) \in \mathcal{R}^n$, I, E are finite sets of indices and Ω is a convex bounded subset of \mathcal{R}^m . A more general problem with parameters as decision variables, with an unspecified time horizon and with the state constraints could also be considered but for the simplicity of presentation we analyze the problem stated above.

We also assume that a solution to system (9) exists and is unique for any $u \in \mathcal{U}$ where

$$\mathcal{U} = \{u \in \mathcal{L}_1^m[T] \mid u(t) \in \Omega \text{ a.e. on } T\}, \quad (10)$$

and any consistent initial condition $x(0)$.

Example Problems restated

Having defined the above one can transcribe the earlier examples in a mathematically formal way. The hybrid optimal control problem from the first example (pendulum) may be stated as follows:

$$\min_{u \in \mathcal{U}} [x_1(t_f)^2] \quad (11)$$

subject to the constraints

$$v_2(t_f) - 0.06 = 0 \quad (12)$$

and

$$\dot{x}_1 = v_1 \quad (13)$$

$$\dot{x}_2 = v_2 \quad (14)$$

$$\dot{v}_1 = w_1 \quad (15)$$

$$\dot{v}_2 = w_2 \quad (16)$$

$$0 = w_1 + x_1 L / p - u \quad (17)$$

$$0 = w_2 + 1 + (x_2 + 1 - p)L / p \quad (18)$$

$$0 = x_1^2 + (x_2 + 1 - p)^2 - p^2 \quad (19)$$

with

$$\mathcal{U} = \{u \in \mathcal{L}_1^1[0, t_f] \mid -0.3 \leq u(t) \leq 0.3 \text{ a.e. on } [0, t_f]\}$$

and $t_f = 3.0$. The parameter p satisfies $p = 1$ for $x_1 \leq 0$ and $p = 0.5$ for $x_1 \geq 0$. The hybrid system has therefore two discrete states q^1 and q^2 with invariant sets: $\mathcal{J}(q^1) = \{(x_1, x_2, v_1, v_2, w_1, w_2, L) \in \mathcal{R}^7 : x_1 \leq 0\}$, $\mathcal{J}(q^2) = \{(x_1, x_2, v_1, v_2, w_1, w_2, L) \in \mathcal{R}^7 : x_1 \geq 0\}$.

In turn, the optimal control problem given as the second example (bouncing ball) can be formally defined as: The optimal control problem transcribed formally is as follows

$$\min_{u \in \mathcal{U}} [x_2(t_f)^2] \quad (20)$$

subject to the constraints

$$x_1(t_f) - 0.5 = 0 \quad (21)$$

and

$$\dot{x}_1 = x_2 \quad (22)$$

$$\dot{x}_2 = -5 + u \quad (23)$$

with

$$\mathcal{U} = \{u \in \mathcal{L}_1^1[0, t_f] \mid -2.5 \leq u(t) \leq 2.5 \text{ a.e. on } [0, t_f]\}.$$

and $t_f = 1.0$. The jump function is given by

$$\mathcal{J}(x(t_i^-)) = \begin{bmatrix} x_1(t_i^+) \\ x_2(t_i^+) \end{bmatrix} = \begin{bmatrix} x_1(t_i^-) \\ -x_2(t_i^-) \end{bmatrix}. \quad (24)$$

The system has only one discrete state q with invariant set defined by $\mathcal{J}(q) = \{x \in \mathcal{R}^2 : x_1 \geq 0\}$.

5 Numerical procedure

The utilized numerical procedure for solving hybrid optimal control problems include the mechanisms for

- numerical integration of system equations (3) between transitions
- proper handling of transitions—precise location of transition times and application of state jumps
- the calculation of adjoint variables for the evaluations of gradients of functionals defining the problem.

The numerical procedure for solving optimal control problems described by higher index DAEs was introduced in (Pytlak, 2011). For the completeness of a presentation the essential ingredients of the procedure are also discussed here.

Suppose that we want to integrate numerically the set of differential–algebraic equations

$$F(\dot{x}, x, u) = 0 \quad (25)$$

by an implicit Runge–Kutta method. If we denote by $x(k)$ the value of x calculated at the k th integration step (at a time t_k), then the next value $x(k+1)$ is evaluated by solving the set of nonlinear equations

$$F(\dot{x}_i(k+1), x(k) + h(k) \sum_{j=1}^s a_{ij} \dot{x}_j(k+1), u(k)) = 0, \quad (26)$$

$$x(k+1) - x(k) - h(k) \sum_{i=1}^s b_i \dot{x}_i(k+1) = 0, \quad (27)$$

where coefficients a_{ij} , b_i depend on a Runge–Kutta method.

In order to define state equations of the discrete time system (26)–(27) we introduce the state vector $X(k)$:

$$X(k) = \begin{bmatrix} \dot{x}_1(k) \\ \vdots \\ \dot{x}_s(k) \\ x(k) \end{bmatrix}, \quad (28)$$

then equations (26)–(27) become

$$\tilde{F}(X(k+1), X(k), u(k)) = 0. \quad (29)$$

System (29) is fully implicit and, under some nonsingularity assumption, can be expressed as explicit. If the Jacobian of \tilde{F} with respect to $X(k+1)$, denoted by \tilde{F}_{X+} , exists and is nonsingular for all $k = 0, \dots, N-1$, then from the Implicit Function Theorem there exists unique function φ such that

$$X(k+1) = \varphi(X(k), u(k)) \quad (30)$$

and

$$\tilde{F}(\varphi(X(k), u(k)), X(k), u(k)) = 0 \quad (31)$$

for $k = 0, \dots, N-1$.

Under differentiability assumptions imposed on \tilde{F} the function φ is differentiable with respect to $X(k)$ and $u(k)$ and we have

$$\varphi_X(k) = -[\tilde{F}_{X+}(k)]^{-1} \tilde{F}_X(k) \quad (32)$$

$$\varphi_u(k) = -[\tilde{F}_{X+}(k)]^{-1} \tilde{F}_u(k). \quad (33)$$

where $\tilde{F}_{X+}(k)$, $\tilde{F}_X(k)$, $\tilde{F}_u(k)$ are evaluated at a point $(X(k+1), X(k), u(k))$ and $\varphi_X(k)$, $\varphi_u(k)$ are evaluated at $(X(k), u(k))$.

If we consider the function

$$\hat{F}^0(u) = \phi(X^u(N)) \quad (34)$$

then its gradient can be calculated by referring to adjoint equations for the functional (34) and the system (30).

The adjoint equations for the functional (34) and the system (30) are considered, for example, in (Pytlak, 1999):

$$p(N) = \phi_X(X^u(N))^T \quad (35)$$

$$p(k) = \varphi_X(k)^T p(k+1), \quad (36)$$

for $k = 0, \dots, N-1$. The adjoint variables p are the means for the gradient evaluation according to the formula

$$\hat{F}_{u(k)}^0(u) = \varphi_u(k)^T p(k+1). \quad (37)$$

Using (32)–(33), the adjoint equations (36) and the formula (37) can be expressed without the knowledge of φ :

$$p(k) = -\tilde{F}_X(k)^T [\tilde{F}_{X+}(k)]^{-T} p(k+1) \quad (38)$$

$$\hat{F}_{u(k)}^0(u) = -\tilde{F}_u(k)^T [\tilde{F}_{X+}(k)]^{-T} p(k+1). \quad (39)$$

Eventually we have the viable formula for the gradient of $\hat{F}^0(u)$ provided that matrices $\tilde{F}_{X+}(k)$ are nonsingular.

For special (but widely used) forms of higher index DAEs the matrices $\tilde{F}_{X+}(k)$ are **nonsingular** provided that $h(k)$ are sufficiently small. For example in (Hairer et al., 1989) it is shown that the statement holds for DAEs in the Hessenberg form up to an index three and the RADAU IIA integration scheme. Therefore, for many higher index DAEs we have a valid technique for computing gradients of $\hat{F}^0(u)$ (and other functions involved in an optimal control problem).

For the numerical integration, our software utilizes the RADAU5 code, which implements the RADAU IIA scheme with number of stages $s = 3$. The coefficients of a method and its good numerical properties are described in (Hairer et al., 1989). The numerical integration of DAEs will succeed if initial states are consistent. The initialization problem is solved in two steps. In the first step, the system of equations, required for the consistent initialization, is formed. This step is carried out with the help of Pantelides' graph based procedure (Pantelides, 1988) together with symbolic differentiation implemented within JModelica.org compiler. In the second step the system of equations is solved with the help of IPOPT solver (Wachter and Biegler, 2006). The consistent initialization procedure is called at the initial time and at points at which control functions exhibit jumps.

RADAU5 code requires the preliminary analysis of higher index DAEs: the user has to identify variables which have so-called index 1, index 2 and index 3 property. These indices can be established during the consistent initialization process—in our approach that information is passed from the procedure which implements the Pantelides' algorithm into the integration procedure. As a result the user of our package does not have to specify indices of equations variables in the DOML script.

In hybrid systems, one needs a procedure, which locates the transition times. To complete this task, our procedure uses subroutines `drchek.f` and `droots.f` (Hiebert and Shampine, 1980). The subroutine `drchek.f` does the preliminary check on the presence of a transition point. Next, the subroutine `droots.f` is called to locate the root with the specified accuracy. The root finding problem is solved with the help of the interpolation procedure which uses Hermite polynomials.

Between transitions, the adjoint equations are solved with the help of a formula (36), the same as for non-hybrid systems. However, the situation at a transition time is different. Let us denote the system equations before transition as

$$X(k+1) = \varphi^1(X(k), u(k), h(k)) \quad (40)$$

and the system equations after transition as

$$X(k+1) = \varphi^2(X(k), u(k), h(k)). \quad (41)$$

In the above equations the dependence of system equations on $h(k)$ is explicitly stated. The reason for that is the following. The step sizes nearby a transition step k_t are not taken freely, but they are taken to satisfy the transition condition

$$\psi(X^-(k_t)) = 0 \quad (42)$$

where $X^-(k_t)$ denotes the state vector before jump. The value of a state after jump is determined by the equation

$$X^+(k_t) = \chi(X^-(k_t)). \quad (43)$$

Now to calculate the adjoint variables at a transition the following linear system of equations needs to be solved for variables $p(k_t)$ and π

$$p(k_t) + \pi(\psi_X(k_t))^T = (\chi_X(k_t))^T (\varphi_X^2(k_t))^T p(k_t+1) \quad (44)$$

$$p(k_t)^T \varphi_h^1(k_t-1) = p(k_t+1)^T \varphi_h^2(k_t). \quad (45)$$

The partial derivative $\varphi_h(k)$ is calculated with the formula

$$\varphi_h(k) = -[\tilde{F}_{X^+}(k)]^{-1} \tilde{F}_h(k) \quad (46)$$

analogical to formulas (32)–(33). The formulae (44)–(45) are derived in (Pytlak and Suski, 2017).

To solve the optimal control problem, we replace control functions by their piecewise constant approximations and follow the optimization procedure outlined below.

Optimization Procedure

1. Choose initial control u_0 and set the iterations number: $k = 0$.
2. For the control u_k , integrate system equations by calling consistent initialization procedure when necessary. Calculate the cost function and evaluate all constraint functions. Evaluate the adjoint equations for each function defining the optimal control problem and on that basis calculate the gradients of all functions involved in optimal control problem.
3. Perform the optimization step. If optimality conditions are satisfied with the assumed accuracy then STOP. Otherwise evaluate u_{k+1} , increase k by one and go to Step 2).

In our code the main optimization loop is handled by the SQP code described in (Pytlak, 1999). The optimization procedure requires solving QP subproblems at each iteration. The interior point method described in (Gertz and Wright, 2003) is used for that purpose.

6 Conclusions

The paper presents the most recent advancement in the DOML-IDOS environment, where a number of optimization algorithms can be used to solve optimal control problems specified in a language directly derived from Modelica (and Optimica). The latest numerical procedure, described here, implements an advanced algorithm for solving hybrid optimal control problems with higher index DAEs. The procedure is based on the RADAU5 program which is the implementation of an implicit Runge–Kutta method. The procedure uses variable stepsizes in order to locate precisely switching points. The procedure is based on the adjoint equations associated with the discretized equations being the result of system equations integration and does not require the transformation of higher index DAEs to ODEs by performing differentiations of some system equations.

So far, the development focus was placed strictly on implementing proof-of-concept, prototype solutions based on most recent dynamic optimization algorithms while ensuring wide Modelica compatibility, environment robustness, more graceful error handling, etc. have been put on hold. The authors are well aware of the need to devote more attention to those lagging issues and indeed intend to do so.

References

- J. Åkesson. *Tools and Languages for Optimization of Large-Scale Systems*. PhD thesis, Department of Automatic Control, Lund University, Lund, Sweden, 2007.
- J. Åkesson. Optimica—an extension of Modelica supporting dynamic optimization. In *Proceedings of the 6th Modelica Conference*, Bielefeld, Germany, March 2008. Modelica Association.
- J. Åkesson, M. Gäfvert, and H. Tummescheit. JModelica—an open source platform for optimization of Modelica models. In *Proceedings of MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling*, Vienna, Austria, February 2009. TU Wien.
- E. M. Gertz and S. J. Wright. Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software*, (29):58–81, 2003.
- E. Hairer, Ch. Lubich, and M. Roche. The numerical solution of differential-algebraic equations by Runge–Kutta methods. *Lecture Notes in Mathematics*, 1409:56–225, 1989.
- K. L. Hiebert and L. F. Shampine. Implicitly defined output points for solutions of ode-s. Technical report, United States Department of Energy, Sandia Laboratories, 1980.
- J. Lygeros, K. H. Johansson, S. Sastry, and M. Egerstedt. On the existence of executions of hybrid automata. In *Proceedings of the 38th IEEE CDC*, pages 2249–2254, Phoenix, AZ, USA, December 1999. IEEE.

- C. C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal on Scientific and Statistical Computing*, 9(2):213–231, 1988. doi:10.1137/0909014.
- R. Pytlak. *Numerical Methods for Optimal Control Problems with State Constraints*. Lecture Notes in Mathematics 1707. Springer Berlin Heidelberg, 1999. ISBN 9783540662143.
- R. Pytlak. Numerical procedure for optimal control of higher index DAEs. *Discrete Contin. Dyn. Syst.*, 29(2):647–670, 2011. ISSN 1078-0947; 0133-0189; 1553-5231/e. doi:10.3934/dcds.2011.29.647.
- R. Pytlak and D. Suski. On solving hybrid optimal control problems with higher index DAEs. *Optimization Methods and Software*, 2017. doi:http://dx.doi.org/10.1080/10556788.2017.1288730.
- R. Pytlak, J. Błaszczyk, A. Karbowski, K. Krawczyk, and T. Tarnawski. Solvers chaining in the IDOS server for dynamic optimization. In *Proceedings of 52nd IEEE CDC*, pages 7119–7124, Florence; Italy, 2013. IEEE. ISBN 978-1-4673-5714-2. URL <http://dblp.uni-trier.de/db/conf/cdc/cdc2013.html#PytlakBKKT13>.
- R. Pytlak, T. Tarnawski, B. Fajdek, and M. Stachura. Interactive Dynamic Optimization Server – connecting one modelling language with many solvers. *Optimization Methods and Software*, 29(5):1118–1138, 2014. doi:10.1080/10556788.2013.799159.
- M. S. Shaikh. *Optimal Control of Hybrid Systems: Theory and Algorithms*. PhD thesis, McGill University, Montreal, Que., Canada, 2004. URL http://digitool.library.mcgill.ca/R/?func=dbin-jump-full&object_id=85095&local_base=GEN01-MCG02.AAINR06340.
- D. Suski and R. Pytlak. The weak maximum principle for hybrid systems. In *Proceedings of the of the 24th IEEE MED*, pages 338–343, Athens; Greece, 2016. IEEE.
- T. Tarnawski and R. Pytlak. DOML - a compiler environment for dynamic optimization supporting multiple solvers. In *Proceedings of the 10th International Modelica Conference*, pages 1095–1104, Lund; Sweden, 2014. Linköping University Electronic Press.
- A. J. van der Schaft and J. M. Schumacher. *An Introduction to Hybrid Dynamical Systems*. Lecture Notes in Control and Information Sciences. Springer, 2000. ISBN 9781852332334.
- A. Wachter and L. T. Biegler. On the implementation of an interior point line search filter algorithm for large scale nonlinear programming. *Mathematical Programming*, (106):25–57, 2006.

Large Scale Training through Spoken Tutorials to Promote and use OpenModelica

Kannan M. Moudgalya¹ Bhargava Nemmaru¹ Kaushik Datta¹ Priyam Nayak¹ Rahul Jain¹
Peter Fritzson² Adrian Pop²

¹Dept. of Chemical Engineering, Indian Institute of Technology Bombay, India,
kannan@iitb.ac.in, {bnemmaru, kaushikdatta18, nayak.priyam22, rahjain1}@gmail.com

²Dept. Computer and Information Sciences, Linköping University, Sweden,
{peter.fritzson, adrian.pop}@liu.se

Abstract

The step-by-step self-teaching approach through audio-video tutorials, known as Spoken Tutorials, has been very successful. About 3.4 million students in India have taken at least one course during the past 6-year period, of which 1.6 million students have attended the rapidly expanding course programme during 2016. This programme has now been expanded by a newly developed course in Modeling and Simulation with Modelica using the OpenModelica open source tool, primarily via the OMEdit graphical user interface. The spoken tutorial programme is exclusively based on free and open source software. This paper gives an introduction to the spoken tutorial approach and presents the recently developed spoken tutorial series for Modelica using OpenModelica. Feedback of participants shows that this series is an effective tool for self-learning of OpenModelica. The paper also presents a new web version that generalises the interactive DrModelica course material, OMWebbook: it enables students to learn Modelica, do text-based modeling exercises, and run simulations without needing to install a Modelica tool. OMWebbook is also planned to be covered in a future update to the spoken tutorial course on Modelica.

Keywords: Spoken Tutorial, tutorial, Modelica, OpenModelica, teaching, self learning, modelling, simulation

1 Introduction

Modelling and simulation is the most cost effective way for a developing country like India to become a developed one. For example, the warning issued by the Indian Space Research Organisation (ISRO) saved 10,000 lives recently (Laxman, 2016). It is well known that satellites help locate arable land, help predict locations in seas with fish population, etc. India's advances in the satellite technology are rooted in modelling and simulation. ISRO's interplanetary missions are said to be most cost and time effective, one of the reasons being the extensive use of simulation (EconomicTimes, 2013). Improving the modelling and simulation ethos amongst the academics can help derive such benefits in other sectors also.

Free and open source software (FOSS) is equally im-

portant to a developing country like India. Use of proprietary software results in an outgo of \$ 1 billion every year, in education and police sectors alone (De, 2009). This precious foreign exchange can be more usefully spent in priority sectors like poverty alleviation and infrastructure development. Because of this, the Indian government has made the use of open source software mandatory, whenever it is of comparable capability to commercial software for the tasks at hand (Govt. of India, 2015). As OpenModelica implements the complete Modelica language (Modelica-Association, 2012) and is an open source implementation, we have selected it to promote the modelling and simulation culture in India. The award winning Spoken Tutorial method is selected for this promotion (Google, 2015; QS&Wharton, 2015).

This paper is organised as follows. We begin with a brief explanation of the Spoken Tutorial methodology in sections 2 and 3. We list the Spoken Tutorials created on OpenModelica in section 4. We then explain in section 5 how to use these tutorials for effective learning without experts. In section 6, we explain the cloud environment OMWebbook. In the final section, we conclude the paper with a discussion on future work.

2 Spoken Tutorials

A Spoken Tutorial is a ten minute long audio-video tutorial, created using the Screencast technology. It is well known that the optimal time for a video tutorial is about 9 minutes (Guo, 2013).

Here we use the tutorial concept in a slightly different way from a conventional 2-3 hour lecture-based tutorial. Spoken Tutorial is a smaller modular piece of active learning: A typical student following such an audio-video tutorial is expected to pause/replay the video and reproduce every command. To enable this, all the files used in the video are also seamlessly made available to the learner. As the software covered is open source, the student can download it, and practise the tutorial side-by-side. There are also exercises to perform. A beginner could easily spend half an hour to practise a 10 minute Spoken Tutorial, while an advanced learner may complete it in less time.

Spoken Tutorials are created to explain general IT con-

cepts. Although only a small amount of information may be covered in ten minutes, one can cover advanced topics as well through a series of tutorials. The following factors distinguish this methodology:

1. Spoken Tutorials are created for self learning. We write the script before making the video. The script has to be certified as understandable by a beginner, before it is taken up for recording.
2. Spoken Tutorials are released through Creative Commons Attribution Share Alike (CC-BY-SA) license from our website (Spoken-Tutorial-Project, 2017b).
3. Spoken Tutorials are created on open source software only. As a result, anyone who wants to learn using Spoken Tutorial, can download and practise with the corresponding software.
4. We make available all the code, data files, etc., that are required in a tutorial, see Figure 6. This allows the learner to reproduce all the commands shown in the tutorial. This also allows a learner to start learning from any tutorial, obviating the need to go in a particular sequence.
5. We use the side-by-side method of learning through Spoken Tutorials. A schematic of the arrangement is given in Figure 7. This arrangement reduces the cognitive overload of learners (Moudgalya, 2014).
6. We dub the spoken part of Spoken Tutorials into all 22 official languages of India. For this, we time the script, which needs to be done only once. The fact that the video is in English helps from the employment perspective of our students. We have dubbed some of our tutorials also into some languages of the Middle East, South East Asia, Latin America and Africa. As our tutorials are available under the CC-BY-SA license, these are available to everyone.
7. We have about 800 tutorials made into English, covering about 40 topics, such as, C, C++, Java, Python, PHP, Perl, Ruby, Scilab, L^AT_EX, LibreOffice and DWSIM. We have about 5,600 dubbed tutorials. This is the largest collection of IT training material in Indian languages.
2. As it was difficult to find free slots, we mapped Spoken Tutorials to lab courses and encouraged learning during lab hours. As they do not have access to good teachers, most students in India find this a valuable resource that helps them understand the subject, score well in exams and get jobs.
3. We convinced university authorities, curriculum boards and syllabus committees to use Spoken Tutorials officially in their lab courses. More than 100 universities, each of which has about 50 to 500 affiliating colleges, have accepted the Spoken Tutorial methodologies, and have sent circulars recommending the use. At present, there are about 20,000 semester long lab courses that officially use Spoken Tutorials.
4. We now insist that we work only with colleges that agree to train ALL their students through Spoken Tutorials. This and the previous point have helped us train a really large number of students in the first full year of implementation, which is 2016, see Figure 1. We have trained a total of 2.8 million people in the past five years, reaching 1.6 million in 2016 alone. 200,000 people have already enrolled for our training during the first 20 days of this calendar year.
5. We provide online tests and certificates to college students who pass them. These are offered free of cost, thanks to the funding from our government.
6. Our website (Spoken-Tutorial-Project, 2017b) also receives a large number of visitors. In Figure 2, we present the statistics of visitors to our web page. It should be noted that most of our learners using the offline learning material, created using the facility described in Figure 4, which has increased the number of learners 50 times, as reported in Figure 2.
7. The web statistics are good from the view of other parameters as well. For example, the average time one spends on our web page (Spoken-Tutorial-Project, 2017b) is about 10 minutes and the bounce rate is about 30%, as can be seen from Figure 3.

We can use the procedure presented in this section to teach and thereby promoting OpenModelica and the Modelica language.

3 Large Scale Training with Spoken Tutorials

We now briefly explain the way we have used Spoken Tutorials to promote IT literacy:

1. We initially offered 2 hour free training to college students and faculty, to be done outside college hours. As Spoken Tutorials are created for self learning, presence of an expert is not necessary. Any interested person could organise a training session, using the resources at the college.

4 OpenModelica Spoken Tutorials

We created ten Spoken Tutorials on OpenModelica, as shown in Table 1. In this Table, we have listed the title of these tutorials and their learning objectives. The average time of these tutorials is about 13 minutes, with the minimum of 8 minutes and a maximum of 15 minutes. Although these are the first set of tutorials we created on OpenModelica, we have numbered them from 4, because of reasons to be explained next.

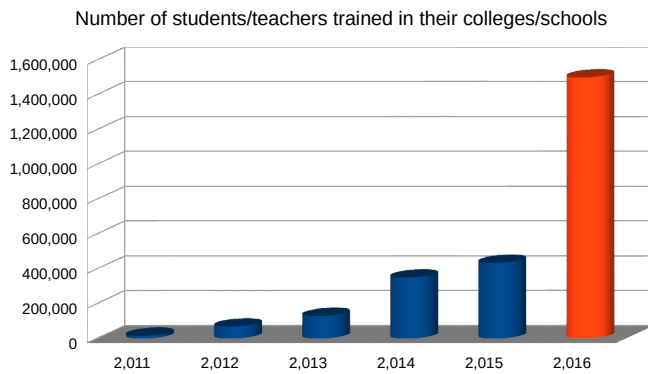


Figure 1. Growth of students/teachers trained. Our insistence that every student should be trained on at least one topic, and mapping to course content, have resulted in a large growth in 2016. The total number trained until now is about 3 million.

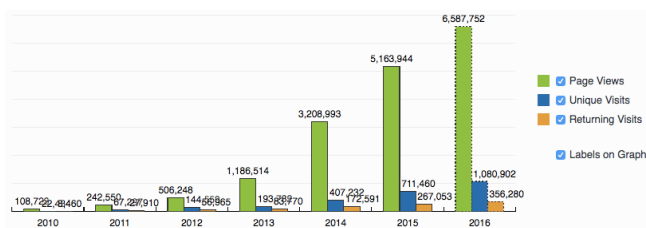


Figure 2. Growth of visits to ST website (Spoken-Tutorial-Project, 2017b). This can be obtained by visiting (StatCounter) and choosing Yearly data

In a recently held workshop (FOSSEE-Team, 2017), all the participants were asked to self-learn OpenModelica using Spoken Tutorials. We received the following feedback from the participants about the use of Spoken Tutorials:

- ... the detailed step-by-step descriptions made it easy to learn the basics of OpenModelica.
- Is is good for beginners. I suggest you to showcase more application use cases for the advanced use.
- ... very good , we have learned much from spoken tutorial.
- Spoken Tutorials are easy to Understand.

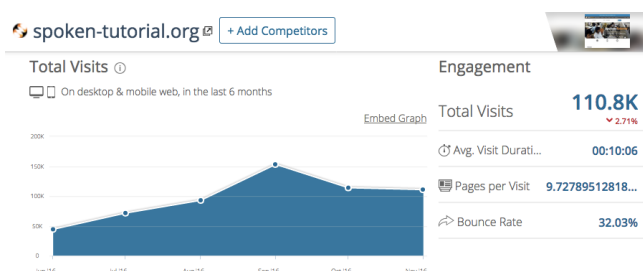


Figure 3. Analysis of visitor statistics of Spoken Tutorial, given by SimilarWeb (Team)

Table 1. Summary of Spoken Tutorials on OpenModelica

No.	Name	Learning Objectives
4	Developing an equation based model	Create a new Modelica class, variables, parameters and equations. Assign initial, minimum and maximum values. Simulate model and see results.
5	Control flow and event handling	Explain if-else statements, time and state events, when statement, and reinit function.
6	Functions and types	Define functions, data types, input and output variables, assignment statements, and the algorithm section. Evaluate a polynomial.
7	Arrays in Modelica	Define vectors, vector indexing, array variables, for and while loops, and nested for loop.
8	Array functions and operations	Explain OMShell, array construction function and perform arithmetic operations on vectors and matrices.
9	Modelica Packages	Create a package of classes, reference classes in a package, import statement, and using Modelica library.
10	Annotations in Modelica	Specify an annotation and define a record. Explain through experiment, model and documentation annotations.
11	Icon and diagram views	Specify icon and diagram views of a class, insert a polygon and an ellipse in icon/diagram view. Introduce coordinate system, grid and components. Explain origin and extent concepts, and line and fill styles.
12	Component oriented modelling	Instantiate classes. Define component orientation, acausal connectors, resistors, sources and the ground. Connecting classes and pins.
13	Block component modelling	Define blocks and connect them. Use MISO blocks and Modelica libraries. Define RealInput and RealOutput connectors. Instantiate sum and product functions.

- ... up to some extent this tutorial make learning easier but we should make some more basic tutorial on OpenModelica.
- In my opinion side by side tutorials are the best way to start learning a new software. In order to master anything it will take a lot of practice of course but

Table 2. Summary of New Spoken Tutorials on OpenModelica

No.	Name	Learning Objectives
1	Introduction to OMEdit	Introduction to OM, OMEdit, opening a class from libraries browser, simulating it and seeing the results in plots. Explain through a heat transfer example in thermal library.
2	Examples through OMEdit	Expose learners to models from electrical and mechanics libraries, using rectifier and double pendulum classes. Simulate and see the results in plots.
3	OM Connectors	Train how to compose existing objects to create a new circuit. Concept of connectors introduced. Explain through resistor, capacitor, inductor, voltage source and ground.

having the basics right makes practice a lot easier.

- *Tutorials are excellent. If you could include tutorials on popular extensions available on Modelica, it would be a great help. For example, the device drivers library.*

Tutorials in Table 1 were created as per the pedagogy we use in classes: teach the underlying language first. But this is not necessarily the best way when it is offered for self learning. Students who depend on commercial simulators are used to the plug and play method of learning, which is possible in OpenModelica if one uses the predefined models that come with the distribution. This will also reduce the fear of first time users.

To address the above issue, we created two more tutorials. The first one explains how heat transfer modelling can be done. The second tutorial is concerned with examples from Electrical Engineering and Mechanical Engineering. These two tutorials are created for the absolute beginners, who would like to learn how to use the already available models.

We created one more tutorial to explain how to connect predefined models. Such a facility is generally available in commercial simulators. Introduction to this capability right at the beginning increases the value of the simulator in the opinion of the learner. In this tutorial, we show how to connect the preexisting blocks to build a new model. We now ask new learners to practise these tutorials, before starting with the ones in the previous section.

These new tutorials are listed in Table 2. Although created later, they are numbered from 1, as we recommend these to be practised first, before proceeding to the ones listed in Table 1.

5 Method to Use OpenModelica Spoken Tutorials

We now explain how to use OpenModelica Spoken Tutorials. One should remember that our method should be such that any volunteer can conduct these workshops. As volunteers may not know answers to questions the learners may have, we insist on only reproducing what is shown in Spoken Tutorials: the learner does not know anything, so why not first do the things suggested in the tutorial? The pros and cons of this approach are summarised in (Moudgalya, 2011). We also offer a timed forum that will help the learner to go through the answers of previous questions and to ask new questions (Spoken-Tutorial-Project, 2017a).

The first thing to do is to create an offline version using the `create the cdcontent` utility, as shown in Figure 4. Of course, this has to be done only by the organiser of the workshop. This will create a zip file, which has to be unzipped and copied on to every computer thereafter. We now summarise the recommended way to use Spoken Tutorials to conduct an OpenModelica self-learning workshop:

1. Open the file `index.html` using Firefox or Chrome. Internet Explorer may not work correctly. One gets the page as shown in Figure 5. From the url inside the red box in this figure, one can see that the learning content comes from the file system.
2. Listen to the side by side spoken tutorial that appears on this page. This tutorial explains how the learner has to use this method. After that, one has to `Select Foss Category` and then `Select Language` and then `Submit` - these options are enclosed in a green box in Figure 5.
3. The learner has to learn all the tutorials in the resulting play list one by one. From the play list, one can select any tutorial. If one scrolls down, one can see if any code file comes with that tutorial. In Figure 6, one can see where the code files are available.
4. Figure 7 shows how to open the Spoken Tutorial and the OpenModelica software side by side. Using the side-by-side method (Moudgalya, 2014), the learner has to reproduce every command described in the tutorial. The learner has to do the assignments also. As a result, the learner ends up spending a lot more time, even though the tutorial itself may only be of ten minute duration.
5. Using the method explained above, one has to learn all Spoken Tutorials in the play list.

The above given explanation is for the offline user. The same procedure will work for online use of Spoken Tutorials.

In the six month time since we created Spoken Tutorials under discussion, we have trained more than 5,000 people on OpenModelica. As this is achieved through the self learning of already created Spoken Tutorials, there is a potential to train many more.

6 Cloud Environment OMWebbook to Ease Learning of OpenModelica

DrModelica (Lengquist-Sandelin et al., 2003) is an interactive electronic document for learning Modelica text-based modeling and simulation using the OpenModelica tool OMNotebook. It is structured like a book, with chapters, sections, model examples, exercises, formulae, and figures (Asghar et al., 2011). The recent versions of DrModelica contain most of the model examples in (Fritzon 2014) in an editable and executable form, including plots of simulated models. DrModelica is intended for self-learning and includes exercises with solutions, where the solutions are temporarily hidden while the student is working on a problem. Thus, it is extremely useful to everyone, and especially to the beginning learners of the Modelica language and the OpenModelica tool.

Recently we have developed a web-based version of OMNotebook, called OMWebbook (<http://omwebbook.openmodelica.org/>) which enables editing models, running simulations, and doing plots in a web-page (Figure 8). The appearance is very similar to the OMNotebook. Thus, the students need not install any Modelica tool on their computer. They do not even need a computer, the exercises can be done using a phone or pad. OMWebbook communicates with a server that performs the actual simulation and plots. Naturally, those who do not have bandwidth to access the Internet can use OMNotebook.

OMNotebook, and also OMWebbook, have recently been enhanced with support for typesetting Mathematical formulae using \LaTeX commands (Asghar et al., 2011). These commands are associated with a formulae cell (Figure 9) but can be hidden (Figure 10) when editing is not

needed. Figure 10 shows a small part of a chemical engineering course book in OMNotebook and OMWebbook with some formulae that have been typeset using this approach.

We propose to develop Spoken Tutorials on DrModelica, OMNotebook and OMWebbook in the near future. This will help beginners to quickly learn how to use these powerful tools and hence will further ease the learning of OpenModelica. Given that modelling and simulation are advanced topics for most students in India, we need all the tools to make them accessible.

7 Conclusions and Future Work

In this work, we have outlined a procedure to improve the modelling and simulation ethos in India and elsewhere through Spoken Tutorial enabled self-learning of OpenModelica. Based on the user feedback, we are in the process of increasing the offering of Spoken Tutorials on other OpenModelica topics. Through this effort, the powerful modelling paradigm of Modelica is expected to reach many people in India.

In the companion paper presented by our group in this conference (Jain et al., 2017), we have explained how the features of OpenModelica have been extended to make it useful to chemical engineers. We seek such contributors in other domains too. We hope to promote the use of the Modelica language and the OpenModelica tool with specific focus on different domains of application.

We also seek collaborators who would want to make available this important technology in their countries.

Acknowledgements

This work has been supported by Swedish Vinnova governmental agency and the Indian DST governmental agency in the Indo-Swedish RTISIM project, and by the National Mission on Education through ICT, Ministry of Human Resource Development, through the Spoken Tutorial project. The OpenModelica development is supported by the Open Source Modelica Consortium.

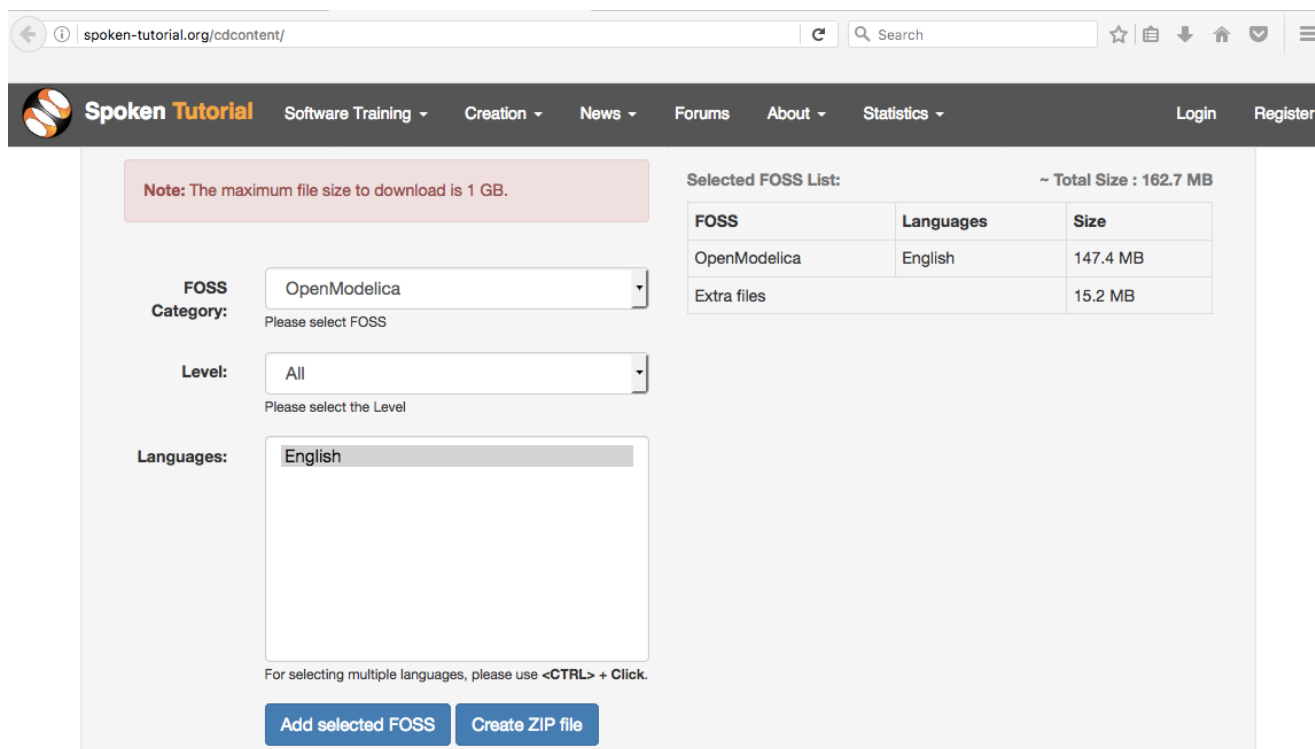


Figure 4. Creating a zip file of all tutorials for offline use, using the cdcontent facility

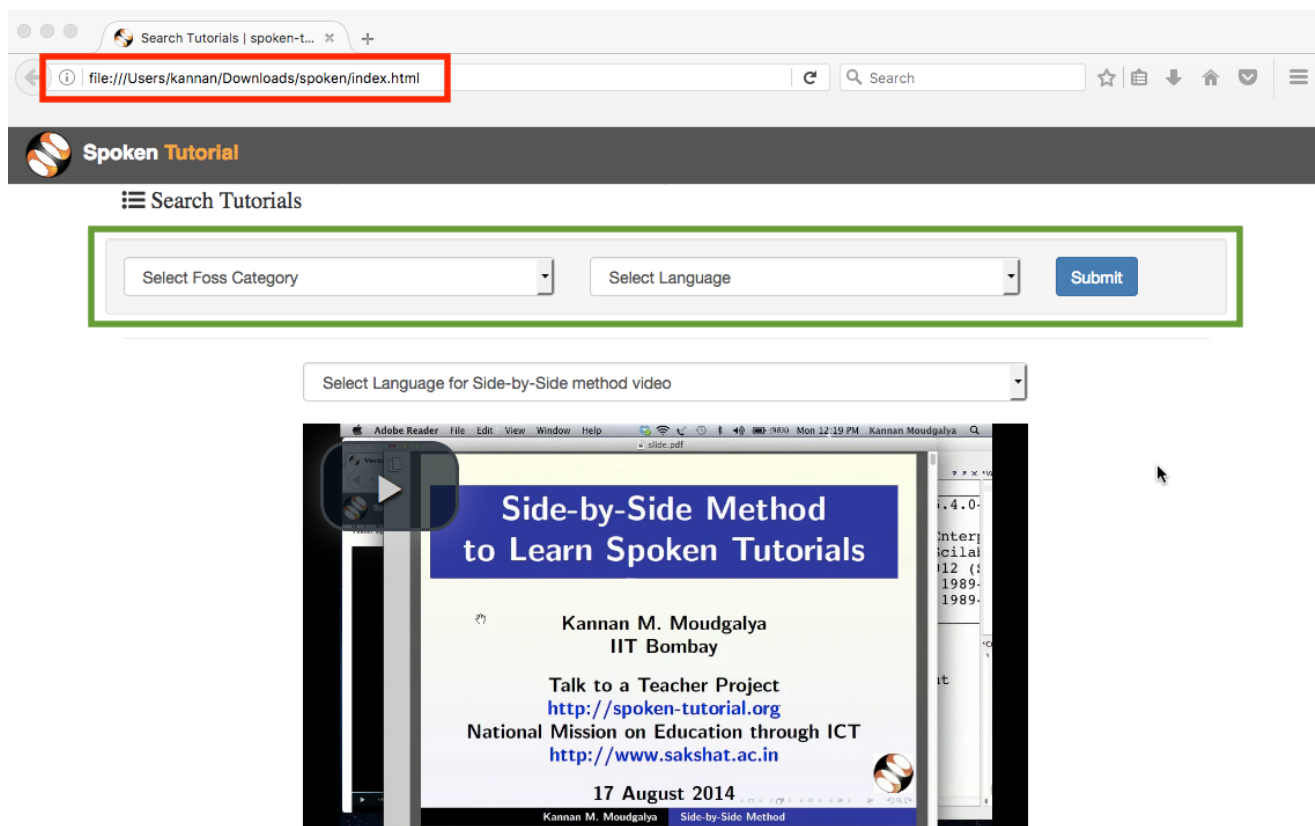


Figure 5. Unzipped content, opened in Firefox or Chrome. Internet Explorer may not work correctly. The URL inside the red box points to a file. One has to first listen to the side-by-side method tutorial. After that, one selects the FOSS, then the language, such as English, and then Submit. This will give a play list. One can then open any of the tutorials and practise side by side, as shown in 7

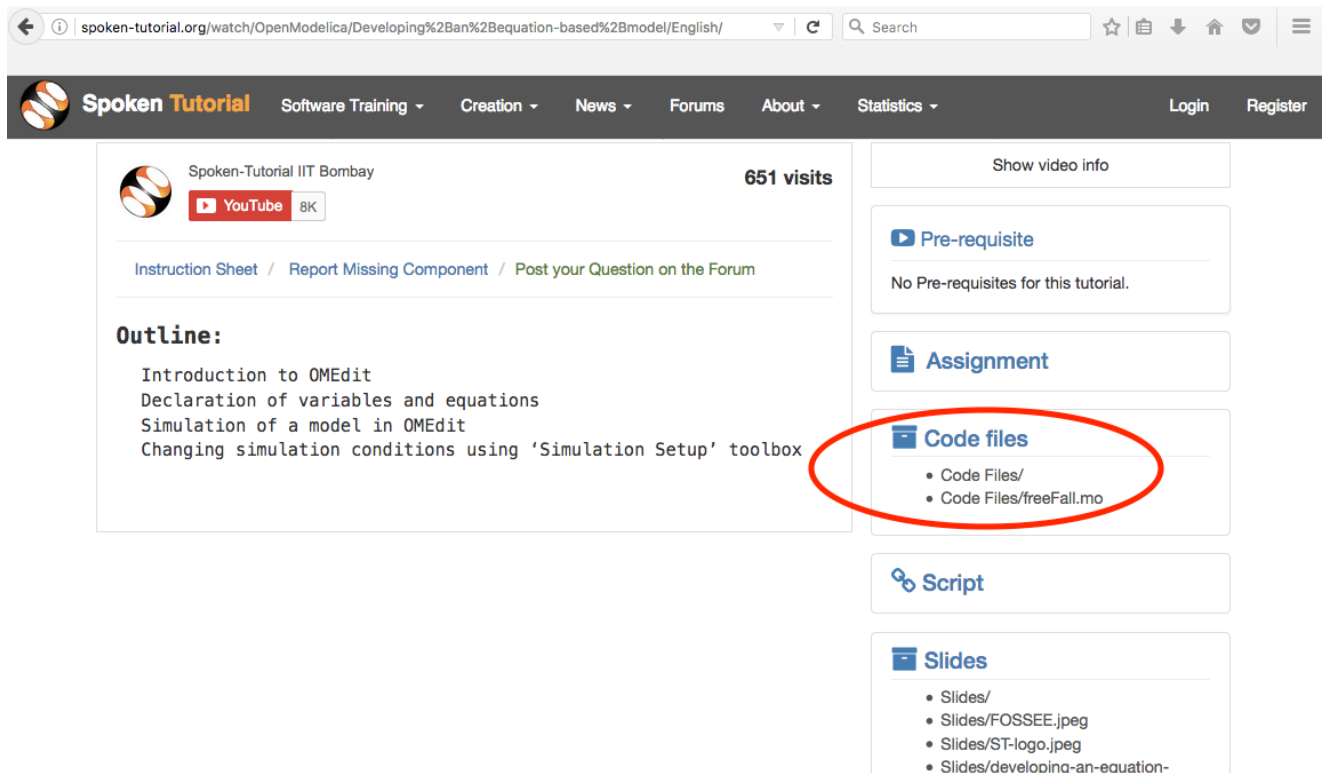


Figure 6. In the selected tutorial, one has to scroll down to locate the code files. Its location is indicated by the red oval in the above figure. Code files help the learner reproduce every command.

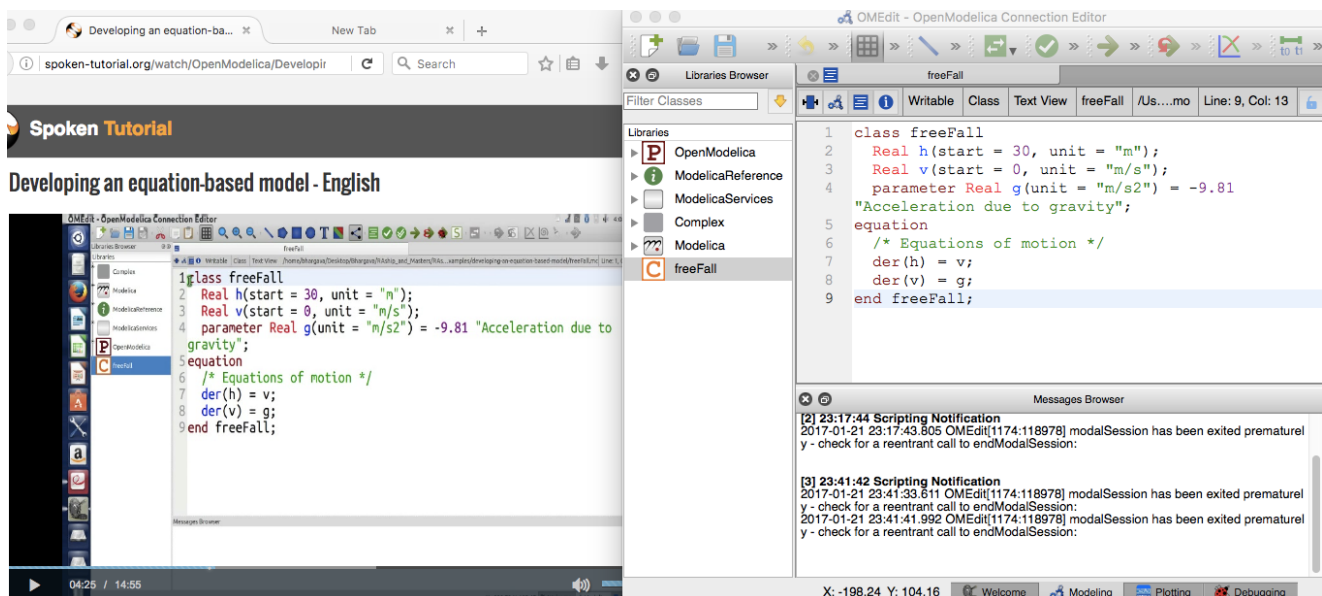
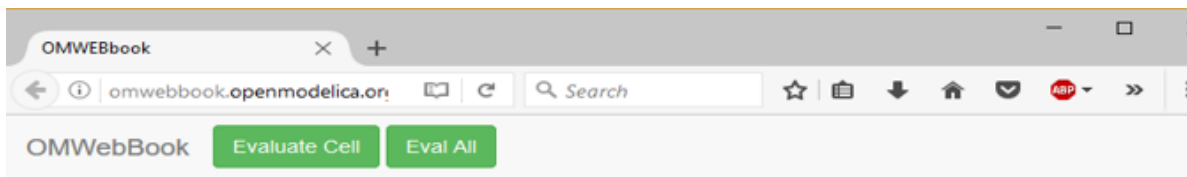


Figure 7. Side-by-side method. On the left hand side, we have the spoken tutorial, developing an equation based model. On the right hand side, OMEdit is displayed. The code file that came with the tutorial is opened in OMEdit.



First Basic Class

1 HelloWorld

The program contains a declaration of a class called HelloWorld with two fields and one equation. The first field is the variable x which is initialized to a start value 1 at the time when the simulation starts. The second field is the variable a , which is a constant that is initialized to 1 at the beginning of the simulation. Such a constant is prefixed by the keyword parameter in order to indicate that it is constant during simulation but is a model parameter that can be changed between simulations.

The Modelica program solves a trivial differential equation: $x' = -a * x$. The variable x is a state variable that can change value over time. The x' is the time derivative of x .

```

1 class HelloWorld
2   Real x(start = 1,fixed=true);
3   parameter Real a = 1;
4   equation
5     der(x) = - a * x;
6 end HelloWorld;
7
```

2 Simulation of HelloWorld

```

1 simulate( HelloWorld, startTime=0, stopTime=4 )
2
```

Plot the results.

```

1 plot( x )
2
```

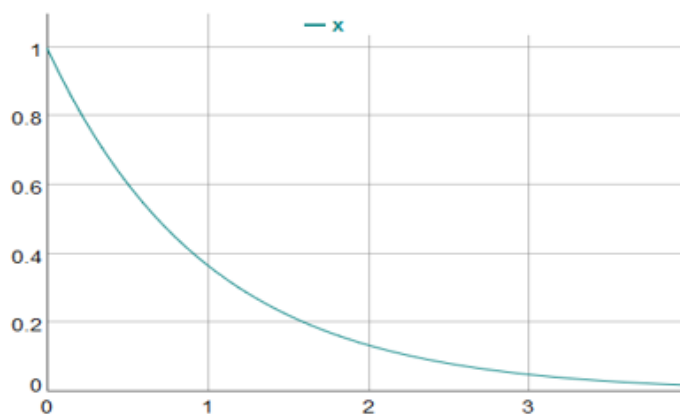


Figure 8. Example using OMWebbook on the simple HelloWorld model in DrModelica

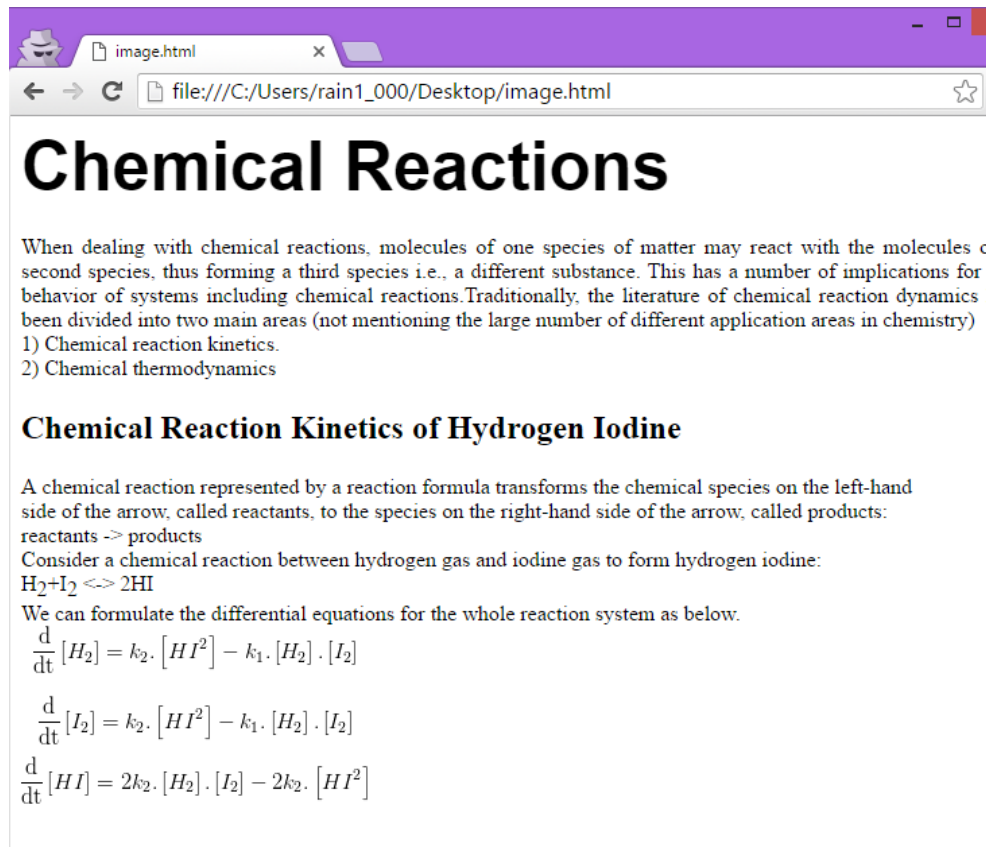


Figure 9. Type-setting mathematical formulae in OMNotebook using L^AT_EX commands

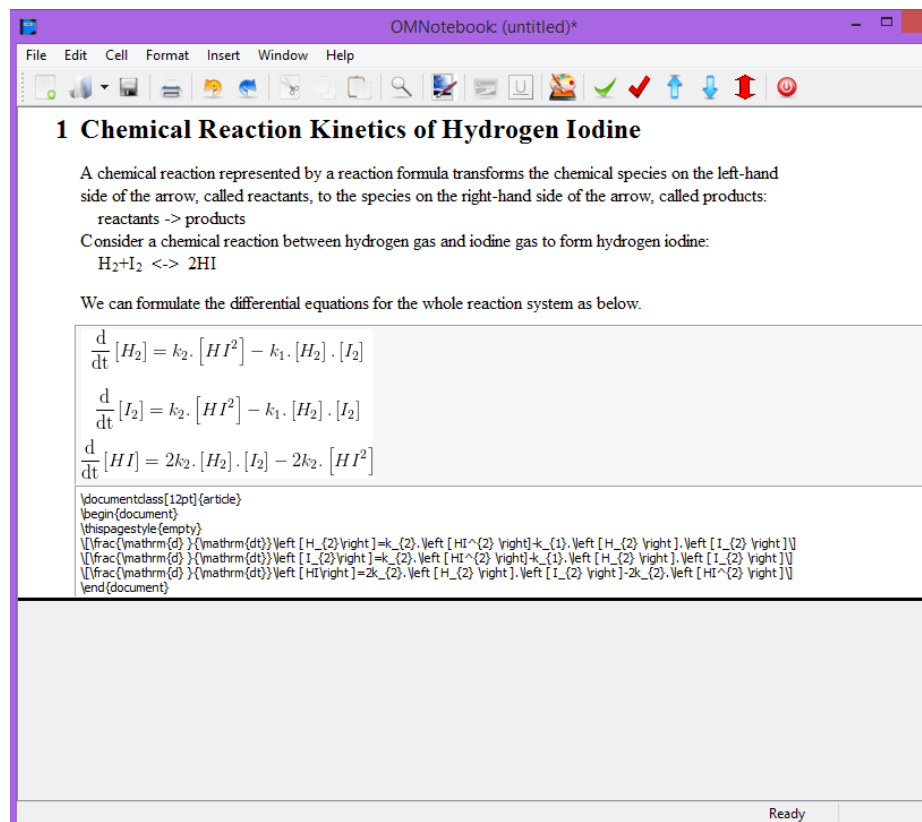


Figure 10. Example of finished typesetting of mathematical formulae in OMNotebook and OMWebbook

References

- A. Asghar, S. Tariq, M. Torabzadeh-Tari, P. Fritzson, A. Pop, M. Sjölund, P. Vasaiely, and W. Schamai. An Open Source Modelica Graphic Editor Integrated with Electronic Notebooks and Interactive Simulation. In *Proc. of the 8th International Modelica Conference 2011*, pages 739–747, Linköping university, Sweden, 2011.
- R. De. Economic impact of free and open source software - a study in india. Technical Report http://www.iimb.ernet.in/~rahulde/RD_FOSSRep2009.pdf, IIM Bangalore, 2009.
- EconomicTimes. Why isro’s mars mission is the cheapest. <http://economictimes.indiatimes.com/slideshows/science-technology/why-isros-mars-mission-is-the-cheapest/harness-software-work-fast/slideshow/24982272.cms>, 31 Oct. 2013.
- FOSSEE-Team. Openmodelica workshop. <http://fossee.in/workshop/om/>, 4-5 Jan. 2017.
- Google. Google mooc focused research awards. <https://research.googleblog.com/2015/03/announcing-google-mooc-focused-research.html>, March 2015. Last seen on 22 Jan. 2017.
- Govt. of India. Policy on adoption of open source software for govt. of india. Gazette Notification, <http://www.indianemployees.com/gazette-notifications/details/policy-on-adoption-of-open-source-software-for-govt-of-india/>, 2015. English notification is given after that in Hindi. Last seen on 22 Jan. 2017.
- P. Guo. Optimal Video Length for Student Engagement. See <https://www.edx.org/blog/optimal-video-length-student-engagement>, 2013. Last seen on 3 April 2017.
- R. Jain, K. M. Moudgalya, P. Fritzson, and A. Pop. Development of a Thermodynamic Engine in OpenModelica. In *12th Int. Modelica Conf.*, Prague, 2017. Modelica Association.
- S. Laxman. Cyclone Vardah: ISRO satellites saved 10,000 lives in Tamil Nadu. Times of India, <http://timesofindia.indiatimes.com/india/cyclone-varadah-isro-satellites-saved-10000-lives-in-tamil-nadu/articleshow/55992320.cms>, 17 December 2016. Last seen on 3 April 2017.
- EL. Lengquist-Sandelin, S. Monemar, P. Fritzson, and P. Bunus. DrModelica - An Interactive Tutoring Environment for Modelica. In *Proceedings of the 3rd International Modelica Conference*, Linköping, Sweden, 3-4 Nov. 2003.
- Modelica-Association. Modelica: A unified object-oriented language for physical systems modeling, language specification version 3.3. <http://www.modelica.org/>, May 2012.
- K. M. Moudgalya. L^AT_EX Training through Spoken Tutorials. *TUGboat*, 32(3):251–257, 2011.
- K. M. Moudgalya. Pedagogical and Organisational Issues in the Campaign for IT Literacy Through Spoken Tutorials. In R. Huang, Kinshuk, and N.-S. Chen, editors, *The new development of technology enhanced learning*, chapter 13, pages 223–244. Springer-Verlag, Berlin Heidelberg, 2014.
- QS&Wharton. Reimagine education 2015: Spoken Tutorial is Placed First in the Nurturing Employability Award Category. <http://application.reimagine-education.com/the-winners-individual/2015/132/2193b0ae3841f24da1464d4b6b70ee0f/Indian+Institute+of+Technology+Bombay%22>, Dec. 2015.
- Spoken-Tutorial-Project. Online forum. See <http://forums.spoken-tutorial.org/>, 2017a. Last seen on 3 April 2017.
- Spoken-Tutorial-Project. Official web page. See <http://spoken-tutorial.org/>, 2017b. Last seen on 3 April 2017.
- StatCounter. Summary log. http://statcounter.com/p5528933/summary/?account_id=2904483&login_id=5&code=9f03e451b379437c7356d2529c726a7a&guest_login=1. Last seen on 12 Dec. 2016.
- Similar Web Team. Get insights for any website or app. <https://www.similarweb.com/>. Last seen on 12 Dec. 2016.

EMOTH

The E-Mobility Library of OTH Regensburg

Alexander Grimm, B.Eng.¹ Prof. Anton Haumer²

¹OTH Regensburg, Germany, alexander.grimm@st.oth-regensburg.de

²OTH Regensburg, Germany, anton.haumer@oth-regensburg.de

Abstract

The importance of E-Mobility is rapidly increasing, not only for private vehicle traffic but also for public transport. In and around Regensburg, Germany there are a lot of automotive companies. Therefore E-Mobility is an important topic in the curriculum of several courses of study at the East-Bavarian Technical University of Applied Sciences Regensburg (OTH).

One Master of Applied Research student at OTH has chosen the topic to develop an open-source simulation tool for electric vehicles – the EMOTH Library – based on Modelica and to refine several aspects of the library during the one and a half year of the master course.

After one semester, the basic version of the library is available and will be presented in this paper.

Keywords: *e-mobility, electric vehicle, modular vehicle model, energy consumption, real driving cycle, driving performance.*

1 Introduction

The City of Regensburg decided to purchase E-buses to serve the old part of the town to decrease emissions and noise pollution in this area visited by many tourists per year. Moreover, the City of Regensburg maintains an E-Mobility Cluster to provide a platform for collaboration between local automotive companies and educational institutions. One project of this cluster is allowed to utilize one of the above mentioned buses for field tests of newly developed components, also offering the possibility to gather measurement data during real driving cycles.

OTH Regensburg joined that cluster and plans to provide an open-source simulation tool based on Modelica to review new components of the electrical drive train in an early stage of development: the EMOTH-Library. This project also permits to validate simulation results against measurements during real driving cycles.

Of course there are several simulation tools for electric vehicles available, also based on Modelica, but we found only commercial available libraries. The drawback of commercial tools is the invest hurdle for the cluster partners and the fact that it might be not that

easy to take components out from a commercial library and improve them to match the specific needs.

The library is based on Modelica and the VehicleInterfaces Library offered by the Modelica Association ([1], [2]). Following the structure and the templates of the VehicleInterfaces Library has the advantage that components can easily be exchanged without having any troubles with the interface definition. A big advantage of the VehicleInterfaces Library is the fact that there are one-dimensional rotational and translational mechanical connectors predefined in the templates as well as three-dimensional mechanical connectors. In the basic version, only one-dimensional effects have been taken into account. During the remaining two semesters of the master course, the basic models available now have to be refined to meet certain requirements, e.g. to enable changing the vehicle's mass at bus stops due to exchange of passengers.

2 Structure and Components of the Library

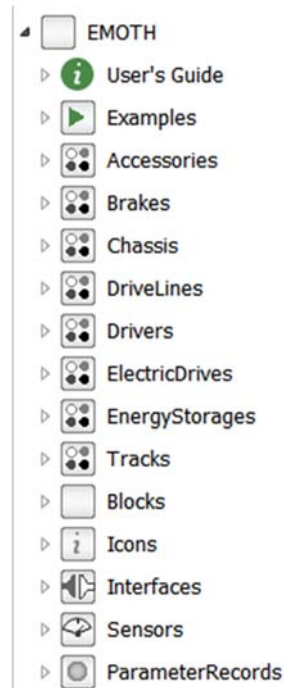


Figure 1 Structure of the EMOTH Library

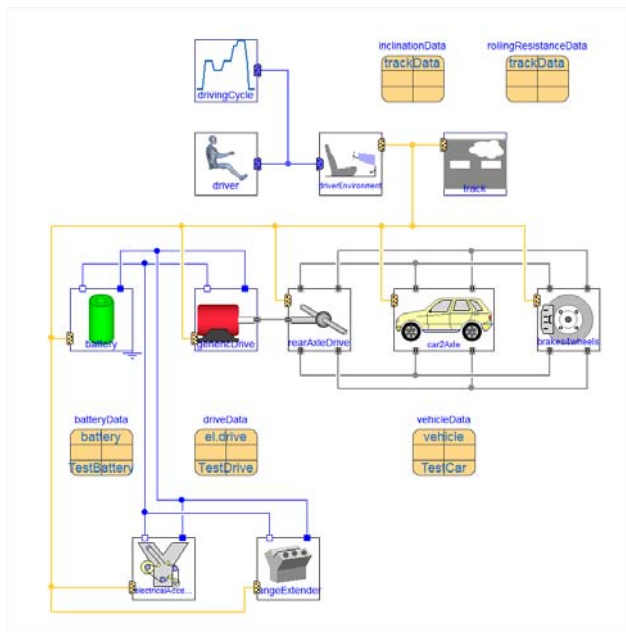


Figure 2 Components used in a complete model

Figure 1 depicts the structure of the EMOTH Library, Figure 2 shows a complete model with all components described in the following sections.

2.1 Chassis

The central component is the chassis (here a car with 2 axles) containing:

- the mass of the car including passengers
- the four wheels with their inertia
- the driving resistances
- connectors of the four half axles
- measurements feeding relevant signals to the bus

According to literature (e.g. [3], [4], [5]) the driving resistances consist of:

- drag resistance according to (1), dependent on relative speed of vehicle with respect to surrounding air,
- rolling resistance dependent on sine of inclination angle γ according to (2),
- inclination resistance according to (3) dependent on cosine of inclination angle γ .

$$F_D = c_W A \rho_A \frac{(v - v_A)^2}{2} \quad (1)$$

$$F_R = c_R mg \cos(\gamma) \quad (2)$$

$$F_I = mg \sin(\gamma) \quad (3)$$

c_W	...	coefficient of drag resistance
A	...	front cross section of vehicle
ρ_A	...	density of air
v	...	vehicle speed
v_A	...	longitudinal wind speed
c_R	...	coefficient of rolling resistance
m	...	total vehicle mass
g	...	gravitational constant

The wheels are taken as ideal wheels from `Modelica.Mechanics.Rotational`, but can easily be exchanged against more sophisticated models.

Since energy consumption is one of the most important aspects of the planned investigations, a one dimensional model of the mass and the rotating wheels is implemented, taking only longitudinal acceleration and velocity into account.

2.2 Brakes

In the first version, the brakes are built with the brake model from `Modelica.Mechanics.Rotational`. A simple controller distributes the brake signal to left and right wheel at front and rear axle. Each brake produces its own braking torque respectively force, according to the parameterization which allows to specify the braking force distribution between front and rear axle.

Driver assistance functions like antiskid braking can be implemented in a more sophisticated controller.

2.3 Drive Line

The drive line model contains the gear box, either with fixed ratio or with gear selection by the driver environment, and a simple model of the differential.

2.4 Electric Drive

The electric drive represents the motor, the power electronics and control. Since a correct parameterized current controlled drive can be represented by a second order block (see [6]), the desired torque is calculated from the throttle signal (in the range 0...1), taking field weakening into account. Throttle signal = 1 is interpreted as maximum torque available at the actual speed. Maximum torque depends on break-down torque of the motor and maximum current of the power electronics. The actual torque is fed to the motor's inertia and the flange, which is connected to the drive line.

Taking losses respectively efficiency of the motor and the power electronics into account, an ideal power converter draws the corresponding current from the DC power connection, taking actual DC voltage into account.

This simplified generic drive model ensures a maximum of simulation performance and can be easily improved by using more sophisticated models of motor, power electronics and control.

2.5 Energy Storage

In the first version, the battery is simplified as constant DC voltage and an inner resistance. However, the calculation of relevant signals – especially the state of charge (SOC) – is already implemented. A more sophisticated battery model, taking into account the SOC – dependent no-load voltage and the transient behavior of the batteries terminal voltage as an answer

to non-constant DC currents, is planned to be implemented in a following development phase.

2.6 Electrical Accessories

Electrical accessories, especially heating and cooling, have significant impact on energy consumption of an electric vehicle. In order to obtain realistic simulation results for the batteries state of charge, the power consumption of the accessories can be defined either constant or by a signal input from measurements over a real driving cycle.

2.7 Range Extender

Range extenders are common practice to increase range without increasing the batteries capacity which in turn means rising mass. In most cases they are a small combustion engine driven at an optimal point of operation, which drives an electric generator charging the battery.

The implemented range extender allows to specify constant charging power and state of charge limits for switching the range extender on and off.

2.8 Track

The track model defines the inclination and the road surface with respect to actual position of the vehicle along the track, either given by a constant value or interpolated from a table. The user has the choice how to extrapolate the table data if the vehicle's position leaves the range of the table definition:

- Hold the first / last point
- Extrapolation using the last two points
- Periodic repetition
- Extrapolation triggers an error.

Periodic repetition allows to define a closed loop, where the vehicle drives as many rounds as desired.

To be able to check the definition of inclination, especially on a closed loop track, the calculation of actual altitude is included. Longitudinal position and velocity of the vehicle is defined along the inclined track.

Additionally, longitudinal wind speed can be given either as a constant or by an input signal from an external table, containing measured data.

2.9 Driver Environment

The driver environment provides the signals that a driver could read from a dashboard to the driver interface. To have a maximum of flexibility, either a throttle and brake model (section 2.10) or a driver model (section 2.11) following the desired driving cycle can be connected to the driver interface.

The throttle and brake commands read from the driver interface are fed to the recuperation controller. The recuperation controller decides upon active braking using the electric drive and charging the battery (recuperation), or using the mechanical brake

system, or to distribute desired braking force between the two alternatives. The user can switch off recuperation.

The first implementation of the recuperation strategy is a very simple one:

- if recuperation is not switched off and
- if the batteries SOC is not above an upper limit and
- vehicle speed is above a lower limit

throttle and brake command (torque demand < 0 means braking) are fed to the electric drive, otherwise throttle signal is sent to the electric drive and brake signal is sent to the mechanical brake system.

2.10 Throttle and Brake

To be able to perform simple experiments, a throttle and brake block has been implemented. Like a human driver, the user can command throttle and brake separately. The commands can be given either by constants or by signal inputs. Additionally, the user can demand to move either forwards or backwards along the defined track (section 2.8).

2.11 Driver

The driver model tries to mimic a human driver. A human driver will watch the environment and the state of the vehicle. The decision somehow is based on a preview of the environment. To take that into account, the driver reads from the driving cycle the reference speed and a preview of the future reference speed. The preview time can be chosen to define the driver's behavior.

Based on his decision, the driver will make his mind of accelerating or braking the vehicle, with some delay representing the human response time. This is modeled by a PI-controller, fed by the difference of preview reference speed and a prediction of actual speed:

$$\epsilon_v(t) = v_{Ref}(t + t_{pre}) - (v(t) + a(t) \cdot t_{pre}) \quad (4)$$

2.12 Driving Cycle

The driving cycle is defined with table data as reference speed versus time. The user is able to define his own driving cycle, or choose one of the following predefined driving cycles:

- UDC urban driving cycle
- EUDC extra-urban driving cycle
- NEDC new European driving cycle

If simulation shall last longer than the time span defined by the driving cycle, one of the following choices can be taken:

- Hold the first / last point
- Extrapolation using the last two points
- Periodic repetition
- Extrapolation triggers an error.

The user can decide whether to terminate the simulation after a specified number of repetitions of the

driving cycle, or to drive until another termination condition is met, e.g. empty battery (SOC below minimum SOC).

2.13 Optional Thermal Connectors

All components have optional thermal connectors that can be switched on or off with the Boolean parameter `includeHeatPort`. In case `includeHeatPort = true`, the user has to connect an external thermal model representing the thermal behavior of the complete vehicle. This allows to develop and investigate the thermal management system.

Components with optional thermal connector:

- brake system
- drive line
- electric drive (motor and power electronics)
- battery (energy storage)

It has been decided that the chassis model has no thermal connector because the energy to overcome inclination resistance is converted rather to potential energy than to thermal energy; energy consumption due to drag resistance and rolling resistance is dissipated to heat, but the heat is generated at the exterior envelope of the vehicle and is most likely not taken into account for thermal management.

However, the power consumptions of driving resistances are fed as signals to the bus, allowing to analyze the power sinks of the biggest influence on energy consumption.

2.14 Bus Concept

According to the concept, all components communicate via their own sub-bus. All sub-buses are collected in the central control bus. This concept eases the distribution of signals through the whole vehicle architecture.

For the communication between the driver and the driver environment, a separate bus called driver interface is implemented.

2.15 Parameterization of the Models

As shown in Figure 1, the parameterization is managed in a flexible way by records:

- `vehicleData`
- `inclinationData` and `rollingResistanceData`
- `driveData`
- `batteryData`

`VehicleData` not only contains vehicle parameters like mass, front cross section, wheel radius and inertia, but also the parameters of the driveline and the brakes.

`InclinationData` and `rollingResistanceData` define inclination and rollingResistance with respect to the position along the track. The user can give constant values or a table definition.

`DriveData` summarizes the parameters of the motor, the power electronics and the control.

`BatteryData` gathers the parameters of the energy storage.

The modular concept allows to define independently a track with inclination and road surface, a vehicle and to try different designs of the drive and / or the battery.

3 Simulation Results

For testing the library components, the parameters summarized in Table 1 have been estimated. The drive parameters are shown in Table 2, the parameters of the battery in Table 3.

Table 1. Vehicle parameters

Vehicle mass	1500 kg
Desired acceleration/deceleration	$5 \frac{m}{s^2}$
Front brake : Rear brake force	50:50
Front cross section	$2 m^2$
Drag coefficient	0,5
Wheel radius	0,3 m
Wheel inertia	$0.25 kg \cdot m^2$
Gear ratio	1:5
Gear efficiency	85 %
Efficiency of differential	95 %

Table 2. Drive parameters

Base speed	4500 rpm
Nominal torque	250 Nm
Efficiency	90 %
Breakdown torque	1000 Nm
Maximum torque	500 Nm
Substitute time constant	5 ms
Inertia	$0.1 kg \cdot m^2$

Table 3. Battery parameters

Nominal DC voltage	400 V
Inner resistance	50 mΩ
Nominal Charge	100 A · h
Minimum SOC	0.1

For the first test, the rolling resistance coefficient was set as $c_R = 0.02$ and inclination of the track was set to 0. The electrical accessories have been estimated with a constant power consumption of 5000 W, and a range extender with a constant power generation of 6000 W. The range extender is started if SOC falls below 40% and is stopped if SOC rises above 80%. Recuperation is chosen for braking if SOC falls below 98%.

The driving cycle has been chosen to meet the NEDC (New European Driving Cycle) shown in Figure 3. Additionally, Figure 3 shows the actual vehicle speed which proves that the vehicle can follow the desired driving cycle nearly perfect.

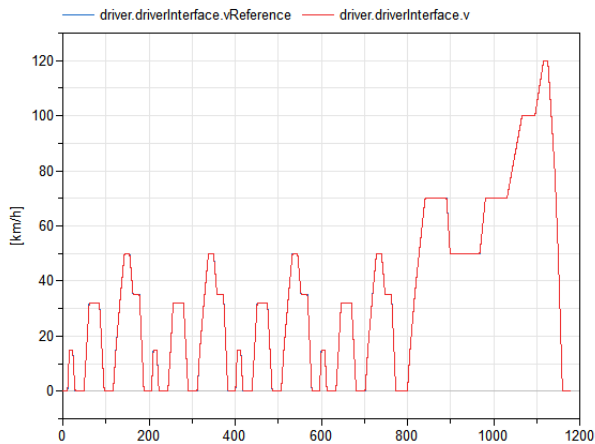


Figure 3 NEDC New European Driving Cycle

The DC power consumption during one cycle is shown in Figure 4, the development of the state of charge in Figure 5. Note that the range extender is not yet started during that cycle since SOC is high enough.

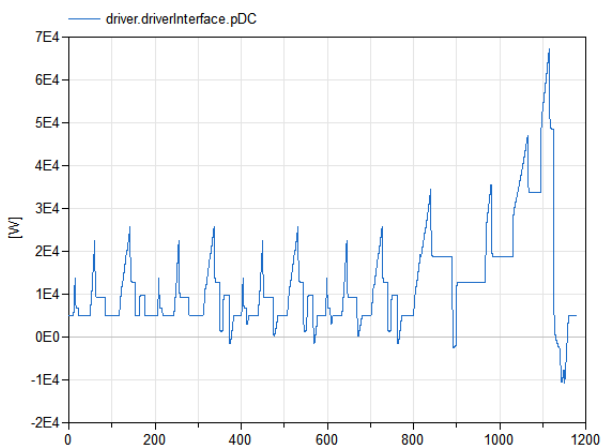


Figure 4 DC power consumption during NEDC

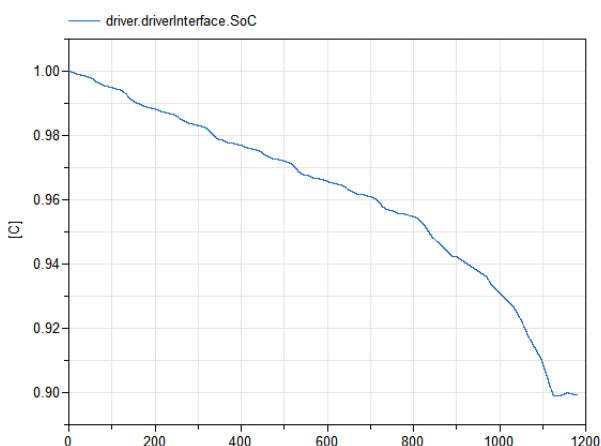


Figure 5 SOC during NEDC

On a notebook with Intel Core i7 processor at 2.3 GHz, 8 GB RAM, Windows 7 64 bit, using Dymola 2017 FD01 64 bit the simulation of the 1180 s cycle took 6 s which shows pretty good performance.

Subsequently, the NEDC was repeated until the battery was exhausted. Figure 6 shows that the range extender gets started at 7021.3 s when SOC falls below 0.4, the battery is discharged slower than before. In Figure 7 the vehicle position is depicted. It can be seen that the battery is exhausted after 14096 s reaching 131.25 km.

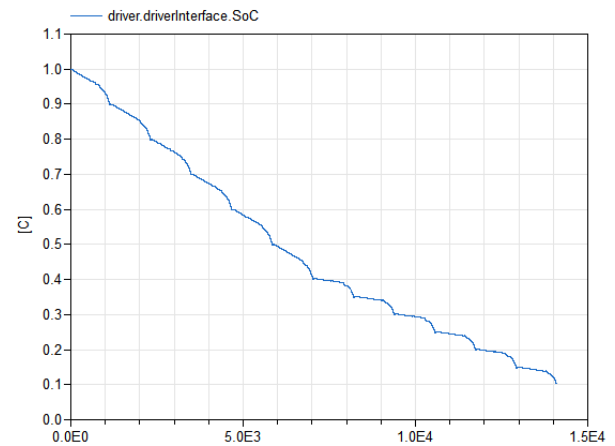


Figure 6 SOC during repeated NEDC

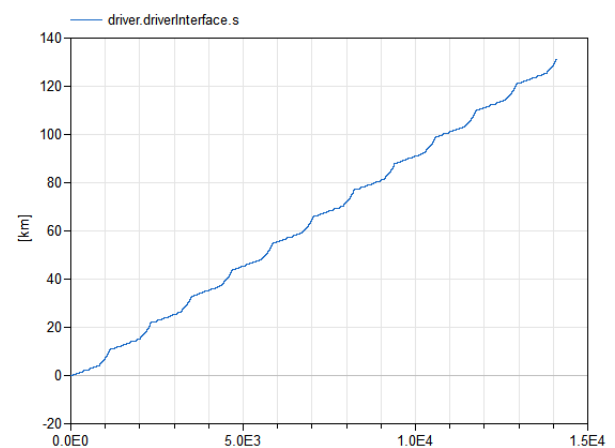


Figure 7 Vehicle position during repeated NEDC

The third experiment investigates acceleration and maximum speed with the chosen electric drive by setting a reference speed rising from standstill to $250 \frac{\text{km}}{\text{h}}$ in 1 s. Of course, the vehicle cannot follow this reference speed but requires maximum torque from the drive. Figure 8 proves that a maximum speed of slightly above $225 \frac{\text{km}}{\text{h}}$ can be achieved. Acceleration from standstill to $100 \frac{\text{km}}{\text{h}}$ takes approximately 6.8 s. Of course the drive may not be operated continuously at maximum torque, a thermal protection function would reduce throttle signal to avoid overheating of the motor and the motor electronics.

Figure 9 reveals both limits of field weakening: During the first phase, torque of the motor remains constant, and DC power consumption rises linearly with speed. After exceeding the first limit, DC power consumption remains constant with respect to speed as torque is lowered reciprocal to speed. When the

reference torque exceeds breakdown torque of the motor dependent on speed, torque has to be reduced more than reciprocal to speed and DC power consumption is reduced, too.

Of course driving with maximum torque respectively maximum power, Figure 10 confirms that the battery gets discharged much quicker than in the previous example.

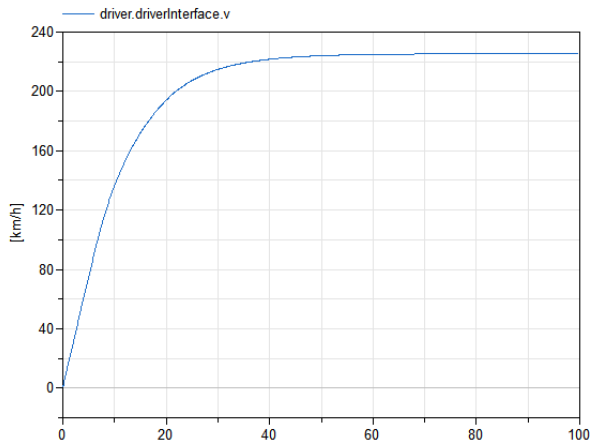


Figure 8 Maximum acceleration and maximum speed

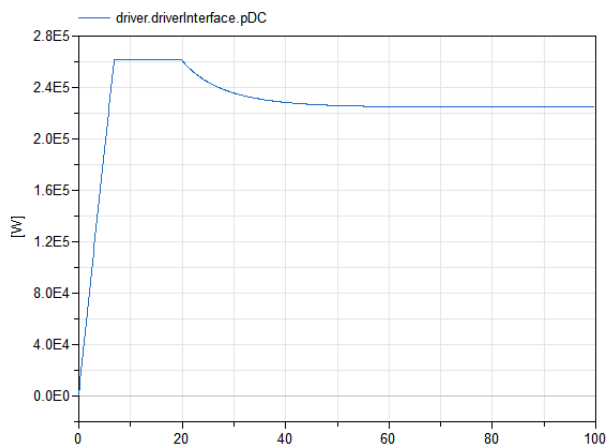


Figure 9 Maximum DC power

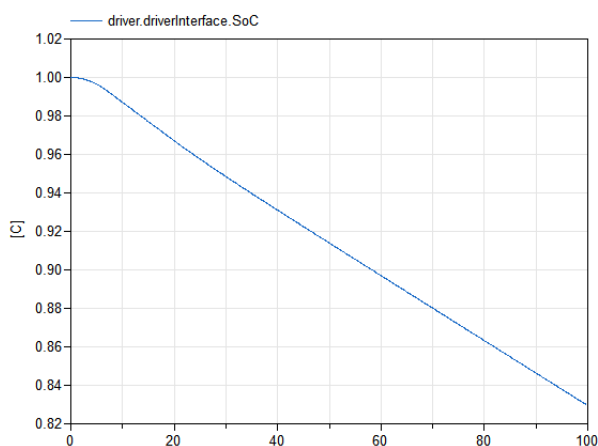


Figure 10 SOC at maximum torque / power

4 Conclusions and Outlook

During the first semester of the three semester Master of Applied Research course the EMOTH Library based on the Modelica Vehicle Interfaces Library has been developed. The library shall provide a flexible framework for longitudinal simulations of electric vehicles to investigate energy consumption during a defined driving cycle including accessories and range extenders as well as the ability of the vehicle to follow the desired driving cycle with prescribed acceleration and deceleration.

For the project members of the E-Mobility Cluster Regensburg it is possible to test new components in an early design stage in the context of the full vehicle.

Up to now, full vehicle models based on the components of the library with estimated but realistic vehicle parameters have been tested successfully with Dymola 2017 FD01. Since the library is conformant to the Modelica Language Specification, it should be possible to run the examples in other Modelica tools, too. In a first test using OpenModelica 1.11 beta 3 the model translates but during simulation errors occur which have to be investigated.

During the following two semesters it is planned to gather the necessary parameters of both the E-bus “EMIL” of Regensburg as well as the E-Smart designed by the Faculty of Electrical Engineering and Information Technology of OTH Regensburg. With these parameters, the results of simulations following real driving cycles shall be validated against measurements. Furthermore, several components of the library shall be refined and improved, such as the electric drive and the battery / energy storage.

It is planned to make the library public available under the Modelica License 2.

References

- [1] VehicleInterfaces on the Modelica website: <https://www.modelica.org/libraries> (visited 2017-01-21)
- [2] Michael Tiller, Paul Bowles, Mike Dempsey: Development of a Vehicle Modeling Architecture in Modelica, 3rd International Modelica Conference 2003, Linköping.
- [3] D.Schramm, M.Hiller and R.Bardini: Modellbildung und Simulation der Dynamik von Kraftfahrzeugen, Springer 2013.
- [4] M.Mitschke and H.Wallentowitz, Dynamik der Kraftfahrzeuge, Springer 2014.
- [5] S.Breuer and A.Röhrbach-Kerl, Mechanik des bewegten Fahrzeuges, Springer-Vieweg 2015
- [6] D.Schröder, Elektrische Antriebe: Regelung von Antriebssystemen, Springer 2009.

Simulating a Variable-structure Model of an Electric Vehicle for Battery Life Estimation Using Modelica/Dymola and Python

Moritz Stüber¹

¹University of Applied Sciences Vorarlberg, Austria

Abstract

A variable-structure model (VSM) of a battery electric vehicle used for simulating the ageing of the battery pack is presented. The operating principle of the software used to simulate the models is described and a brief summary of the state of science and technology regarding the simulation of VSMs is given. By comparing the performance of the VSM to a conventional model, it is found that the simulation time does not necessarily decrease when replacing a model with a variable-structure version. However, the VSM has advantages regarding the handling of the result files and the possibility to analyse the results.

Keywords: *Variable-structure Model, VSM, Modelica, Dymola, Simulation*

1 Introduction

In recent years, the use of electrified or electric power trains in passenger cars has gained renewed research interest. However, most currently available battery electric vehicles (BEVs) have a rather low driving range caused by the low energy density of the battery pack in comparison to conventional fuel. The battery pack is not only the single most expensive part of the BEV, but also subject to significant degradation. Consequently, car manufacturers have to simulate the state of health (SOH) of the battery for two main reasons: on the one hand, it has to be made sure that a vehicle still meets the requirements when the battery has aged. On the other hand, it is necessary to estimate possible warranty costs caused by battery packs that reach the end of their life ahead of time.

With age, the capacity of the battery decreases and the impedance increases. Since this affects the electrical quantities within the vehicle, there should be no separation between the model used for simulating the driving behaviour and the ageing model. Instead, the model should combine electric, mechanical and thermal models and thus be capable of calculating feedback effects. Due to the fact that the battery life has to be simulated for 8 to 15 years, simulating this complex model takes a substantial amount of time. Therefore, a speed-up of the simulation is desirable.

Vehicles are idle for the majority of their life. During this time, the complexity of the model used for simulation can be much lower than during driving. Implementing this change in the level of detail of the model leads to

a so-called variable-structure model (VSM). Simulating a VSM is potentially faster and/or more accurate than a conventional simulation, but VSMs are not yet supported by commonly used modeling languages and simulation environments.

In this paper, the results of investigating the question “Does the simulation time of the ageing process of a battery used in electric vehicles decrease if the system is systematically modeled as a variable-structure model?” (Stüber 2016) are presented.

In order to answer this question, four steps were taken. First, a conventional model capable of estimating the battery life of a BEV was assembled using Modelica/Dymola. Then, a variable-structure version of this model was implemented and a software capable of simulating the VSM using Dymola’s Python interface was written. Last, a series of simulations was performed in order to investigate the influence of the model and solver settings on the time needed to execute the simulation.

In the next section, a brief introduction to the modeling and simulation of variable-structure models is given.

2 Variable-structure Models

In the context of modeling and simulation, variable-structure models are models that consist of *several* sets of equations describing the same physical system. Each set of equations is called a “mode” of the model; exactly one mode is active at all times during the simulation. The changes between the modes are denoted “transitions”. For each mode, the transitions define which mode becomes active next, the condition c_i for changing and information i_j on how to initialize the next mode (Mehlhasse 2015, chapter 3.2)—compare Figure 1.

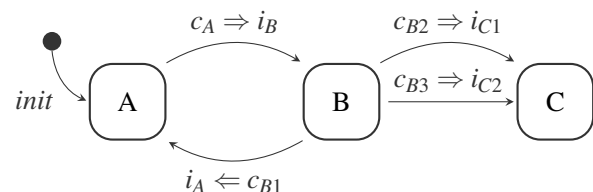


Figure 1. Graphical representation of a VSM with three modes A, B and C

As the name suggests, variable-structure models exhibit varying *structural* properties, which can either refer to the properties of a real system or to the properties of the system of equations used to mathematically describe it.

In *variable-structure systems (VSSs)*, the change in structure is a property of the physical system. Failure situations like breaking mechanical or electrical connections or so-called agent-based systems represent examples of VSSs.

On the other hand, in *variable-structure models (VSMs)*, the change in structure is a result of abstraction; in other words the result of *creating a model* of a system. For example, if detailed information about the switching process is not relevant for an experiment, ideal switches are used. Since ideal switches can attach or detach whole parts of a model, a change in the structure of the underlying set of equations occurs: variables and relations can change and the system of equations can grow or shrink in size.

Variable-structure models can be used to implement changes in the behaviour or the required level of detail of the system under investigation. Components can be added and removed during the simulation. This is necessary for simulating agent-based systems or changing the discretization of a model by changing the number of identical components; as well as for implementing ideal switches, breaking connections or limiters, leading to the dynamic addition or removal of parts of the model (Mehlhasse 2015, chapter 4; Zimmer 2010, chapter 1.2). Furthermore, changing the solver and the solver settings during a simulation is possible when simulating VSMs.

Despite the multitude of use cases for VSMs, only very few simulation environments support their definition and execution. According to Zimmer (2010, chapter 1.3), there are two main reasons for this: first, current modeling languages lack the expressiveness required to accurately define the structural variability. Second, it is technically very challenging to simulate the resulting models.

Three different concepts have been developed for implementing a simulation engine that can handle variable-structure models: maximal state-space, hybrid decomposition and dynamic causalization.

- In a *maximal state-space*, “state events switch on and off algebraic conditions, which freeze certain states for certain periods” (Breitenacker 2008, page 9). The maximal state-space-model is static and can therefore be simulated using conventional tools.
- In contrast, when using the *hybrid decomposition*-approach, the VSM is split into its modes, which have a static structure and can be executed sequentially. The order of execution is controlled at a meta-level, either within the simulation environment or externally.
- The most flexible, but also the most challenging approach from a technical point of view, is called

dynamic causalization. Here, the model is re-causalized if necessary, which is impossible when using the usual translation–compilation–execution sequence.

Because structural changes *always* cause events and VSMs thus represent a generalization of hybrid models, modeling languages that support them need to provide a generalized way to *define* events in order to achieve the required expressiveness.

The most recent, Modelica-based¹ attempts to implement a modeling language and a corresponding simulation environment that support the simulation of VSMs are Sol, DySMo and MoVasE as well as an unreleased prototype of Dymola.

Sol Sol (Zimmer 2010) is an experimental language which is intended to serve as a proof of concept for simulating variable-structure models using dynamic causalization. Conditional index changes as well as the local definition of modes within components are supported. Sol therefore allows the modeling and simulation of “almost arbitrary structural changes” in a truly object-oriented manner (Zimmer 2013), but the proposed language constructs and simulation techniques have not been integrated into commonly used languages and tools yet.

Dymola Elmqvist, Mattsson, and Otter (2014) presented an approach to simulate VSMs in Dymola. It represents an extension to the capabilities of the synchronous state machines defined in Modelica 3.3 and was implemented in a prototype version of Dymola 2015. Instead of defining additional language elements, the semantics of the existing language was extended.

Using this approach, a large, but limited class of VSMs could be simulated. Because it is not necessary to process the model definition using an interpreter, like in Sol, the simulation is significantly faster. However, variable-structure models with varying index could not be simulated. This possibility was added later by extending the Pantelides algorithm (Mattsson, Otter, and Elmqvist 2015), but it has not been added to the official version of Dymola yet.

DySMo (Dynamic Structure Modeling) is a Python application that allows the simulation of VSMs (Mehlhasse 2015, chapter 7). Each mode is represented by an executable model with static structure that terminates if the condition for a transition is triggered. Upon termination, a variable is set that defines

¹Tools that support VSMs to a certain degree, but rely on different modeling concepts have been developed outside the context of simulating physical systems, for example Hydra (functional programming), JAMES (systems biology) and ANYLOGIC (large-scale agent-based systems). They are not suited for simulating the BEV model and thus not considered further.

the cause for the transition. DySMo then reads and stores the results, initializes the next mode depending on the cause for the transition and starts the next simulation. All modes and transitions have to be maintained manually.

MoVasE Esperon, Mehlhase, and Karbe (2015) propose a methodology to append structural changes to existing models by externally defining conditional component exchanges. The tool MoVasE (**Modelica Variable-structure Editor**) implements the proposed solution. The aim of MoVasE is to provide a platform for facilitating the investigation of VSM-design. In contrast to DySMo, MoVasE does not require the user to create and maintain all modes manually. By defining the structural variability through conditional component exchanges, many modes can be created and maintained. However, the flexibility of this approach is still limited with regard to the dynamic addition and removal of components.

In conclusion, the approaches taken by Sol and Dymola solve the most important technical problems, but they have not been integrated into standard languages and tools yet. Therefore, script-based approaches (DySMo, MoVasE) are necessary for studying the benefits and drawbacks of using variable-structure models.

Three levels of complexity can be distinguished when creating VSMs: in the simplest case, the modes are defined on the highest level of the model, as shown in Figure 1. Second, individual components of the model can exhibit modes. In the most complex case, the modes are a result of the addition and removal of components, like in agent-based systems.

From the point of view of the simulation engine, a variable-structure model *always* consists of modes defined at the highest level of the model, which is called the *factorized* version of a VSM (Mehlhase 2015, chapter 5.1.2). Depending on the tool used for simulation, it might be necessary to manually create the factorized version of the VSM. Additionally, it has to be made sure that the VSM is *valid*. This includes avoiding chattering or unphysical transitions and unphysical factorized modes. A set of guidelines that is intended to help with the creation of valid VSMs was formulated by Mehlhase, Esperon, and Karbe (2015).

In addition to many small examples that were used to verify a certain language/tool (Zimmer 2010, chapter 11; Mehlhase 2015, chapter 8), several examples of the successful application of variable-structure models for solving real-world problems have been published (Krüger, Mehlhase, and Schmitz 2012; Mehlhase, Esperon, Bergmann, et al. 2014; Möckel, Mehlhase, and Nytsch-Geusen 2015). In these examples, a significant reduction of the overall time needed to simulate the system could be achieved due to a big difference in the complexity of the modes and a low number (≤ 40) of mode switches.

So far, no attempt to use a VSM of a BEV for estimating its battery life has been published. In the next section, the models used by the author are described, followed by a description of the observed advantages and disadvantages.

3 Implementation

In order to assess the usefulness of using a VSM of a BEV for battery life estimation, both a conventional and a variable-structure model were assembled and simulated. Implementing the *components* used for assembling the models was *not* part of this work; they are part of the commercial *Electrified Powertrains Library* and the *Battery Library* developed by Dassault Systèmes².

The conventional model consists of nine main components: the driving cycle, the driver model, a control unit, the models of the energy supply, the electric power train, the auxiliary loads, the chassis and the environment, as well as the charger model. In contrast, the VSM has two global modes that reflect the “operating modes” of the vehicle (Figure 2). The model that represents the “driving” mode does not contain the charger and its control logic, while the model that represents the “idle” mode *only* contains the battery model, the charger model and the environment model.

The inputs of both models are the desired speed of the vehicle, the time frames the charger is plugged in, and the ambient temperature over time. The VSM additionally has a schedule for switching between modes based on the driving behaviour. All inputs are loaded from externally stored files which are generated using a script. The script allows the convenient definition of usage scenarios and ensures that the resulting profiles are consistent.

The system of differential algebraic equations (DAEs) of mode “idle” has 915 scalar unknowns and equations and 25 continuous time states. Mode “driving” consists of 3062 scalar unknowns and equations and has 29 continuous time states; the conventional model of the BEV has 3219 scalar unknowns and equations and 38 continuous time states. Since the system of ordinary differential equations (ODEs) that needs to be integrated differs only slightly, it can be expected that the speed-up is small, but noticeable. It would be straightforward to use more complex models of the power train during driving because a template is used that allows the simple exchange of models. This would likely increase the performance gain of the VSM compared to a conventional model, but also increase the overall simulation time in both cases.

The VSM is simulated using a software written in Python that allows the definition and simulation of VSMs with globally defined modes using Modelica and Dymola. The software is comparable to DySMo and described in section 6.

²<http://org-www.3ds.com/products-services/catia/products/dymola/industry-solutions/>

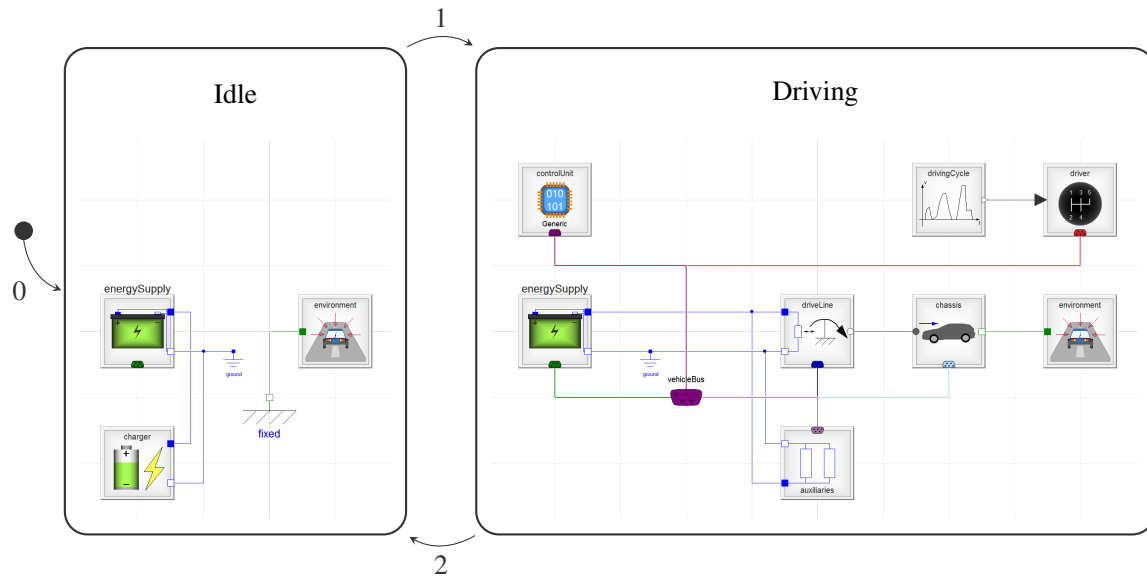


Figure 2. Variable-structure model of the BEV

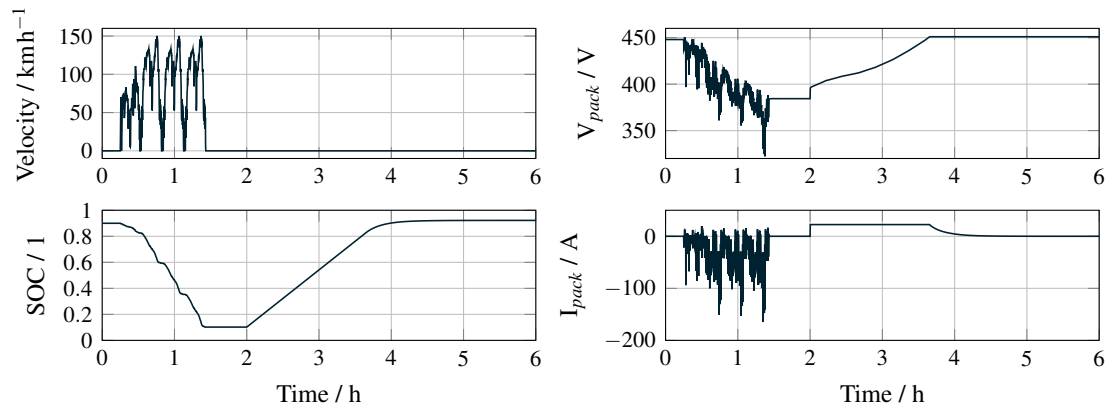


Figure 3. Selected results of the experiment used for model validation

4 Simulation Results

The parameters of the models were chosen with the intention to reflect typical values for BEVs. The models were checked for plausibility by simulating a short driving cycle followed by charging. The driving phase starts after 15 min standstill and takes approximately 70 min. During this time, a distance of 103 km is covered and the battery is discharged from 90 % state of charge (SOC) to 10 % SOC, which corresponds to a total consumed energy of approximately 16.2 kWh. About half an hour after the driving cycle is completed (at $t = 2$ h), the charger is plugged in. In Figure 3, selected results of the simulation can be seen. There is no noticeable difference between the results of the conventional simulation and the results of the VSM.

Knowing that the models are properly parameterized, a simulation of the SOH spanning several years could be performed. Two usage profiles were created, which are shown in Figure 4 and Figure 5. Both span a week and are used repeatedly if longer scenarios are needed. In the

first scenario, the battery is always charged fully, whereas in the second profile, the SOC only varies from approximately 60 % to 40 %.

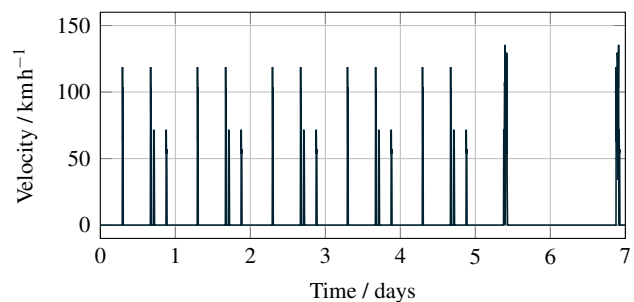


Figure 4. Demanding usage scenario: desired velocity. When using this driving cycle, 658 km are driven per week; the yearly mileage amounts to 34238 km.

The higher usage and the storage at higher SOC should result in faster ageing of the battery in the first scenario.

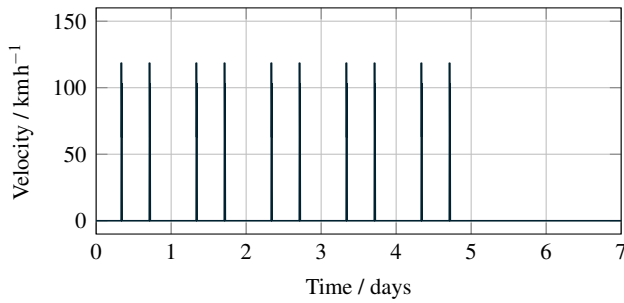


Figure 5. Relaxed usage scenario: desired velocity. When using this driving cycle, 289 km are driven per week; the yearly mileage amounts to 15040 km.

The simulation results are shown in Figure 6.

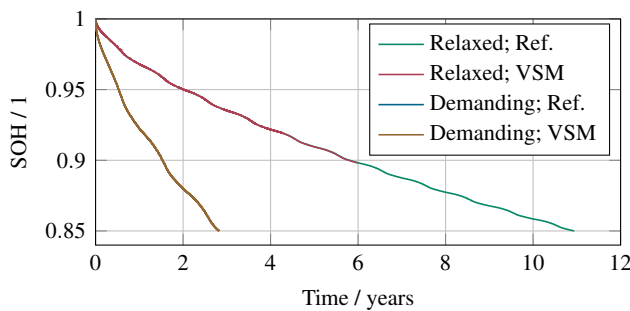


Figure 6. State of health of the battery

For the assessment of the battery's ageing behaviour, the models were simulated until the SOH falls below 0.85. This value was chosen because a compromise between the time needed for simulation and the length of the calculated trajectory had to be found and a ΔSOH of 0.15 is regarded meaningful. Moreover, in this case, more data does not mean more information because all input and model parameters are estimated values anyway.

There is no significant difference between the result of the conventional model and the VSM: the relative error is in the range of $\pm 0.02\%$.

5 Advantages and Disadvantages of the Variable-structure Model

In Figure 7 and Figure 8, the elapsed CPU time during a 2-week simulation of the BEV model using the “relaxed” driving cycle (Figure 5) is shown. The CPU time comprises the time needed for initializing the system(s) of equations and the time needed for integration. The same solver settings are used for both the conventional and the variable-structure model.

In both figures, the difference between driving and standing can be seen clearly due to the step wise increase of the elapsed time resembling a staircase. This is the result of using a solver with variable step size: during driving, only a small step size can be used due to the dynamic

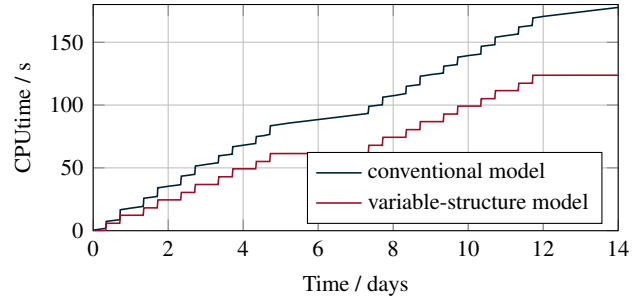


Figure 7. CPU time; dense output enabled

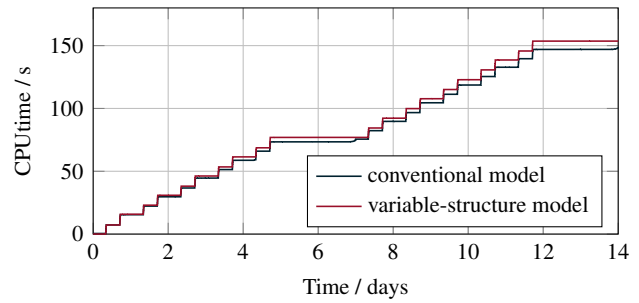


Figure 8. CPU time; dense output disabled

changes of the state variables. In contrast, the step size can be greatly increased when the vehicle is idle.

There is a significant difference in the elapsed time depending on whether dense output is enabled or not. If dense output is enabled, the conventional model takes longer to calculate, depending on the size of the output interval. The effect is especially noticeable during the idle phases because during this phase, the *necessary* step size calculated by the step size control algorithm is usually *bigger* than the desired output interval.

However, when dense output is disabled, the conventional simulation becomes *faster* than the variable-structure model. A reason for this is that each system of equations (mode) needs to be *initialized*.

In Figure 9, a more detailed account of how much time is spent on which part of the simulation is given. On the right hand side, two pie charts visualize the data listed in the table on the left. The area of the pie charts corresponds to the total duration of the simulation, whereas the slices denote the time spent on the initialization of the systems of equations, the integration itself and the post-processing of the data. Additionally, time is needed for writing the result files and “consumed” by the operating system for other processes. There is a striking difference between the conventional and the variable-structure model: in the former, the integration takes 99.9 % of the time, but only 52.2 % of the time needed for simulating the latter is actually spent on the integration. Therefore, the VSM takes longer to simulate, even though the integration finishes 0.5 h earlier. A large, *unnecessary* part of the overhead is caused by Dymola's Python interface: since the `simulateExtendedModel()`-command only

Task	Reference	VSM
Initial Compilation	4.5 s	7.4 s
Initialization	1.3 s	1.0 h
Integration	7.7 h	7.2 h
Recompilation	—	5.6 h
Reading .mat-files	≈ 0.0 s	19.3 s
Post-processing	≈ 0.0 s	1.2 h
<i>Total Duration</i>	7.7 h	15.1 h

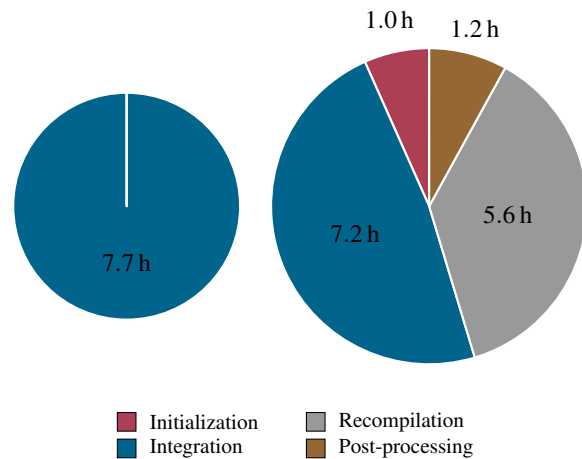


Figure 9. Comparison reference model–VSM: time measurements for a simulation of the demanding scenario for 3 years. In the variable-structure model, 5304 mode switches were performed.

supports passing real numbers for initialization, modifiers have to be used for passing the necessary vectors and attributes. This causes a recompilation of the model at each mode switch. By finding a workaround for this, the overall simulation time of the VSM could be reduced drastically. A further reduction could be achieved by improving the implementation of post-processing.

When working with VSMs, besides the restriction to use the same *model* for all phases, also the restrictions to use the same *settings* and *result files* for all phases no longer exist. Therefore, the question “Does it *make sense* to use a variable-structure model of a BEV for estimating its battery life?” may still be answered with “yes”, even if the time needed to calculate the output trajectory does *not* decrease very much.

One problem that arises when estimating the battery life using a full vehicle model is the size of the result file, which depends on the settings for dense output and the length of the simulation. In order to limit the amount of data stored in the result file, the output interval needs to be set to a constant, high value (for example 24 h when simulating 15 years). Additionally, it is necessary to select the set of variables that has to be stored in advance, for example by using Dymola’s `__Dymola_selections`-annotation. This means that all detailed information calculated during the course of the simulation is irretrievably lost and not available for analysis. When simulating a VSM, this problem is much less likely to occur as every mode has its own result file and the individual simulation times are much shorter³. Moreover, it is possible to store the results in a high resolution when the vehicle is driving and in a low resolution otherwise. Therefore, it becomes possible to perform a detailed analysis of the driving behaviour at the end of the battery’s life.

³Strictly speaking, the memory limitations could also be avoided by implementing the possibility to split up result files when performing a conventional simulation in Dymola.

6 PyVSM

In this section, the software used for implementing and simulating the VSM of the BEV is described. It is called PyVSM and supports the simulation of factorized variable-structure models using Dymola’s Python interface. PyVSM is intellectual property of Dassault Systèmes Deutschland GmbH.

The basic idea of PyVSM is to use Dymola for simulating the modes and Python for switching between them. Therefore, it is required that each mode of the *factorized* VSM is a complete Modelica model that can be simulated using Dymola. Modes, transitions and solver settings of the VSM are defined using JSON-files. When a condition of a transition becomes true, the simulation of the currently active mode terminates and the initialization of the next mode is initiated by PyVSM based on the results of the previous mode.

In Figure 10, the processing steps taken to simulate a VSM in PyVSM are shown in more detail. First, the JSON-file used for the definition of the VSM is read. Then, for each mode, a directory used during simulation is created and the Dymola-interface is instantiated with the working directory set to the previously created folder. The actual simulation of the VSM starts by executing the initial transition in order to set the initial values for the simulation. The active mode is set according to the definition of the transition and the simulation is started. Upon termination, the relevant results are loaded to PyVSM, including the variable `transitionID`. Its value corresponds to the numerical identifier of the transition that has to be executed next. If and only if it is 0, the end of the simulation is reached. After this, the individual results of each mode are concatenated and post-processed. This includes the calculation of characteristic values such as the time needed for initialization or the step size and the resampling of the data to the specified interval length. Last, the post-processed data is saved and plots are generated if desired.

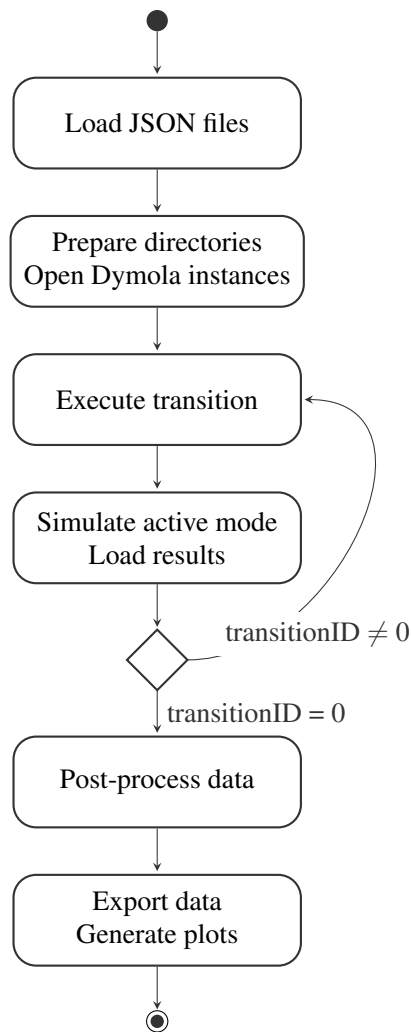


Figure 10. UML-activity diagram of PyVSM

PyVSM is implemented in Python using object-oriented programming techniques. It is capable of simulating factorized variable-structure models using Modelica/Dymola, but, being a prototype, misses advanced features such as automatic validity checks of the input files or a graphical user interface. Additionally, externally controlling the simulation via Dymola's Python interface generates an overhead. Nonetheless, PyVSM provides modelers with the possibility to simulate models that would be difficult or even impossible to implement in a conventional simulation environment in a straightforward manner.

7 Conclusion

A variable-structure model (VSM) of a BEV was implemented with the aim of making the simulation of the battery ageing faster by switching between a complex model used when the vehicle is driving and a simpler model used when the vehicle is idle. Since VSMs are not yet supported by Modelica/Dymola, a software had to be written that provides means to define the structural variability and performs the mode switches.

When simulating the variable-structure BEV model, it

is found that while the CPU time needed for integration decreases, thus matching the expectations, the overall simulation time *increases* due to overhead generated by switching between models. This is caused by the properties of the model on the one hand (low difference in the complexity of the modes, many (> 5300) mode switches) and the implementation of the software used for simulation on the other hand. The overhead comprises the excess time needed for compiling, initializing the systems of equations, reading the result files and post-processing them as well as the unnecessary recompilation of the modes at all transitions. Nonetheless, the VSM allows a more detailed analysis of the simulation result due to memory limitations occurring when using a conventional model.

Acknowledgements

This work was supervised by Markus Andres, who contributed by discussing results and giving helpful advice on plans for further work. Marco Keßler and Markus Andres proofread the manuscript. Further help regarding the used battery models and the experiment set-up was provided by Lukas Rohr. The project was carried out during a paid internship at Dassault Systèmes Deutschland GmbH.

References

- Breitenecker, Felix (2008). "Development of Simulation Software – from Simple ODE Modelling to Structural Dynamic Systems". In: *Proceedings of the 22nd European Conference on Modelling and Simulation (ECMS 2008)*. DOI: 10.7148/2008-0005-0022.
- Elmqvist, Hilding, Sven Erik Mattsson, and Martin Otter (2014). "Modelica extensions for Multi-Mode DAE systems". In: *Proceedings of the 10th International Modelica Conference*. DOI: 10.3384/ecp14096183.
- Esperon, Daniel Gomez, Alexandra Mehlhase, and Thomas Karbe (2015). "Appending Variable-structure to Modelica Models (WIP)". In: *Proceedings of the Conference on Summer Computer Simulation*. SummerSim '15. Chicago, Illinois: Society for Computer Simulation International.
- Krüger, Imke, Alexandra Mehlhase, and Gerhard Schmitz (2012). "Variable Structure Modeling for Vehicle Refrigeration Applications". In: *Proceedings of the 9th International Modelica Conference*. DOI: 10.3384/ecp12076927.
- Mattsson, Sven Erik, Martin Otter, and Hilding Elmqvist (2015). "Multi-Mode DAE Systems with Varying Index". In: *Proceedings of the 11th International Modelica Conference*. DOI: 10.3384/ecp1511889.
- Mehlhase, Alexandra (2015). "Konzepte für die Modellierung und Simulation strukturvariabler Modelle". PhD thesis. Technische Universität Berlin, Fakultät IV – Elektrotechnik und Informatik. DOI: 10.14279/depositonce-4514.

- Mehlhase, Alexandra, Daniel Gomez Esperon, Julien Bergmann, et al. (2014). “An example of beneficial use of variable-structure modeling to enhance an existing rocket model”. In: *Proceedings of the 10th International Modelica Conference*. DOI: 10 . 3384 / ECP14096707.
- Mehlhase, Alexandra, Daniel Gomez Esperon, and Thomas Karbe (2015). “Challenges when Creating Variable-structure Models”. In: *Proceedings of the 5th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, pp. 101–110. DOI: 10 . 5220 / 0005521601010110.
- Möckel, Jens, Alexandra Mehlhase, and Christoph Nytsch-Geusen (2015). “Exploiting Variable-structure Models in the Context of Building Simulations within Modelica”. In: *Proceedings of BS2015*. International Building Performance Simulation Association. URL: <https://www.researchgate.net/publication/301229350>.
- Stüber, Moritz (2016). “Simulating a Variable-structure Model of an Electric Vehicle for Battery Life Estimation Using Modelica/Dymola and Python”. Master’s Thesis. University of Applied Sciences Vorarlberg.
- Zimmer, Dirk (2010). “Equation-Based Modeling of Variable-Structure Systems”. PhD thesis. Swiss Federal Institute of Technology, Zürich. DOI: 10 . 3929 / ethz-a-006053740.
- (2013). “A new framework for the simulation of equation-based models with variable structure”. In: *SIMULATION* 89.8, pp. 935–963. DOI: 10 . 1177 / 0037549713484077.

Model Reduction Techniques Applied to a Physical Vehicle Model for HiL Testing

R. Gillot*

S. Gallagher**

A. Picarelli*

M. Dempsey*

*Claytex Services Ltd. Edmund House, Rugby Road, Leamington Spa, CV32 6EL
{romain.gillot, alessandro.picarelli, mike.dempsey} @claytex.com

**Ford Motor Company Ltd, Dunton Technical Centre, SS15 6EE
sgalla20@ford.com

Abstract

To build a full vehicle model entirely based on physical equations is a challenge (Dempsey M., 2006). To have this model to run fast enough so that it is suitable for Hardware-in-the-Loop testing is even more challenging. The level of detail in the physical representation of the vehicle can always be increased at the cost of simulation time. Even if the performance of the hardware is constantly improving, we still have to compromise.

As part of the MORSE (Model based Real-time Systems Engineering) project, model reduction techniques are developed and applied to a vehicle model. The results in terms of accuracy and simulation speed are then investigated.

Keywords: vehicle model, model reduction, real-time simulation, Hardware-in-the-Loop testing

1. Introduction

MORSE (Model based Real-time Systems Engineering) is a 2-year project in collaboration with Ford and AVL, co-funded through InnovateUK's Towards Zero Prototyping competition. The aim of the project is to develop predictive engine and vehicle models enabling virtual calibration of driveability control features and validation of On Board Diagnostics (OBD) fault paths. In order to satisfy these requirements, we need physical models with a high level of detail. We need, for example, a clutch with a detailed friction model, a gear set with torque reactions, a differential with force and torque reactions, compliant drive shafts, Pacejka tyre model, linear engine mounts, detailed suspensions, a crank angle resolved engine model. We use these models for Software-in-the-Loop (SiL) and Hardware-in-the-Loop (HiL) testing. Whilst simulation time is not a major concern for SiL, the models do have to run in real time and with no overruns to be used in the HiL environment. This is why we need model reduction techniques that will help us simplify our models to improve simulation speed while matching

the behaviour of the full model. The idea is to have two different models for two applications: the fully detailed

model for SiL testing and a reduced version, automatically generated and parameterized from the first one in order to match its results, for HiL testing. In this paper, we present the full vehicle model and its associated level of detail. Then we introduce the model reduction techniques and show how they are applied to each subsystem. The subsystems and their reduced equivalents are tested and the results compared. Finally the full vehicle model as well as the reduced vehicle model are run over a series of Tip-In/Tip-Out manoeuvres in the HiL environment and the trade-off between accuracy and simulation performance is investigated.

2. The Vehicle Model

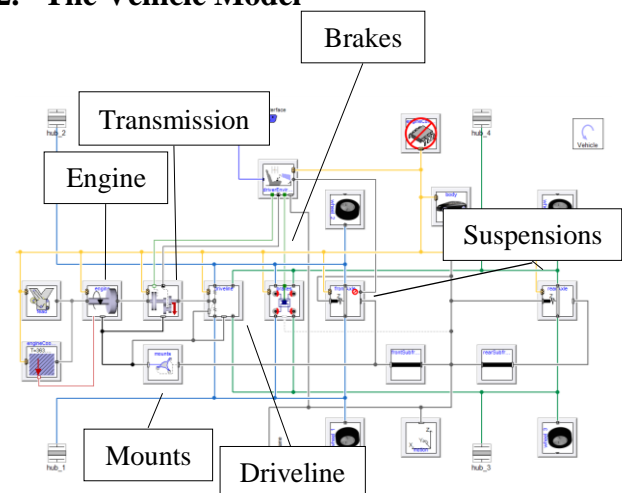


Figure 1. Detailed view of the vehicle model with all the subsystems.

In order to perform the driveability analysis, a certain level of detail is required in the vehicle model.

We require a mounts model using linear springs and dampers to constrain the motion in the three directions (x, y, z) as well as a transmission model with a clutch based on coulomb friction with a reliable handling of the stuck phase and a gear set that models the gears, gear

meshes and mesh losses and takes into account the torque reactions.

In the driveline, the drive shafts need to be compliant and to include backlash. The differential, in the same manner as the gear set, models the gear contact and considers the torque reactions.

The suspensions have a vertical degree of freedom (fore-aft motion can be included) and use a linear spring and damper (other spring and damper models are available if required).

Another critical component in driveability studies is the modelling of tyres; our models thus utilize the Pacejka slip model since it is the most commonly used model to investigate tyre dynamics.

The engine is not studied in this paper and sits outside of the vehicle model, a torque source coupled to a flywheel are used to transmit the torque from the engine to the transmission.

3. Model Reduction

a. Transmission

The physical gear set (Figure 2) is a multibody model (Dempsey M., 2009) that uses physical representations of gears, shafts, bearings and synchronizers. Gear engagement is achieved through translational mechanics flanges (in green in the Figure 2) providing a clamp load to the left or right flanges on the synchronizer dependant on the sign of the clamp force.

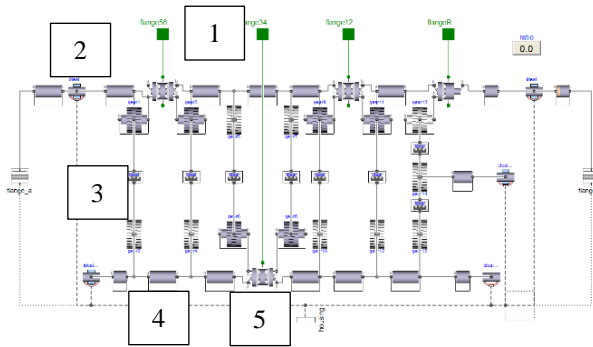


Figure 2. Physical gear set model (1: Translational flange, 2: Bearing, 3: Gear, 4: Shaft, 5: Synchronizer).

This is how the model reduction tool works internally: The physical gear set is run on a test rig (Figure 3) in 1st gear. The speed source ramps up from 0 to 6000 rpm. A load is attached to the gear set. The experiment is repeated several times, varying the load each time (from 30 to 360 N.m). The transmission is thus run over a range of speeds and loads.

This procedure is repeated for all the remaining gears. We now have a loss map for the transmission for all the operating points. This data is stored in a set of data records (one for each gear) through an automated procedure.

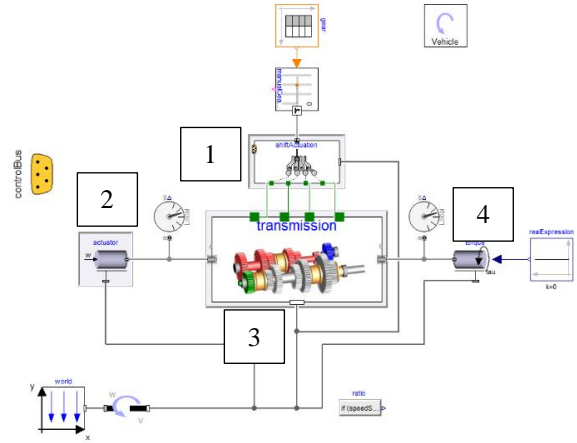


Figure 3. Gear set test rig (1: Shift mechanism, 2: Speed source, 3: Gear set, 4: Load).

The function then extends the reduced gear set model from the PTDynamics library (Figure 4) and populates the lumped losses component with the data records we just created. This lumped losses component interpolates the tables in the data records to give the losses depending on gear, speed and load.

The inertia of the whole physical gear set for each gear is also calculated. The reduced gear set has a lumped inertia component that will be populated with the data we just derived.

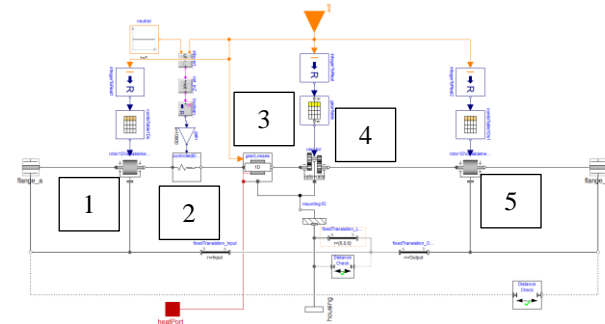


Figure 4. Reduced gear set model (1: Input shaft lumped inertia, 2: Clutch, 3: Lumped losses, 4: Ideal variator, 5: Output shaft lumped inertia).

The gear ratio is applied using an ideal variator which means there is a first order transfer function between the ratio input and the applied ratio. A clutch (item no. 2 in the Figure 4) is used in this model since the ideal variator does not give good results when in neutral.

$$\begin{aligned}\omega_a &= \omega_b * ratio \\ 0 &= \tau_a * ratio + \tau_b\end{aligned}$$

Where ω_a is the angular velocity at flange_a (input flange), τ_a is the torque at flange_a and ratio is the gear ratio. When in neutral gear, the equations become:

$$\begin{aligned}\omega_a &= 0 \\ 0 &= \tau_b\end{aligned}$$

The first equation forces the angular velocity at the input flange of the gear set to be zero and as a consequence the angular velocity of all the components rigidly connected to it, including the engine, to also be zero. The clutch in this gear set is always engaged except in neutral. The equation in the variator sets the angular velocity at the output flange of the clutch to zero but the input flange is free to rotate as the clutch is disengaged.

Let us run a fully detailed 6-speed gear set and its reduced version we derived using the model reduction function and compare the results. To do so, we run the models in all the gears, feeding in a torque of 80 N.m (see Figure 5).

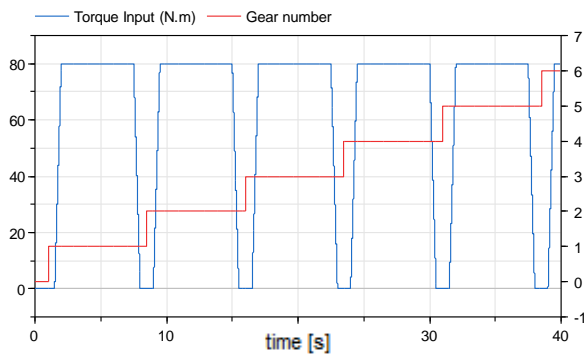


Figure 5. Test rig gear and torque inputs.

We can now have a look at the torque at the input and output of the two gear sets with different levels of detail:

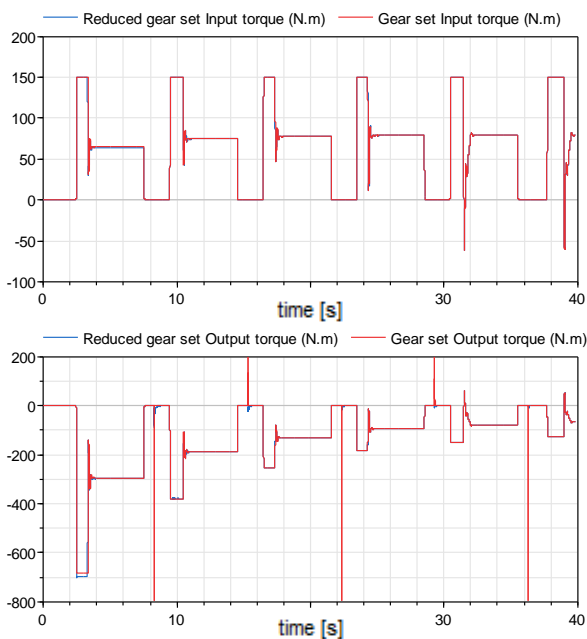


Figure 6. Input and output torque of a 6-speed gear set and its reduced equivalent.

The results of the reduced model match very well those of the full gear set, there is only a small discrepancy at around 3s. This is because in the table of lumped losses that we got using the function, the torque ranges only up to 350 N.m. The torque being outside of this range at the beginning of the test, Dymola has to extrapolate from the table of losses which leads to a small inaccuracy. The torque range will be extended in future work.

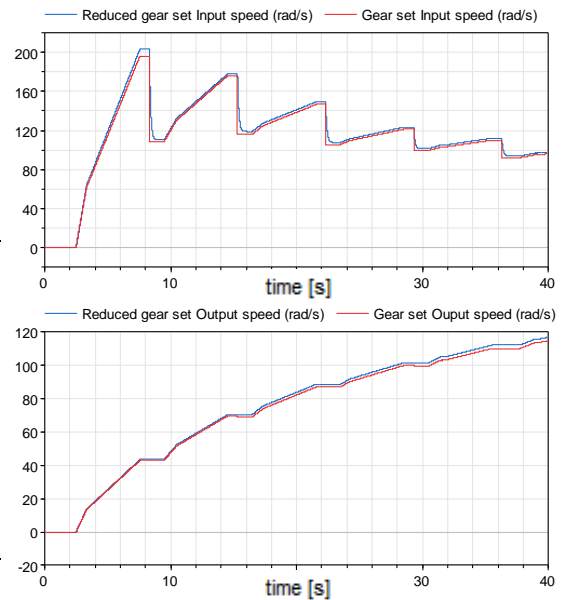


Figure 7. Input and output shaft speed of a 6-speed gear set and its reduced equivalent.

The speed curves match well too. The speed of the reduced gear set is slightly overestimated though. This comes from the inaccuracy in the torque curve at the beginning of the simulation (see Figure 6) which therefore calculates an acceleration that is too big. The relative error in angular velocity then gets carried until the end of the simulation but its magnitude does not increase.

The test lasts for 40s and the solver used is Radau II – order 5 stiff with a tolerance of 1e-5, this solver will be used to test all the subsystems in the following paragraphs. The improvements in terms of simulation performance are shown in the following table:

	Full gear set	Reduced gear set
Simulation time (s)	4.33	0.95
State events	204	36
Jacobian-evaluations	302	155

Most of the events happen during gearshift (see figure 8 below) so the savings in simulation time depend a lot on

the type of test we run (i.e. how frequently we change gear).

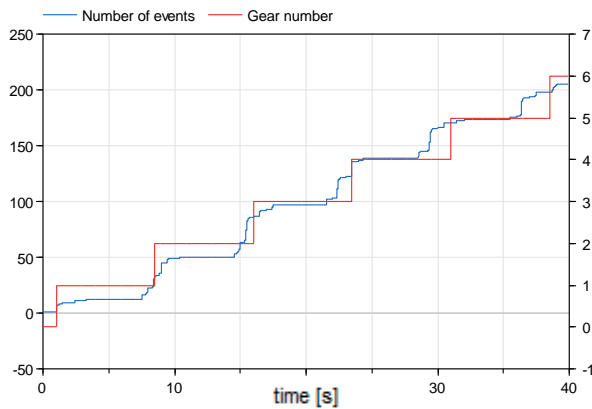


Figure 8. Correlation between number of events and gearshift.

We presented in this section a model reduction tool which is automatic, creates a reduced model that gives very similar results to the full model and runs faster. The new model is however a one-dimensional rotational model which is then not suited for studies where force or torque reactions are of prime importance.

b. Driveline

Here we take advantage of the fact that we are, in the scope of the MORSE project, only performing straight-line manoeuvres. The results we get on the left side of the car (wheel angular velocity, suspension's spring force and position, driveshaft torque etc.) are thus very similar to the ones on the right-hand side, allowing some simplifications. We can use ideal force and torque sources to replace the physical actuators (translational and rotational springs and dampers) on one side of the vehicle. We arbitrarily chose to reduce the components on the right side. In the case of the driveline, we then reduce the right driveshaft, and keep the left one unchanged.

The differential required adaptation since it now only needs to transfer torque to one driveshaft. A standard open differential would transfer all the torque coming from the transmission to the right driveshaft as there is no load on it. This new model splits the torque independently of what is connected to its flanges. This approximation works because we only test the vehicle in a straight line and we assume that the road is ideal (i.e. uniform friction coefficient, no bumps or holes).

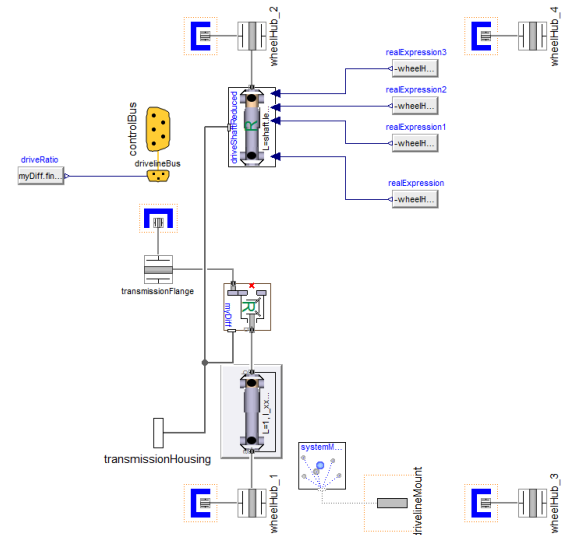


Figure 9. Driveline model with a complete left-hand side driveshaft (bottom) and a reduced right-hand side driveshaft (top).

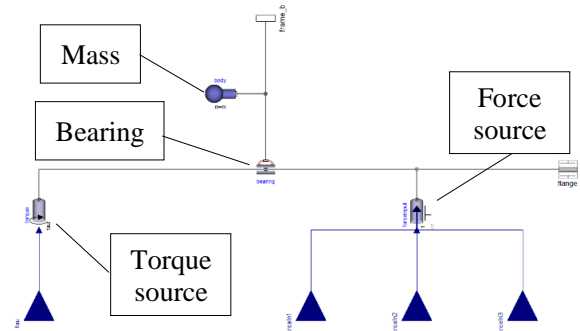


Figure 10. Reduced driveshaft using force and torque sources.

The reduced driveline uses ideal force and torque sources to replicate the behaviour of the other non-reduced driveshaft. The inputs to these force and torque sources are set to the sensed values in the non-reduced driveshaft.

We can switch between the full and reduced driveline by just double-clicking on the driveline subsystem at the vehicle level (see figure 1) and choosing between the two models. There is a Boolean parameter that is used to conditionally enable or disable components. The reduced model's parameters are linked to the parameters from the full one so we do not need extra parameterization when switching between models.

We test the reduced driveline with a trapezoidal torque input and observe the torque and angular velocity at the wheel hubs:

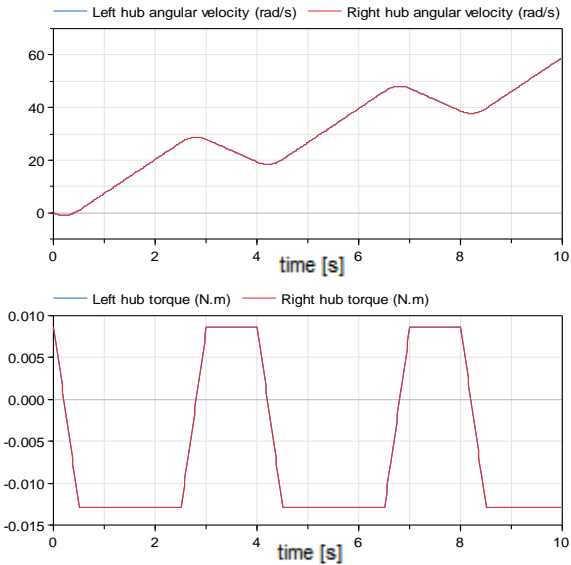


Figure 11. Angular velocity (top plot) and torque (bottom plot) at both front wheel hubs.

The results match perfectly. The benefits in terms of simulation speed and number of events are not shown in this section since the driveline itself is a subsystem that runs relatively quickly. The results will be investigated when testing the full vehicle model.

c. Suspensions

In this paper, we consider a one degree of freedom independent suspension with anti-roll bar. An optional steering connection can be used but we leave the model empty here since we only want to run the vehicle in a straight line. This empty steering model holds the steering frame in a fixed position. The linear anti-roll bar model uses the difference in z-heights to calculate a roll angle and apply a reaction torque.

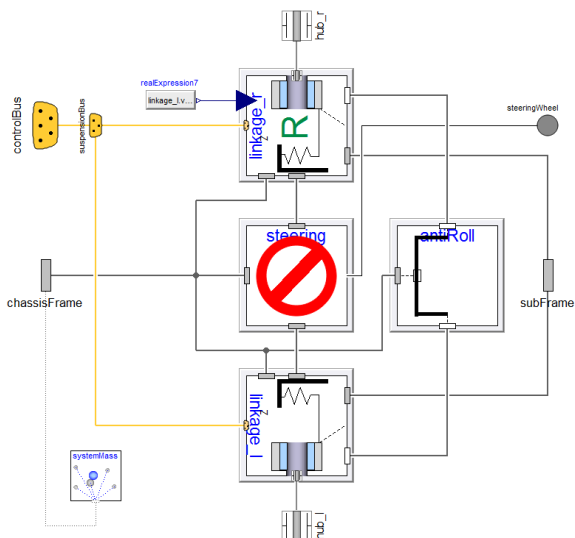


Figure 12. Front suspension model. The left linkage (bottom) is a physical suspension model while the right one (top) is reduced.

The left suspension is kept physical while the right one is reduced.

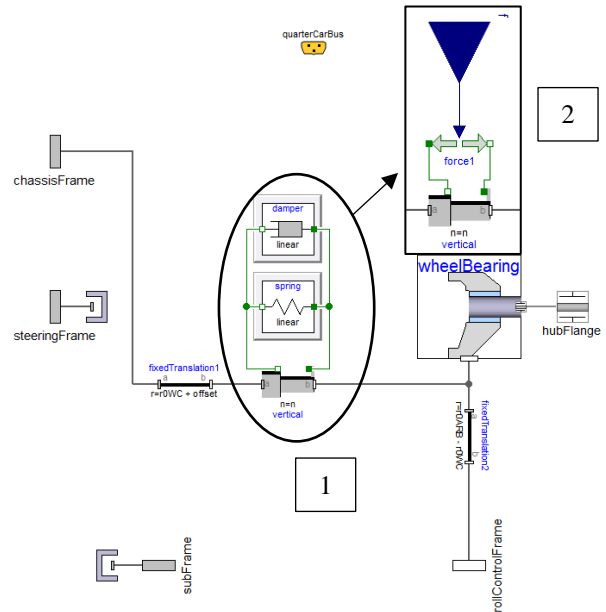


Figure 13. Physical suspension with spring and damper (1) and its reduced equivalent using a force source (2).

The suspension model only allows a vertical degree of freedom. It uses a linear spring and a linear damper. The fast oscillations that can happen when running this model are computationally very expensive.

In the reduced suspension model, the spring and damper are replaced by an ideal force source fed with the force value read at the full suspension model's flange. The rest of the model, which is not computationally very intensive, is kept identical between the left and right side of the vehicle.

We test the suspensions on a test bed with a trapezoidal position input at both hubs.

In this ideal experiment, where the desired behaviour of the suspensions is exactly similar on both sides, the results of the reduced model match perfectly those of the full model. When tested in a vehicle, the forces and torques applied to the left and right suspension hubs will be slightly different, even during a straight-line manoeuvre (the effective rolling radius is never equal in all the wheels, the repartition of the vehicle mass is never perfect, etc.). The reduced model will ignore these differences and produce the exact same results on both sides. The inaccuracies being extremely small, they are completely acceptable for the applications targeted in the MORSE project.

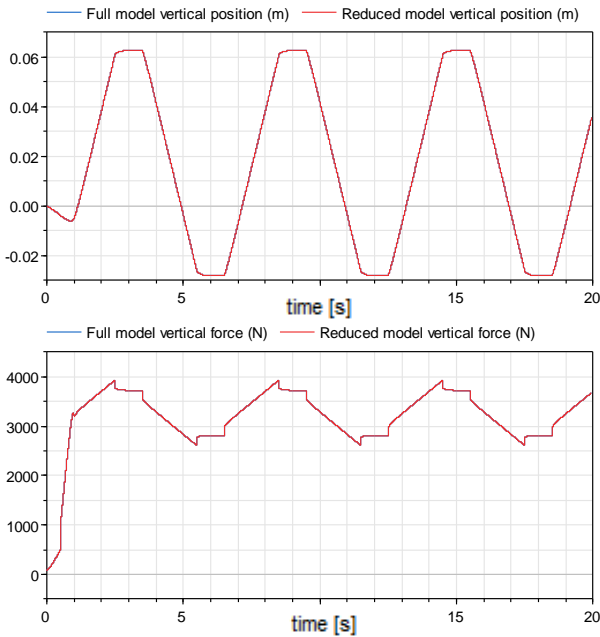


Figure 14. Suspension's hub vertical position (top graph) and vertical force (bottom graph) for both the complete and reduced model.

d. Wheels

The wheels are reduced in the same way, a force and a torque actuator are used to account for the tyre dynamics in the front left and rear right wheels.

Once again we have to point out that in reality, each one of the tyres would behave slightly differently, even in a straight line. The reduced model ignores these differences and replicates exactly on the right side of the car what happens on the left side.

The reduced wheel model is not presented in detail in this paper since it is generated following the idea as the drive shafts and the suspensions.

4. Results

a. In Dymola

Hardware specifications: computer with Windows 10, processor is Intel® Core™ i7-4790K @ 4.00 GHz Quad-core.

In this section, we run a vehicle with several levels of model reduction on a series of Tip-in/Tip-out manoeuvres in 2nd gear. The levels of model reduction are as follows: Level 1: Full vehicle model. Level 2: Vehicle with reduced transmission only. Level 3: Vehicle with reduced transmission and reduced driveline. Level 4: Vehicle with reduced transmission, reduced driveline and reduced chassis (suspensions and wheels). Level 5: Vehicle with reduced transmission, reduced driveline and reduced chassis and only allowing longitudinal motion.

It is important to note that the interface of all the vehicle models is the same as they need to be able to dialog with the ECU without missing information. The simple vehicle model is thus capable of sending and receiving the same signals as the most detailed one.

The model is run first in Dymola. The simulation lasts for 56s. The solver settings are: Step size = 0.0005s, tolerance=1e-5, inline integration method = implicit Euler. This has indeed proven to be the quickest inline integration method for our application. The step size has been calculated to be the biggest time step that gives correct results when running the crank angle resolved engine model, which is the subsystem that requires the smallest sample rate.

The conditions of the tests are different from the conditions of the tests of the individual subsystems since we wanted to run the vehicle on a real manoeuvre like Tip-In, Tip-Out.

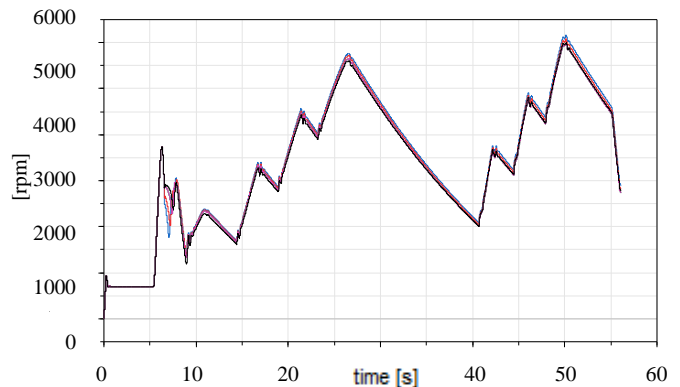


Figure 15. Engine speed. Blue: Level 1, Red: Level 2, Green: Level 3, Magenta: Level 4, Black: Level 5.

The maximum error in engine speed for each level of reduction is respectively: 1.24%, 2.69%, 2.68% and 3.05%.

The biggest error occurs at around 7s when we engage the clutch after engaging 1st gear (i.e. at pull-away when the vehicle starts moving). The magnitude of the error after that moment does not increase, it remains constant until the end of the simulation.

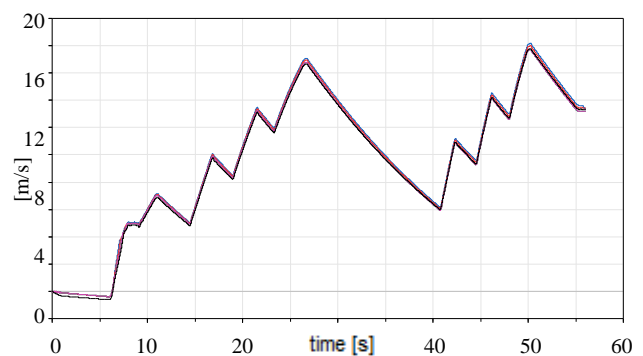


Figure 16. Vehicle speed. Blue: Level 1, Red: Level 2, Green: Level 3, Magenta: Level 4, Black: Level 5.

The maximum error in vehicle speed for each level of reduction is respectively: 1.01%, 2.41%, 2.40% and 2.60%.

We see a slightly negative vehicle speed at the beginning of the experiment, when the engine is idling and the vehicle is in neutral. This is attributable to the Pacejka tyre model which is inaccurate at very low vehicle speeds.

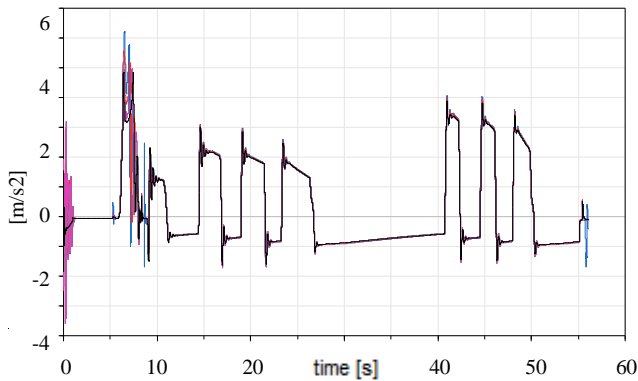


Figure 17. Vehicle acceleration. Blue: Level 1, Red: Level 2, Green: Level 3, Magenta: Level 4, Black: Level 5.

The acceleration plot shows good correlation between the models. There are oscillations at the beginning due to non-optimal initialisation and during clutch engagement at 7s.

In the table below, a time overrun happens when a time step in Dymola lasts longer than the corresponding amount of time in real life. For example, if we choose a step size of 0.5 ms, it should take less than 0.5 ms for the hardware to perform all the calculation before moving to the next step. Otherwise, all the equations do not have time to be solved before the next step and the results cannot be trusted anymore so this has to be avoided.

Simulation performance summary:

	Simulation time (s)	Number of events	Overruns
Level 1 (Full model)	62.2	46	>50
Level 2	42.5	26	>50
Level 3	36.2	28	>50
Level 4	33.8	27	>50
Level 5 (Fully reduced model)	16	16	12

We can see from the table above that each level of reduction improves the simulation time. The number of

overruns of the first four models is quite high and even though it seems to decrease when we reduce the model, it remains too high for the model to be tested in HiL.

b. In Hardware-in-the-Loop

Hardware specifications: dSPACE DS1005 PPC with 4 cores available.

The simplest and the most detailed vehicle models are run on the HiL rig with a step size of 0.0005 s. Due to time constraints, we only tested two vehicle models in HiL, the most detailed one and a reduced one.

The most detailed vehicle is the full vehicle model (i.e. the Level 1 vehicle in the last section).

The simplest vehicle is essentially the fully reduced vehicle (i.e. the Level 5 vehicle in the last section) with an elasto-plastic based friction model to reduce the number of events and thus the number of time overruns. We could see indeed that if the Level 5 vehicle was running very fast in Dymola it still generated a few overruns. The elasto-plastic clutch uses a single state and defines the friction in a continuous way without introducing events (Dupont P., 2002).

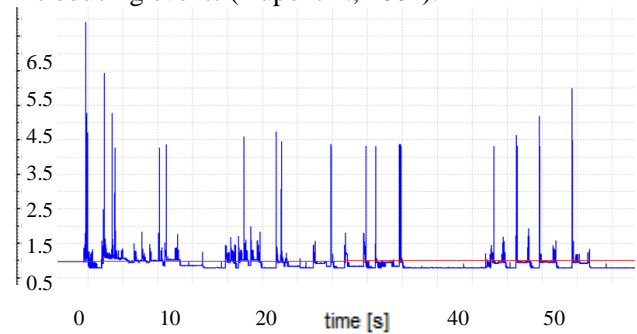


Figure 18. Most detailed vehicle's turnaround time (ms) (blue) and target step size (red).

Maximum turnaround time: 7.4 ms
Minimum turnaround time: 0.28 ms
Number of overruns: 55

This very detailed vehicle model is on average too slow on the current hardware and cannot achieve real-time performance. It also generates a high number of overruns.

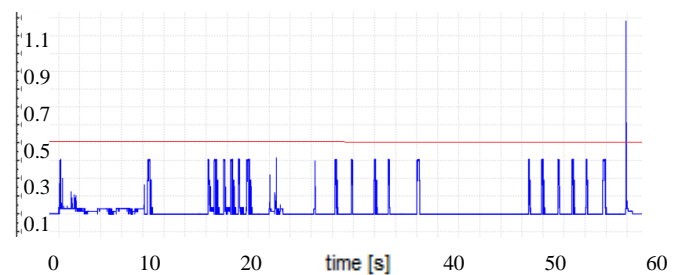


Figure 19. Simplest vehicle's turnaround time (ms) (blue) and target step size (red).

Maximum turnaround time: 1.18 ms
 Minimum turnaround time: 0.096 ms
 Number of overruns: 1

This simplified vehicle model is achieving real-time performance and generates only one overrun. The cause of the overrun is under investigation.

A comparison of the key variables is not very relevant here since, due to the high number of overruns, the results of the most complex model are rapidly drifting. However, despite these inaccuracies, the results are still matching well. The manoeuvre starts at time=0s, what happens before this time can be ignored.

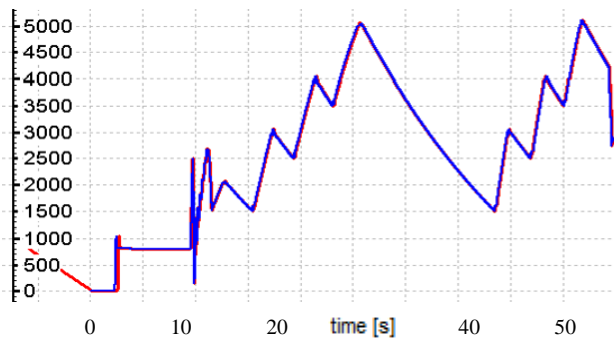


Figure 20. Reduced vehicle's engine speed (rpm) for single core (blue) and multicore (red) implementation.

The results are slightly different from what we got in Dymola because of a change in vehicle parameterisation (tyre and aerodynamic drag have been increased in the vehicle tested in Dymola, hence smaller vehicle speeds). Due to time constraints, a second test of the model in the HiL environment has unfortunately not been possible. The point here is not to compare the results between Dymola and HiL but rather to compare the results of the different vehicles.

The multicore capability will also be investigated in more detail in future work; it does not show a real benefit here since the controller we used is the software version and thus is not very CPU demanding. The crank angle resolved engine model has been, as part of this project, split into three sub models (Gallagher S., 2016): the mechanics part (included in the vehicle model), the combustion part (one for each cylinder) and the air path. We thus have 5 s-functions for the engine and vehicle to run and 4 cores available. Along with these models are the driver model and the CAN buses that also have to be run on these 4 cores. At the time of writing of this paper, it was still undecided how the repartition between the cores would be done.

5. Conclusion and Future work

Model reduction techniques for all the vehicle subsystems have been implemented and tested. The accuracy of the results is satisfying and the improvement in performance significant. The level of detail in the

chassis and driveline has been maintained the same as in the full models. While the reduced transmission has lost the 3D capability, it still outputs the correct speed and torque for all operating points. However, the fully detailed model is still very much needed. It is important to be able to model a vehicle with a physical representation and a high level of detail to accurately predict the vehicle behaviour to then be able to calibrate the reduced model.

A series of assumptions and simplifications have of course had to be made. The main one is that the results would be the same on the left and right hand sides of the vehicle since we test it in a straight line. This assumption is acceptable in the MORSE project as the small inaccuracies are acceptable. Moreover, we thought it was more interesting to compromise on the left/right discrepancy but to keep the same level of detail in the model rather than reducing the capability of the subsystems to maintain the models physical on both vehicle sides which is not of prime importance in a straight-line test.

More testing needs to be done in the Hardware-in-the-Loop environment: we need for example to test the vehicle over other manoeuvres than Tip-in/Tip-out, to include the detailed Dymola engine model and to explore further the multicore capability.

The reduced models (except the gear set) are multi-body and could be simplified further to one-dimensional subsystems if needed in order to still be able to achieve real-time performance once we will have integrated the Dymola engine in the vehicle.

References

- Gallagher S. *et al.* (2016) *Model-based Real-time Systems Engineering*, Loughborough, England, Powertrain Modelling and Control Conference.
- Dempsey M. *et al.* (2006) *Coordinated automotive libraries for vehicle system modelling*, Vienna, Austria, Proceedings of the 5th International Modelica Conference.
- Dempsey M. *et al.* (2009) *Investigating the Multibody Dynamics of the Complete Powertrain System*, Como, Italy, Proceedings of the 7th Modelica Conference.
- Dupont P. *et al.* (2002) *Single State Elasto-Plastic Friction Models*, IEEE Transactions of Automatic Control.

Towards Virtual Validation of ECU Software using FMI

Lars Mikelsons¹ Roland Samlaus¹

¹Robert Bosch GmbH

Abstract

Connected, Automated, Electrified. These three trends in the automotive industry require rethinking of the use of simulation respectively models. The use of models for evaluation of new concepts or stimulating the unit-under-test (in HiL testing), already firmly rooted in the development process of software functions, will not be sufficient to realize visions like autonomous driving or update-over-the-air. One key enabler for such technologies is virtual validation, i.e. the validation or release of software functions in a pure virtual setup. That is, simulation is not only a tool to shorten the development cycle, but one of the key technologies to release future software functions, e.g. highly automated or autonomous driving. In this contribution a feasibility study for the validation of FMI-based virtual ECUs (vECUs) in a co-simulation setup is presented. Thereby, the powertrain and the vECU are represented by FMUs, while the tool CarMaker is used for vehicle dynamics. On the base of the gained experience requirements for the FMI standard are formulated that would allow to go for virtual validation of future software functions. *Keywords: FMI, virtual validation, ECU, vECU, autonomous driving*

1 Introduction

The use of models and simulation is firmly rooted in the development process of automotive software as well as hardware components. However, in the development of software functions typically simulation is mostly used to evaluate new concepts or to stimulate the unit-under-test, e.g. HiL in testing. More precisely, the model of a software or hardware component is typically used during development (Junghanns et al., 2014). Models and simulation are rarely used for virtual validation, i.e. validation or even release of a software function in a pure virtual setup. There exist examples where software validation was done virtually, e.g. ESC homologation (Holzmann et al., 2012). However, the validation of ECU software is mostly performed using real prototypes. In fact, although in many cases models are exchanged between OEMs and suppliers, it is not a standard workflow to use them for the application of software functions. While for "old fashioned" software functions not using existing models may lead to a more costs, not using models is not an options when it comes to concepts like autonomous driving or update-over-the-air. According to (Wachenfeld and Winner, 2015) and (Winner et al., 2010), following ba-

sic statistics, between 100 million and 5 billion kilometers of test driving are required in order to ensure that software for autonomous driving is at least as save as a human driver. Note that, the test procedure has to be repeated after every single update or modification. Clearly, it is not possible to use real prototypes for those test drives due to required time and costs (Google states that its 20 self driving cars drive 16.000 kilometers per week (goo, 2015)). The same argumentation holds for update-over-the-air except that typically the problem arises from the number of variants and configurations that need to be tested. Thus, here virtual validation has to be employed. Typically, for technologies like autonomous driving one has to couple models from different domains (xDomain vehicle simulation), e.g. powertrain, vehicle dynamics or powernet. One approach for xDomain vehicle simulation is to use Modelica in order to model all involved domians in the same tool respectively language. Though, in big companies the models for the different domains are generated in different business units that prefer different simulation tools (best suited for their specific problems). Hence, co-simulation is typically the way to go. Designing such a co-simulation setup for virtual validation leads to several challenges. Typical questions that arise are

- What is the required level of detail for my models?
- How do I parametrize my models?
- How to validate a model?
- How big is the discretization error?
- How big is the coupling error?
- How can I integrate the software code into the simulation?
- Which portions of the ECU code do I have to integrate (where to cut)?

In this contribution only the last two questions are focused. In fact, this contribution presents a feasibility study for integrating ECU code as an FMU into a co-simulation setup. Thus it shows a possibility to integrate ECU code (including the formulation of further requirements on FMI) and discusses the problem of identifying the portion of the software stack required for a specific validation task. Note that, the used software function is part of a function for highly automated driving (HAD). Future work aims at treating this HAD function as sketched in this paper.

In industry there are different meanings for vECU. Some people just mean cross-compiled application code, others mean ECU code running on a virtual OS and last but not least a vECU can also include virtual hardware. In section 2 a brief overview is given and the used approach for the vECU used in section 3 is described. In section 3 the co-simulation setup and generated results are discussed. Starting from a yaw rate controller implemented in ASCET a vECU is generated. This vECU, is then integrated in a co-simulation setup consisting of Model.CONNECT from AVL as a co-simulation middleware, an FMU containing a powertrain model generated with GT Suite from Gamma Technologies and CarMaker from IPG for vehicle dynamics simulation. Moreover, required additional features in the used tools and standards are discussed. The paper closes with a summary and an outlook.

2 Virtual ECUs

Virtual ECUs (vECU) aim at running target ECU code on standard x86 systems by virtualization. This section introduces use cases for virtual ECUs supporting the ECU developer in creating software with higher quality faster than with regular development processes. The basic software architecture for ECUs is explained and it is distinguished between three types of virtual ECUs that differ in the extent of the re-used target code. Finally the virtual ECU used in the feasibility study is presented.

2.1 The AUTOSAR software architecture

The AUTOSAR (Automotive Open System Architecture) standard defines an architecture (see figure 1) for embedded software on ECUs. The idea is to "cooperate on standards - compete on implementation". AUTOSAR systems can be divided in six main components (see 1):

1. **Application Software (ASW)** is the software implementing the unique features of an ECU, e.g., the behavior of the electronic stability program (ESP) or HAD functions.
2. **Runtime Environment (RTE)** is the communication layer which distributes the signals directly between ASW components or using the base software's (BSW) communication stack. The idea of AUTOSAR ASW components is that they can be distributed freely on different ECUs. The RTE will then either dispatch the data from one component directly to another component, if they are deployed on the same ECU, or the data is sent via the communication stack in the base software.
3. **Base Software (BSW)** is software that provides basic functionality of an ECU. Typical software components are communication stacks such as CAN, automotive ethernet, or flexray. Other examples are memory access and diagnosis functions. The extent of the used base software on an ECU depends on the

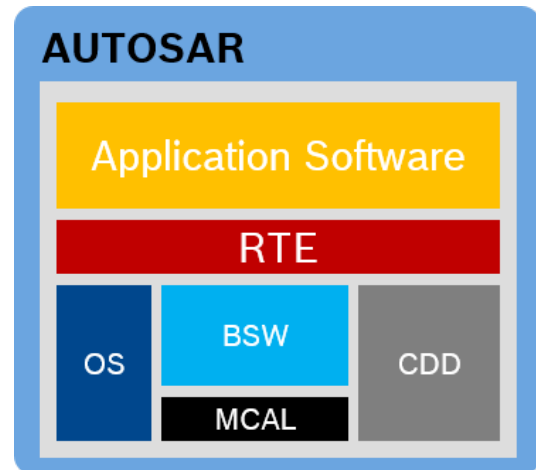


Figure 1. The AUTOSAR software architecture

use-case, e.g., ECUs for wipers need less functionality than engine control units.

4. **Operating System (OS)** is a real-time system that is responsible for executing code at the right time and with a defined maximum duration. Therefore, runnables that contain the executable code, are scheduled using scheduling tables. The runnables are assigned to recurring tasks of a defined maximum duration. For simulation it is often desired to accelerate the execution of code. Therefore the tasks are executed as fast as possible. Furthermore, the OS is responsible for handling interrupts, e.g., when data is received from a sensor. This pauses the execution of runnables until the interrupt has been handled.
5. **Microcontroller Abstraction Layer (MCAL)** is the driver layer and specific for the used ECU hardware. This should be the single software component which is hardware dependent. For simulation on x86 systems the MCAL layer has to be exchanged.
6. **Complex Device Drivers (CDD)** contain special code which is not commonly used and this not part of the AUTOSAR specification, e.g., drivers for magnetic valves.

2.2 Categories of virtual vECUs

vECUs can roughly be classified into three categories:

1. vECUs that contain only the ASW and RTE (and optionally an OS). This aims at quick testing of basic functions of the ASW without using base software components like communication. If the ASW code is AUTOSAR compatible, vECUs for this use case can be easily created since there are no hardware specific parts. However, no realistic estimation of the execution behavior on real ECUs can be derived with this kind of vECUs.

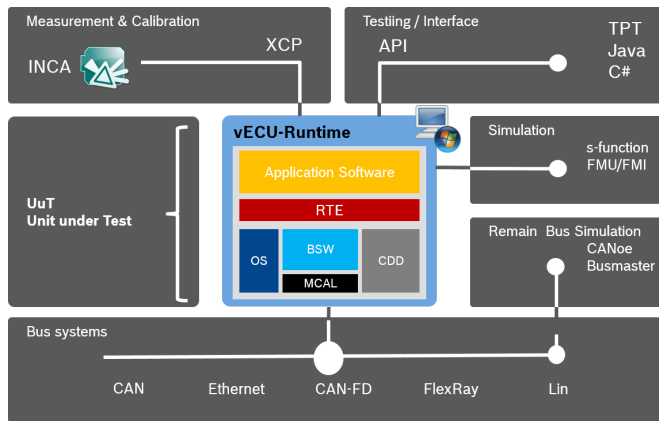


Figure 2. Use cases for vECUs

2. vECUs that consists of ASW, RTE, BSW, OS and a virtual MCAL (for x86 systems). A more realistic behavior of the real ECU can be simulated with this kind of vECU. The scheduling of tasks is considered and the functionality of the BSW can be tested. As an example rest-bus simulation can be performed to mock additional ECUs in the network to test for correct reaction of the simulated vECU.
3. If a more realistic behavior of the vECU is desired, virtual hardware can be used. All software components of the real ECU can be re-used, including the target MCAL layer. The MCAL is used with detailed hardware models which simulate real timing behavior. Another benefit is the ability to perform fault injection which can be problematic when done with real hardware since the injected faults could cause damage to the devices. A drawback of using hardware models is reduced simulation speed since the models are usually highly detailed.

2.3 Use cases for virtual ECUs

Virtual ECUs can be used for faster test of application software. Based on the vECU category used, BSW functionality and timing behavior can also be considered. Typical use-cases for vECUs are displayed in figure 2.

The XCP protocol is used by tools like ETAS INCA to measure and calibrate parameters of the ECU software, e.g. to optimize the gasoline injection for a certain engine type. This can also be done virtually with vECUs. Test APIs allow for automatic testing of the ECU software. Examples for commonly used testing tools are TPT and ECU-TEST. It is also possible to write custom tests with arbitrary programming languages like Java. The vECU can be exported as an FMU or S-Function for co-simulation with physical and plant models. Virtual busses (CAN, LIN, ...) can be connected to virtual ECUs using the MCAL layer. This allows to analyze messages on the busses and to perform rest-bus simulation with tools like CANoe or Busmaster to simulate additional ECUs in a network.

2.4 vECU for feasibility study

For the feasibility study a category 1 vECU has been used. The vECU consists of an OS, the RTE and application software. The application software has been generated as AUTOSAR 4 compatible code from an ASCET model. Based on the application's AUTOSAR description the RTE has been generated. No BSW or MCAL has been integrated at this point. This will be done as a next step in order to send messages via CAN. This will enable to investigate how a software function can be deployed on more than one ECU.

2.5 vECU tool evaluation

Several tools of different vendors including ETAS, QTronic, Dassault, dSPACE and Mentor Graphics have been evaluated for the creation of vECUs at Bosch. For this contribution ETAS EVE is used since it is best suited for the use case presented here (e.g. best integration into the existing ECU build tool chain).

3 Feasibility study

In this section it is shown how FMI is used to integrate and finally validate a software function in a co-simulation setup. The vision is to validate HAD functions or software for autonomous driving in the future. In this contribution, not a complete function consisting of e.g. object recognition, trajectory generation and follow-up control is used but only the lateral control since the goal is demonstrate the use of FMI to integrate ECU software into a simulation for virtual validation (and not to investigate the numerical properties).

In section 3.1 the co-simulation architecture is described, while 3.2 gives a brief overview on the used models and simulation results. In section 3.3 further requirements on the FMI standard and the used tools are derived.

3.1 Co-Simulation Architecture

In order to setup a co-simulation one of the first things to do is to define the integration platform, i.e. the tool that executes the master algorithm. In some cases, especially when not all involved tools offer an FMI export, it may be the case that there is not one defined master algorithm. Moreover, direct tool couplings (that do not rely on FMI) written by different tool vendors tend to have different numerical properties and to produce out-of-sync signals. An approach to face those issues is to a co-simulation middleware, that does not contain a model but only serves as a master and coordinator. Consequently, the co-simulation has a clean architecture with tool couplings that are consistent with each other (see figure 3). Moreover, it is easily extendable and typically offers more options to configure the co-simulation than simulation tools do. There exist several open-source (e.g. PyFMI) as well commercial (e.g. TISC from TLK Thermo, Cosimate from ChiasTek or Silver from QTronic that also includes vECU generation) co-simulation middlewares. In this contribu-

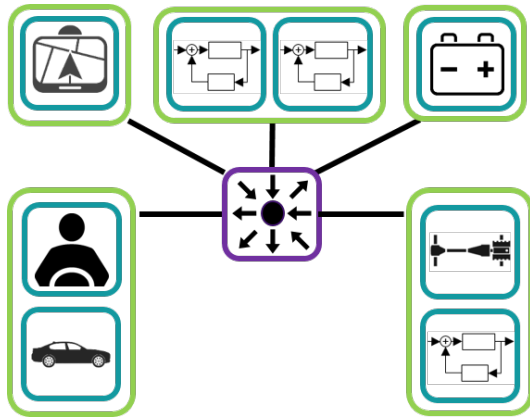


Figure 3. Co-simulation architecture with typical vehicle domains using a co-simulation middleware (purple rectangle)

tion Model.CONNECT from AVL is used.

3.2 Co-Simulation Setup

As already described above the co-simulation consists of the following participants (see figure 4)

- CarMaker from IPG: Vehicle dynamics and driver model for longitudinal control
- FMU: Powertrain model generated from GT Suite from Gamma Technologies (see ??)
- FMU: vECU generated with EVE from ETAS GmbH

Throughout this section it is assumed that the software under consideration can be validated by using the ISO double lane change as maneuver. The software shall be validated if the deviation in the yaw rate is rate is not bigger than 0.021/s and the deviation in the position in the y-direction is not bigger than 30cm. The vehicle model (and accordingly the software function) is parametrized according to a luxury car, however details are neglected here. Advanced co-simulation algorithms were not used, i.e. the models communicate using a parallel scheme using zero order hold extrapolation. It is expected that this has to be changed when using a more complex powertrain model and a more complex software function. Input for the software function is the actual yaw rate, the desired yaw rate, the vehicle velocity and the steering angle. Note that, the desired yaw rate is read from a table, that will be replaced by a trajectory generator in the future.

Figure 5 indicate the the simulation result lies within the specified error bounds. In fact the maximal error in the yaw rate is 0.0151/s and 29.8cm in the y-direction of the position of the vehicle. Thus (under the assumptions stated above), the software function can be judged as validated.

3.3 Derived Requirements for Tools and Interfaces

While it can be seen from the previous section that a virtual validation of ECU software using FMI is possible in

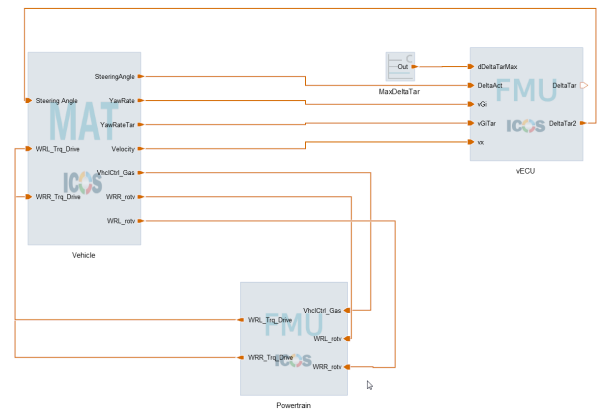


Figure 4. Setup of the co-simulation (screenshot from Model.CONNECT from AVL) with two FMUs (vECU from ETAS EVE and powertrain from GT Suite) and CarMaker

principle, there are some issues that prevent or will (in the case of more complex functions) prevent FMI from being suited for that use case. Beside issues described in (Link et al., 2015) the following requirements were derived:

- For complex software functions lots of physical signals have to be connected to the vECU. Thus, FMI should support vectors for easier workflows and better models (w.r.t. clarity).
- When it comes to software functions that are distributed over multiple ECUs, signals have to be exchanged between them. In many cases these signals are not just scalars or vectors (see above), but structs. A typical example is the ADASIS protocol (Ress et al., 2008) that is used to transmit the e-horizon from an e-horizon provider to an e-horizon reconstructor. Thus, in order to use FMI for vECUs it should support structs.
- In cases where not only the functionality, but also the timing shall be validated the communication has also to be modelled, e.g. using virtual CAN. Currently, the user (FMU generator and/or integrator) has to care about the communication between FMUs (at least for virtual busses etc.). In future versions it would be desirable to have the communication mean as part of FMI. Note, that this will be the case for the Advanced Co-Simulation Interface (ACI) (Krammer et al.).

Beside the requirements on FMI there are also some issues on the tool side. Among these are

- According to the list above tools (simulation tools and co-simulation middlewares) should support vectors and structs.
- When more complex models are used and especially in cases where numerically demanding couplings are in place it is desirable to use a master algorithm that

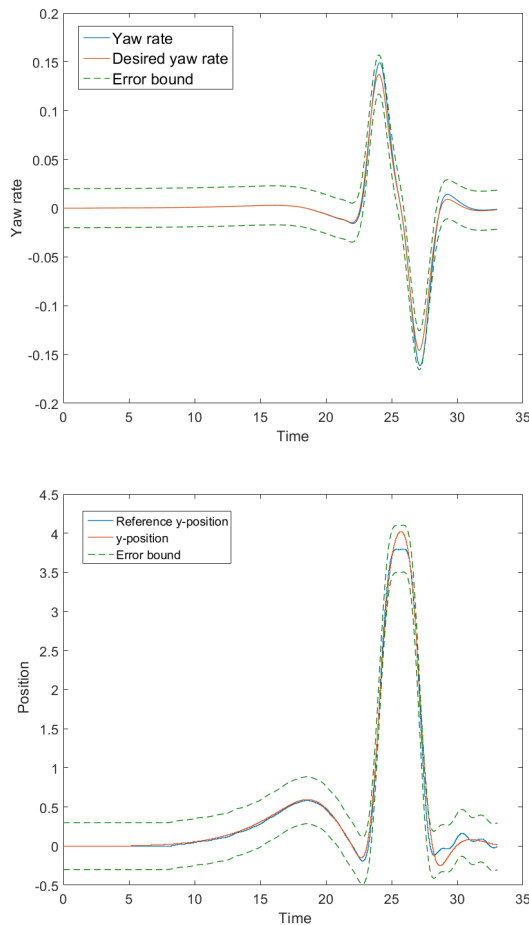


Figure 5. Variables used for validation including error bounds for the double lane change including error bound

can iterate, i.e. repeat macro steps. This is currently not supported by the used vECU, but will be supported in the future. However, many simulation tools do not support this (optional) FMI feature. This situation should be changed.

- Tool: If common open source implementations of virtual MCALs would reduce development overhead and enable switching between different vECU tools.

4 Summary & Outlook

This paper presents a feasibility study for the use of FMI for the validation of future ECU software. Besides a brief overview over the concept of vECUs different use cases for the use of vECUs are considered. For a functional validation (without timing) it is shown that the integration of ECU Code into a co-simulation works in principle. However, for more complex functions than the yaw rate controller used here, some issues arise that were collected in section 3.

In future work a more complex (HAD) function will be used. Consequently, more complex models have to be used. Moreover, it is desired to do also timing investiga-

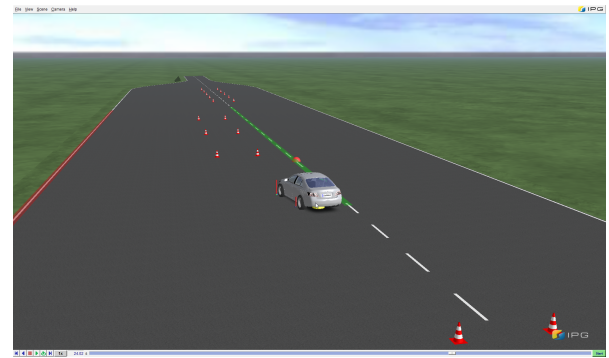


Figure 6. Screenshot showing the double lane change maneuver in CarMaker

tions. Thus, virtual CAN will be used for signal exchange. Therefore, the vECU has to include the communication stack. Last but not least an interface to connect calibration software (e.g. INCA from ETAS) will be created.

References

- Google self-driving car project monthly report. August 2015.
- Henning Holzmann, Karl Michael Hahn, Jonathan Webb, and Oliver Mies. Simulation-based esc homologation for passenger cars. *ATZ worldwide*, 114(9):40–43, 2012.
- Andreas Junghanns, Jakob Mauss, and Michael Seibt. Faster development of autosar compliant ecus through simulation. *ERTS-2014, Toulouse*, 2014.
- Martin Krammer, Nadja Marko, and Martin Benedikt. Interfacing real-time systems for advanced co-simulation - the acosar approach.
- Kilian Link, Leo Gall, Monika Mühlbauer, and Stephanie Gallardo-Yances. Experience with industrial in-house application of fmi. In *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*, number 118, pages 17–22. Linköping University Electronic Press, 2015.
- Christian Ress, Dirk Balzer, Alexander Bracht, Sinisa Durekovic, and Jan Löwenau. Adasis protocol for advanced in-vehicle applications. In *15th World Congress on Intelligent Transport Systems*, page 7, 2008.
- Walther Wachenfeld and Hermann Winner. Die freigabe des autonomen fahrens. In *Autonomes Fahren*, pages 439–464. Springer, 2015.
- H. Winner, G. Wolf, and A. Weitzel. Freigabefälle des autonomen fahrens/the approval trap of autonomous driving. *VDI-Berichte*, (2106), 2010.

Parameter Estimation based on FMI

Rüdiger Kampfmann Danny Mösch Nils Menager

Bosch Rexroth AG, Lohr am Main, Germany

{ruediger.kampfmann, fixed-term.danny.moesch, nils.menager}@boschrexroth.de

Abstract

In order to stay competitive the requirements on machinery in the producing industry have enormously increased. Within the automation industry these demands, like higher throughput or better energy efficiency, result in increasing complexity of the installed plants. Additionally, Industry 4.0 and the Internet of Things continuously increase the amount of software. Using model-based development methods is one approach to deal with this complexity. But model-based methods can also be utilized during the operational phase of a plant in order to generate additional value for the plant operator. Introducing smart services based on the usage of physical models enables new control and diagnosis features, e.g. the utilization of inverse plant models for feedforward control or comparing the output of a model with measurements of the plant in order to prove for correct behavior. For all these services the accuracy of the considered models is crucial. With an inexact model neither the future behavior can be foreseen nor the control quality can be improved. The used models don't have to be built up from scratch, existing models already created for sizing can be reused. However, these models cannot be used directly. First a reparametrization is necessary, because effects like friction or manufacturing tolerances cannot be taken into account correctly during sizing. For this special kind of problem dedicated optimization algorithms are available for parameter estimation, which take randomly distributed measurement errors and the special structure of this problem class into account.

In this paper a work flow for parameter estimation based on open source tools is presented, in which the considered models are provided as Functional Mock-up Unit. Afterwards the performance of this work flow is demonstrated on a real industrial problem: A three arm Delta Robot.

Keywords: *Parameter Estimation, Levenberg-Marquardt Algorithm, FMI, Least Squares Optimization, Log-likelihood Method*

1 Outline

The paper is structured as follows. First the considered optimization problem is derived from an approach based on probability theory. Afterwards suitable algorithms for this problem class are discussed with a special focus on the Levenberg-Marquardt algorithm, which is used in this contribution. Then the used software tools are presented: The Functional Mock-up Interface for the description of

the dynamic systems and the software library Ceres for the solution of the underlying optimization problem. Afterwards the whole architecture of the used toolchain is presented. Finally this toolchain is applied to a real problem.

2 Mathematical Background

In this contribution it is assumed that the simulation model of a real plant is described in the following way:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t), \mathbf{p}) \quad (1)$$

$$\mathbf{y}(t, \mathbf{p}) = \mathbf{g}(t, \mathbf{x}(t), \mathbf{p}) \quad (2)$$

$$\mathbf{x}(t_{\text{start}}) = \mathbf{x}_0 \quad (3)$$

The dynamic system consists of a set of ordinary differential equations (1), a set of algebraic equations (2) and an initial condition (3). The time interval $T = [t_{\text{start}}, t_{\text{end}}]$ is considered. The functions $\mathbf{x} : T \rightarrow \mathbb{R}^{n_x}$, $\mathbf{y} : T \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_y}$, $\mathbf{u} : T \rightarrow \mathbb{R}^{n_u}$ denote the states, the outputs and the inputs, respectively. The vector $\mathbf{p} \in \mathbb{R}^{n_p}$ represents the parameters. Additionally,

$$\mathbf{f} : T \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x},$$

$$\mathbf{g} : T \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_y},$$

$$n_x, n_u, n_p, n_y \in \mathbb{N}.$$

The input \mathbf{u} from the real plant is assumed to be known exactly over the whole interval, whereas $\boldsymbol{\eta}_i$ denotes the measured output vector of the real plant at time t_i for $i = 1, \dots, n_t$. The most obvious approach for parameter estimation is just to minimize some norm $\|\cdot\|_q$ with $q \in [1, \infty)$ or $q = \infty$ of the deviation between the measured and the simulated outputs by varying the parameters \mathbf{p} , i. e.

$$\arg \min_{\mathbf{p} \in \mathbb{R}^{n_p}} \sum_{i=1}^{n_t} \|\boldsymbol{\eta}_i - \mathbf{y}(t_i, \mathbf{p})\|_q^q$$

or

$$\arg \min_{\mathbf{p} \in \mathbb{R}^{n_p}} \sum_{i=1}^{n_t} \|\boldsymbol{\eta}_i - \mathbf{y}(t_i, \mathbf{p})\|_\infty,$$

respectively. That $q = 2$ is a reasonable choice is going to be the result of the following subsections. Therefore a little bit of probability theory has to be consulted. Ensuing from (Krengel, 1988) the maximum-likelihood approach is introduced first. In the next subsection the idea is used to formulate the underlying optimization problem which is the foundation of the presented process of parameter estimation.

2.1 Maximum Likelihood Approach

Let $\mathbf{X} \in \mathbb{R}^n$ be a random vector with independent and identically distributed components and concrete realizations $\mathbf{x} \in \mathbb{R}^n$ of \mathbf{X} . Each component X_i has the density function $v(\cdot|\mathbf{p}) = v(\mathbf{x}_i|\mathbf{p})$, which depends on a parameter set \mathbf{p} . It describes the probability of \mathbf{x}_i given the parameters \mathbf{p} . Since they are independent, the joint density can be written as

$$v(\mathbf{x}|\mathbf{p}) = \prod_{i=1}^n v(\mathbf{x}_i|\mathbf{p}).$$

To formulate the optimization problem the *likelihood function* $L(\cdot|\mathbf{x}) = L(\mathbf{p}|\mathbf{x})$ is defined as

$$L(\mathbf{p}|\mathbf{x}) := v(\mathbf{x}|\mathbf{p}),$$

which is now a function of the parameters \mathbf{p} given the data \mathbf{x} . $L(\cdot|\mathbf{x})$ is not a proper probability density function, since its integral over all parameters is not necessarily equal to 1. Therefore, it also should not be considered a conditional probability density function, which might be supposed by the vertical bar.

Thus, it is obvious to choose the optimization problem

$$\arg \max_{\mathbf{p} \in \mathbb{R}^n} L(\mathbf{p}|\mathbf{x})$$

to get the *maximum likelihood estimator* \mathbf{p}_{ML} which makes the sample data \mathbf{x} most likely. In some cases it will simplify the optimization process if the *Log-likelihood function* $\ln L(\mathbf{p}|\mathbf{x})$ is chosen instead of $L(\mathbf{p}|\mathbf{x})$ as will be seen later. In fact, it does not make a difference whether choosing $L(\mathbf{p}|\mathbf{x})$ or $\ln L(\mathbf{p}|\mathbf{x})$, since the logarithm is a monotonic function that does not influence the maximum.

2.2 The Underlying Optimization Problem

Each measured data vector $\boldsymbol{\eta}_i$ at time t_i can be expressed as the real output $\mathbf{y}(t_i, \mathbf{p}^*)$ with the exact but naturally unknown parameter set \mathbf{p}^* plus a measurement error $\boldsymbol{\varepsilon}_i$, i. e.

$$\boldsymbol{\eta}_i = \mathbf{y}(t_i, \mathbf{p}^*) + \boldsymbol{\varepsilon}_i. \quad (4)$$

It is assumed that the measurement errors $\boldsymbol{\varepsilon}_i$ are statistically independent and underly a certain distribution. The most common assumption is to choose a normal distributed error vector $\boldsymbol{\varepsilon}_i$ with statistically independent components, known (diagonal) covariance matrix $\boldsymbol{\Sigma}_i$ and expectation $E(\boldsymbol{\varepsilon}_i) = \mathbf{0}$, i. e.

$$\boldsymbol{\varepsilon}_i \sim N(\mathbf{0}, \boldsymbol{\Sigma}_i) \quad \text{with} \quad \boldsymbol{\Sigma}_i = \text{diag}(\sigma_{i,1}^2, \dots, \sigma_{i,n}^2). \quad (5)$$

In an applied sense that means that the measurements $\boldsymbol{\eta}_i$ do not influence each other and the errors $\boldsymbol{\varepsilon}_i$ do not contain a systematic error.

With these requirements the density function

$$v(\boldsymbol{\varepsilon}_{i,j}) = \frac{1}{\sqrt{2\pi}\sigma_{i,j}} \exp\left(-\frac{\boldsymbol{\varepsilon}_{i,j}^2}{2\sigma_{i,j}^2}\right)$$

for each measurement error $\boldsymbol{\varepsilon}_{i,j}$ is obtained. Since the $\boldsymbol{\varepsilon}_{i,j}$ are statistically independent for all j and for all i , too, it holds

$$\begin{aligned} v(\boldsymbol{\varepsilon}) &= \prod_{i=1}^{n_t} v(\boldsymbol{\varepsilon}_i) \\ &= \prod_{i=1}^{n_t} \prod_{j=1}^{n_y} v(\boldsymbol{\varepsilon}_{i,j}) \\ &= \prod_{i=1}^{n_t} \prod_{j=1}^{n_y} \frac{1}{\sqrt{2\pi}\sigma_{i,j}} \exp\left(-\sum_{k=1}^{n_t} \sum_{l=1}^{n_y} \frac{\boldsymbol{\varepsilon}_{k,l}^2}{2\sigma_{k,l}^2}\right), \end{aligned} \quad (6)$$

where $\boldsymbol{\varepsilon} = (\boldsymbol{\varepsilon}_1 \dots \boldsymbol{\varepsilon}_{n_t})$. Because of (4) we also get $\boldsymbol{\eta}_i \sim N(\mathbf{y}(t_i, \mathbf{p}^*), \boldsymbol{\Sigma}_i)$ and thus

$$v(\boldsymbol{\eta}_{i,j}) = \frac{1}{\sqrt{2\pi}\sigma_{i,j}} \exp\left(-\frac{(\boldsymbol{\eta}_{i,j} - \mathbf{y}_j(t_i, \mathbf{p}^*))^2}{2\sigma_{i,j}^2}\right).$$

For the same reason as in (6) and with $\boldsymbol{\eta} = (\boldsymbol{\eta}_1 \dots \boldsymbol{\eta}_{n_t})$ this leads to the conditional density function

$$v(\boldsymbol{\eta}|\mathbf{p}) = \prod_{i=1}^{n_t} \prod_{j=1}^{n_y} \frac{1}{\sqrt{2\pi}\sigma_{i,j}} \exp\left(-\frac{(\boldsymbol{\eta}_{i,j} - \mathbf{y}_j(t_i, \mathbf{p}))^2}{2\sigma_{i,j}^2}\right).$$

The Log-likelihood function is then defined by

$$\begin{aligned} \ln L(\mathbf{p}|\boldsymbol{\eta}) &= -\frac{n_t n_y}{2} \ln 2\pi - \sum_{i=1}^{n_t} \sum_{j=1}^{n_y} \ln \sigma_{i,j} \\ &\quad - \frac{1}{2} \sum_{i=1}^{n_t} \sum_{j=1}^{n_y} \frac{(\boldsymbol{\eta}_{i,j} - \mathbf{y}_j(t_i, \mathbf{p}))^2}{\sigma_{i,j}^2}. \end{aligned}$$

Since the first and the second term are constant they can be omitted from the optimization.

Finally, the whole constrained nonlinear optimization problem can be formulated:

$$\arg \min_{\mathbf{p} \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{n_t} \|\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{\eta}_i - \mathbf{y}(t_i, \mathbf{p}))\|_2^2 \quad (7a)$$

subject to

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t), \mathbf{p}), \quad (7b)$$

$$\mathbf{y}(t, \mathbf{p}) = \mathbf{g}(t, \mathbf{x}(t), \mathbf{p}) \quad (7c)$$

$$\mathbf{x}(t_{\text{start}}) = \mathbf{x}_0 \quad (7d)$$

observing the box constraints

$$\mathbf{p}_{\text{lower}} \leq \mathbf{p} \leq \mathbf{p}_{\text{upper}} \quad (7e)$$

given

$$\mathbf{u}(t_i) \quad \text{with} \quad t_i \in [t_{\text{start}}, t_{\text{end}}] \quad \text{for} \quad i = 1, \dots, n_t$$

and related measured outputs $\boldsymbol{\eta}_i$. The result is the parameter set \mathbf{p}_{ML} which is the most likely for the given measured output values. It should be noted again that it is important for the chosen approach to have measurement errors following (5). The method is not reasonable for different distributions, although others can be established.

3 Optimization Algorithms

Eliminating the ODE constraints (7b)-(7d) through numerical integration from (7a) yields a common nonlinear optimization problem. For this kind of problem a couple of different algorithms for the efficient solution exist. Some algorithms exploit derivative information and some do not. The derivative-free optimization algorithms have the advantage that they obviously do not require derivatives with respect to the varied optimization variables, which may be costly to compute. Another advantage is that under certain circumstances global convergence is achieved, neglecting limited computational time and rounding errors. In (Gedda et al., 2012) a tool chain for parameter estimation with FMI and derivative-free methods already has been presented. Nevertheless, derivative-free optimization algorithms show very poor convergence speed.

However, optimization algorithms, which use derivatives of the objective function, have also distinct advantages. The main benefit is the fast convergence rate. These methods compute iteratively starting from an initial guess a descent direction and thus reduce the objective function in every step until a certain stop criterion is reached. The better the initial guess the faster the convergence speed. The disadvantages of these methods are on the one hand that the problem has to be sufficiently smooth and that derivatives have to be computed and on the other hand that these methods may get stuck in a local minimum, if the initial guess is too bad. The last disadvantage can be overcome with multi start-ups, i.e. run several optimization from different initial guesses (see (Raue et al., 2013) for more information). For the purposes of this contribution, whereas existing models from the sizing should be reused, good initial guesses are known, because rough parameter sets are already needed for correct dimensioning. Thus sticking in local minimum is not a problem at all. The considered models are also sufficiently smooth with respect to the parameters. Also in (Raue et al., 2013) a benchmark of different algorithms was conducted to an estimation problem from systems biology, demonstrating the slow convergence speed of derivative-free optimization algorithms compared to the ones which rely on derivatives. Since good initial guesses are known and the fast convergence speed the demonstrated toolchain is based on derivative based optimization algorithms.

3.1 Levenberg-Marquardt Algorithm

The investigated optimization problem (7a) has a special structure. It is a nonlinear least squares problem. This kind of problem is widely spread in scientific and engineering areas. Thus structure exploiting optimization algorithms have been developed, which solve this problems efficiently. The Levenberg-Marquardt algorithm is one of these algorithms and is used within this contribution. It can be seen as a conjunction of the Gauss-Newton method together with the idea of Trust-Region approaches.

To give a short overview (Björck, 1996), some abbreviations are introduced first:

$$\begin{aligned} \mathbf{h}_i(\mathbf{p}) &:= \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{\eta}_i - \mathbf{y}(t_i, \mathbf{p})) \\ H(\mathbf{p}) &:= \sum_{i=1}^{n_t} \|\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{\eta}_i - \mathbf{y}(t_i, \mathbf{p}))\|_2^2 \\ \mathbf{y}'_i(\mathbf{p}) &:= \frac{\partial \mathbf{y}(t_i, \mathbf{p})}{\partial \mathbf{p}} \end{aligned}$$

Therein denotes \mathbf{y}'_i the Jacobian matrix of \mathbf{y} at time t_i with respect to \mathbf{p} . The Gauss-Newton method solves the linearized least squares problem

$$\arg \min_{\Delta \mathbf{p} \in \mathbb{R}^{n_p}} \frac{1}{2} \sum_{i=1}^{n_t} \|\boldsymbol{\Sigma}_i^{-1}(\mathbf{h}_i(\mathbf{p}_k) - \mathbf{y}'_i(\mathbf{p}_k)\Delta \mathbf{p}_k)\|_2^2$$

in each iteration step k to get a new approximation

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \Delta \mathbf{p}_k$$

of the exact parameter set \mathbf{p}^* .

It is the idea of the Levenberg-Marquardt algorithm to add a regularization term to the linearized objective function,

$$\arg \min_{\Delta \mathbf{p} \in \mathbb{R}^{n_p}} \frac{1}{2} \sum_{i=1}^{n_t} \|\boldsymbol{\Sigma}_i^{-1}(\mathbf{h}_i(\mathbf{p}_k) - \mathbf{y}'_i(\mathbf{p}_k)\Delta \mathbf{p}_k)\|_2^2 + \frac{\lambda_k}{2} \|\Delta \mathbf{p}_k\|_2^2.$$

With the regularization the problem has always a solution even with rank deficient Jacobians \mathbf{y}'_i . The parameter λ_k also controls the step length $\|\Delta \mathbf{p}_k\|_2$ as well as the direction $\Delta \mathbf{p}_k$. It can be observed (Marquardt, 1963) that for a large λ_k the direction is almost a gradient step with only small step size, whereas a small λ_k leads to a direction close to a Gauss-Newton step. Therefore it is reasonable to choose a small λ_k near the actual minimum where the linearized problem is a rather good approximation. Hence, choosing λ_k is a significant task in each iteration to reach a preferably fast convergence rate. There are different ways to update the parameter λ_k . A central role plays the ratio between actual reduction and (by the linearized problem) predicted reduction

$$\psi_k(\Delta \mathbf{p}_k) = \frac{H(\mathbf{p}_k) - H(\mathbf{p}_k + \Delta \mathbf{p}_k)}{H(\mathbf{p}_k) - \sum_{i=1}^{n_t} \|\boldsymbol{\Sigma}_i^{-1}(\mathbf{h}_i(\mathbf{p}_k) - \mathbf{y}'_i(\mathbf{p}_k)\Delta \mathbf{p}_k)\|_2^2},$$

whose value decides whether \mathbf{p}_k will be updated or not and how λ_k will be changed.

4 Functional Mock-up Interface

The Functional Mock-up Interface (FMI) is a tool independent standard to support model exchange between different simulation environments (Blochwitz et al., 2011). A model which is shared via FMI is referred to as Functional Mock-up Unit (FMU). An FMU consists of a XML-File, describing the whole model variables and parameters, and a compiled library containing the model equations and additionally required functions, i.e. functions for initialization or data exchange. The models are described as hybrid

ODEs supporting state and time events. The FMI standard supports two modes to share these models. On the one hand there is Model Exchange, whose models are described in a form similar to the equations 1 and 2, and on the other hand the Co-Simulation mode, which delivers the FMU additionally with its own integrated ODE solver. In this contribution the Co-Simulation mode is used due to the fact that no extra ODE solver is required for simulating an FMU. The models considered in the estimation procedure are built up in a simulation environment and exported as FMU. In 7.2 an example is described. For the optimization derivatives with respect to the parameters are required. With the actual standard 2.0 of the FMI, only derivatives with respect to inputs and states are supported (Blochwitz et al., 2012). Hence finite differences are used. As far as the authors know, parameter sensitivities are currently considered by the FMI steering committee.

5 Ceres Solver

For the solution of the nonlinear least squares problem (7a), (7e) the Ceres Solver (Agarwal et al.) is utilized. The Ceres Solver is an open source C++ library for solving large optimization problems. Beneath general unconstrained optimization problems it was developed to solve nonlinear least squares problems with bound constraints. There are several reasons why the Ceres Solver was chosen. The solver is published under the New BSD license, so there are almost no license restrictions for commercial use. In addition to the Levenberg-Marquardt Algorithm 3.1 this software library is equipped with other state of the art algorithms and has reached a certain maturity since it is used in commercial applications for more than four years and still has an active community.

Furthermore the library is developed in C++ and has already been migrated to Android and iOS. Hence an migration to Bosch Rexroth embedded systems should be possible with little effort. Ceres can compute the required derivatives of the objective function by finite differences or the user can provide them. Because the actual FMI version is not supporting parameter derivatives, they are computed by Ceres via finite differences. With upcoming features of the next FMI version, this can be easily adapted. Ceres is one of the few libraries which is also capable to derive covariance estimations for the solution. Hence, confidence intervals for the computed parameters can be computed directly.

Within Ceres a problem class needs to be implemented which corresponds to the desired residual function (7a). The model to be investigated and the measured data have to be provided therefor. An additional class method handles possible solver options and is responsible for the actual optimization.

6 Structure of the Tool Chain

Figure 1 shows the structure of the parameter estimation toolchain. The stimulation of the real plant and the real

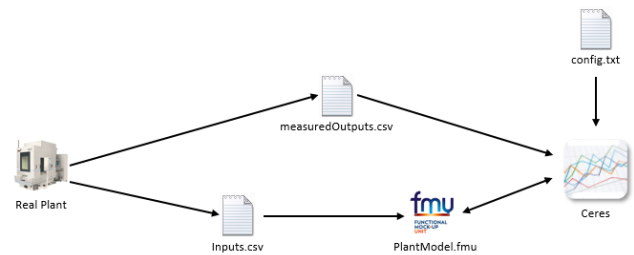


Figure 1. Structure of parameter estimation toolchain

measured outputs have to be provided as CSV-file. The whole estimating procedure is configured by a configuration file. In this file the desired model, the parameters to estimate and the paths of the CSV-files are denoted. Also an initial guess and the variances for each measurement noise have to be stated. Additionally, upper and lower bounds for the parameters can be specified. Within Ceres a problem class was defined which takes the configuration file and manages the whole parameter estimation procedure. This problem class directly interfaces the FMU. No additional library for calling the FMU functions is used. An evaluation of the residual function implies a simulation of the FMU. In every step the inputs are written to the FMU. Thereafter, the residual function is built up by comparing the measured outputs with the outputs of the FMU. Hence the whole simulation is triggered by Ceres. The derivatives of the residual function with respect to the parameters are computed directly by Ceres via finite differences which corresponds to multiple simulation runs. Ceres then conducts the chosen optimization algorithm by varying the parameters. Subsequently, an a posteriori evaluation of the covariance matrix of the estimated parameters can be conducted. Out of this matrix confidence intervals for each parameters can be derived directly. For the import of the FMU into Ceres an own light weight framework was implemented.

7 Application of the Tool Chain

In this section the capability of the toolchain is demonstrated on a Delta Robot. This type of robot was developed in the 1980s (Clavel, 1988) and is widely used for pick and place applications. It is built up out of parallel bars and has 3 degrees of freedom for translational manipulation and one for manipulating the orientation. Hence it has to be driven by 4 motors. Since the robot should move as fast as possible, knowing the exact dynamic behavior is advantageous, i. e. an accurate model can be exploited for feedforward control in order to enhance the dynamic behavior. The dynamic of the robot is mainly influenced by frictional effects and mass parameters. The mass parameters underly a certain manufacturing tolerance and the friction is hard to be known beforehand. Therefore, estimating these parameters is a good use case for the toolchain.

7.1 Real Set-up of the Robot

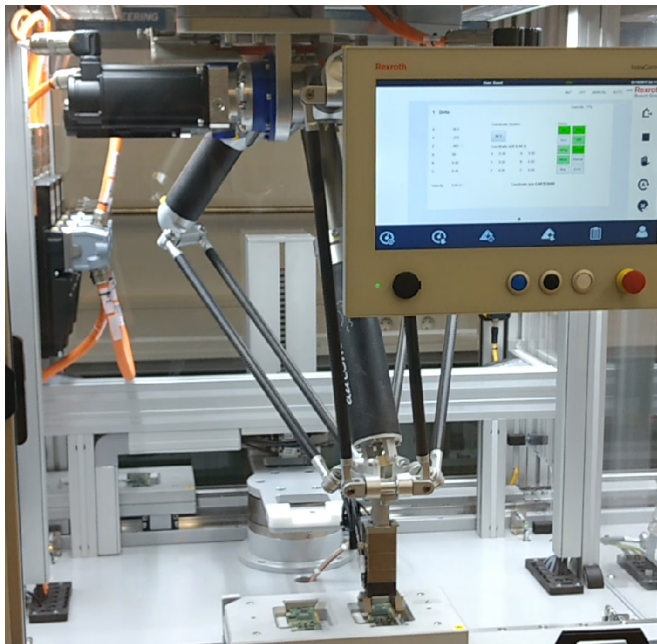


Figure 2. Real delta robot

Figure 2 shows the investigated robot. The kinematic is manufactured by Autonox24 and is driven by 4 Rexroth synchronous motors. Three MSK040B-0600 for the translational movement and one MS2N03-B0BYN for the orientation axis are used. The movement of the robot is controlled by a Rexroth IndraControl VPB 40.3 industrial PC. No special trajectories were considered. The robot just executes a usual pick and place cycle and the motor torque is measured via actual motor current. Through the recorded motor torques the parameters of the robot should be identified. Since the dynamics of the orientation axis is well known, no measurements for this axis have been taken into account.

7.2 Delta Robot Model

The physical model of the robot was built up in the modeling language Modelica using Dymola. The mechanical model consists of Modelica Standard Library (MSL) components. Mainly joints and body components from the multi body library are used.

7.3 Real Set-up of the Robot

Figure 3 shows the animation of the MSL components within Dymola. All parallel bars were considered. No simplifications of the mechanical structure were made. Additionally the drive train of each axis was modeled in the way that motor and gear inertia, gear efficiency and Coloumb and viscous friction are considered. Therefore own Modelica components were added to standard rotational mechanics components. As input of the model the position, velocity and acceleration of each axis were used. The resulting motor torques were declared as output. It is assumed that the inertia and friction properties of all 3

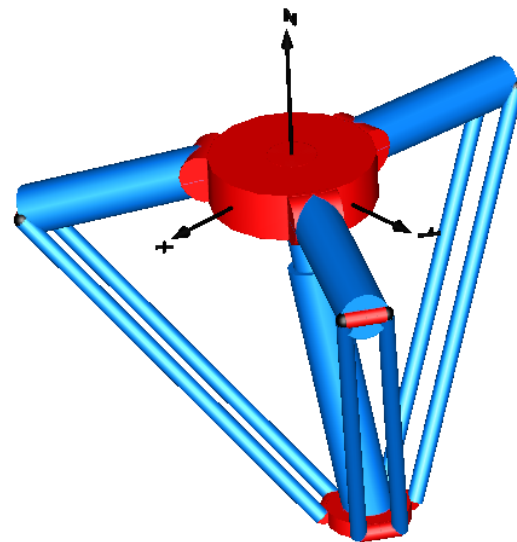


Figure 3. Animation of the multi body model

considered axes are equal. Hence, the following parameters should be identified:

- Mass of lower arm
- Mass of upper arm
- Mass of base plate
- Motor and gear inertia
- Gear efficiency
- Parameter for Coloumb friction
- Parameter for viscous friction

The model was exported as an FMU 2.0 for Co-Simulation containing the CVODE solver (Hindmarsh et al., 2005).

7.4 Estimation of Parameters

For the measurement the robot moves the usual pick and place cycle at four different speeds. 6250 time points were considered. The complete cycle lasts 62.278 seconds. For each of the three axes the position, velocity, acceleration and torque were recorded. For the identification procedure Ceres compares the measured motor torque with the one resulting from the FMU.

Table 1 shows the results of the parameter estimation procedure. The residual of the objective function (7a) was reduced significantly from 2.12×10^6 to 1.25×10^6 . The computed parameter sets especially the friction and gear efficiency parameter seem reasonable. Also the computed confidence intervals, i.e. the intervals in which the real parameters are located with a probability of $\beta = 0.95$, imply that the computed estimations are reliable. Unfortunately it is not possible to validate the estimation results by scaling the components, because the robot cannot be disassembled.

Parameter	Unit	Initial Value	Estimated Value	Confidence Interval $\beta = 0.95$
Mass of lower arm	[kg]	0.1	0.08	± 0.00350
Mass of upper arm	[kg]	1.5	1.74	± 0.0211
Mass of base plate	[kg]	0.87	1.1	± 0.0212
Motor and gear inertia	[kg m ²]	0.000144	0.000155	$\pm 2.65 \times 10^{-6}$
Gear Efficiency	[1]	1.0	0.907	± 0.0136
Coloumb Friction	[Nm]	0.12	0.105	± 0.00158
Viscous Friction	[N m s rad ⁻¹]	0.001	0.00128	$\pm 1.92 \times 10^{-5}$

Table 1. Parameter estimation results

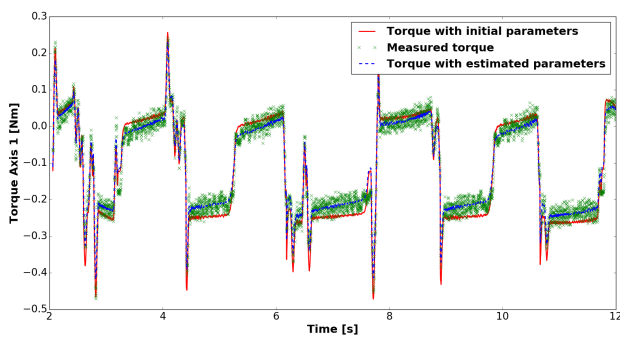


Figure 4. Results for arm 1

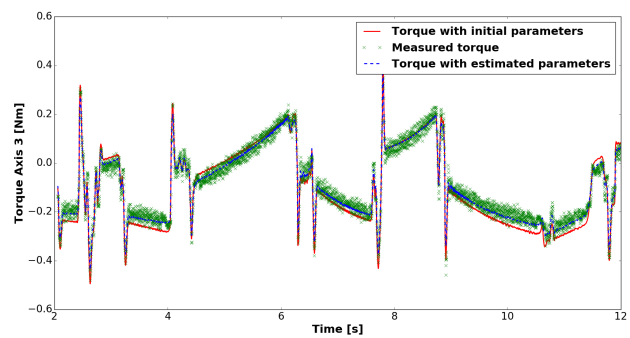


Figure 6. Results for arm 3

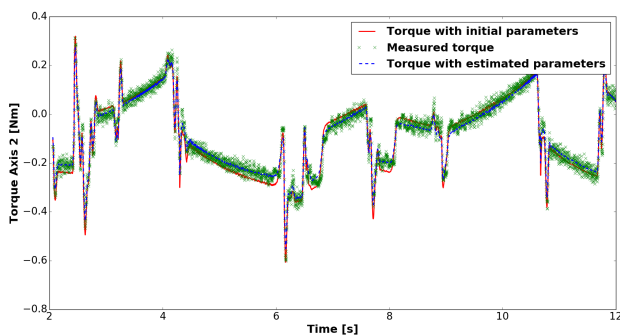


Figure 5. Results for arm 2

Figure 4 to 6 show a section of the results for each axes. With the estimated parameters the accuracy of the model has been improved significantly.

8 Summary and Outlook

A tool chain for parameter estimation with a state of the art Open Source software library and the Functional Mock-up has been presented. The capabilities of this tool chain were demonstrated on a real industrial robot. The results are very promising such that this approach should be pursued. On the one hand a migration of the whole tool chain to embedded systems seems meaningful. For example the estimation procedure can be used for auto calibration of feedforward controllers using inverse models.

On the other hand with Industry 4.0 and the Internet of Things new use cases occur. The intelligent plants equipped with sensors record all their data. With these

measurements and the presented tool chain parameter estimations could be conducted automatically. Upon these well known parameters and accurate models new smart services for diagnosis or control purposes can be enabled.

References

- Sameer Agarwal, Keir Mierle, et al. Ceres solver. <http://ceres-solver.org>.
- Åke Björck. *Numerical methods for least squares problems*. SIAM, 1996.
- Torsten Blochwitz, Martin Otter, Martin Arnold, Constanze Bausch, H Elmqvist, A Junghanns, J Mauß, M Monteiro, T Neidhold, D Neumerkel, et al. The functional mockup interface for tool independent exchange of simulation models. In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical University; Dresden; Germany*, number 063, pages 105–114. Linköping University Electronic Press, 2011.
- Torsten Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, et al. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 076, pages 173–184. Linköping University Electronic Press, 2012.
- Reymond Clavel. A fast robot with parallel geometry. In *Proc. Int. Symposium on Industrial Robots*, pages 91–100, 1988.

Sofia Gedda, Christian Andersson, Johan Åkesson, and Stefan Diehl. Derivative-free parameter optimization of functional mock-up units. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 076, pages 819–828. Linköping University Electronic Press, 2012.

Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

Ulrich Krengel. *Einführung in die Wahrscheinlichkeitstheorie und Statistik*, volume 8. Springer, 1988.

Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

A. Raue, M. Schilling, J. Bachmann, A. Matteson, M. Schelke, D. Kaschek, S. Hug, C. Kreutz, B. D. Harms, F. J. Theis, U. Klingmüller, and J. Timmer. Lessons learned from quantitative dynamical modeling in systems biology. *PLoS ONE*, 8(9):e74335, Sept. 2013. doi:10.1371/journal.pone.0074335.

Generic FMI-compliant Simulation Tool Coupling

Edmund Widl¹ Wolfgang Müller²

¹Center for Energy, AIT Austrian Institute of Technology, Austria, edmund.widl@ait.ac.at

²Institute of Analysis and Scientific Computing, TU Wien, Austria, wolfgang.mueller@student.tuwien.ac.at

Abstract

The Functional Mock-up Interface (FMI) specification provides a simple yet effective definition for co-simulation APIs. Even though the number of simulation tools supporting the export of Functional Mock-up Units (FMU) is growing steadily, there is a considerable number of well-established tools that do not. This paper addresses this issue by introducing a generic and adaptable way of coupling established simulation tools in an FMI-compliant manner. The proposed concept has been implemented as part of the FMI++ library, which is used as basis for FMI-compliant wrappers for the TRNSYS simulation tool and the MATLAB environment. These examples demonstrate the potential of the proposed approach to include well-established simulation tools with minimal effort. This not only enables researchers and engineers to include a diverse range of tools more easily into their work flow, but is also an incentive for tool developers to provide FMI-compliant wrappers.

Keywords: *FMI for Co-Simulation, tool coupling, front-end/back-end concept, TRNSYS, MATLAB*

1 Introduction

The list of simulation tools offering FMI (Blochwitz et al., 2011) support is rapidly growing¹, demonstrating the feasibility of the approach and underlining the importance of such a specification for Co-Simulation (CS) and Model Exchange (ME). However, many established simulation tools do not yet offer APIs for co-simulation, let alone one that follows the FMI specifications. This paper explores a generic approach that facilitates the integration of FMU CS export functionalities for such tools.

The proposed approach uses a front-end that is exposed to the master algorithm as FMI component, and an appropriately linked back-end that is used by the slave application. Due to its design this approach can not only be used by tool developers who have access to the (possibly closed) source code of the core application but also by users who have only limited access to or knowledge of the underlying application layers. The only requirement is the possibility for users to provide custom objects based on C/C++ code (or languages with adequate bindings to C/C++) that can be embedded within and exchange data with the simulation environment.

2 The FMI-compliant front-end/back-end concept

The basic concept comprises two components: The *front-end* component to be used by the simulation master and the *back-end* component to be used by the slave application. Between these two components a proper *data management* has to be established that is responsible for the communication and data exchange between both ends. The corresponding interfaces are tailored to suit the requirements of the FMI specification. They implement the necessary functionality required for a master-slave concept, i.e., synchronization mechanisms and exchange of data. See Figure 1 for a schematic view of this concept.

2.1 Front-end component

The front-end component is the actual gateway for a master algorithm to communicate and exchange data with an external simulation application. Its interface (see Figure 2) is designed such that it can be easily used as an FMI component (FMI model type `fmiComponent`), implementing functionalities close to the requirements of the FMI specification, for instance functions `initializeSlave(...)`, `doStep(...)` or `setReal(...)`. The front-end is responsible for the following tasks:

2.1.1 Information retrieval

The front-end component parses the FMI model description and stores the relevant information. This includes general simulator attributes (e.g., executable name, event handling capabilities) as well as specific model information (e.g., simulator-specific input files, variable names and types, input/output relations).

2.1.2 Variable initialization

Once the model description information is retrieved, the memory for the variables has to be allocated. This is done with the help of the dedicated data management (see Section 2.3 below).

2.1.3 Variable handling

The front-end has to manage the mapping between the model specific variable names and the associated value references according to the FMI specification. The latter are used to refer to and access the variables through the FMI API. In addition, the front-end has to ensure during runtime that variables are accessed properly, e.g., prohibiting write requests for output variables.

¹See <https://fmi-standard.org/tools/>.

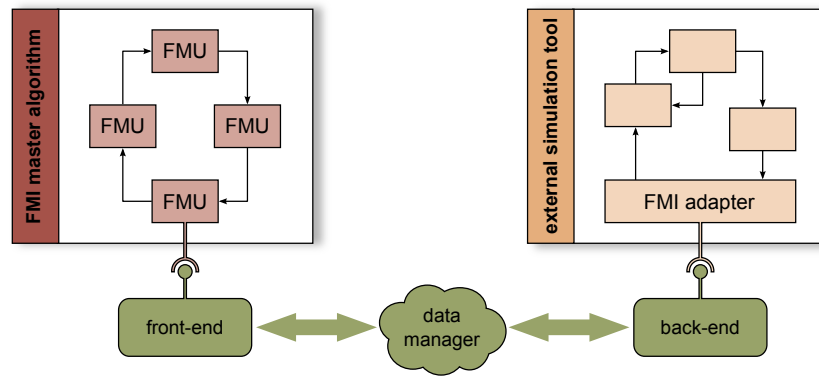


Figure 1. Schematics of the FMU CS export using the front-end/back-end concept: A simulation tool couples via an internal component to the back-end. The co-simulation master algorithm uses an instance of the front-end as FMI component. Synchronization and data exchange between the two ends is handled via a dedicated data manager.

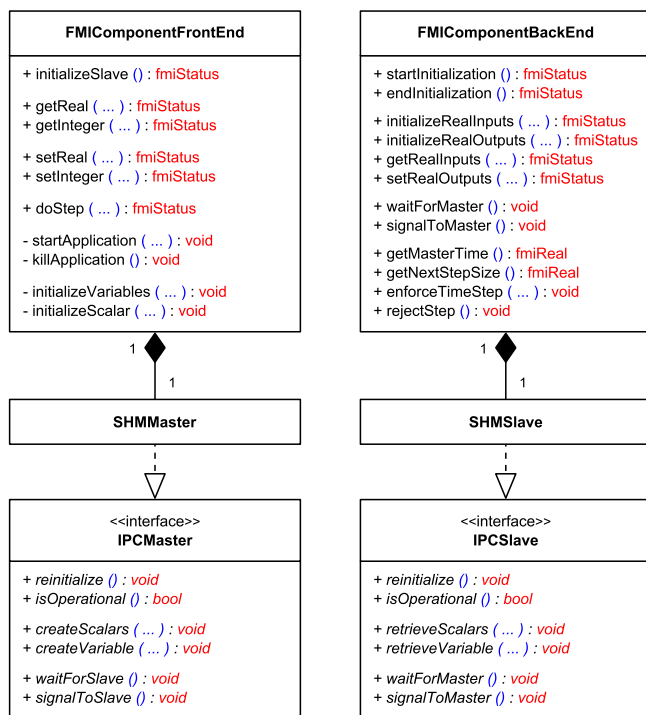


Figure 2. UML diagram of the most important features of the front-end and back-end components and the classes responsible for their data management (via shared memory in this specific case). The function arguments are not shown due to space constraints.

2.1.4 Application handling

The front-end is responsible for starting the external simulation application. It also has to establish a synchronized communication and data exchange, which is again done with the help of the dedicated data management (see Section 2.3 below)

2.2 Back-end component and FMI adapter

The back-end component functions as counterpart to the front-end component and is intended to be incorporated within the slave application as part of a dedicated simu-

lation component, referred to as the *FMI adapter* (see Figure 1). The back-end interface is designed to make the connection with the front-end as simple as possible, focusing on synchronization and data exchange (see Figure 2). The adapter has to carry out the following tasks with the help of the back-end:

2.2.1 Information retrieval

The adapter has to be a part of the model that is loaded in the external simulator. As such it has to be able to retrieve and store information about the model it is embedded in at run-time, most importantly the names and types of the inputs and outputs that should be shared within the co-simulation.

2.2.2 Establishing the data exchange

Once the names and types of all inputs and outputs are known, the adapter has to connect to the front-end and establish the synchronized data exchange. This is done with the help of the back-end component, which retrieves pointers to automatically synchronized variables via the dedicated data management (see Section 2.3 below).

2.2.3 Data exchange during simulation

The adapter has to be designed such that it knows at which points of the simulation it has to send/receive data to/from the front-end. Using the previously retrieved pointers it can read/write data with the help of the back-end component.

2.3 Data management

The *data manager* is the crucial link between the front-end and the back-end and handles all issues regarding Inter-Process Communication (IPC). It is split in two instances (see Figure 2), implementing the purely abstract interface definitions provided by *IPCMaster* and *IPCSlave*, which are intended to be used by the front-end and the back-end, respectively.

2.3.1 Data handling

Both interfaces are primarily designed for handling FMI scalar variables (XML type *fmiScalarVariable*),

Listing 1. Implementation of function `fmiDoStep(...)` according to the FMI 1.0 specification.

```

1  fmiStatus fmiDoStep( fmiComponent c, fmiReal currentCommunicationPoint,
2                      fmiReal communicationStepSize, fmiBoolean newStep )
3  {
4      FMIComponentFrontEnd* fe = static_cast<FMIComponentFrontEnd*>( c );
5
6      return fe->doStep( currentCommunicationPoint, communicationStepSize, newStep );
7  }

```

i.e., variables that are associated not only to a value represented by a basic data type (e.g., `fmiReal`) but also to model-related attributes (e.g., name, value reference or causality). The corresponding functionality is provided via `createScalars(...)` and `retrieveScalars(...)`.

In addition, the data manager allows to handle and access data with functions `createVariable(...)` and `retrieveVariable(...)` for internal communication between both ends (e.g., size of next time step, boolean flag for rejecting the next step).

2.3.2 Synchronization

Since the data exchange between both ends has to be synchronized, the data manager is not only responsible for allocating memory. It also has to have a way to control the access to the data, in order to prevent non-deterministic behavior.

This is realized via the functions `waitForSlave()` and `signalToSlave()` for the front-end and the functions `waitForMaster()` and `signalToMaster()` for the back-end. In both cases, variables that were instantiated via the data manager should not be read or written unless the blocking functions `waitForSlave()` or `waitForMaster()` return. Likewise, once a component is done reading or writing data, it is required to signal this via `signalToSlave()` or `signalToMaster()`, respectively, and wait again.

2.3.3 Flexibility

The abstract interfaces `IPCMaster` and `IPCSlave` have been designed such that the actual data transfer and synchronization can be achieved in various ways. For instance, shared memory access or communication via local or network sockets is feasible. In principle, this mechanism could even be used to build web applications.

3 Implementation

The above concept has been implemented for the FMI CS specification version 1.0 and version 2.0 as part of the FMI++ library². The FMI++ library is an open-source software toolbox written in C++ that provides high-level functionality for handling FMUs. As such it intends to bridge the gap between the basic FMI specification and typical requirements of simulation tools. While some of

the functionality offered by the FMI++ library for importing FMUs is comparable to what it is available in other software libraries (such as the FMU SDK³ or the FMI Library⁴), the implementation of the concept for generic tool coupling as explained above is unique.

A data manager has been implemented that uses shared memory access to share data, including semaphores for the synchronization of both ends, relying on features provided by the Boost⁵ library collection. In this case, both ends of the data management can physically access the same data. If the co-simulation master and the external application were executed on different machines (distributed simulation environment) both ends would have to allocate their own memory and keep their contents synchronized, e.g., by means of the Message Passing Interface (MPI Forum, 2009).

The implementation of the front-end, the back-end and the data management are generic, i.e., it is independent of the external application. FMI adapter implementations obviously depend strongly on the designated application, even though reasonably sophisticated simulation environments should offer the possibility to design it model-independent. Bindings for the generic back-end implementation to FMI adapters in other languages than C/C++ can be automatically created with the help of the SWIG tool (Beazley, 2003).

In addition, a thin layer implementing all functions according to the FMI specification is needed, which calls the corresponding front-end component functions, see lines 4 and 5 of the code snippet in Listing 1 for an example. Since version 1.0 of the FMI specification defines the model name (FMI model description attribute `modelIdentifier`) as a prefix to all functions in the final shared library, this thin layer has to be recompiled for each individual exported model. However, this does not require any changes in the source code, as the actual functionality remains unaltered.

4 Examples

The concept explained above is very flexible and can be used within a broad context of applications. For developing an FMI adapter, only three requirements need to be satisfied by any tool:

³Available at <http://www.qtronic.de/en/fmusdk.html>.

⁴Available at <http://www.fmi-library.org/>.

⁵Available at <http://www.boost.org/>.

²Available at <http://fmipp.sourceforge.net/>.

- **Modularity:** The targeted tool has to offer a mechanism for including user-defined code (including the possibility to access memory), in order to define an FMI adapter.
- **Execution control:** User-defined code has to be able to impact the tool's execution. This can be achieved either actively (e.g., by accessing methods that directly control the execution) or passively (e.g., by halting the execution through a blocking function).
- **Language compatibility:** The language of the user-defined code has to be compatible or have bindings to an existing front-end/back-end implementation.

In the following, this is demonstrated for two distinct tools.

4.1 TRNSYS FMI adapter

4.1.1 Implementation

TRNSYS (Klein et al., 1976) is a popular and well established thermal building and system simulation environment that comes with a validated components library. It uses instances of so-called *types* to model the individual components of a building or a system. Unfortunately, it does not provide an API that allows to use it as a slave application within a co-simulation. However, TRNSYS fulfills all the prerequisites to use the above discussed concept to implement an FMI adapter that overcomes this limitation:

- **Modularity:** In addition to providing a rich library of validated types, TRNSYS also offers the possibility to include user-defined types. Since TRNSYS is based on Fortran, these types are not object-oriented components in a strict sense but follow a sufficiently similar design pattern based on specialized function calls.⁶
- **Execution control:** The overall simulation execution is steered by the TRNSYS core, which calls the individual instances of the types included within a model. During these calls the instances are told at which stage the simulation currently is, especially whether it is the initialization phase, a standard call during a time step or the last of a time step. This information can be used by TRNSYS types to take actions accordingly.
- **Language compatibility:** Due to the TRNSYS simulation core being implemented in Fortran, user-defined types can be implemented using C/C++. Even though the ability of Fortran programs to call compiled C/C++ functions is limited, for the task at hand all conditions are met to establish sufficient interoperability.

⁶ Basically, every TRNSYS type is implemented as a function. Individual simulation components based on the same type are handled via the same function call, using a unique ID and appropriate memory storage utilities that allow to differentiate between the instances.

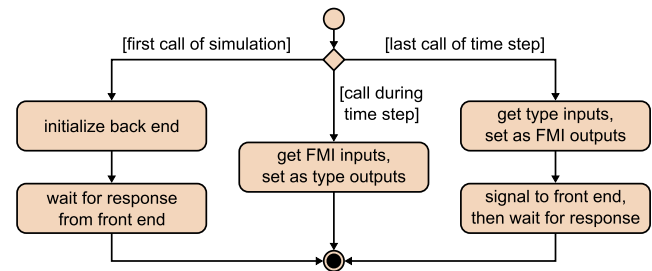


Figure 3. Schematic view of the functionality of the TRNSYS FMI adapter type in dependence on the simulation step.

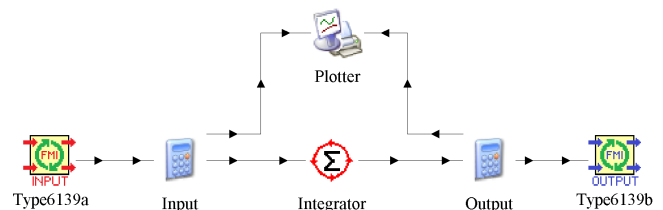


Figure 4. Example of a simple TRNSYS model containing blocks of Type6139 for FMU export.

Therefore the implementation of the front-end and back-end concept discussed in Section 3 can be used to develop a TRNSYS type that acts as FMI adapter. Figure 3 depicts the internal use of the back-end component within this type and its interaction with the simulation master. Please note that inputs to the TRNSYS FMI adapter type are the FMU's outputs and vice versa. Due to the strict fixed step size simulation paradigm of TRNSYS the adapter enforces time steps accordingly using the back-end component's `enforceTimeStep(...)` function. The front-end handles this information accordingly and rejects calls of `doStep(...)` in case they do not conform. The model description flag `canHandleVariableCommunicationStepSize` is set accordingly.

This FMI adapter has been implemented on top of the FMI++ library and is available online⁷. The provided FMI adapter – referred to as *Type6139* – can be included within a TRNSYS model like any other type, with ordinary inputs and outputs coming from and going to other types. In addition, the names of the input and output variables have to be provided (as part of the *Special Cards* in the type's *Proforma*) according to the definition that is also used in the model description. Apart from the additional input and output block of this type, TRNSYS models are constructed in the usual way. Given such a model, an FMU can be generated with the help of a dedicated Python script.

4.1.2 Example application

The example uses a simple thermal model from TRNSYS (see Figure 4) that implements the following first order

⁷ Available at <http://trnsys-fmu.sourceforge.net/>.

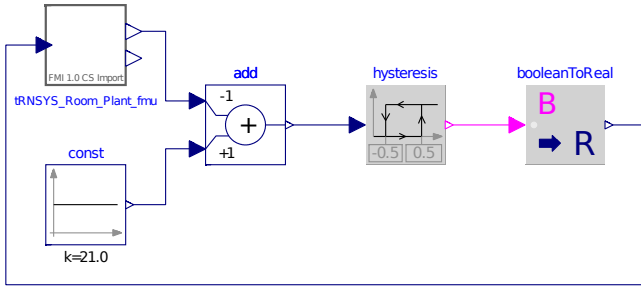


Figure 5. Dymola model.

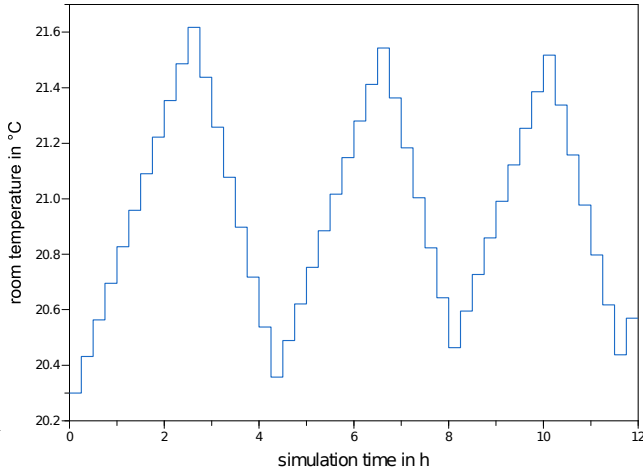


Figure 6. Example TRNSYS FMU output.

ODE:

$$\dot{T}_{\text{room}} = \begin{cases} -Q_{\text{loss}} & \text{if heater is off,} \\ Q_{\text{heater}} - Q_{\text{loss}} & \text{if heater is on.} \end{cases} \quad (1)$$

T_{room} is the room air temperature, Q_{loss} the difference between losses to the environment and inner loads, and Q_{heater} is the power of the heating unit. Both Q_{loss} and Q_{heater} are normalized w.r.t. the thermal capacity of the room air. The model was exported as an FMU with one input variable (associated to the on/off signal of the heater, called `control_signal`) and one output variable (associated to the room temperature, called `room_temperature`).

To test its functionality, the FMU was used as a plant model in a simple closed-loop control system implemented in Dymola, see Figure 5. Depending on the room temperature provided by FMU output variable `room_temperature`, the controller turns the room's heating on or off by setting the FMU input variable `control_signal` to either 0 or 1. More precisely, the model implements a hysteresis controller that turns the heater on as soon as the room temperature falls below 20.5°C and turns it off when it exceeds 21.5°C.

Figure 6 shows the results of the simulated Dymola model. Depicted is the room temperature as computed by TRNSYS, which is kept within 21.0°C ± 0.5°C by the Dymola controller. Due to the fixed simulation step size of 15 minutes, the switching of the controller state does

not happen at the exact edges of the controller's dead-band (i.e., at 20.5°C and 21.5°C). Please be aware that this is not a shortcoming of the FMU itself, but due to TRNSYS's restriction to fixed simulation time steps. Such simulation artifacts are unavoidable in fixed-step co-simulation and have to be taken into account by the modeler (e.g., by choosing an adequate simulation step size).

4.2 MATLAB FMI adapter

4.2.1 Implementation

Despite the popularity and widespread use of the numerical computing environment MATLAB, there is so far only comparably little support within the context of FMI. The Modelon FMI Toolbox⁸ and the FMI Kit for Simulink⁹ offer the export of Simulink models as FMUs for Model Exchange, but so far there is no tool available that allows to provide MATLAB's full functionality via an FMI-compliant co-simulation interface. In the following, a description is given of how the proposed front-end/back-end concept can be utilized to solve this issue.

- **Modularity:** Since MATLAB is a multi-purpose, multi-paradigm computing and programming environment, there are potentially many possible ways to implement an FMI adapter. Within the context of this work, an object-oriented approach has been chosen that relies on a base class called `FMIAdapter`, which provides the full functionality of the FMI adapter. In order to utilize its functionality, the abstract methods `init(...)` and `doStep(...)` have to be implemented by a derived class.
- **Execution control:** In contrast to Simulink, MATLAB defines itself no general notion of time. With the proposed concept, calls to the FMU's `doStep(...)` function are associated to a call to method `doStep(...)` of class `FMIAdapter` (or rather the class derived from it). For such a function call, the current communication point and communication step size are provided as input arguments.
- **Language compatibility:** MATLAB provides many ways for interfacing. Within the context of this work, the SWIG tool has been used to create S-Function bindings to a generic back-end implementation in C++ that can be called from within MATLAB. Even though these bindings can be used directly from MATLAB scripts, it is recommended to utilize their functionality through class `FMIAdapter`.

The MATLAB FMI adapter has been implemented on top of the FMI++ library (for Windows with 32-bit MATLAB) and is available online¹⁰. As mentioned above, it requires to implement the abstract methods `init(...)` and

⁸Available at <http://www.modelon.com/>.

⁹Available at <http://www.3ds.com/>.

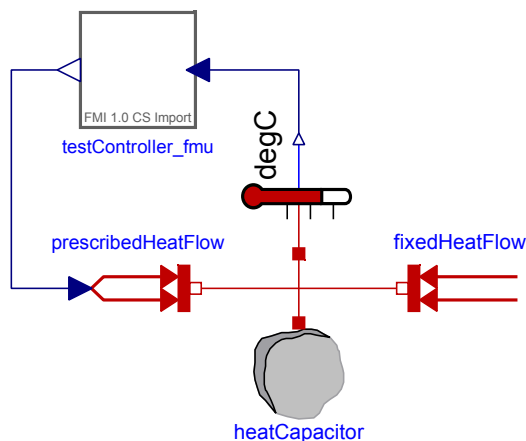
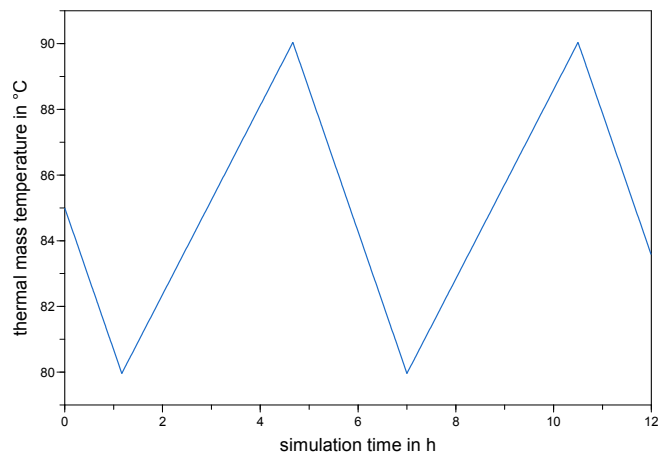
¹⁰Available at <http://matlab-fmu.sourceforge.net/>.

Listing 2. MATLAB implementation of the FMI adapter for a simple on/off controller.

```

1  classdef SimpleController < fmipputils.FMIAdapter
2
3      methods
4
5          function init( obj, currentCommunicationPoint )
6              obj.defineRealInputs( { 'T' } );
7              obj.defineRealOutputs( { 'Pheat' } );
8          end
9
10         function doStep( obj, currentCommunicationPoint, communicationStepSize )
11             realInputValues = obj.getRealInputValues();
12             T = realInputValues(1);
13             if ( T >= 90 )
14                 obj.setRealOutputValues( 0 );
15             elseif ( T <= 80 )
16                 obj.setRealOutputValues( 1e3 );
17             end
18         end
19
20     end
21
22 end

```

**Figure 7.** Example Modelica thermal system model.**Figure 8.** Example Dymola output.

doStep(...) of class FMIAdapter with the help of an inherited class, see for instance the example code in Listing 2.

Method init(...) is intended to initialize input and output variables needed for co-simulation. For instance, input and output variables of type fmiReal can be initialized with the help of methods defineRealInputs(...) and defineRealOutputs(...), whose input arguments are cell arrays containing the associated variable names. Method doStep(...) is called at every simulation step (as requested by the master algorithm). During such a call, methods getRealInputValues() and setRealOutputValues(...) can be used to get input and set output values for instance.

Since the init(...) and doStep(...) methods may contain any MATLAB-compliant code, virtually any MAT-

LAB functionality can be made available with the help of this concept. In order to create an FMU from such an implementation, the dedicated script createFMU.m has to be called from within MATLAB. Its inputs arguments are only the intended FMI model identifier of the FMU and the path to the class file implementing the FMI adapter. Additional MATLAB files may also be specified, e.g., containing data or further function definitions.

It is also noteworthy that an FMI adapter's functionality can be tested and debugged directly from within MATLAB. Unless explicitly activated, instances of FMI adapters do not try to connect to a back-end component. In this state, the input (output) variables defined by calling the init(...) method can be set before (read after) a call to the doStep(...) method from within MATLAB with a set of dedicated methods.

4.2.2 Example application

This example uses a simple on/off controller implemented in MATLAB, to control a thermal system implemented in Modelica. The Modelica model consists of a thermal mass that is connected to a constant negative heat flow (heat sink) and a heater, see Figure 7. The underlying equations of this model are analogous to the previous example, cf. Equation 1. The temperature of the thermal mass is sent as input to the controller, which can set the heater's power output. The MATLAB implementation of the controller is shown in Listing 2. Method `init(...)` defines in line 6 an input variable called `T`, associated to the temperature of the thermal mass, and in line 7 an output variable called `Pheat`, which controls the heater's power output. Method `doStep(...)` retrieves the value of previously defined input variable (lines 11 and 12) and sets the values of the previously defined output variables according to its simple internal logic (lines 14 and 16, respectively).

To test its functionality, the MATLAB controller was exported as FMU and imported into the Modelica model. Figure 8 shows the simulation results. Shown is the temperature of the thermal mass as computed by Dymola, which is kept within the range specified by the controller implementation.

5 Conclusion and Outlook

This work presented a generic approach for FMI-compliant tool coupling for a broad spectrum of tools. The approach is based on the concept of a generic front-end and back-end, with the front-end being directly accessed by a master algorithm as an FMI component. The back-end, which is synchronized to the front-end via a data manager, is associated to the coupled tool. The tool itself interacts with the back-end via a dedicated FMI adapter.

The proposed concept has been implemented as part of the FMI++ library according to the Functional Mock-up Interface version 1.0 specification and adapted to two

distinct tools, TRNSYS and MATLAB. In both examples the same front-end, data manager and back-end have been used, with customized FMI adapters to meet the requirements of the specific tools. With the help of two simple co-simulation setups the functionality of both approaches has been shown.

Future work will comprise the extension of the FMI++ implementation to support optional functionality, e.g., handling of input derivatives.

Acknowledgments

Part of this work emerged from the Annex 60 project, an international project conducted under the umbrella of the International Energy Agency (IEA) within the Energy in Buildings and Communities (EBC) Programme. Annex 60 develops and demonstrates a new generation of computational tools for building and community energy systems based on Modelica and the Functional Mock-up Interface standard.

References

- D.M. Beazley. Automated scientific software scripting with SWIG. *Future Generation Computer Systems*, 19(5):599 – 609, 2003.
- T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J.-V. Peetz, and S. Wolf. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In *Proceedings of the 8th International Modelica Conference*, 2011.
- S. A. Klein, J. A. Duffie, and W. A. Beckman. TRNSYS: A transient simulation program. *ASHRAE Transactions*, 82:623 – 633, 1976.
- The MPI Forum. MPI: A Message-Passing Interface Standard. Technical Report Version 2.2, Sept. 2009. URL <http://www.mpiforum.org/>.

FMI and IP Protection of Models: A Survey of Use Cases and Support in the Standard

Erik Durling* Elias Palmkvist Maria Henningsson*

*Modelon AB, Sweden, Corresponding author: erik.durling@modelon.com

Abstract

FMI is increasingly being adopted as a standard for exchanging simulation models within and between organizations. Such models often represent significant investments for the model creator. There is thus a large interest in protecting intellectual property while collaborating and sharing simulation models in the form of FMUs. This paper presents a collection of use cases and issues related to IP protection of model contents, that have been identified in interviews with industrial representatives. The requirements in each use case are described, along with an investigation of how well the use cases can be managed within the current version of the FMI standard, including a proposed extension of the standard.

Keywords: *FMI, IP protection, model exchange*

1 Introduction

The promise of major benefits in model-based systems engineering and virtual development lies in reusing models in different contexts. To develop, parameterize, validate, and maintain models represent a significant investment, and to maximize the return the models need to be utilized as much as possible.

FMI is becoming the de facto industry standard for exchanging models between different tools. Two main directions in the FMI domain is currently integration and democratization. Integration means software, processes, and standards for co-simulation of multiple models from different tools or different organization. Democratization means effort to spread the usage of advanced simulation models for experts using expert tools to much larger groups of engineers to use for design space exploration, boundary conditions for other systems, or software development and testing.

Both these directions involve exposing models that often represent significant investments and contain sensitive data to a larger user base within and outside of the original organization. The question about protecting IP (Intellectual Property) is often raised in discussions about exchanging models between partners with commercial interests.

Although there exist solutions and best practices for sharing models with existing technologies, FMI is still a new standard, and there is a general need for knowledge

about applying similar solutions with FMI (Köhler et al. 2016). One of the arguments for using FMI is that it allows protecting the internal contents of models. But it is important for the part sharing a model to understand what is exposed, and what measures that can be taken to protect what should not be shared.

The purpose of this study has been to make an inventory of use cases and concerns related to IP protection of FMUs, and to evaluate to what extent this is supported by the current standard. This overview can be of interest for users who need to understand the risks and mechanisms for exposing and protecting the content of their models.

The study also intends to raise the need for a standardized way of managing IP protection mechanisms of FMUs, or at least to provide information to the model importer about embedded mechanisms to restrict execution of the model.

The paper starts by outlining how the listed use cases were elicited. The list of use cases is presented in Section 3. An evaluation of how well the use cases are supported by the current (2.0) FMI standard is found in Section 4.

2 Methodology

The study was carried out in two phases. During the first phase, information was gathered about the needs that exist for protecting IP when sharing models within the general area of model based systems engineering.

Interviews were carried out with 16 engineers at Modelon and Volvo Car Group, with experience of sharing models within automotive, energy and aerospace industries. The interviews were typically with a single person at a time, and lasted in the range of 30 to 60 minutes. To obtain unbiased information, open questions were asked to let the stakeholder present their own view of the issues they found important. The questions concerned exchanging models in general, and not specific to the FMI standard.

In addition to the interviews, an anonymous online survey was sent out to additional external stakeholders. The purpose of this survey was mainly to obtain an impression of the importance and priorities among the identified use-cases. The survey also included specific

questions to identify experience and concerns specific to the FMI standard.

During the second phase, the FMI-standard was evaluated in terms of each of the use cases that had been identified. The following questions were considered when evaluating the standard:

- Is the use case relevant to be considered within the scope of the standard?
- Is there any support for the use case within the standard?
- Are there any obstacles or gaps within the standard that prevents solutions for the use case from being implemented?

3 Use Cases: Needs for Protecting IP When Sharing Models

This section describes the different use-cases that have been identified in this study, following a short summary of the roles involved.

The question of IP protection is typically considered when models are to be shared between different organizations with commercial interests. The purpose is to protect valuable or sensitive knowledge or data from being accessed by someone who is not trusted. The need for protection generally comes from the part sharing (exporting) the model, but there are issues related to this that may affect the receiver (importer) of the model.

A common scenario is that a component supplier delivers a component model to an OEM (Original Equipment Manufactory). This component model is integrated by the OEM as part of a system model. The opposite also occur, where the OEM supplies a system model, to let the supplier test their component as part of a system environment.

There are also situations where there is a need to protect models that are shared inside the same organization. Reasons for this can be to maintain control over what models are being used in the organization. Another reason can be to prevent potential leaks by limiting access to sensitive information. This situation could also apply during projects with external partners, where the information is not secret to the people in the project, but there is a need to protect the information from being shared outside the project.

In general, the main concerns regard export of models. This mainly covers two main issues: hiding the model content, and controlling who can use the model. But protecting the models can also lead to challenges for the receiver of the model, in terms of usability, that need to be considered.

3.1 Use Case 1: Protect Model Contents

The basic use case is that the part who shares a model would like to hide what is inside from the receiver. The sharing part needs to export the model in such a way that the contents are protected. There are a number of aspects

that may be valuable or sensitive and needed to be protected.

3.1.1 Model Structure

There is often a need to protect the structure or design of the model. This consists of equations and algorithms that describe the relationship between inputs and outputs.

The model may be implemented using unique methods for describing the specific component. Examples of this could be algorithms, or representations of equations, or clever ways to select dynamic states. This can make the model design valuable in itself.

The model could also represent unique knowledge about the component that is modeled, and could reveal sensitive information about the actual component design and properties.

Some models might be created to support multiple application. In this case, information about the other types of application that is supported might be sensitive. It could be that the receiver should only have access to information that concerns their specific application.

3.1.2 Internal Variables

Internal variables (or "signals") may reveal sensitive information about the inner workings of a model, and could facilitate reverse engineering.

The names of the internal variables could also be sensitive and reveal information about the model structure and design, or ways to apply the model that the receiver should not be aware of.

3.1.3 Parameters

Values of internal design parameters, boundary conditions and start values, may reveal information that would not be available to a user of the actual component or system that the model represents. This data may be the result of expensive research, and considered valuable knowhow that a supplier is reluctant to share.

Parameter names may also reveal information about model structure, in the same way as internal variables. It could also be that the parameter values are only sensitive with a specific parameter name. Generic parameter names may not reveal any useful IP.

3.1.4 Black Box or Grey Box

The simplest approach is to hide everything inside the model (black box), which may be sufficient in some cases. However, in many cases it is necessary to expose parts of the model contents (grey box), in order to make the model usable. In this case, the exporter typically would like to expose only the sub-set of the content that is necessary for the receiver to have access to.

For example, exposing part of the model structure or internal variables could aid in simulation debugging. And some parameters may need to be tweaked in order to use the same model for multiple scenarios.

3.1.5 External Dependencies

A model could contain external dependencies, for example parameter files or additional model libraries. These parts could contain IP that may not be covered by the protection applied to the main model, and may require specific consideration.

3.1.6 Reverse Engineering

Sharing a model always comes with a risk of reverse engineering, either of the model itself or the component that the model represents. The only way to be completely protected against this is to not share any model. The required level of protection against this depends on the value of the model contents and the risk of the contents being revealed. A common strategy of handling this is to make sure that the cost of reverse engineering is higher than the value of the contents.

3.2 Use Case 2: Limit Access to Users

A common scenario is that only a set of expected users should have access to a model, for example to maintain control or prevent reverse engineering. A model could contain information that should not fall into the wrong hands, or be used for applications other than the exporter's intention.

3.2.1 Limit Access to Specific User(s)

There are many scenarios where a model is only meant to be shared with a limited number of users, or different users should have different level of accessibility to the model. It is common that the right to use a model is given to a single organization by a partner. In sensitive cases, some models may even be restricted to specific groups within an organization, to minimize the risk that it ends up in the hands of the wrong people, for example a competitor. There are also commercial scenarios. For example, a model library may be sold for use on a single computer only.

Models exported from some tools may be restricted to users who have a license for the exporting tool. Such limitations may represent a big obstacle for some scenarios of model sharing. It may not be feasible for users integrating models from many different sources to have a license for all the tools. This could also be a problem when an exported model need to be deployed to a large group of end-users, since the licensing fee would become unreasonable. Some OEMs have also expressed concerns that licensing solutions on exported models could lead to vendor lock-in.

3.2.2 Limit the Model Over Time

There are also scenarios where one would like to limit the model access to a specific time frame. A reason could be that the model could contain information or be used for applications that is only relevant during a limited time, and the use of the model may even be contracted between the two partners. This could for

example be during the course of a specific project, or during a trial period of a commercial model library.

Having a time limitation on a model could also be a benefit when it is being developed and need to be maintained over time, since it reduces the risk that an old version of the model is used.

The time frame could differ depending on the use case, from a couple of days for a sales demonstration, a few months between model release versions, or during a project that last for years.

3.2.3 Information About the Protection

Models with limited access pose a challenge from the model receiver's perspective. Without sufficient information about what type of protection is applied, debugging could be difficult when the user or the importing tool should identify that the model is not working due to this protection.

This is especially important for a user working with aggregates of models from multiple model suppliers, where there may be number of different types of access limitations that need to be managed. The workflow for these users are improved if it is possible to easily understand how each model is restricted and what is required for getting access to it.

3.3 Use Case 3: Provide Information to the Model Importer

When parts of a model are hidden, or protected, there is an increased need for information to keep the model useful.

3.3.1 Documentation

For a model with hidden contents, the user must rely on the documentation for information on how to use the model and what results to expect. It can be crucial to understand what aspects of the physical systems are modeled and at what degree of accuracy, especially when integrating the model as part of a larger system, or to understand simulation results. This could dictate what parts are needed outside the model and how to interpret the interface. It may also be difficult to determine what range of operating conditions the model is valid for, since it likely is not obvious what simplifications or assumptions have been made. In general, it is important that both parts have agreed on the interface of the model inputs and outputs.

3.3.2 Debugging

It can be very challenging to debug a model without knowledge about how it is constructed, without the ability to measure internal variables or to get usable error messages. This can be a challenge also when using the model as part of a larger system. The user may have to rely on support from the model supplier for solving the issues. This could also pose a challenge for OEMs that need to be able to trace issues found in simulation

results back to the source model, many years after the results were produced.

3.3.3 Network Dependencies

Some protection solutions may require the model to communicate with remote network resources, for example to gain access to using the model or to exchange results with a simulation server. Information about these dependencies and adequate error messages can be important for helping the user identify any issues related to this.

3.4 Use Case 4: Binary Platform Support and Source Code

There is a conflict between protecting a model from reverse engineering while at the same time allowing the model to be used on multiple platforms (different operating systems or processor hardware). Exporting a model in a compiled binary format is a common way to protect the sensitive content. However, this will limit the model to the specific platform that the binary is compiled for. To support multiple platforms, the solution is often to export the model as code (commonly C-code) and let the receiver compile the model on the specific platform. While binary export is often considered sufficient protection against reverse engineering, c-code is generally not considered sufficient, since this is more easily interpreted by a human. Solutions for this could place requirements both on the exporting and importing tools.

It is important to note however, that the content of a binary also can be interpreted, while the effort to do so is generally much higher than doing this for higher level source code.

3.5 Knowledge Need

A general need for knowledge about the IP-risks specific to FMI was identified during this survey. When exporting a protected model that contains IP, it is important for the exporter to understand what is exposed when the model is exported, and what risks may need to be avoided. This will help making correct decisions about what measures need to be taken, but is also necessary for the exporter to feel trust in the solution used.

It is worth noting that new technologies have a start-up phase in general, where potential users will be naturally skeptical before information about the technology is widely known, and best practices have been established.

It may also be important also for people that are only working indirectly with models understand how the risks are handled. For example, a lack of knowledge about the technology could represent an obstacle in and negotiations about sharing models between partners. A wider acceptance may be needed among all affected parts of an organization before it is regarded as safe.

3.6 Use Case 5: Authentication

Authentication concerns the need to ensure the integrity of a model. This question is not mainly about hiding content, but instead of protecting it from being changed. Although, it is sometimes discussed in relation to IP protection, since the challenges is somewhat related.

Some of the common needs are:

- Verifying that the model comes from the expected source.
- Verify that the model has not been altered after it was exported. This could be important in order to provide reliable support as a model supplier, or when the model is deployed in safety critical systems.
- Verify that the model is compatible with some external dependencies, for example that the specific version of a model is used together with the corresponding version of parameter data.

Authentication plays a role both as a sanity check to avoid mistakes, but also as a means of protecting against intentional intrusion.

4 Support for Protecting IP Within the FMI Standard

This section describes how and to what degree the use cases described in Section 3 are supported by the FMI-standard. Some examples are used to demonstrate how the standard supports certain use cases.

4.1 The Content of an FMU

An implementation following the FMI-standard is called an FMU. This section describes what parts of a model is exposed when being packaged as an FMU. An FMU is a zip-file, with a certain file structure, that contains the following parts:

- The model description XML-file:
Contains meta information about the model that will be exposed to the simulation tool and user.
- Binaries:
This is the actual implementation of the model, compiled for a specific (or multiple) target platform. This binary exposes the standard FMI API functions, for reading and writing variables and performing simulation time steps.
- Source code:
C-source code for the model can be provided as an alternative, or in addition, to a model binary.
- Additional data/resources:
This could be data stored in any format as a resource in the FMU. This would typically be parameter data. It is also possible for an FMU to access external resources outside of the FMU itself.

The FMI standard specifies the format of the model description XML-file, the API function interface of the binaries or source code, and the structure of the zip-file.

The FMI-standard allows two different type of models, one that contains a solver to simulate the model (Co-Simulation or CS-FMU) and one that requires an external solver to simulate (Model-Exchange or ME-FMU). The functions and exposed content differ somewhat between the two FMU flavors.

4.1.1 Content in the Model Description XML File

The model description XML-file contains necessary meta information to the user and simulation tool, in order to make the model useful.

The information required to be included is the type of FMU, name of the model, and a GUID (Global Unique Identifier).

In addition to this, XML-file is required to contain tags for model variables and model structure, which will contain a set of variables that are exposed. But there is no requirement from the standard that all model content, in terms of variable names or values, should be exposed in the model description XML-file.

In practice, at least the top level input and output variables are exposed. The model description could also contain references to all, or a subset, of the internal model variables and parameters. But it is up to the exporting tool whether all variables should be exposed, or none (black box) or a sub-set (grey box), and also what names to give the variables.

For the exporting user, it could be very helpful to get clear information from the exporting tool about what variables are exposed. Although this information is available in the xml-file, it can be very impractical to obtain the information by reading the file directly, especially for large models.

The variables defined in the model description file is a mapping between variable names and variable references. The variable reference (a number) is used to access the variable value with the FMI function calls in the binary or source. It is possible to include variables in the binary/source, that can be accessed by reference (a number), without having any mapping to a variable name in the model description file. This allows for “secret” variables. This also allows for defining “anonymous” variable names, that do not reveal any sensitive information about what the variables represent.

In order to avoid algebraic loops when using the FMU as part of a system, it could be necessary to provide a list of outputs and the variables that the outputs depend on to the importing tool.

Additional information can be included in the model description file that is typically not sensitive, like the exporting tool or experiment settings.

4.1.2 FMU Binaries

The FMU binaries typically represents a compiled implementation of the whole model. As discussed in

Section 3.4, compiling a model as a binary is commonly considered sufficient protection of the model implementation. It is however up to the exporting tool to ensure that what is stored in the binary is not exposed in an open way.

An FMI binary is only required to expose the FMI API functions. These will provide access to values of at least the variables defined in the model description file. For an ME FMU, it will also be possible to access the values of each of the internal (continuous time) state variables, their derivatives, and any event indicators. But the names of such internal state variables will not be exposed.

The FMI-standard provides support for logging, so that the FMU can generate messages for warnings and errors to the simulation environment. The message generated from the FMU, using the logging interface, could depend on hardcoded messages that might include internal model information (like variable names and values) that is not exposed in the model description XML-file. The standard allows using variable references when logging, which will avoid exposure of hidden names, but could still expose hidden value references and their values. It is up to the exporting tool to ensure that such internal messages are not exposing sensitive model content.

One way to support multiple platforms is to include multiple binaries in the same FMU. This may be an option if it is not possible to provide source code for the model (as described in section 4.1.3). This requires that the exporting tool is able to compile or package binaries supported by all different platforms. FMUs for multiple platforms could be supported through cross-compilation or with tools for merging multiple binaries into the same FMU. Note that the model description needs to match all of the binaries (including the GUID).

In some cases, the FMU binary just represents an FMI gateway, as an interface to an external application or interface (like another simulation tool or network sockets).

4.1.3 FMU Source Code

An FMU could include the source code for the model, in addition to, or instead of, the binaries. This is a way to allow the model to be compiled to a general target platform, to avoid supplying a binary for each platform where the FMU is to be used. Many suppliers are however reluctant to provide source code for their models, since this exposes the implementations of the models in a more open way than compiled binaries do. Depending on how the code has been generated, this may expose algorithms, parameters, and model equations that represent valuable IP.

A common way to deal with this is to apply code obfuscation, which makes the code very difficult to read. The effort of reverse-engineering would be similar to a compiled binary.

4.1.4 External Data in an FMU

An FMU may contain additional resources. This would typically be parameter files. This means that even if internal model parameters are not exposed through the FMI interface and model description, parameter data could still be openly readable through these resource files. To avoid exposing sensitive data, it could be necessary to apply encryption or some form of obfuscation of these resources.

An FMU can contain external dependencies for example to facilitate parameterization. The standard allows the FMU to contain additional data such as files with data tables. But the FMU is also allowed to access and use external files not included in the actual FMU.

The FMI-standard only specifies the communication interface between the model and the simulation tool. Access to external data is not covered by the standard. Handling of external data files needs to be considered separately, to avoid unintentional exposure of sensitive data.

4.2 Limit Access to the FMU

The purpose of limiting the access to the FMU is either to restrict the usage of the model or restrict the access to the content in the FMU. Reasons to protect the FMU is discussed in section 3.2.

There is nothing included in the FMI standard that either specifies or restricts how to limit the access to the model binary or source code. This means that it is possible to include any protection mechanism in the source code or binaries of the FMU.

4.2.1 Examples of Access Protection

Common examples of protection that could be applied are:

- **Server Solutions:** Only share access to the interface. The model content is protected on the server. The user can only access the FMI function calls. This type of solution will effectively protect the model files from unintended distribution and reverse engineering.
- **Encryption:** The main reason for encryption is to prevent the wrong user from accessing the model. In general, the model is exposed once it has been decrypted. There are many variations of workflows and encryption solutions, for example licensing of the decryption and password protected zip-file.
- **Licenses:** This can be applied to ensure that the model can only be used for a certain time, or to restrict the model to only be used by a given group of people. This licensing protection would be integrated into the binaries and will thus not protect the content of the XML-file.
- **Limitation over time:** The binaries can be generated to only work during a restricted timeframe. This is a way to protect the model from being executed. But

it does not protect the content of the model description XML-file.

4.2.2 Information About Applied Protection

In section 3.3, the importance of the available information to the recipient is discussed. The FMI standard does not specify a way to provide information to the model receiver about the type of protection applied or requirements for accessing the model. It is up to the exporter to inform the receiver, either in or outside the FMU. The standard also does not define any requirements or interfaces for protecting the access to an FMU.

The model description xml of FMI 2.0 may contain an optional flag that describes information about the intellectual property licensing. This provides information about how the FMU may be used, but not how it is protected in terms of technical licensing.

One proposal is to extend the standard with information about the type of technical licensing that is applied to an FMU. This could be added in the form of new attributes in the model description XML: "protection-type" and "protection-trigger", and an additional function in the header file "fmi2checkProtection". The "protection-type" attribute should contain information about what type of protection the FMU has, like "license-file", "time" etc. The "protection-trigger" contains information about how the protection is triggered, like "instantiation", "initialization", "2017-01-01" (for protection over time). To check if the FMU can be used at the current state, the function fmi2checkProtection can be called to perform a "validation check".

4.3 Authentication

Use cases concerning authentication were discussed in section 3.6. Authentication is not covered specifically in the current standard. However, implementation of authentication solutions in the model (binaries/sources) does not necessarily require any specific support from the standard. Two examples are given to demonstrate how the use cases can be implemented without specific support from the standard:

- **Verify the source of the FMU:** To verify that the FMU comes from the correct source, the checksum of the FMU could be digitally signed by the exporter, and provided in addition to the FMU. The signed checksum could then be used by importing tool to verify the integrity of the FMU.
- **Verify that the XML has not been altered:** The modelDescription.xml is most likely part of an FMU to be altered. It would be possible to include a function in the model binary that calculates and verifies the hash of the XML, and prevents the model from running if this is different from expected.

5 Conclusions

We have presented a survey of common use cases and concerns regarding IP protection when sharing models, and we have discussed to what extent this can be addressed within the current FMI standard.

The most common use cases concern export of models, mainly in terms of having control and information of what is exposed of the model content, as well as limiting access to unintended users. But there are aspects of this that also affect the importer, mainly in terms of usability and platform support.

Furthermore, a general need for knowledge dissemination was identified, regarding the risks and mechanisms of protecting the model content, specific to the FMI standard. One purpose of this article has been to address this need.

No obstacles were identified within the standard. All of the use cases described can be managed within the standard. Tools that export FMUs are free to include any conceivable solution for restricting the execution of the binaries, and are free to exclude all sensitive information from the model description file.

A risk for the model exporter is that sensitive information may be exposed in unintended ways, like through the logger, or through external dependencies not controlled by the standard at all.

For most use cases, it is more a question of support by the tool rather than support by the standard. The amount of information that is exposed depends a lot on the tool and specific export settings.

This leaves much freedom for tool vendors and model exporters, which can translate to challenges for model importers. The lack of standardized ways of imposing IP protection on models can make it difficult to deal with a multitude of different licensing or encryption mechanisms. Without a standardized interface it is hard to troubleshoot issues related to licensing issues. We therefore propose for a future version of the FMI standard to add an optional flag in the model description XML scheme to provide information about embedded protection that will limit execution.

Acknowledgements

This study was carried out within the research project Second Road Phase 2, coordinated by Volvo Cars Corporation and funded through the Swedish research agency VINNOVA. Time and input from all interviewees is gratefully acknowledged.

References

- FMI for Model Exchange and Co-Simulation, Version 2.0: <https://www.fmi-standard.org/>
- Köhler J., Heinkel H.-M., Mai P., Krasser J., Deppe M., Nagasawa M. Modelica-Association-Project “System Structure and Parameterization” – Early Insights. *The First Japanese Modelica Conferences, May 23-24, Tokyo, Japan, 2016*. doi: 10.3384/ecp1612435
- Köhler J., King J., Kübler M. Simulation of Complete Systems at ZF using Modelica Standards, *The First Japanese Modelica Conferences, May 23-24, Tokyo, Japan, 2016*. doi: 10.3384/ecp1612424
- “Smart Systems Engineering” project of the iViP Association: <http://www.prostep.org/en/projects/smart-systems-engineering.html>

Model-based virtual sensors by means of Modelica and FMI

M. González^{1,2} O. Salgado¹ J. Croes^{2,3} B. Pluymers^{2,3} W. Desmet^{2,3}

¹IK4-Ikerlan Technology Research Center, Control and Monitoring Area, Spain

²KU Leuven, Department of Mechanical Engineering, Belgium

³Member of Flanders Make, Belgium

Abstract

This paper presents an application case for the estimation of forces using Modelica and the FMI. For that purpose model-based virtual sensors are used. These techniques are presented and the development of the virtual sensor for Modelica and the FMI is discussed. The work has been done in Python where the package pyFMI is used with models exported with the FMI 2.0 for model exchange. The technique is used for the estimation of forces and the friction coefficient in a vertical transportation system. The model of this test bench is explained and the results of the estimation of forces and the friction coefficient are discussed. These estimations provide a valuable tool for the condition monitoring of guiding systems.

Keywords: *FMI, virtual sensors, pyFMI, Extended Kalman Filter*

1 Introduction

The condition of the guiding system influences significantly the riding quality and performance of transportation systems such as railways or elevators. The proper design and the correct maintenance of the guides is therefore of high importance. Both the design and monitoring of the guiding system require an accurate assessment of the loading condition. However the direct measurement of forces is not feasible, as a dedicated sensor is too costly and intrusive. Virtual sensors are an attractive option to overcome these difficulties.

Virtual sensors process available measurements to estimate other variables of interest that cannot be measured. Mainly two virtual sensor approaches are suggested in the literature: data-driven methods and model-based methods. Data-driven methods use a machine learning perspective to recognize patterns in the behavior of the system. These methods require previous observations of the system in order to learn the different states and conditions of the asset. A review of data driven virtual sensors can be found in (Kadlec et al., 2011). Some common approaches include developing autoregressive models of the system as in (Samara et al., 2013), using artificial neural networks ((Bizon et al., 2014),(Gonzaga et al., 2009)) or using moving window methods as in (Liu et al., 2009). The required data training may be a handicap in systems where data cannot be acquired continuously or in which faulty conditions cannot be measured.

On the other hand model-based methods combine physics-based models and measurements of the system by means of estimation algorithms. The model provides knowledge of the dynamics of the system, which in combination with off-the-shelf sensors can be used to estimate variables of interest otherwise difficult to measure. These approaches are valuable tool in several applications such as control techniques, condition monitoring or model updating (Isermann, 2005).

The performance of these techniques depends on the capability of the model to accurately represent the physics of the system (Isermann, 2005). In addition a great modeling flexibility and simplicity is required to avoid errors and speed up the process. Using Modelica has thus a great added value in the development of model-based virtual sensors. The acausal nature of Modelica allows efficiently modeling heterogeneous systems reusing already developed and tested models. However, it doesn't allow the user to manipulate the solution at each time step, as required by estimation algorithms. In order to use Modelica for state estimation the models have to be exported and manipulated at each time step (Brembeck et al., 2011).

Several modeling environments include model exchange capabilities. However, they are usually developed ad-hoc to interface with one particular tool in a certain context. Therefore they are commonly limited to certain tools and are version dependent. The Functional Mock-up Interface (FMI) is a tool independent standard that can efficiently solve this. Furthermore the FMI 2.0 includes some features that aid the development of state estimation algorithms (e.g. directional derivatives).

The combination of Modelica with other programming languages by means of the FMI provides thus a suitable approach for the implementation of model-based virtual sensors. The main focus of the FMI is simulation, but it has already been applied for estimation. For instance in (Brembeck et al., 2011) and (Brembeck et al., 2014) is used to implement nonlinear state observers within Dymola. In (Bonvini et al., 2014) an Unscented Kalman Filter (UKF) is implemented in Python using the FMI 1.0 for model exchange and is used for Fault Detection and Diagnosis. In this paper the FMI 2.0 for model exchange is used to develop an Extended Kalman Filter (EKF) for state and parameter estimation in Python. The suitability of Modelica and FMI for state estimation is tested with a highly nonlinear model which includes events, rotations

and friction.

The rest of the paper is organized as follows. Section 2 gives an overview of Model-based virtual sensors and explains the algorithms used in the current application. Section 3 explains how these algorithms are implemented using pyFMI with the FMI 2.0 for Model exchange. Section 4 describes the proposed application case along with the proposed model, a test bench of a vertical transportation system where contact and friction forces are estimated. The results of these estimations are shown in section 5. The final conclusions and the future work are drawn in section 6.

2 Model-based virtual sensor approaches

The core of model-based virtual sensors consists on the use of state estimation algorithms. These algorithms use the difference between the real measurements and the prediction of a physics-based model to correct the output of the model. The most common state estimation algorithms are the Luenberger observer (LO), the sliding mode observer (SMO) and the Bayesian estimators. LO and SMO are simpler to implement than Bayesian estimators but under noisy measurement conditions the Bayesian algorithms are proved to perform better (Zhang et al., 2009), (Esteban et al., 2016). Thus the presented work is focused only on Bayesian estimators.

2.1 Kalman Filter

The Kalman Filter (KF) is the optimal linear estimation filter (Simon, 2006). In the case of Gaussian noise, it provides the maximum a posteriori estimate with the smallest achievable covariance. With non Gaussian noise, it is optimal in giving the minimal mean square error. It is the most widely used Bayesian estimator and has been successfully used in a number of applications (Simon, 2006). The KF uses a linear model defined in state space form as the one shown in equation 1. In the stochastic Bayesian derivation of the KF, both the process and measurement equations are assumed to be disturbed by zero mean white Gaussian noise (\mathbf{w} and \mathbf{v} in equation 1) of covariance \mathbf{Q} and \mathbf{R} respectively. The states are assumed to be Gaussian variables with a covariance \mathbf{P} and mean the state estimation ($\hat{\mathbf{x}} \sim \mathbf{N}(\hat{\mathbf{x}}, \mathbf{P})$).

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, t) + \mathbf{w} \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}, \mathbf{u}, t) + \mathbf{v}\end{aligned}\quad (1)$$

The most common formulation of the KF requires the dynamic system of equation 1 to be described in a discrete form of equation 2, where for a linear model the matrices \mathbf{F} , \mathbf{G} and \mathbf{H} are constant. Generally the discretization of a state space model assumes a zero-order hold for the input \mathbf{u} and continuous integration for the noise \mathbf{v} . As explained in (Simon, 2006) the discretization involves the computation of the integral of a matrix exponential or any

equivalent discretization such as Euler or Runge-Kutta.

$$\begin{aligned}\mathbf{x}_k &= \mathbf{F}_{k-1} \cdot \mathbf{x}_{k-1} + \mathbf{G}_{k-1} \cdot \mathbf{u}_{k-1} + \mathbf{w}_{k-1} \\ \mathbf{y}_k &= \mathbf{H}_k \cdot \mathbf{x}_k + \mathbf{v}_k\end{aligned}\quad (2)$$

The KF algorithm is shown in figure 1. In each k-time step the system model is evaluated and compared against measured data. This is done in two steps: prediction and update. In the prediction step an a-priori estimation of the states mean and covariance is obtained from the system's model. In the update step this a priori estimation is corrected using the system's output. This estimation process is done recursively: all the prior information is summarized in the initial mean and covariance of each step ($\hat{\mathbf{x}}_0^+$, \mathbf{P}_0^+). Therefore the computational effort in each time step is the same regardless the number of measurements.

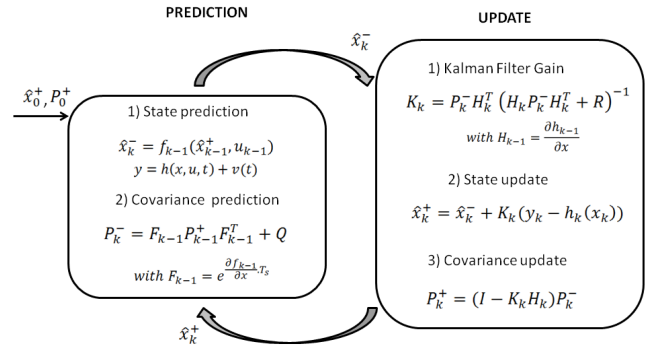


Figure 1. Kalman Filter algorithm

Despite being widely used, the KF is limited to linear systems, which also makes the joint estimation of states and parameters not applicable (Simon, 2006). Several suitable extensions of the KF to non-linear systems, such as the EKF or the UKF can be found in (Simon, 2006). As the current system is highly nonlinear (events, rotations) the well known Extended Kalman Filter is used instead.

2.2 Extended Kalman Filter (EKF)

The EKF is the most widely used extension of the KF for nonlinear systems and for the joint estimation of states and parameters. If the model of equation 1 is nonlinear the \mathbf{F}_{k-1} and \mathbf{H}_{k-1} matrices of equation 2 are no longer constant, but change at each K-step instead. Then the EKF linearizes and discretizes the model around the KF estimate, propagating a linear approximation of the covariance (Simon, 2006). The standard KF shown in figure 1 is then applied at this linearized point. As the estimation is based on the linearization of the system a small step size is required if the system is highly nonlinear. On the other hand, the ease of implementation and the reduced computational cost of the EKF make it an attractive option for the estimation of states in nonlinear systems.

2.3 Parameter identification and Virtual Sensors

State estimation algorithms can be augmented to estimate not only the states of the system but unknown parameters too. Based on the continuous state-space system representation, an augmented version of the system can be obtained if the unknown parameters are included in the states vector (equation 3) and their directional derivatives are included in the system matrices (equation 4). Then a random walk model is used for the unknown parameters: they are assumed to remain constant except for an additive noise (Naets et al., 2015) (equation 5). The discretization of these matrices can later be done in the same way as for the non-augmented model.

$$\mathbf{x}^{aug} = \begin{bmatrix} \mathbf{x} \\ \mathbf{p} \end{bmatrix} \quad (3)$$

$$\mathbf{A}^* = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \mathbf{x}} & \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (4)$$

$$\dot{\mathbf{p}}(t) = \mathbf{0} + \mathbf{w}_p(t) \quad (5)$$

The joint estimation of parameters makes the system non-linear. Once defined in the proper way, this augmented vector can be estimated by means of the EKF or any other nonlinear filter.

Once all the states and parameters of the model are known, we can use the model to obtain some other variables of interest (i.e. a virtual sensor). This is a post-processing step in which the model is evaluated in the estimated set of states, parameters and inputs and the variables of interest are treated as another model output. By means of the estimated state covariance the degree of uncertainty of the virtual sensors can be estimated as well (equation 6).

$$P_{VS} = \frac{\partial f(x, u)}{\partial VS} \cdot P_x \cdot \frac{\partial f(x, u)}{\partial VS} \quad (6)$$

3 State and parameter estimation with Modelica and FMI 2.0 for model exchange

This section explains the implementation of an Extended Kalman Filter that uses physics-based models developed in Modelica and exported by means of the FMI 2.0. The model used for this work is developed in OpenModelica as it provides a powerful model editor that facilitates the development of models and has the advantage of being an open-source tool. The posterior translation of the Modelica models to FMUs is done by means of JModelica.org. This tool provides full functionality to export models for model exchange with the FMI 2.0, including the possibility of requesting directional derivatives. The possibility of

requesting directional derivatives is particularly useful in the development of the EKF as they are more reliable than numerical derivatives.

There are several FMI libraries aimed at programming languages suitable for the development of state estimation algorithms. In this work pyFMI is used, which has the advantage of being open-source. Thus the presented Extended Kalman Filter is written in Python. In addition to pyFMI, which allows the simulation of FMUs, Python offers several other scientific computing packages that aid the development of custom made algorithms and applications (e.g. Numpy, Scipy, Matplotlib).

To make models compatible with the EKF, the inputs of the filter also have to be defined as inputs in the model, while the measurements of the system have to be defined as model outputs. Estimated parameters are simply defined as parameters, and the newly estimated value of the parameter is set in the model at the beginning of each step. In addition, care must be taken when modeling, so that the states of the model agree with the expected ones during the whole estimation.

As explained in section 2.1, the first step of the Kalman Filter requires the prediction of the model $\mathbf{x}_{k+1}^- = f(\mathbf{x}_k^+, \mathbf{u}_k, k)$. To get this prediction the model is initialized with the states and parameters estimated in the previous step and is simulated from the current step to the next one. In addition to setting the new states and parameters, in models with events these have to be updated after setting states and parameters. To reduce the computational time the results are handled in memory.

For the second step of the filter the EKF requires the matrices of the system in state space form, i.e. the system has to be linearized before it can be used with the estimation algorithm. To achieve this the FMI functionality of obtaining the directional derivatives of the system is used. This function is directly implemented in pyFMI and thus obtaining the system matrices is straightforward:

$$A, B, C, D = \text{model.get_state_space_representation}()$$

The same is not true for the directional derivatives of the parameters, required for the estimation of parameters along with states (section 2.3). As the FMI does not provide directional derivatives for model parameters, these derivatives are computed numerically according to the symmetric difference quotation shown in equation 7.

$$\dot{f}(x) = \left. \frac{f(x+h) - f(x-h)}{2h} \right|_{h \rightarrow 0} \quad (7)$$

4 Application for the estimation of forces in guiding systems

The proposed application case is the guiding system of a vertical transportation system. T-shaped guiding rails are used to minimize horizontal motion ensuring travel in a uniform vertical direction (Janovský, 1999). Inappropriate

installation of these guides and their surface roughness are the main causes of vibration in the car frame (Janovský, 1999). These guides are usually composed of several rail segments aligned together. The proper alignment is, however, extremely difficult, and in general out of plane or out of angle misalignment are common. Such deviations increase the contact forces and induce abrupt forces in the car frame at the rail segment joints, reducing the ride quality and efficiency of the system.

The interaction between the car frame and the guiding system is given in four discrete contact points, by means of so-called sliding shoes. These sliding shoes are U-shaped polymeric pads that grab the guide rail's web. The contact forces of the guiding system are applied in these shoes both in x and y axis. Forces in x axis may be in the positive or negative direction, whereas forces in y axis are only directed towards the car-frame. A scaled test bench of a vertical transportation system available at IK4-Ikerlan is used to validate the methodology. This test bench is a useful tool to study the behavior of such system's using sensors not available in real installations. Additionally it allows us to study the effect of defects that we could not put in a real installation. The test bench is a scaled 'rucksack' type rigid car frame, traveling in vertical direction and constrained horizontally by two T-shaped guiding rails (see figure 2). The system has 12 states corresponding to the 6 degrees of freedom of this car frame (x, y, z, roll, pitch and yaw). Without loss of generality, the driving force (T) is assumed to be known and acts as a system input. In the scaled test bench under study this force is measured with a load cell attached to the driving cable (see figure 2). The numbering followed in this paper for the four contact points is shown in figure 2. The current system has a maximum travel length of 1.8 meters, a nominal velocity of $0.4 \frac{m}{s}$, nominal acceleration of $0.3 \frac{m}{s^2}$ and a nominal jerk of $1 \frac{m}{s^3}$.

4.1 System's model

Models available in the vertical transportation literature are mainly focused on the assessment of the vertical dynamic of elevators (Isasa, 2010). However vertical dynamics are affected by the friction forces, which are directly related to the rail forces acting on the horizontal plane. Horizontal and vertical dynamics are therefore coupled and should be assessed as a whole. As a first step this paper studies the possibility of using the horizontal dynamics to assess the condition of the guiding system, opening the way to studying the system as a whole. The Modelica Standard Library is used to model the described system. The inertial properties of the cabin given in table 1 are obtained from its CAD model. The contact stiffness can be obtained from classical structural analysis, assuming the guide as a flexible beam with flexible supports. The actual stiffness will thus be a function of the vertical position of the cabin. However, in order to simplify the estimation we use a constant stiffness for the whole guide.

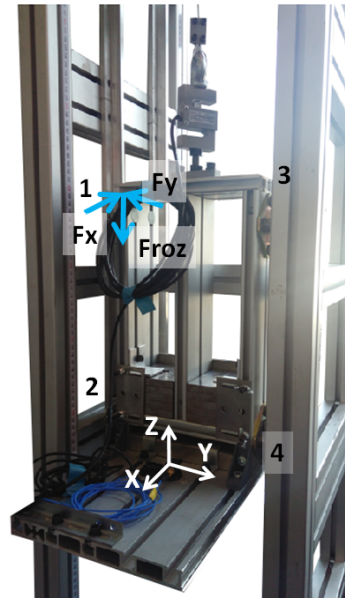


Figure 2. Described system and relevant parameters

Table 1. Model parameters, positions measured from car's floor coordinate system

PARAMETER	DESCRIPTION	UNITS	VALUE
M	carframe's mass	[Kg]	14.287
I_{11}, I_{22}, I_{33}	Carframe's inertias	[Kg.m ²]	0.28, 0.38, 0.20
K_x, K_y	contact stiffness	[N/m]	600000
D_x, D_y	contact damping	[N/(m/s)]	10
$ClearanceY$	sliding shoe clearance	[m]	0.0
r_1^0	position of shoe 1	[m]	(-0.085, -0.124, 0.297)
r_2^0	position of shoe 2	[m]	(-0.085, -0.124, 0.067)
r_3^0	position of shoe 3	[m]	(-0.085, 0.124, 0.297)
r_4^0	position of shoe 4	[m]	(-0.085, 0.124, 0.067)
$r_{c.g}^0$	carframe's C.G	[m]	(-0.0923, 0.0043, 0.08824)
r_T^0	position of cable	[m]	(-0.085, 0.0, 0.435)

4.1.1 Car frame

Due to the low contact forces and the high stiffness of the car frame, the latest can be modeled as a rigid body. The movement of the body is represented in the coordinate system (C.S) attached to the floor of the car frame, as it is the location where the required sensors are installed. The rotation is constraint far away from the Gimbal lock position due to the guide rails and consequently Quaternion representation is not required. Hence rotations of the car frame are represented using Euler angles (α , β and γ).

4.1.2 Guiding rails: contact and friction model

The sliding shoes are the interface between the car frame and the guiding rails. As such, the forces imposed by the guide rail system on the car frame will be applied via the sliding shoes. The sliding shoes grab the guiding rail's web, contacting it in three flanges. Contact in these three flanges at the same time is highly unlikely and commonly only one or two of the flanges of the sliding shoe are in contact with the rail. From figure 3 it can be seen that movement of the shoe in the x direction will always result in a force opposite to the movement. Thus, for modeling purposes, contact in x axis can be assumed to behave as

a spring-damper system. Movement of the sliding shoe in the y axis will depend on the direction of movement. Thus contact in y direction is modeled with a modelica standard ElastoGap model. The direction of the guides web is also taken into account in the actuated prismatic in order to make the model more general.

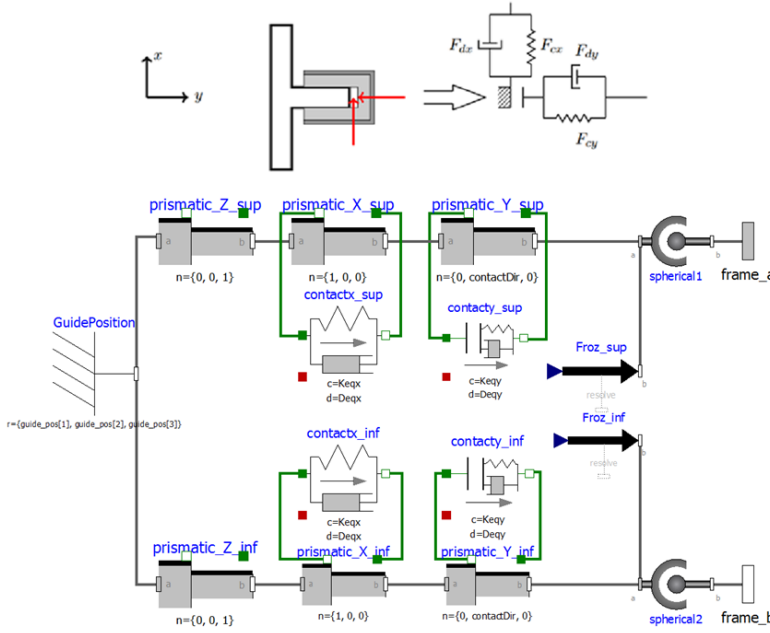


Figure 3. Model of the guiding rail

In this application high displacements between the sliding shoes and the guiding rails are expected. The friction behavior at small displacements is not relevant and the use of dynamic friction models such as Bouc-wen, LuGre or Dahl is not required. A coulomb friction model is used instead. In order to simplify the mathematics of the state estimation algorithm, instead of using an event driven friction element from the Modelica Standard Library the friction is added with a MSL Multibody World-Force model. The value of this force the absolute value of the contact forces times a user given friction coefficient.

```
Froz_sup.force = mu_eq * (abs (
contactx_sup.f) + abs (contacty_sup.f)
) * {0, 0, 1};
```

The advantage of this approach is its simplicity. Avoiding events simplifies greatly the estimation, as the Jacobians of the system change more smoothly. On the other hand, this simplification requires that the direction of the force has to be specified at each simulation, and the direction of the force when the car is stopped is a-priori unknown. Physically the value of the friction coefficient is greater than zero, however we directly include the direction of the force in this parameter. Therefore, a negative value of this parameter just indicates that the direction of the friction force will be negative. With this approach we can find out the direction of the force in the estimation phase.

5 Estimation results

In this section the results of the application of model-based virtual sensors for the evaluation of forces in guiding rails is presented. The estimation approach from section 2.2 is applied to the described system. The measurements used for the EKF are the lateral and vertical accelerations and the vertical position of the car. These measurements are taken with a triaxial piezoelectric accelerometer (lateral acceleration), with a DC response accelerometer (vertical acceleration) and with a draw-wire encoder (cabin position). The accelerometers are located in the coordinate system at the center of the car as depicted in figure 2. Table 2 shows the assumed measurement noise matrix \mathbf{R} (defined in section 2.1).

The measurement of the lateral acceleration provides information of the dynamic change of the contact forces, induced by the roughness and defects of the guiding rails. In addition the model provides information regarding the dynamic behavior of the car and the order of magnitude of the forces. However, the misalignment of the guiding rail results in a DC component of the contact forces which cannot be estimated, as neither the lateral acceleration nor the model have information on that regard. This misalignment affects the vertical dynamics of the system, as friction increases with it. Therefore, we are able to account for this effect within the friction coefficient μ . Additionally, the friction coefficient of each sliding shoe has a significant variability, as it depends on several factors such as, frequency of use of the system, lubrication and temperature. Consequently this parameter is estimated jointly with the states of the system as explained in section 2.3. This parameter contains thus information both on the actual coulomb coefficient and on the misalignment of the guiding system. In contrast to what happens in the actual system, where each sliding shoe has a different friction coefficient, here only one equivalent friction coefficient is assumed for all the contacts (μ_{eq}).

The parameters of the filter's design are shown in table 2. The \mathbf{P} and \mathbf{Q} matrices shown in the table include the covariance of the states and unknown parameter of the system. The last term of these matrices is the covariance of the unknown friction coefficient (μ_{eq}). Table 2 also shows the initial value of the states of the system in the following order: cabin.body1.phi[1], cabin.body1.phi[2], cabin.body1.phi[3], cabin.body1.phi_d[1], cabin.body1.phi_d[2], cabin.body1.phi_d[3], cabin.body1.r_0[1], cabin.body1.r_0[2], cabin.body1.r_0[3], cabin.body1.v_0[1], cabin.body1.v_0[2], cabin.body1.v_0[3] and the initial expected value of the friction coefficient μ_{eq} .

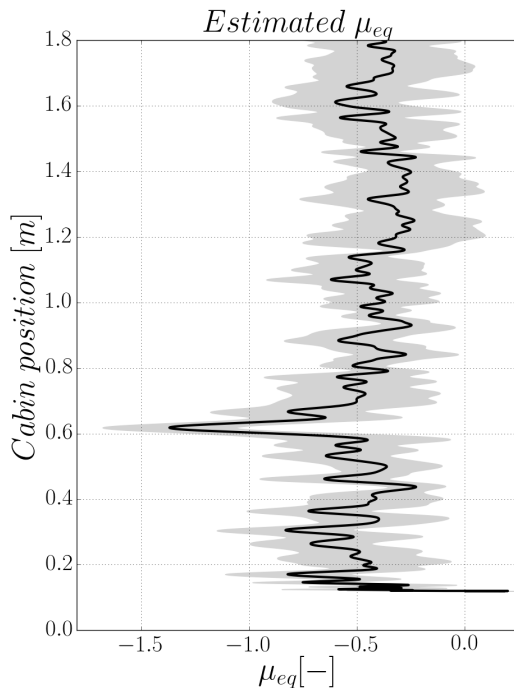
In this application the estimated Virtual sensors and comparison variables have a significant noise. For visualization purposes a smoothing has been performed.

Figure 4 shows the parameter estimated along the position of the cabin. The gray area around the estimated μ_{eq} is the 99.7% (3σ) confidence interval of the value of

Table 2. Design parameters of the filter

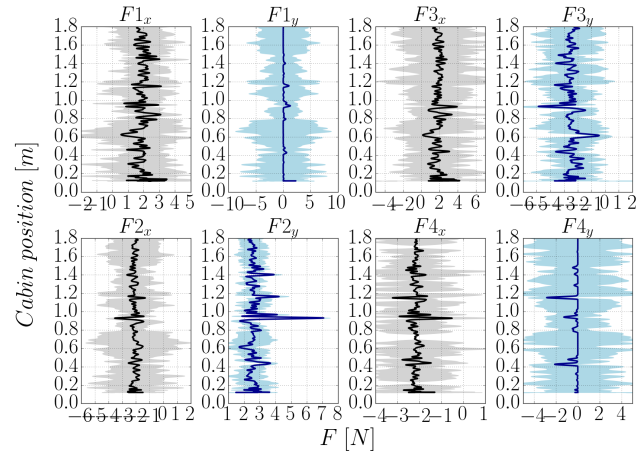
PARAMETER	VALUE
States	α [rad], β [rad], γ [rad], $\dot{\alpha}$ [rad/s], $\dot{\beta}$ [rad/s], $\dot{\gamma}$ [rad/s], x [m], y [m], z [m], \dot{x} [m/s], \dot{y} [m/s], \dot{z} [m/s]
Measurements	z [m], \ddot{x} [m/s ²], \ddot{y} [m/s ²], \ddot{z} [m/s ²]
P	diag([1e-09, 1e-09, 1e-09, 2e-09, 2e-09, 2e-09, 1e-09, 1e-09, 1e-09, 1e-09, 1e-09, 1e-09, 1.5e-06])
Q	diag([0e+00, 0e+00, 0e+00, 1e-03, 1e-03, 1e-03, 0e+00, 0e+00, 0e+00, 1e-05, 1e-05, 1e-05, 5.0])
R	diag([1e-08, 1e-16, 1e-16, 1e-16])
$x_initial$	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.12, 0.0, 0.0, 0.0]
F_s [Hz]	1652
μ_{eq0} [-]	0.2

this parameter. This confidence interval comes from the co-variance estimated for this parameter. As explained in section 4, this parameter includes information regarding the actual friction coefficient of coulomb and regarding the misalignment of the guiding rails. The negative sign of the estimated parameter is not related to the physical meaning of the friction coefficient, but to the direction of the friction force instead. In addition to the friction coefficient

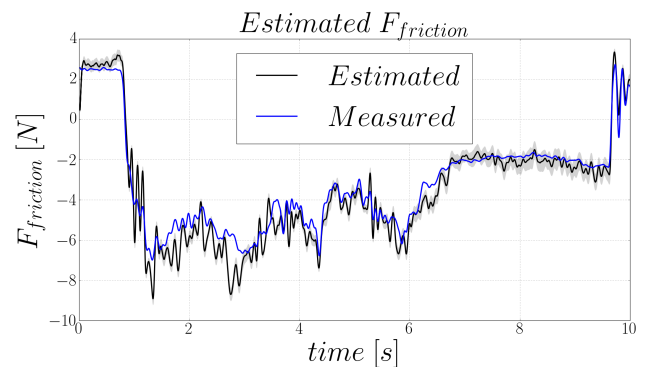

Figure 4. Estimated friction coefficient and the estimated 99.7% confidence interval of the estimation

cient, the contact forces also provide information regarding the condition of the guiding system. More exactly, they provide information regarding sharp differences in the position of the guide rails. Such dynamic changes will

be due to sharp changes of guiding rail segment or imperfections on the rail. These defects induce abrupt forces in the sliding shoes that have a negative impact on the riding quality. Figure 5 shows the estimated contact forces in X and Y directions along with the 99.7% confidence interval (light gray for X direction and light blue for Y direction). As expected, the variance of the estimated forces is relatively high. This is mainly because the model and the measurements do not provide information regarding the DC component of this forces.


Figure 5. Estimated contact forces. In black the forces in X direction and 99.7% confidence interval, in blue forces in Y direction and 99.7% confidence interval.

Finally figure 6 shows the comparison between the estimated friction force and the measured one. The estimated friction force is computed from the sum of the friction forces in each sliding shoe as explained in section 4.1.2. The measured one is the direct subtraction of the cable's tension and cabin acceleration.


Figure 6. Comparison of estimated and measured friction force

The guiding system is not the most common source of failure but it is one of the most critical systems. Failure in guiding rails leads to large down times of the whole system and it is difficult to evaluate its condition. It is complicated thus to assess both the alignment and the smoothness

of the guides, as required to ensure comfort and energy efficiency. Different condition monitoring alternatives are suggested in the literature for guiding rails. In general vibration data processing is used to assess the condition of the guides (Wan et al., 2015). However these methods require well trained data which in may be difficult to obtain. Model-based virtual sensors, on the other hand, provide a suitable approach to monitor the condition of system using off-the-shelf sensors and without training data. The trade off is that an accurate model of the system is required. Nevertheless it was shown that Modelica and the FMI simplify the development of model-based virtual sensors.

The presented estimations provide a useful indicator of the condition of the guiding system. Changes in the friction coefficient indicate misalignment and abrupt changes in the forces indicate local damages in the guides. For instance the estimation from figure 4 shows a significant deviation in the friction coefficient when the cabin is at 0.6m. This deviation indicates thus a misalignment of the guides at that position. Additionally, the abrupt change of the contact forces at 0.9m indicates a local defect in the web of the rail. However, further work is required to develop a methodology to set a threshold for the value of these variables. Once the threshold is defined we can use this to assess the condition of the system, aiding the alignment of the guides and finding early damage in the rails.

6 Conclusions

An EKF with parameter identification capabilities has been developed in Python using the package pyFMI and models exported with the FMI 2.0 for model exchange. Modelica and FMI are very useful to cope with the complexities arising from the use of Model-based virtual sensor with complex systems. The combination of these tools reduces modeling effort and simplifies the implementation of the virtual sensor. As an example of the efficiency of this combination the estimation of forces in a vertical transportation system scaled test bench is presented. The EKF is used to simultaneously estimate states and parameters in a scaled vertical transportation system test bench. Additionally the forces acting on the guiding system are estimated. This estimations provide a mean to assess the condition of the guiding system. This approach opens the way to condition based maintenance strategies for guiding systems. Such maintenance schemes can improve riding quality, safety and efficiency of vertical transportation systems, fulfilling thus the requirements of modern smart systems. Future steps in the investigation include:

- Assessment of the estimated variables and parameters: a theoretical optimal value of the friction coefficient and of the contact forces should be used to set a threshold that aids assessing the condition of the guides.
- In this work the tension in the cable has been used as input of the system. Even though off-the-shelf sen-

sors exist for that purpose it would be better to use just sensors available in the system or easier to use, such as the input of the controller and the currents in the machine. To achieve this the model of the system has to be extended to include not only the cabin of the test bench, but the controller, the electric machine, the pulley, the cable and the counterweight as well. The model will be extended to include all the parts of the system. The electric machine and control systems will be included in the estimation.

- As the system grows in complexity, the use of other state estimation algorithms such as the Unscented Kalman Filter or the Moving Horizon Estimator will be explored. Finally the estimation will be used to monitor the condition of the system.

7 Acknowledgments

The authors gratefully acknowledge the European Commission for its support of the Marie-Sklodowska Curie program through the ITN ANTARES project (GA 606817) and the support from the KU Leuven research fund.

References

- K. Bizon, G. Continillo, S. Lombardi, E. Mancaruso, and B.M. Vaglieco. Ann-based virtual sensor for on-line prediction of in-cylinder pressure in a diesel engine. In *24th European Symposium on Computer Aided Process Engineering*, volume 33 of *Computer Aided Chemical Engineering*, pages 763 – 768. Elsevier, 2014. doi:<http://dx.doi.org/10.1016/B978-0-444-63456-6.50128-9>.
- M. Bonvini, M. Wetter, and M. Sohn. An fmi-based framework for state and parameter estimation. In *Proceedings of the 10th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, number 096, pages 647–656. Linköping University Electronic Press, 2014.
- J. Brembeck, M. Otter, and D. Zimmer. Nonlinear observers based on the functional mockup interface with applications to electric vehicles. In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical University; Dresden; Germany*, number 63, pages 474–483. Linköping University Electronic Press, 2011.
- J. Brembeck, A. Pfeiffer, M. Fleps-Dezasse, M. Otter, K. Wernersson, and H. Elmqvist. Nonlinear state estimation with an extended fmi 2.0 co-simulation interface. In *Proceedings of the 10th International Modelica Conference-Lund, Sweden-Mar 10-12, 2014*, volume 96, pages 53–62. Linköping University Electronic Press, 2014.
- E. Esteban, O. Salgado, A. Iturrospe, and I. Isasa. Model-based approach for elevator performance estimation. *Mechanical Systems and Signal Processing*, 68-69:125 – 137, 2016. ISSN 0888-3270. doi:<http://dx.doi.org/10.1016/j.ymssp.2015.07.005>. URL <http://www.sciencedirect.com/science/article/pii/S0888327015003246>.

- JCB Gonzaga, L.A.C. Meleiro, C Kiang, and R. Maciel Filho. Ann-based soft-sensor for real-time process monitoring and control of an industrial polymerization process. *Computers & Chemical Engineering*, 33(1):43–49, 2009.
- I. Isasa. *Model validation applied to locally nonlinear lift structures*. PhD thesis, Mondragon Unibertsitatea, 2010.
- R. Isermann. Model-based fault-detection and diagnosis status and applications. *Annual Reviews in Control*, 29(1):71 – 85, 2005. ISSN 1367-5788. doi:<http://dx.doi.org/10.1016/j.arcontrol.2004.12.002>. URL <http://www.sciencedirect.com/science/article/pii/S1367578805000052>.
- L. Janovský. *Elevator mechanical design*. Elevator World Inc, 1999.
- P. Kadlec, R. Grbić, and B. Gabrys. Review of adaptation mechanisms for data-driven soft sensors. *Computers & chemical engineering*, 35(1):1–24, 2011.
- Xueqin Liu, Uwe Kruger, Tim Littler, Lei Xie, and Shuqing Wang. Moving window kernel pca for adaptive monitoring of nonlinear processes. *Chemometrics and intelligent laboratory systems*, 96(2):132–143, 2009.
- F. Naets, J. Croes, and W. Desmet. An online coupled state/input/parameter estimation approach for structural dynamics. *Computer Methods in Applied Mechanics and Engineering*, 283:1167 – 1188, 2015. ISSN 0045-7825. doi:<http://dx.doi.org/10.1016/j.cma.2014.08.010>. URL <http://www.sciencedirect.com/science/article/pii/S0045782514002795>.
- P. Samara, J. Sakellariou, G. Fouskitakis, J. Hios, and S. Fassois. Aircraft virtual sensor design via a time-dependent functional pooling narx methodology. *Aerospace Science and Technology*, 29(1):114–124, 2013.
- D. Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.
- Z. Wan, S. Yi, K. Li, R. Tao, M. Gou, X. Li, and S. Guo. Diagnosis of elevator faults with ls-svm based on optimization by k-cv. *Journal of Electrical and Computer Engineering*, 2015: 70, 2015.
- Y. Zhang, Z. Zhao, T. Lu, L. Yuan, W. Xu, and J. Zhu. A comparative study of luenberger observer, sliding mode observer and extended kalman filter for sensorless vector control of induction motor drives. In *2009 IEEE Energy Conversion Congress and Exposition*, pages 2466–2473. IEEE, 2009.

Dymola-JADE Co-Simulation for Agent-Based Control in Office Spaces

Ana Constantin¹ Artur Löwen² Ferdinanda Ponci² Kristian Huchtemann¹ Dirk Müller¹

¹Institute for Energy Efficient Buildings and Indoor Climate, RWTH Aachen University, Germany, {aconstantin, khuchtemann, dmueller}@eonerc.rwth-aachen.de

²Institute for Automation of Complex Power Systems, RWTH Aachen University, Germany, {aloewen, fponci}@eonerc.rwth-aachen.de

Abstract

This paper presents an application of coupling Modelica under Dymola and JADE to test novel agent-based control for office spaces. The office space with a coupled energy system and weather boundary conditions are modeled in Dymola. The agent platform is programmed in JADE, where the agents communicate with each other to control the technical equipment used to deliver thermal energy to the room. Heating experiments, run for a one room scenario, using a radiator, show better system reaction to the comfort desires of the user compared to a control with a thermostatic valve, while having similar energy consumption. While the agents run in real time, the simulation in Dymola runs faster. We focus on the particularities of the connection for co-simulation to insure smooth transferability of the experiments from simulation to a field test, where the energy system as well as the agent platform would be running in real time.

Keywords: *agent-based control, JADE, co-simulation*

1 Introduction

The energy saving potential of buildings is estimated to be around 30% (IEA, 2015b) with the proper policies and technologies compared to continuing the current business-as-usual scenario, which accounts for over 30% of total final energy consumption for all sectors of the economy (IEA, 2015a). One of these novel technologies being currently researched are multi-agent systems (MAS), used for building energy and comfort management (BECM) (Shaikh et al., 2014).

An agent can be defined as an entity that perceives its environment through sensors and according to a goal function acts upon its environment through actuators (Russell and Norvig, 2003). For applications of indoor climatization agents of this type could be embedded on energy generation, distribution or delivery equipment (e.g. heat pump, circulating pump or valve on a heat exchanger). Through negotiation, for example over the cost of energy supplied to the room, the agents can strive to find optimal solutions for the trade-off between thermal energy and indoor comfort. As all novel technologies MAS applications are first tested on a simulation level. According to

the state of the art (Labeodan et al., 2015) we chose JADE for the programming of the agents. JADE is a Java based software framework which is compliant with the specifications of the Foundation for Intelligent Physical Agents (FIPA). The simulation setups containing the models for the building, technical equipment as well as the boundary conditions for the weather are done using Modelica under the simulation environment Dymola. Dymola and JADE are connected in a Co-Simulation by using a TCP/IP interface.

We present in this paper the methodology for building our simulation tests with a focus on the particularities of setting up the co-simulation. We describe the use case we are using, information on how the agent platform works and how the models are built and the communication with the models takes place. Afterwards, exemplary results for a one room scenario are presented and discussed. The paper closes with a conclusions section.

2 Method

The first steps in our work for developing MAS for BECM was to develop an ontology and by focusing on the application on non-residential buildings adding a data model. However, this is not within the scope of the paper and further details will be skipped. The data models for the energy equipment were built based on the data generally available in manufacturer data sheets, to insure a wide applicability of the methodology to devices of the same type but by different manufacturers. To develop a coherent control strategy we used a use case base approach. We developed use cases for one and two room scenarios, with different type of users or for plug-in of a new component. For exemplary purposes we focus on a one room use case in this paper.

2.1 Use Case

We consider an office room, with one user. The user is also an owner of the office, so his / her comfort desires will be taken into consideration. The room is supplied with thermal energy by a system consisting of a pump and a heat pump, which deliver energy to a radiator. The energy can be increased either by increasing the supply temperature of the heating fluid (action of the heat pump) or by in-

creasing the volume flow through the radiator (action of the pump). These two actions do influence one another. In this paper we only focus on heating the room. Figure 1 presents the agents involved in the one room use case.

The following agents act in this use case:

- the personal comfort agent (ComfortAgent): receives from the person working in the room the function for preferred room temperature as a function of outside air temperature, as well as the productivity function as a function of the room air temperature. The agent calculates the set temperature for maximum productivity and the comfort curve, which gives the costs of comfort increases and the savings of comfort decreases.
- the heat pump agent (EnergySupplyHeatPump Agent): has knowledge of the current operation point of the heat pump and the corresponding energy costs.
- the pump agent (EnergySupplyPumpAgent): knows the current operation point of the circulating pump and the corresponding energy costs.
- the room agent (RoomAgent): monitors the current room air temperature against the desired temperature and starts a negotiation between comfort and energy supply agents if the room temperature has to change
- sensor and actuator agents: temperature sensors (room air, outside air, supply and return of the medium through the radiator) and the actuator on the valve of the radiator.

We present the steps of the use case when using the control strategy "relaxation of comfort costs", meaning the costs for the user's productivity loss because of decreased comfort are taken into account against the costs for energy supply to the room. While the energy costs are calculated according to the price of the extra energy supplied to the room, the comfort costs are calculated according to (Sepaenen et al., 2006).

1. The room agent monitors the room temperature, received from the sensor, against the desired room temperature, received from the comfort agent (1). If the room air temperature lies outside a tolerance interval around the desired temperature a call for proposal (CFP) is sent to the relevant energy supply agents that are connected to the supply circuit of the room and might produce this energy: heat pump and pump (2). Additionally a request for the comfort costs is sent to the comfort agent (1*).
2. The energy supply agents send their costs and the amount of energy they can provide for the room (3).
3. The room agent compares the costs for supplying the energy against the comfort costs of the user. The option with the lowest costs is executed: either energy

is supplied (if the comfort costs are lower than the costs for supplying the energy), or not.

4. If energy has to be provided to the room, the room agent sends a request to the energy supply agent with the lowest costs (4): this might be just one agent or both agents, if one agent cannot supply all the energy on its own. If the pump is chosen, it will send a request to the valve (4*). Once the action is executed, the valve sends a confirmation back to the pump (4*).
5. The energy supply agents send messages to the room agent on how the command was executed (5): success or failure.
6. The room agent informs the comfort agent on how the action for improving the room conditions was executed (6).

2.2 Agent platform

We used the Java based programming language JADE (Java Agent DEvelopment Framework) to develop the agents. The decision for JADE was made based on its implementation of the FIPA specifications. The base component of a typical agent platform is the agent, with each agent having a unique name. Agents execute tasks and interact with each other through the exchange of messages. They are located on a platform, which provides them with basic services such as message delivery.

In this application we decided to attach agents to every system component: technical equipment, actuators and sensors. In this way we diverged from the given definition for an agent, as a sensor in itself cannot act upon its environment. However, the state of the art in agent systems for building energy management (see for example (Dibley et al., 2012)) have agents responsible for sensors (reading and sending values). For a first implementation we decided on having agents for each sensor to enable a closer look at the communication flow between such components.

The agents communicate with each other and with the attached devices. The agents have different roles to play and according to their goals the corresponding actions can be grouped into so called behaviors. Exemplary we present here the behaviors of an agent attached to a circulating pump. The *EnergySupplyAgentPump* connects to a pump device to receive information on the device's current state, answers to a CFP, receives and executes requests to change the operation point of the device:

- *HandleSensorDataSubscription*: receives and handles the notifications from the sensor agents the pump agent subscribed to: supply and return temperature sensors for the rooms on the same supply circuit as the pump.
- *EnergySupplyContractNetResponder*: handles incoming CFPs, by sending a proposal, to handle

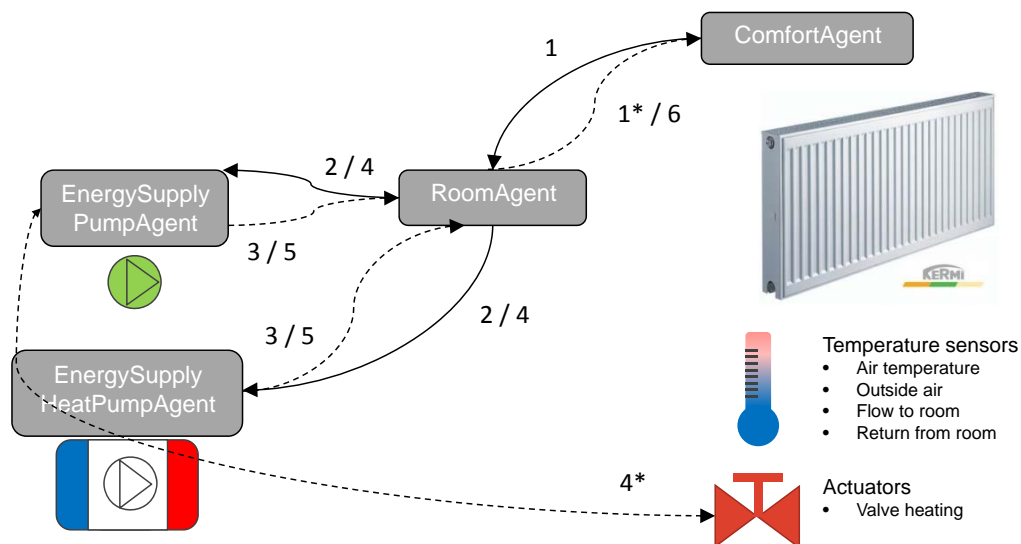


Figure 1. Agents involved in a one room use case.

accepted proposals through *HandleAcceptProposal* and to handle rejected proposals.

- *HandleAcceptProposal*: handles the acceptance of the proposal, that was requested and accepted by the room agent. It starts the *InitiateRequestToValveActuator* behavior to send a request to the valve agent, which communicates with the valve device to control the volume flow, if the pump does not allow to set the volume flow directly.
- *InitiateRequestToValveActuator*: this behavior is initiated by *HandleAcceptProposal* to send a request to the valve agent as previously detailed.

It is important to understand, that as long as the behaviors are not connected to each other, meaning a behavior is started from another behavior, that they are running in parallel. An agent attached to a pump can at the same time process messages received from a temperature sensor as well as a CFP received from a room agent.

2.3 Modeling

The simulation models were developed using blocks from the Modelica Standard Library, the open-access library of the Institute for Energy Efficient Buildings and Indoor Climate (EBC), AixLib, and an institute internal Modelica library for components for energy systems.

The following models were used and/or developed:

- Technical equipment: heat pump, boiler, circulating pump, radiator
- Building: room with walls, windows and doors
- Internal gains from persons present in the rooms
- Boundary conditions, mainly weather

The models available in the EBC libraries for the technical equipment were until now a heterogeneous aggregation of all the elements needed to describe a real component, including the physical phenomenon in the device along with control dedicated blocks. We decided to split the models from the controllable pieces of technical equipment into three sub-models (see Figure 2):

- **physical model**: describes the physical component of the device on which the energy transformation is taking place, along with the energy sources which support this transformation. For example for a boiler model the physical model includes a water volume, a hydraulic resistance and an energy source for the gas flame.
- **internal control**: describes the way in which energy flows are directed at the physical components. This control mechanisms are a proprietary part of the device, and the user has no influence on them. In the boiler example this is a PI controller which controls the thermal output of the gas burner in order to achieve a given supply temperature. The efficiency of the energy transformation in burning the gas is also taken into account.
- **external control**: describes the control strategy that the user sets for the device. Continuing with the boiler example, this is the algorithm for choosing a set temperature for the volume flow: it might be constant, it might be given by a heating curve (state of the art), it might also include a night and a day mode.

The arrows in Figure 2 describe the flow of information between the three blocks, comprising set and measured values.

We are partial to this modeling approach, as it gives a good overview of the model and what it can do, and it

leads to the re-usability of the sub-models. By using the *replaceable model* functionality in Modelica the external control block can be changed between simulations, so all other elements stay the same in the model. As such the probability of making a mistake when setting up identical setups with different control strategies is very low. The external control as such can be programmed in Modelica or in another software. In the latter case the external control block contains the Dymola implementation of the interface between the two softwares.

2.4 Simulation setup

The office space has a floor area of 19 m^2 , with a height of 3 m and a large window area of 8 m^2 . The construction is well insulated: $U_{\text{wall}} = 0.2 \frac{\text{W}}{\text{m}^2 \cdot \text{K}}$ and $U_{\text{window}} = 1.2 \frac{\text{W}}{\text{m}^2 \cdot \text{K}}$. The office has one outer wall and the boundary conditions at the other five inner walls are set as adiabatic. The energy delivery system to the room is a radiator, equipped with a valve for volume flow control. The energy distribution system consists of a circulating pump and the energy generation system is a heat pump.

The reference system, for comparison evaluation of the control strategy is build the same way, with the exception that the control of energy flow to the room is done by a thermostatic valve.

For the weather data we used the test reference year, TRY 05, for the area of Aachen provided by the German meteorological service (DWD).

3 Co-Simulation

3.1 TCP/IP Communication

We decided on using a TCP/IP communication between Dymola and JADE, as routines for TCP/IP communication have been developed in previous projects for both Dymola and JADE. Furthermore in the planned field test one communication possibility between agents and devices is via TCP/IP, which means that only few modifications have to be carried out for moving the testing from simulation to a

field test.

Details on a first version of the routines for the TCP/IP communication can be found in (Schneider et al., 2015). As set up parameters the routines only need an IP-address and a port.

Dymola does not build a socket on its own, but can communicate with an open socket. The socket is build in JADE. We extended the routines by allowing for several different TCP/IP channels to be built and used in one simulation. Additionally, we changed the state of the receiving routine for a socket from blocking to non-blocking. In this way messages can be sent to Dymola only when a command has to be executed. The rest of the time Dymola listens on the socket for a given amount of time and if no message is received it proceeds with the simulation. This is also important for the transferability of the testing from a simulation to a field test, where commands are send only when needed.

We wanted to have a standardized content for the messages being sent between Dymola and JADE. For this we decided on using the Java Script Object Notation (JSON) standard (ECMA, 2013). Figure 3 shows the structure of an object in JSON. The extra overhead to the communication by the fact that the messages are human readable is not an issue to our applications, were a handful of values are exchanged every couple of minutes. A further advantage of the standard is that it offered us an easy way of integrating the data model, which is also built using tuples of label, type and value.

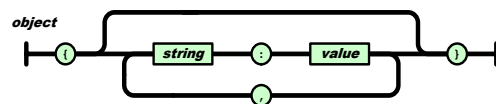


Figure 3. Object structure according to JSON standard (ECMA, 2013)

For a circulating pump the message sent by the device to the JADE agent looks like this:

```
{deviceId: pump1, building: Build1, supplyCircuitId: SC1, room: None, vFlow: 5.3239e-005, head: 0.49068, isOn: 1, mode: 1, health: 1, control: 1, end: false}
```

The *building*, *supplyCircuitId* and *room* parameters describe the position of the device in the energy system.

The values for volume flow (*vFlow*) and *head* describe the current operation point of the pump. The values are in SI units, as well as the algorithms inside the agents are in SI units to make the flow of information between simulation and agent platform as lean as possible. We firmly believe that when communication between two softwares takes place, all measured or calculated values have to be provided in SI for transparency. Changes from SI to other units should be done in the software themselves if needed.

The values for *isOn*, *mode*, *health* and *control* describe the status and settings of the pump, according to the data model.

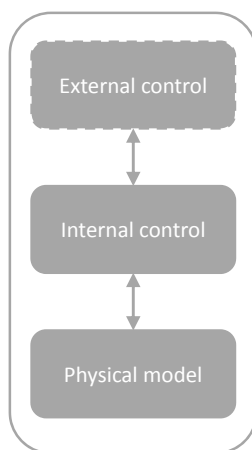


Figure 2. Modeling approach for a piece of controllable technical equipment

The *end* parameter is used to signal the end of the simulation and triggers a shutdown of the agent platform in JADE.

3.2 Real time vs simulation time

The agent platform runs in real time, while the simulation can run in real time or the time it normally takes to run the simulation which we call "simulation time". Depending on the complexity of the model the simulation time can be quite different, from a few seconds to a few minutes for a whole simulation day. The advantage of having Dymola run in "simulation time" is that more experiments can be done in a shorter period of time.

Dymola communicates with JADE at discrete time intervals, set in the discrete blocks by the parameter *sample rate*. The challenge when running a co-simulation lies matching the dynamics of the controller with the ones of the controlled system. For us it meant identifying an adequate length of the time interval such that the response time from JADE to Dymola, for example in the case of a CFP which leads to a command to Dymola, should be reasonable. We define reasonable as 5 minutes time in the simulation, from the moment an uncomfortable temperature has been identified and the moment the an agent has received a command to act in this case.

The communication rhythm, i.e. the sample rate, between Dymola and JADE influences the simulation-time two fold:

- the shorter the time interval for communicating, the longer the simulation, as sampling leads to events and the TCP/IP interface is programmed using algorithms which both slow down the simulation
- the commands from JADE can arrive more quickly, at the very next communication step, which can sometimes lead to an over-reaction in the system. For example if the temperature didn't have time to be influenced by the latest agent actions and already the room agent is initiating a new CFP

Our method of finding an adequate sampling rate is trial-and-error based and has to be executed for each simulation model in part, as increased model complexity leads to increased CPU time. It requires a series of simulations using the same model and different values for the sampling rate. For each simulation the time in the simulation between registering an uncomfortable temperature and a subsequent agent action is measured and compared to the reasonable time frame. Additionally when waiting for an action confirmation (meaning set point change) from the simulation, we assume the set point of the device should change between two samples. As such the timeout for an agent waiting for confirmation from the device should be at least 2.5 the sample time translated in simulation time.

4 Results and discussion

We present exemplary results for a one room use case, done using the described simulation setup with a room (R1) where the agent based control is implemented. As a reference case we use a second room (R2) where a thermostatic valve controls the temperature in the room, considered to be state-of-the art for room temperature control. Both rooms are connected to the same supply circuit thus a change in the supply temperature of the heat pump affects both rooms. Changes in the opening of the valves only affect the room in question. The users are assumed to be present over the whole duration of the simulation. As such the room agents are continuously monitoring the room conditions against the preferences expressed by the comfort agents.

We present a simulation for a heating scenario, done over the first three days of January.

Expected results:

- we expect the valve to open when the room temperature needs to increase, and the supply temperature from the heat pump to drop when the room temperature needs to decrease
- we expect the valve in room R1 to open almost fully during the test, as the costs of the pump are lower than the costs of the heat pump
- we expect the value of the supply temperature to decrease under the value for a control strategy based on a heating curve (in this case around 45°C), as the volume supply through the pump increases

Achieved results

Figure 4 presents the air temperature and valve opening for each room, along with the supply temperature of the heat pump. We observe that the results are as expected. The supply temperature of the heat pump increases when the room temperature needs to increase, only when the valve is fully opened and as such no further action is possible from the valve.

The set temperature of 20°C is given for both rooms. The evolutions of the room air temperatures are similar. However the valve openings are quite different, on account of the different control strategies. The thermostatic valve has a proportional term of 1 K, which means the valve fully closes once the room air temperature is 1 K above the set temperature (comfort requirement R2). If the room air temperature is lower than the set temperature the valve can, depending on the type of valve, open fully only when the difference between set and current temperatures is around 5 K, which explains why the valve doesn't open more and the temperature drops below the set temperature. The MAS holds the room temperature in an interval of ± 0.5 K around the set temperature. This is the comfort preference of the user, as variations of this degree are considered unnoticeable by the user (comfort requirement R1). In the case of the user in R2, the drop

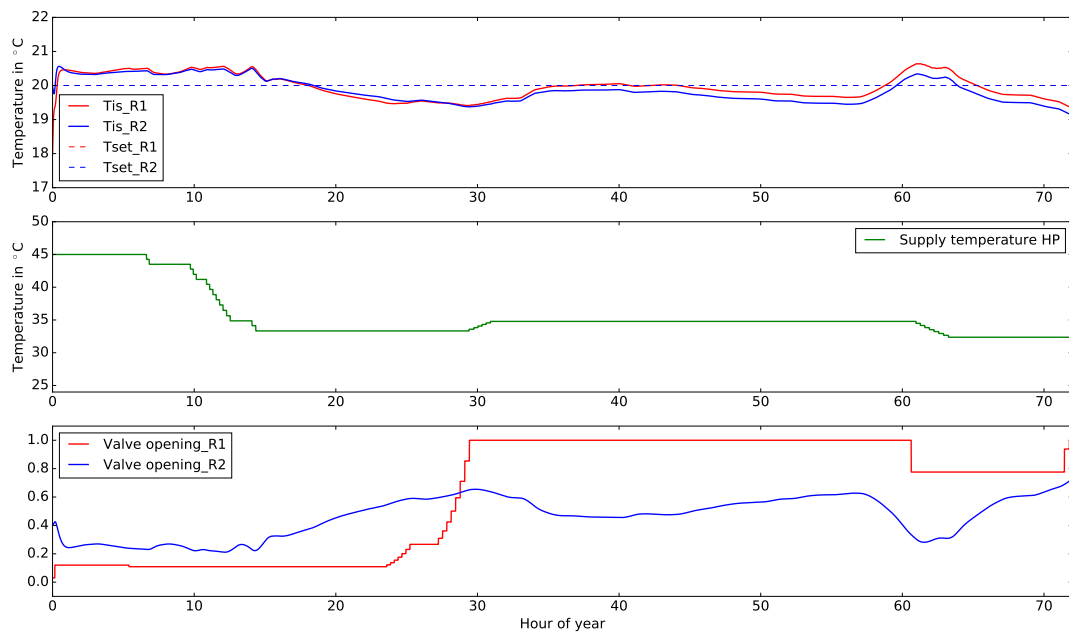


Figure 4. Air temperature, supply temperature of heat pump (HP) and valve opening for room 1 (R1) and room 2 (R2) in a heating experiment

below the set temperature is considered a violation of the comfort requirement.

Key performance indicators (KPI) relating to energy consumption and comfort for the two rooms are presented in table 1. While the energy consumption is 3% higher for room R1 than room R2, the discomfort, as calculated according to the different comfort requirements, is a lot lower, 97% less than for room R2. However using a common comfort criterion for both rooms, like the Predicted Mean Vote (PMV) according to (Fanger, 1970) the comfort levels in both rooms are more similar. The energy delivery to the room is similar while having different valve openings, because in the case of room 2 the temperature difference between supply and return for the radiator is higher. If both rooms had different energy generation systems, the costs for the energy generation would have been higher for room 2.

Table 1. KPI One Room Use Case

Room	KPI	Value	Unit
R1	Energy Consumption	20.2	kWh
R2	Energy Consumption	19.5	kWh
R1	Lost comfort	0.6	K · h
R2	Lost comfort	18	K · h

5 Conclusions

We presented in this paper the realization of a co-simulation between Dymola and JADE, for implementing

an agent based control for an office space. The building, boundary conditions and the technical equipment with the exception of the control strategy are modeled in Dymola. The multi-agent system containing the control strategy for the system is setup in JADE.

The communication between Dymola and JADE is done using a TCP/IP connection. Measured values describing the current state of the simulation, including temperatures and operation points of the technical equipment, are sent to JADE with a fixed sample rate. Commands from the agents in JADE are sent to Dymola over non-blocking sockets at the next available communication step. Care has to be taken when setting up the sample rate and the timeout for action confirmations from the simulation as the agents are running in real-time and the simulation can run faster.

We exemplified our concept with a simulation for a one room scenario, using a second room controlled by a thermostatic valve as a reference case. While the energy consumption and comfort are similar the agent system reacts better to temperature changes and can lead to lower energy costs. Future work will focus on experiments with multiple room as well as a field test.

References

- Michael Dibley, Haijiang Li, Yacine Rezgui, and John Miles. An ontology framework for intelligent sensor-based building monitoring. *Automation in Construction*, 28:1–14, 2012. ISSN 09265805. doi:10.1016/j.autcon.2012.05.018.
- ECMA. Ecma-404: The json data interchange format, 10 2013.

URL <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>.

- P.O. Fanger. *Thermal Comfort*. Danish Technical Press, 1970.
- IEA. Building energy performance metrics: Supporting energy efficiency progress in major economies. Technical report, International Energy Agency, 2015a.
- IEA. Energy technology perspectives 2015. Technical report, International Energy Agency, 2015b.
- T. Labeodan, K. Aduda, G. Boxem, and W. Zeiler. On the application of multi-agent systems in buildings for improved building operations, performance and smart grid interaction a survey. *Renewable and Sustainable Energy Reviews*, 50: 1405–1414, 2015.
- Stuart J. Russell and Peter Norvig. *Artificial intelligence A modern approach*. Prentice Hall/Pearson Education, 2003.
- Georg Ferdinand Schneider, Jens Oppermann, Ana Constantin, Rita Streblov, and Dirk Mueller. Hardware-in-the-loop-simulation of a building energy and control system to investigate circulating pump control using modelica. In *The 11th International Modelica Conference*, 2015.
- Olli Seppanen, William Fisk, and QH Lei. Effect of temperature on task performance in office environment. Technical report, Ernest Orlando Lawrence Berkley National Laboratory, 2006.
- P. H. Shaikh, N. B. M. Nor, P. Nallagownden, I. Elamvazuthi, and T. Ibrahim. A review on optimized control systems for building energy and comfort management of smart sustainable buildings. *Renewable and Sustainable Energy Reviews*, 34:409–429, 2014.

Failure Modes of Tearing and a Novel Robust Approach

Ali Baharev Arnold Neumaier Hermann Schichl

Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
ali.baharev@gmail.com

Abstract

State-of-the-art Modelica implementations may fail in various ways when tearing is turned on: Completely incorrect results are returned without a warning, or the software fails with an obscure error message, or it hangs for several minutes although the problem is solvable in milliseconds without tearing. We give three detailed examples and an in-depth discussion why such failures are inherent in tearing and cannot be fixed within the traditional approach.

Without compromising the advantages of tearing, these issues are resolved for the first time with staircase sampling. This is a non-tearing method capable of robustly finding all well-separated solutions of sparse systems of nonlinear equations without any initial guesses. Its robustness is demonstrated on the steady-state simulation of a particularly challenging distillation column. This column has three solutions, one of which is missed by most methods, including problem-specific tearing methods. All three solutions are found with staircase sampling.

Keywords: *decomposition methods, diakoptics, large-scale systems of equations, numerical instability, sparse matrices, staircase sampling*

1 Introduction

Definitions. **Traditional tearing**, cf. (Elmqvist, 1978; Elmqvist and Otter, 1994; Mattsson et al., 1999; Carpanzano, 2000; Cellier and Kofman, 2006; Täuber et al., 2014), is the representation of a sparse system of nonlinear equations

$$f(x) = 0, \text{ where } f: \mathbb{R}^n \mapsto \mathbb{R}^n, \quad (1)$$

in a permuted form where most of the variables can be computed sequentially once a small auxiliary system has been solved. More specifically, given permutation matrices P and Q such that after the transformation

$$\begin{bmatrix} g \\ h \end{bmatrix} = Pf, \quad \begin{bmatrix} y \\ z \end{bmatrix} = Qx, \quad (2)$$

$g_i(y, z) = 0$ can be rewritten in the equivalent explicit form

$$y_i = \tilde{g}_i(y_{1:i-1}, z) \quad (3)$$

using appropriate symbolic transformations. Here the shorthand $p:q$ is used for the index set $p, p+1, \dots, q$ where

$p \leq q$. Equation (3) implies that the sparsity pattern of the Jacobian of Pf is

$$J = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \text{ where } A \text{ is lower triangular,} \quad (4)$$

J is therefore **bordered lower triangular**. We will use the abbreviation **BLTF** which stands for bordered lower triangular form. We refer to a particular choice of P, Q, g, h, y , and z satisfying equations (3) and (4) as an **ordering**. Given an ordering, the system of equations $f(x) = 0$ can be written as

$$\begin{aligned} g(y, z) &= 0 \\ h(y, z) &= 0. \end{aligned} \quad (5)$$

The requirement (3) that $g_i(y, z) = 0$ can be made explicit in y_i essentially means that we can obtain y from z by a nonlinear triangular solve. Substituting the result $y = \bar{g}(z)$ into h yields $h(\bar{g}(z), z) = 0$ or

$$r(z) = 0. \quad (6)$$

That is, the original nonlinear system (1) is reduced to the (usually much) smaller system $r(z) = 0$. A commonly used objective is to find an ordering that minimizes the **border width** $d := \dim z$ of J . For a given z , we call the value of $r(z)$ the residual vector or simply the **residual**.

Advanced tearing methods. There are other, more sophisticated variants of tearing, summarized in Table 1. These try to reduce the size of the final system (6) by relaxing the requirements of (3) (by allowing implicit equations for example) and/or allowing A in (4) to have a form other than lower triangular. These enhancements share that the computation of y for a given z only involves fast and numerically stable algorithms such as solving implicit univariate equations or small systems of equations. Another recent approach tries to balance between minimizing the border width and preserving the sparsity during the elimination (Magnusson and Åkesson, 2017). The reader is referred to (Baharev et al., 2017a) for an in-depth discussion of the variations on tearing. To keep the examples short and simple, we only discuss the failure modes of traditional tearing in the present paper.

Importance: initializing and solving DAE systems. The problem of solving nonlinear systems of equations arises in the daily engineering practice, e.g., when consistent initial values for differential algebraic equation (DAE) systems are sought (Pantelides, 1988; Unger et al., 1995),

Table 1. The sparsity pattern of the Jacobian in the different variants of tearing, classified by the largest subproblem size and the level of subproblem nesting. The present paper discusses the failure modes of traditional tearing only.

Largest subproblem	Maximum level of subproblem nesting	
	1	ℓ
univariate equations only	bordered lower triangular (traditional tearing)	—
$k \times k$ systems of equations	bordered block lower triangular (tearing with diagonal blocks)	nested bordered block lower triangular (hierarchical tearing)

or when solving steady-state models of technical systems. A steady-state solution can be used as a consistent initial set of the DAE system (Kröner et al., 1997). Tearing usually also helps to speed up the solution process of DAE systems thanks to the reduced problem size (Elmqvist, 1978; Elmqvist and Otter, 1994; Mattsson et al., 1999; Carpanzano, 2000; Cellier and Kofman, 2006).

Even though mature equation-based component-oriented modeling environments are available, e.g., Modelica (Mattsson et al., 1998; Tiller, 2001; Fritzson, 2004) for multi-domain modeling of heterogeneous complex technical systems, and gPROMS, ASCEND (Piela et al., 1991) and EMSO (de P. Soares and Secchi, 2003) for chemical process modeling, simulation and optimization, etc., the steady-state initialization is still not satisfactorily resolved in the general case. Often, steady-state initialization failures can only be resolved in very cumbersome ways, requiring user-provided good initial values for the variables (Vieira and Jr, 2001; Bachmann et al., 2007; Sielemann and Schmitz, 2011; Sielemann et al., 2013; Ochel and Bachmann, 2013).

2 Demonstrative examples

Here we show the behavior of the latest release of Dymola (Version 2017 FD01 (32-bit), 2016-10-11) and OpenModelica (v1.11.0 (64-bit); February 6, 2017) on three examples. Examples 1 and 2 demonstrate that applying tearing can lead to completely incorrect results or to initialization failure. However, correct results are obtained for both examples when tearing is turned off. Example 3 is about performance: It shows that tearing can slow down the solution process drastically. Dymola can easily hang for minutes on problems that are otherwise solvable in milliseconds without tearing. The causes are discussed in Section 3. The examples trigger failure only if the tearing is performed according to the specified ordering. The Modelica source files are available in the GitHub repository of the (Online Supplement).

Example 1: The residual is overly sensitive to the changes in the tear variable. We solve the following 20×20 linear system in a Newton step:

$$x_{i-1} + 10x_i + x_{i+1} = 1.2 \quad i = 1:20, \quad (7)$$

where $x_0 := 0.1$ and $x_{21} := 0.1$ to keep the formulas simple. The only tear variable is x_1 ; the residual is given by the last equation ($i = 20$). The exact solution is $x_i = 0.1$ for $i = 1:20$. Both Dymola and OpenModelica return completely incorrect results, for example, $x_{20} = 32.03$ and $x_{20} = 85.82$, respectively, but claim that the simulation was successful.

Example 2: The residual is insensitive to the changes in the tear variable. We solve the following 20×20 linear system in a Newton step:

$$x_{i-1} + x_i + 15x_{i+1} = 17 \quad i = 1:20, \quad (8)$$

where $x_0 := 1$ and $x_{21} := 1$ to keep the formulas simple. The only tear variable is x_1 ; the residual is given by the last equation ($i = 20$). The exact solution is $x_i = 1$ for $i = 1:20$. Dymola fails with an unhelpful error message, and does not return any result. OpenModelica emits some confusing intermediate warnings and reports at the end of the computations that “simulation process finished successfully”. But it returns incorrect results; for example, x_1 still equals the initial guess, as if nothing had happened.

Example 3: Unacceptable border width, leading to very poor performance. We solve the following $N \times N$ linear system in a Newton step:

$$\sum_{i=1}^N x_i = N \quad (9)$$

$$x_i + x_N = 2 \quad i = 1:N-1, \quad (10)$$

and we assume that the only variable that can be eliminated is x_N from equation (9); this can be due to the nonlinearities of the original problem (whose Newton step we see here). All other variables are tear variables, and all other equations are residuals. For $N = 300$, the problem is solved by Dymola in 74 seconds and by OpenModelica in 37 seconds. As we argue in Sec. 3.3, the problem is solvable in milliseconds: For $N = 300$ (the largest dimension permitted in the free trial version we used), the AMPL modeling environment (Fourer et al., 2003) is faster than Dymola and OpenModelica by factors of more than 1200 and 600, respectively. The performance of the Modelica implementations rapidly deteriorates as the problem size increases: For $N = 500$, Dymola hangs for more than 6 minutes, and OpenModelica takes more than 1.5 minutes.

The examples are intentionally chosen to be easy. In challenging real-life examples, like the one in Sec. 5.3, it is hard to identify and understand the reasons of the failures, because different failure modes usually occur simultaneously and interact with each other. However, the simplicity of the present examples allows us to gain (in Section 3) important insights into the reasons *why* tearing fails or causes very poor performance. The examples were chosen to demonstrate the reasons in isolation, one at a time. This is also the reason why we picked linear examples; they should be regarded as the linear system solved in a Newton step.

3 In-depth discussion of the examples

Pathological input problems are ignored throughout this paper, for example when the system (1) has conflicting equations and as a consequence it is infeasible, when (1) is singular, when it is poorly scaled, or when the problem has a huge number of solutions, etc. While these edge cases are interesting and important, a non-tearing approach can fail in these cases too, and therefore such failures are not specific to tearing. Throughout this paper we only focus on those failure modes that are specific to tearing: We assume that the input problem (1) is feasible and properly scaled, has at most a small number of real solutions, and that these solutions can be found with an appropriate non-tearing approach using 64-bit floating-point arithmetic. Traditional tearing can fail even if all these assumptions are met.

3.1 Example 1: uncontrollable residual

The residual can become practically uncontrollable because it depends too sensitively on the tear variables. To see this we consider the system

$$x_{i-1} + 10x_i + x_{i+1} = 1.2 \quad \text{for } i = 1:20 \quad (11)$$

where $x_0 := 0.1$ and $x_{21} := 0.1$ to keep the formulas simple. The exact solution is $x_i = 0.1$ for $i = 1:20$. The coefficient matrix of the system (11) is a strictly diagonally dominant tridiagonal matrix (cf. Fig. 1 top), hence solving (11) with Gaussian elimination produces excellent results even without pivoting (Golub and van Loan (1996, Ch 3.4.10)). As it was demonstrated in Section 2, traditional tearing fails on this easy problem.

We order the coefficient matrix of (11) into BLTF with minimal border width by moving x_1 to the border, see on the bottom of Fig. 1. Given an initial guess for x_1 , the formula for the forward substitution along the diagonal is:

$$x_{i+1} = -x_{i-1} - 10x_i + 1.2 \quad \text{for } i = 1:19, \quad (12)$$

and the residual $r := -x_{19} - 10x_{20} + 1.1$ is a univariate function of x_1 , that is, we have to solve the univariate equation $r(x_1) = 0$ for x_1 . Because of the factor 10 in (12), the error in our guess for x_1 is multiplied roughly by a factor of 10 in each step of the elimination according to (12). There are 19 steps in (12), meaning that the error in x_1 will

be magnified roughly by a factor of 10^{19} till we compute the residual. This has catastrophic consequences. There is no machine representable number for x_1 such that after eliminating all the other variables according to (12) r is sufficiently close to zero: The two closest 64-bit floating-point numbers enclosing 0.1 give approximately 85.82 and -101.03 for x_{20} , respectively, due to the roughly 10^{19} factor magnifying the error in x_1 . In other words, (11) is literally unsolvable in 64-bit floating-point arithmetic with traditional tearing, whereas solving it with Gaussian elimination is numerically stable even without pivoting. The failure is not due to a single ill-conditioned elimination step but the sequence of well-conditioned steps becoming ill-conditioned when they are chained together as in (12).

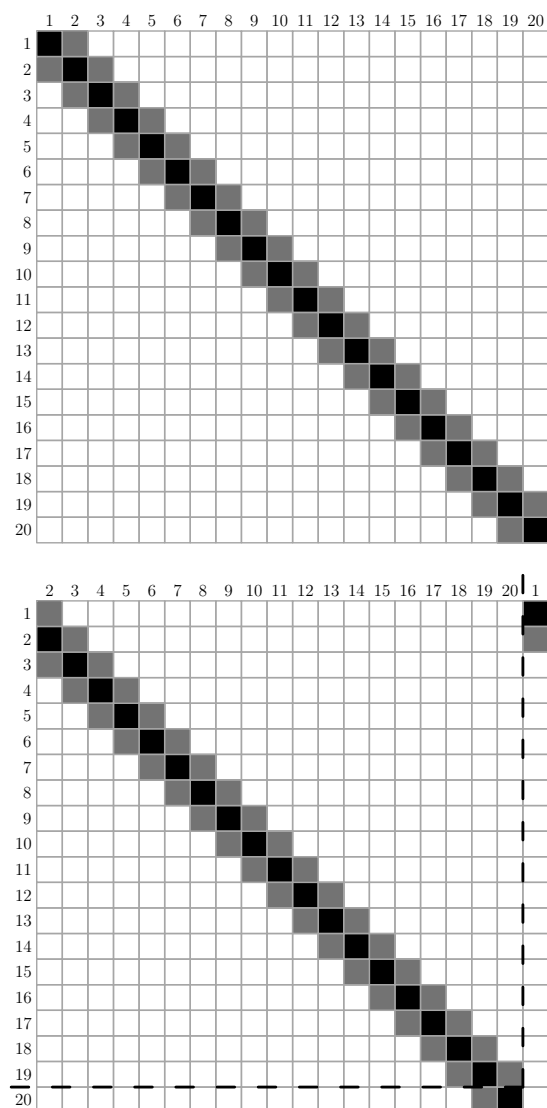


Figure 1. Top: The sparsity pattern of the coefficient matrix of problem (11). Black entries correspond to 10, gray entries to 1. Bottom: The same matrix ordered to bordered lower triangular form. The leading lower triangular submatrix, surrounded by dashed lines, is singular to working precision in 64-bit floating point arithmetic.

A similar behavior in the nonlinear case makes the problem practically unsolvable with iterative solvers, even if the original problem is easy to solve without tearing. Distillation columns are real-life examples where such failures happen, c.f. Doherty et al. (2008).

For those familiar with linear algebra: The condition number estimate of the coefficient matrix of (11) is 1.5 (symmetric, strictly diagonally dominant tridiagonal matrix), whereas the condition number estimate of the leading lower triangular matrix of the BLTF is $9 \cdot 10^{16}$, meaning that it is singular to working precision in 64-bit floating point arithmetic. See also Golub and van Loan (1996, Ch 3.3, and 3.5.4).

3.2 Example 2: insensitive residual

It can also happen that the residual shows practically no response to changes in the tear variables. Such an example is the following:

$$x_{i-1} + x_i + 15x_{i+1} = 17 \quad i = 1:20, \quad (13)$$

where $x_0 := 1$ and $x_{21} := 1$ to keep the formulas simple. It is easy to see that the solution is $x_i = 1$ ($i = 1 : 20$). Solving (13) with a non-tearing approach is not a challenge.

Making x_1 the only tear variable, and moving it to the border makes the resulting BLTF have minimal border width, see Fig. 2. Given an initial guess for x_1 , the formula for the forward substitution along the diagonal is:

$$x_{i+1} = \frac{1}{15}(-x_{i-1} - x_i + 17) \quad \text{for } i = 1:19, \quad (14)$$

and the residual

$$r := -x_{19} - x_{20} + 2 \quad (15)$$

is a univariate function of x_1 , that is, we have to solve the univariate equation

$$r(x_1) = 0 \quad (16)$$

for x_1 . As it can be seen from (14), the error in our estimate for x_1 is divided roughly by a factor of 15 in each step of the recursion, that is, the error attenuates in an exponential rate. As a consequence, we get $r = 0.0000000000$ (with 10 decimals) for both $x_1 = -1$ and $x_1 = 3$. This is unacceptable, since x_1 and many of the eliminated variables are still very far from the solution. The reason of the failure is that the value of $r(x_1)$ provides no information about the desired update of x_1 : The final equation (16) is satisfied even with grossly erroneous x_1 values.

In the nonlinear case, similar issues can lead to failures of the tearing approach. Distillation columns are again real-life examples where such failures happen. In fact, distillation columns are difficult for tearing methods because one part of the column can magnify the error in the tear variables with exponential rate (similarly to (12)), while the remaining part attenuates it with an exponential rate (similarly to (14)). This in turn can trigger two failure

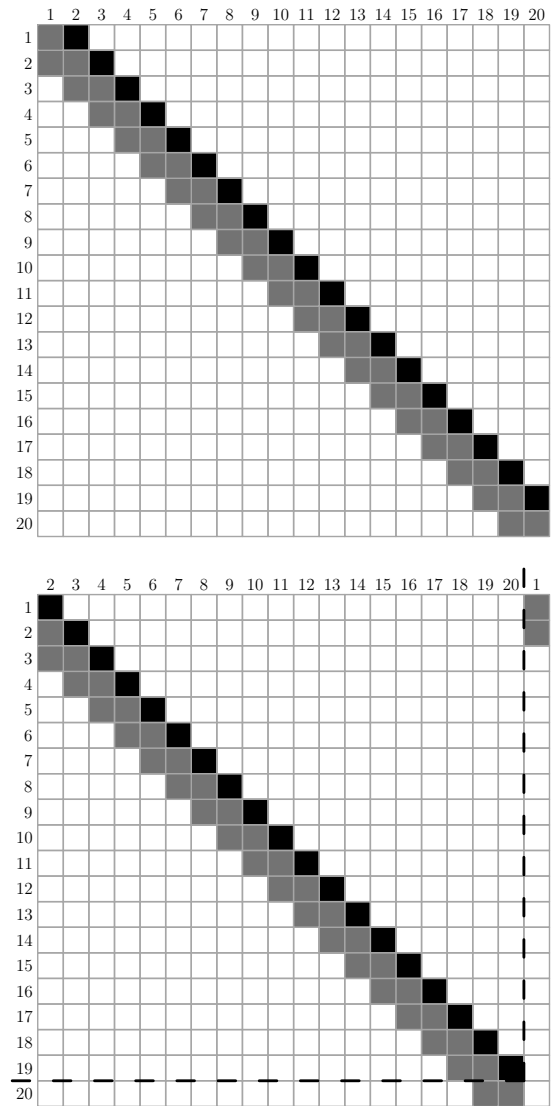


Figure 2. Top: The sparsity pattern of the coefficient matrix of problem (13). Black entries correspond to 15, gray entries to 1. Bottom: The same matrix ordered to bordered lower triangular form.

modes of tearing at the same time: the one described in this section, and the one from the previous section.

As already stated, (13) is not a challenging problem; it is just that traditional tearing fails. For those familiar with linear algebra: Although problem (13) is mildly ill-conditioned, the condition number estimate is $7 \cdot 10^{11}$, one can still get the result with several accurate significant digits in 64-bit floating point arithmetic (Golub and van Loan (1996, Ch 3.3, and 3.5.4)).

3.3 Example 3: unacceptably wide border

Wide border due to tearing incautiously. The primary motivation behind tearing is to speed up the solution process (Dymola User Manual, Ch. 8.8.2, pp. 433-434). However, tearing can significantly hurt performance, especially if it is applied without any caution; Example 3

at equations (9) and (10) in Sec. 2 is just one example of that. Here the problem is that the border width is proportional to the size of the original problem. In case of Dymola, it slows down the model generation and compilation drastically; for $N = 500$, the software hangs for more than 6 minutes. OpenModelica hangs for issues that are most likely independent of tearing, but we failed to track down the exact causes.

In comparison, solving the instance $N = 300$ with AMPL takes only 61 milliseconds, although AMPL adequately performs all relevant tasks, namely:

- (i) reading and parsing the model file written in the AMPL modeling language,
- (ii) instantiating the model,
- (iii) flattening,
- (iv) compiling the C code,
- (v) generating binary opcodes for the virtual AMPL stack machine,
- (vi) launching the external solver and executing the code to compute the solution, and
- (vii) writing back the results for the modeling environment.

AMPL and the Modelica implementations have to go through basically the same steps during the solution process of Example 3; the computational work to be done is essentially the same for each modeling environment.

For those familiar with linear algebra: The problem here is that tearing results in catastrophic fill-in (Duff et al., 1986, Ch. 7). AMPL and the solver it invokes, IPOPT (Wächter and Biegler, 2006) with MA27 from (HSL, 2017), avoid this by not perform tearing, and by using proper sparse data structures and sparse linear algebra. As far as we can tell, the state-of-the-art Modelica implementations seem to perform $O(n^2)$ or more operations, and this can hurt performance already on relatively small problems. See also (Duff et al., 1986, Ch. 5.8) regarding the so-called $O(n^2)$ traps.

Wide border due to trying to create a maximum-weight diagonal. Let A denote the coefficient matrix of (11). The reason why tearing failed in Section 3.1 is that the largest entries of A became off-diagonal after A was ordered to BLTF, and the elimination happened along the diagonal. The straightforward attempt to fix this is to mimic complete pivoting (Golub and van Loan (1996, Ch. 3.4.8)): We order A into BLTF but instead of having a minimal border width, our objective is to have a maximum-weight diagonal on the lower triangular part. Indeed, such approaches were proposed in the past, see for example Westerberg and Edie (1971a,b) and Gupta et al. (1974).

Although creating a maximum-weight diagonal mitigates the issue of uncontrollable residual, it can easily lead to the opposite problem, to the issue of the insensitive residual: The example of Sec. 3.2 has a maximum-weight diagonal and tearing fails on that easy problem. Also, compare Fig. 1 with Fig. 2 where the subdiagonal

became maximum-weight. In short, creating a maximum-weight diagonal can turn one failure mode to another.

However, there is another issue that creating a BLTF with maximum-weight diagonal can also cause: It can produce a BLTF whose border width is proportional to the size of the input matrix, whereas if we minimized the border width, the border width would be a small constant, independent of the problem size. Table 2 and Figure 3 show examples of such disastrous cases. Since the final system (6) is dense, this means that tearing turns (in the course of the elimination) a sparse problem into a dense problem whose size is proportional to the original problem. Such dense problems become intolerably expensive to solve as their size grows.

4 Failing due to a single elimination step

In the previous section we discussed failure modes where a sequence of eliminations led to the failure. In this section we show additional examples where tearing fails due to a single elimination step, because it leads to an undefined operation (Sec. 4.1) or to a floating-point exception (Sec. 4.2), or it is multivalued (Sec. 4.3). We comment on the usual workaround as seen in the state-of-the-art Modelica environments, and propose a novel and better alternative in Sec. 4.4.

4.1 Undefined elimination step

For the sake of demonstration let us assume that in an elimination step in (3) we want to eliminate x_3 from

$$x_1 - x_2 x_3 = 0, \quad (17)$$

so we rearrange (17) as

$$x_3 := \frac{x_1}{x_2}. \quad (18)$$

However, this symbolic transformation is invalid if $x_2 = 0$. If x_2 happens to be 0 during the iteration in the tear variables, eliminating x_3 according to (18) would lead to division by zero, whereas the original equation (17) does not suffer from this issue. It is not only division that is problematic: Another example of this kind of failure is a negative argument to the logarithm function during the iteration in the tear variables (when working over the set of real numbers). In general, arguments outside the domain of the functions involved lead to failure of traditional tearing.

4.2 Floating-point exception in an elimination step

Floating-point exceptions can easily occur in systems involving an exponential. For example, let the tear variables be $x_{41} := 440$ and $x_{43} := 0.0$, and the elimination steps are:

$$\begin{aligned} x_{42} &:= \exp(x_{41} + 273.15) \\ x_{44} &:= x_{42} x_{43}. \end{aligned} \quad (19)$$

Table 2. Unacceptably wide border when the matrix is ordered to BLTF with maximum-weight diagonal on the lower triangular part.

Matrix	Minimal border width	Border width with max-weight diagonal	Pattern in Figure (3)
Tridiagonal	1	$\frac{1}{2}n$	top row
Pentadiagonal	2	$\frac{2}{3}n$	(not shown)
Arrowhead	1	$n - 1$	bottom row

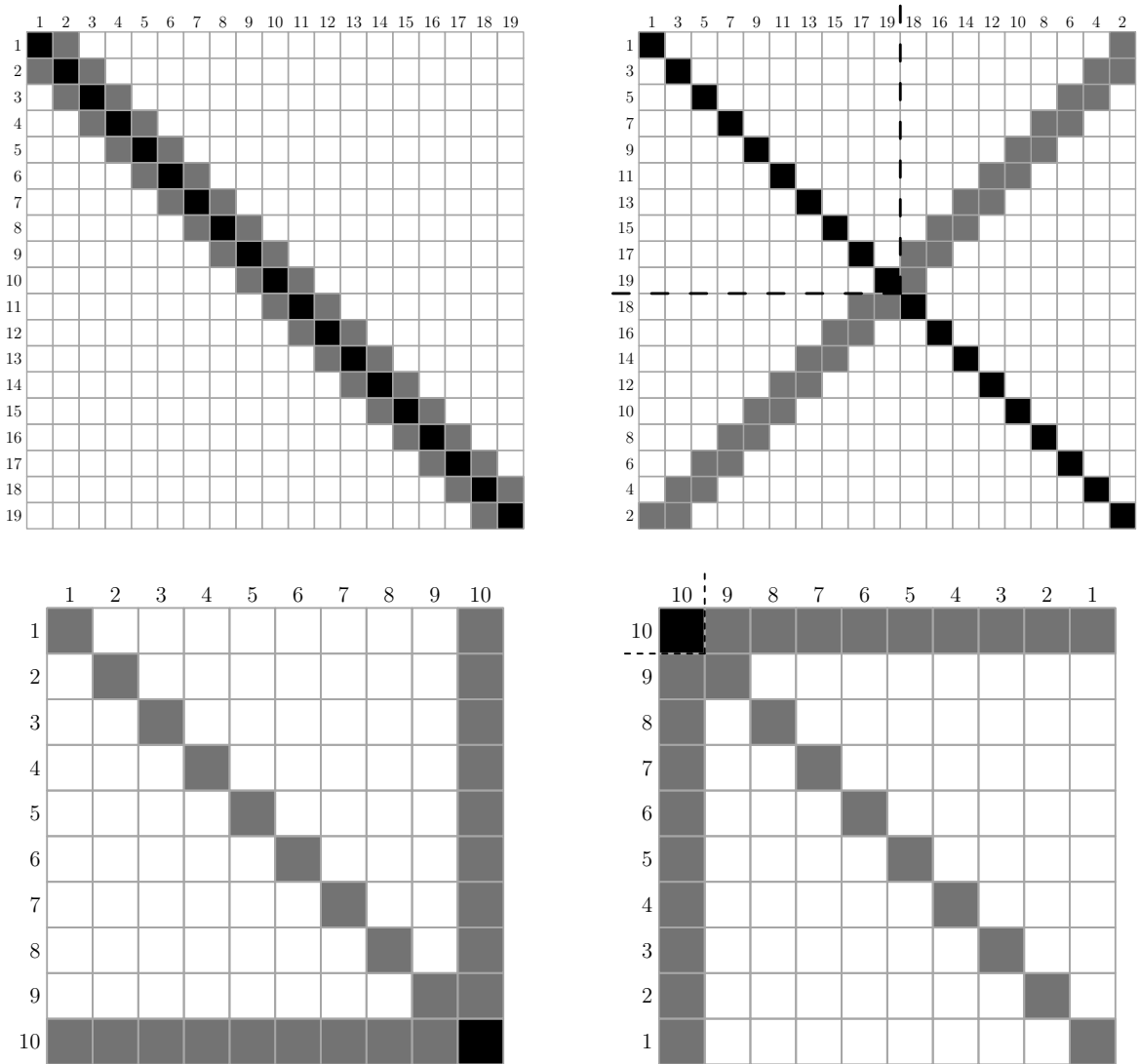


Figure 3. The left column shows the input matrices, the right column the corresponding matrices ordered to BLTF with maximum-weight diagonal; black entries correspond to 10, gray entries to 1. The tridiagonal matrix of size n will have a border width $\frac{n}{2}$ (top row). In the worst case, when the optimal ordering is the arrowhead matrix (bottom row), the border width is $n - 1$.

Unlike previously, the elimination steps are mathematically sound here. However, in 64-bit floating-point arithmetic we get: $x_{42} = \text{inf}$ and $x_{44} = \text{nan}$, where inf and nan stand for infinity and not a number, see (IEEE 754). The eliminations cannot be continued as x_{42} does not have any correct significant digits, and x_{44} is not a number. Unfortunately, similar failures are not at all uncommon in practice, especially with thermodynamic models.

Getting a floating-point exception due to a single elimination step is an extreme case. More common is that the error in the tear variables is amplified during a sequence of elimination steps, as already discussed in Sections 3.1, and this leads to an interaction between failure modes by triggering a floating-point exception. For example, let us assume that x_{41} is not a tear variable, but an eliminated one, and the sequence of eliminations leading up to x_{41} yields the value 440 for x_{41} due to amplification of the error in the tear variables. This is similar to Example 1 of Sec. 3.1 where the value of x_{20} was three orders of magnitude off compared to the true value. In Example 1 it was the residual that became uncontrollable, here it is a floating-point exception that causes the ultimate failure.

4.3 Multivalued elimination step

In the equation

$$x_1^2 + 2x_1x_2 - 1 = 0, \quad (20)$$

the elimination of x_1 requires to solve this equation for x_1 . However, there are two possibilities to perform this:

$$x_1 = -x_2 + \sqrt{x_2^2 + 1} \quad \text{and} \quad x_1 = -x_2 - \sqrt{x_2^2 + 1}. \quad (21)$$

To continue the remaining eliminations, we would have to know which solution for x_1 will remain feasible, or continue with both possibilities for x_1 . If we ignore the fact that the elimination step is multivalued, we either risk losing solutions, or we risk that we continue with that value for x_1 that becomes infeasible in later eliminations. This failure mode is simply ignored in state-of-the-art Modelica implementations, for example, “[the solver] *hopefully* returns the solution closest to the guess value” (Dymola User Manual, Ch. 8.9.2, p. 442); the emphasis is ours. If we want to find all well-separated solutions of a nonlinear system of equations, the kind of applications discussed in (Baharev et al., 2016), this is unacceptable.

4.4 Avoiding floating-point exceptions, undefined and multivalued elimination steps

The commonly seen workaround in state-of-the-art Modelica tools to avoid undefined and multivalued elimination steps is to choose only linearly appearing variables with non-zero coefficients as tear variables. Although this workaround is easy to implement, it is also overly limiting in the choice of the tear variables. This can lead to a BLTF with unacceptably wide border and eventually to very poor performance, because it excludes variables

that are perfectly eligible to become a tear variable. The same holds for disallowing division by variables in an attempt to avoid undefined elimination steps and floating-point exceptions. As for multivalued eliminations, it is left to chance whether a feasible solution is found or not, see Sec. 4.3.

It is moderately easy to avoid single elimination steps that can potentially become problematic depending on the actual values of the variables involved: In Baharev et al. (2017b) we proposed a novel pre-processing technique based on interval arithmetic for recognizing single-step eliminations that are *guaranteed* to be single-valued and numerically well-behaved, irrespective of the actual value of the variables involved. (Of course, it does not prevent a sequence of eliminations from becoming ill-conditioned.) Our approach offers more flexibility in the choice of tear variables than the common workaround, and it does that without risking any numerically troublesome operation. The increased flexibility in the choice of the tears can help to reduce the border width of the BLTF significantly. As for multivalued elimination steps, nothing is left to chance in our approach. For those familiar with linear algebra: Our method is, in some sense, a nonlinear extension of threshold pivoting (Duff et al., 1986, Ch. 4.4).

5 A novel robust approach

Staircase sampling was inspired by (Baharev and Neumaier, 2014), and proposed in (Baharev et al., 2016) to mitigate all of the issues listed in Sections 3 and 4. A detailed presentation of this method is outside the scope of this paper; here we only sketch the basic idea.

The subsystem

$$g(y, z) = 0 \quad (22)$$

in (5) is an underdetermined system of equations; it has infinitely many solutions per our assumptions in the first paragraph of Section 3. The aim of staircase sampling is to find a small set of points such that every solution of (22) is close to one of the points in this set. We call this small set of points the *sample*: It is an approximation to a sample from the infinitely many solutions of (22). The sample is built up incrementally, similarly to the usual tearing approach. Staircase sampling requires finite and reasonable lower and upper bounds on all of the variables; this is needed to allow an adequate sampling of the search space.

Staircase sampling starts with an *entire set* of values for z (a scattered set of points between the variable bounds), and not just with a single value for z as in the usual tearing approach. The algorithm then proceeds similarly to the common tearing algorithms, and it performs eliminations. A minor difference compared to (3) is that staircase sampling solves small nonlinear systems in the elimination steps, that is, it performs block elimination. The fundamental difference is that after each block elimination step, the points are redistributed, and a subvector of y is recomputed as necessary. The goal of this redistribution algorithm is to improve the spatial distribution of the

points between the variable bounds: It discards points that are too close, and it inserts new points where the search space has become deserted. The details of the redistribution algorithm are discussed in (Baharev et al., 2016). Staircase sampling returns a set of scattered points, satisfying (22) fairly well. In the current implementation, those points are chosen that violate (5) the least, and they are used as a starting point for large-scale sparse solvers that target solving (1) directly. In the future, interpolation and extrapolation on the complete set of scattered points will be used to find starting points that approximately satisfy (1).

5.1 How staircase sampling resolves the failure modes of traditional tearing

We now compare staircase sampling to traditional tearing from the point of view of the failure modes; the items are enumerated in the same order as in Sections 3 and 4.

1. As shown in Sections 3.1 and 3.2, the error in our initial estimate for z can grow or attenuate exponentially even in simple cases, ultimately leading to the failure of traditional tearing. Staircase sampling breaks this exponential change in the redistribution step; the error accumulation in an exponential rate is not possible.
2. As a consequence of the previous point, we can minimize the border width in our ordering algorithm without having to worry about the exponential error growth rate during the eliminations. An exact ordering algorithm to minimize the border width is given in (Baharev et al., 2017b). Furthermore, staircase sampling works on so-called staircase triangular matrices, and those matrices allow more flexibility in the orderings than the BLTFs do.
3. A single block elimination step can also fail, however, this is usually not an issue. Staircase sampling works with a set of points, losing some of them is typically not a problem: New points are inserted after each block elimination step in the redistribution algorithm, which makes up for the lost points.
4. Staircase sampling builds up a set of solution vectors, not just a single solution vector at a time as in traditional tearing. As a consequence, multivalued elimination steps are handled naturally.

5.2 A note on plotting vectors in 2 dimensions

Here we explain how the starting points and solution vectors will be plotted in the next section. We first select a subset of the variables according to an appropriate rule; for example, we select the methanol composition in each device of the system that we are simulating. Let us assume that we have selected a 20-dimensional subset. We then draw each 20-dimensional vector as a curve in 2 dimensions by connecting the points (x_i, i) ($i = 1:20$) with

adjacent indices, as shown in Fig. 4. The connecting lines have no meaning, but allow us to plot without ambiguity several vectors in one figure.

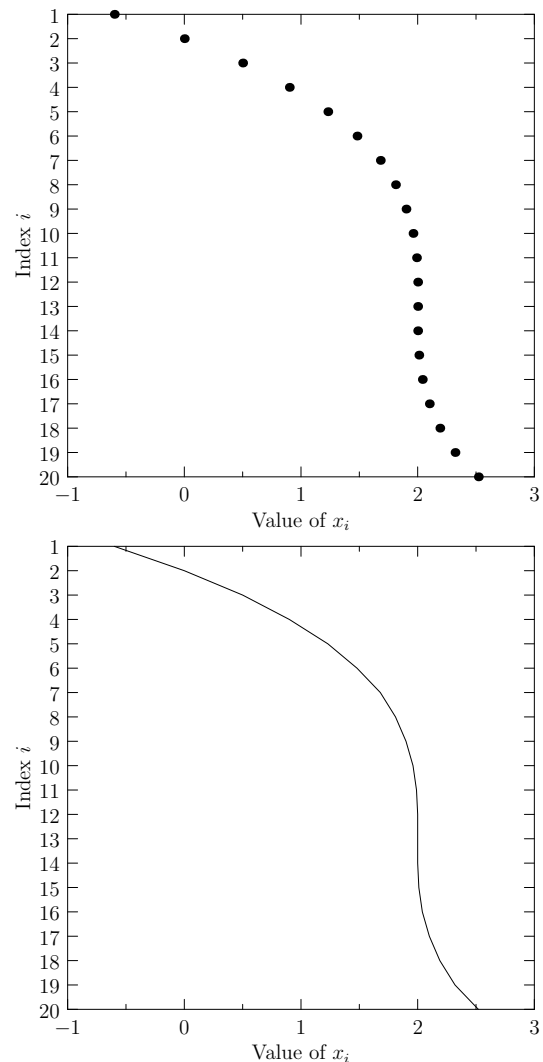


Figure 4. Top: Plotting the 20-dimensional vector x in 2 dimensions by placing a dot at (x_i, i) for $i = 1:20$. Bottom: To indicate that the dots belong to the same vector, we connect the neighboring points with linear lines, and we may omit the dots. The connecting lines have no meaning, but allow us to plot without ambiguity several vectors in one figure.

5.3 Demonstration test case

The robustness of staircase sampling is demonstrated on a particularly challenging distillation column. The model and its parameters correspond to the Auto model (Guttinger et al., 1997). The problem has three steady-state solutions: two stable steady-state branches and an unstable branch. Both the inside-out procedure (Boston and Sullivan, 1974) and the simultaneous correction procedure (Naphthali and Sandholm, 1971) were reported to miss the unstable steady-state solution, see (Vadapalli and Seader, 2001) and (Kannan et al., 2005). However, all steady-state branches were computed either with the AUTO software

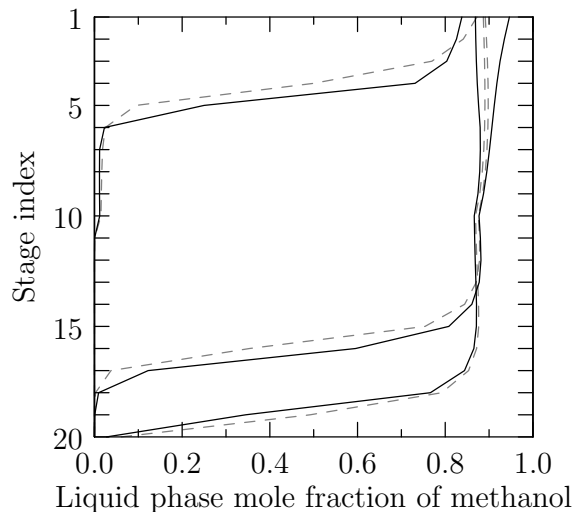


Figure 5. The three steady-state solutions (dashed gray lines) and those generated starting points (solid black lines) that are the closest to them. The gradient-based solver IPOPT converges to the nearest solution when started from the corresponding starting point.

package (Doedel et al., 1995) or with an appropriate continuation method (Güttinger et al., 1997; Vadapalli and Seader, 2001; Kannan et al., 2005). The initial estimates were carefully chosen with the ∞/∞ analysis (Bekiaris et al., 1993; Güttinger and Morari, 1996), and special attention was paid to the turning points and branch switching.

Our goal with staircase sampling was to find all solutions automatically, without any initial estimates, without relying on any domain-specific knowledge, and without any human interaction. This goal was achieved: All three steady-state solutions are found when IPOPT (Wächter and Biegler, 2006) is run from the starting points generated with staircase sampling. Figure 5 shows the three steady-state solutions of a 20-stage column and those starting points that are the closest to them; see also Sec. 5.2 as to how the solution vectors are plotted. For a more detailed discussion of this example, and for other examples see (Baharev et al., 2016).

Despite these promising results, the practical applicability and limitations of staircase sampling are yet to be explored, and a benchmark suite with real-world problems would be needed for that.

Acknowledgement

The research was funded by the Austrian Science Fund (FWF): P27891-N32. Support by the Austrian Research Promotion Agency (FFG) under project numbers 846920 and 853930 is thankfully acknowledged.

References

B. Bachmann, P. Aronßon, and P. Fritzson. Robust initialization of differential algebraic equations. In *1st International Workshop on Equation-Based Object-Oriented Model-*

ing Languages and Tools (Berlin; Germany; July 30; 2007), Linköping Electronic Conference Proceedings, pages 151–163. Linköping University Electronic Press; Linköpings universitet, 2007.

A. Baharev, H. Schichl, and A. Neumaier. Decomposition methods for solving nonlinear systems of equations. Submitted, 2017a. URL http://reliablecomputing.eu/baharev_tearing_survey.pdf.

A. Baharev, H. Schichl, and A. Neumaier. Ordering matrices to bordered lower triangular form with minimal border width. Submitted, 2017b. URL http://reliablecomputing.eu/baharev_tearing_exact_algorithm.pdf.

Ali Baharev and Arnold Neumaier. A globally convergent method for finding all steady-state solutions of distillation columns. *AIChE J.*, 60:410–414, 2014.

Ali Baharev, Ferenc Domes, and Arnold Neumaier. A robust approach for finding all well-separated solutions of sparse systems of nonlinear equations. *Numerical Algorithms*, pages 1–27, 2016. doi:10.1007/s11075-016-0249-x. URL <https://doi.org/10.1007/s11075-016-0249-x>.

N. Bekiaris, G. A. Meski, C. M. Radu, and M. Morari. Multiple steady states in homogeneous azeotropic distillation. *Ind. Eng. Chem. Res.*, 32:2023–2038, 1993.

J. F. Boston and S. L. Sullivan. A new class of solution methods for multicomponent, multistage separation processes. *Can. J. Chem. Eng.*, 52:52–63, 1974.

Emanuele Carpanzano. Order reduction of general nonlinear DAE systems by automatic tearing. *Mathematical and Computer Modelling of Dynamical Systems*, 6(2):145–168, 2000.

François E Cellier and Ernesto Kofman. *Continuous system simulation*. Springer Science & Business Media, 2006.

R. de P. Soares and A. R. Secchi. EMSO: A new environment for modelling, simulation and optimisation. In *Computer Aided Chemical Engineering*, volume 14, pages 947–952. Elsevier, 2003.

E. J. Doedel, X. J. Wang, and T. F. Fairgrieve. AUTO94: Software for continuation and bifurcation problems in ordinary differential equations. Technical Report CRPC-95-1, Center for Research on Parallel Computing, California Institute of Technology, Pasadena CA 91125, 1995.

M. F. Doherty, Z. T. Fidkowski, M. F. Malone, and R. Taylor. *Perry's Chemical Engineers' Handbook*, chapter 13, page 33. McGraw-Hill Professional, 8th edition, 2008.

I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Clarendon Press, Oxford, 1986.

Dymola User Manual. Volume 2. Dymola 2017 FD01, Dassault Systèmes AB, 2016.

H. Elmqvist and M. Otter. Methods for tearing systems of equations in object-oriented modeling. In *Proceedings ESM'94, European Simulation Multiconference, Barcelona, Spain, June 1–3*, pages 326–332, 1994.

- Hilding Elmqvist. *A Structured Model Language for Large Continuous Systems*. PhD thesis, Department of Automatic Control, Lund University, Sweden, May 1978.
- Robert Fourer, David M. Gay, and Brian Wilson Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Brooks/Cole USA, 2003.
- Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press, 2004.
- G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, USA, 3rd edition, 1996.
- gPROMS. Process Systems Enterprise Limited, gPROMS. <https://www.psenderprise.com>, 2017. [Online; accessed 21-Jan-2017].
- Prem K. Gupta, Arthur W. Westerberg, John E. Hendry, and Richard R. Hughes. Assigning output variables to equations using linear programming. *AIChE Journal*, 20(2):397–399, 1974.
- T. E. Güttinger and M. Morari. Comments on “multiple steady states in homogeneous azeotropic distillation”. *Ind. Eng. Chem. Res.*, 35:2816–2816, 1996.
- T. E. Güttinger, C. Dorn, and M. Morari. Experimental study of multiple steady states in homogeneous azeotropic distillation. *Ind. Eng. Chem. Res.*, 36:794–802, 1997.
- HSL. A collection of Fortran codes for large scale scientific computation., 2017. URL <http://www.hsl.rl.ac.uk>.
- IEEE 754. IEEE standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, Aug 2008. doi:10.1109/IEEESTD.2008.4610935.
- A. Kannan, M. R. Joshi, G. R. Reddy, and D. M. Shah. Multiple-steady-states identification in homogeneous azeotropic distillation using a process simulator. *Ind. Eng. Chem. Res.*, 44:4386–4399, 2005.
- A. Kröner, W. Marquardt, and E.D. Gilles. Getting around consistent initialization of DAE systems? *Computers & Chemical Engineering*, 21(2):145–158, 1997.
- F. Magnusson and J. Åkesson. Symbolic elimination in dynamic optimization based on block-triangular ordering. *Optimization Methods and Software*, 2017. doi:10.1080/10556788.2016.1270944. Published online: 17 Jan 2017.
- S. Mattsson, H. Elmqvist, and M. Otter. Physical system modeling with Modelica. *Control. Eng. Pract.*, 6:501–510, 1998.
- S. E. Mattsson, M. Otter, and H. Elmqvist. Modelica hybrid modeling and efficient simulation. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 4, pages 3502–3507, 1999.
- L. M. Naphthali and D. P. Sandholm. Multicomponent separation calculations by linearization. *AIChE J.*, 17:148–153, 1971.
- L. A. Ochel and B. Bachmann. Initialization of equation-based hybrid models within OpenModelica. In *5th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools (University of Nottingham; Nottingham, UK; April 19, 2013)*, Linköping Electronic Conference Proceedings, pages 97–103. Linköping University Electronic Press; Linköpings universitet, 2013.
- Online Supplement, 2017. URL <https://github.com/baharev/failure-modes-of-tearing>.
- C. C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal on Scientific and Statistical Computing*, 9(2):213–231, 1988.
- P. C. Piela, T. G. Epperly, K. M. Westerberg, and A. W. Westerberg. ASCEND: An object-oriented computer environment for modeling and analysis: The modeling language. *Computers & Chemical Engineering*, 15(1):53–72, 1991.
- M. Sielemann and G. Schmitz. A quantitative metric for robustness of nonlinear algebraic equation solvers. *Mathematics and Computers in Simulation*, 81(12):2673–2687, 2011.
- M. Sielemann, F. Casella, and M. Otter. Robustness of declarative modeling languages: Improvements via probability-one homotopy. *Simulation Modelling Practice and Theory*, 38:38–57, 2013.
- P. Täuber, L. Ochel, W. Braun, and B. Bachmann. Practical realization and adaptation of Cellier’s tearing method. In *Proceedings of the 6th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pages 11–19, New York, NY, USA, 2014. ACM.
- M. Tiller. *Introduction to physical modeling with Modelica*. Springer Science & Business Media, 2001.
- J. Unger, A. Kröner, and W. Marquardt. Structural analysis of differential-algebraic equation systems — theory and applications. *Computers & Chemical Engineering*, 19(8):867–882, 1995.
- A. Vadapalli and J. D. Seader. A generalized framework for computing bifurcation diagrams using process simulation programs. *Comput. Chem. Eng.*, 25:445–464, 2001.
- R.C. Vieira and E.C. Biscaia Jr. Direct methods for consistent initialization of DAE systems. *Computers & Chemical Engineering*, 25(9–10):1299–1311, 2001.
- A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.
- A. W. Westerberg and F. C. Edie. Computer-aided design, Part 1 Enhancing Convergence Properties by the Choice of Output Variable Assignments in the Solution of Sparse Equation Sets. *The Chemical Engineering Journal*, 2:9–16, 1971a.
- A. W. Westerberg and F. C. Edie. Computer-Aided Design, Part 2 An approach to convergence and tearing in the solution of sparse equation sets. *Chem. Eng. J.*, 2(1):17–25, 1971b.

Towards Adjoint and Directional Derivatives in FMI utilizing ADOL-C within OpenModelica

Willi Braun¹ Kshitij Kulshreshtha² Rüdiger Franke³ Andrea Walther² Bernhard Bachmann¹

¹FH Bielefeld University of Applied Science, {wbraun, bernhard.bachmann}@fh-bielefeld.de

²Universität Paderborn, kshitij@math.upb.de, andrea.walther@uni-paderborn.de

³ABB AG, Mannheim, ruediger.franke@de.abb.com

Abstract

Algorithmic differentiation has proven to be an efficient method for evaluating derivative information for implementations of mathematical functions. In the context of the Functional Mockup Interface (FMI) the reverse mode of algorithmic differentiation shows immense promise.

FMI is increasingly used for model-based applications, such as parameter estimation or optimal control. The paper motivates the exploitation of algorithmic differentiation and proposes an extension of FMI for the evaluation of adjoint directional derivatives.

Attempts to interface algorithmic differentiation libraries with Modelica tools have been made. Instead of generating code for the target language which is instrumented with algorithmic differentiation library API and then compiled, in this new approach the intermediate representation used by the library is generated directly. This avoids compilation of the target language that often takes a large fraction of the overall simulation time. It also avoids model execution in order to create such an internal representation at runtime. The initial results are presented here.

Keywords: OpenModelica, ADOL-C, Derivatives, Jacobian

1 Introduction

Algorithmic differentiation (Griewank and Walther, 2008) is a technique to compute derivatives of functions expressed as computer programs efficiently, and accurately upto machine precision (Griewank et al., 2012). Ruge et al. (2014) first investigated the use of the algorithmic differentiation tool ADOL-C (Walther and Griewank, 2012) in conjunction with OpenModelica (Fritzson et al., 2006). ADOL-C is designed for the C++ programming language and uses operator overloading to create an internal representation of the computation, called a trace, when a program instrumented with the datatype `adouble` is executed. In Ruge et al. (2014) such instrumented code written in C++ was generated in addition to the usual model code for the C-Runtime and was compiled and linked with the ADOL-C library in addition to the C-Runtime Library of OpenModelica.

In this work we endeavoured to generate the trace for

the use by the ADOL-C library to compute derivatives directly from within the OpenModelica compiler. Since the compiler has all the required information about the computation of the model it can present this information in the manner we need, without having to generate C++ code and executing it. On the other hand the ADOL-C library did not have any other mechanism for creation the internal data structures associated with a trace, other than executing C++ code instrumented with the datatype `adouble`. The challenge was therefore two-fold: firstly to teach the ADOL-C library to accept a trace in another format, and secondly to generate this format from the OpenModelica compiler while processing the model.

This paper is organized as follows: Section 2 outlines the motivation for the exploitation of algorithmic differentiation in FMI. Further more it presents a proposal for an extension of FMI offering the evaluation of adjoint directional derivatives. The needed details of algorithmic differentiation as well as the implementation of ADOL-C are described in section 3. Whereas section 4 focuses on the implementation work in OpenModelica. Finally, the first results are shown in section 5.

2 Adjoint directional derivatives in FMI

FMI emerged as a new standard resulting from the ITEA2 project MODELISAR, in 2010. The standard is a response to the industrial need to connect different environments for modeling, simulation and control system design. Commonly, different tools are used for different applications, whereas simulation analysis at the system integration level requires tools to be connected. FMI provides the means to perform such integrated simulation analysis.

FMI specifies an XML format for model interface information and a C API for model execution. The XML format, specified by an XML schema, contains information about model variables, including names, units and types, as well as model meta data. The C API, on the other hand, contains C functions for data management, e.g., setting and retrieving parameter values, and evaluation of the model equations. The implementation of the C API may be provided in source code format, or more commonly as a compiled dynamically linked library.

Starting from version 2.0, the Functional Mock-up Interface (FMI) is well suited to hook Modelica models to numerical solvers for model-based applications, such as parameter estimation or optimal control (Franke et al., 2015). This way numerical routines can focus on the solution process, while FMI abstracts implementation details of the model. It is likely that such applications of FMI will increase. FMI should provide appropriate model derivatives.

Consider the solution of a least squares problem for parameter estimation as example. A general model has the form

$$\begin{aligned} v_{unknown} &= h(v_{known}, v_{rest}), \\ h: \mathbb{R}^{nKnown} \times \mathbb{R}^{nRest} &\rightarrow \mathbb{R}^{nUnknown}. \end{aligned} \quad (1)$$

The task is to obtain $nKnown$ parameters such that a sum of squared residuals for $nUnknown$ model outputs y and $k = 1, \dots, K$ given data points y_k is minimized:

$$R = \sum_{k=1}^K \|y_k - h(v_{known}, v_{rest,k})\|^2 \rightarrow \min_{v_{known}}. \quad (2)$$

The solution must fulfill the necessary condition

$$\begin{aligned} \frac{\partial R}{\partial v_{known}} &= \text{zeros}(nKnown) = \\ &-2 \sum_{k=1}^K [y_k - h(v_{known}, v_{rest,k})] \frac{\partial h(v_{known}, v_{rest,k})}{\partial v_{known}}. \end{aligned} \quad (3)$$

The solution process, for instance applying Newton's method, involves the successive computation of (3), including model derivatives.

The existing API of FMI 2.0 provides the function `fmi2GetDirectionalDerivative` to obtain a column of the Jacobian matrix $\partial h(v_{known}, v_{rest}) / \partial v_{known}$ or a linear combination of columns of the Jacobian matrix. One computation of directional derivatives gives:

$$\Delta v_{unknown} = \frac{\partial h(v_{known}, v_{rest})}{\partial v_{known}} \Delta v_{known}. \quad (4)$$

The signature of the API function is:

```
fmiStatus fmi2GetDirectionalDerivative(
    fmiComponent c,
    const fmi2ValueReference vUnknown_ref[],
    size_t nUnknown,
    const fmi2ValueReference vKnown_ref[],
    size_t nKnown,
    const fmi2Real dvKnown[],
    fmi2Real dvUnknown[])
```

The computation of (3) requires $K \times nKnown$ calls to `fmi2GetDirectionalDerivative`. This is inefficient if multiple parameters shall be estimated ($nKnown > 1$).

This is why the FMI interface should be extended with a new function `fmi2GetAdjointDerivative`, along with a capability flag `providesAdjointDerivatives`. The new function

computes:

$$\Delta v_{known} = \left(\frac{\partial h(v_{known}, v_{rest})}{\partial v_{known}} \right)^T \Delta v_{unknown}. \quad (5)$$

It has the signature:

```
fmiStatus fmi2GetAdjointDerivative(
    fmiComponent c,
    const fmi2ValueReference vUnknown_ref[],
    size_t nUnknown,
    const fmi2ValueReference vKnown_ref[],
    size_t nKnown,
    const fmi2Real dvUnknown[],
    fmi2Real dvKnown[])
```

The new function allows to obtain one row of the Jacobian matrix, or a linear combination of rows of the Jacobian matrix, with only one model evaluation in reverse mode of algorithmic differentiation. The computation of (3) becomes significantly more efficient. Only K calls to `fmi2GetAdjointDerivative` are needed, one call for each data point k and arbitrary numbers of $nKnown$ parameters or $nUnknown$ model outputs, when passing the values of the residuals as seeds

$$\Delta v_{unknown,k} = y_k - h(v_{known}, v_{rest,k}). \quad (6)$$

3 Algorithmic differentiation using ADOL-C

In order to apply algorithmic differentiation (AD) on a program we model the program structure as a sequence of instructions, which perform specific mathematical functions. This is called an evaluation procedure in Griewank and Walther (2008). The evaluation procedure can be then evaluated forwards or reverse to compute the derivatives in the so called forward mode and reverse mode of AD. Griewank and Walther (2008, Chapter 3 and 4) describe this process in great detail and give bounds on the complexity and memory requirements. The main import of the complexity analysis is that the reverse mode is very well suited to compute gradient vectors for functions in much less complexity than they can be computed otherwise, either numerically or symbolically. The same applies to computing rows of the Jacobian matrix.

ADOL-C implements the AD process by overloading the operators and mathematical functions in the C++ programming language for a special datatype `adouble`. Such supported operations are called elementary operations. Each of these overloaded operators and functions when executed records the elementary operation currently being performed, the locations of the operands in working memory, and the locations of the results in working memory. This record is created normally on runtime, when a program, instrumented with the ADOL-C headers, datatypes and some instructions on when to begin and end the recording, is executed. ADOL-C is then able to use this record, called a *trace*, to evaluate function values, first

and higher order derivatives in forward and reverse mode at any given point of evaluation. The trace can be made persistent even after the program has terminated. Traditionally this trace is stored in binary format as raw data in three different files, one for the list of operations, one for the list of operand locations, and one for storing any constant values that might occur. The organisation of the trace is as follows. For each operation a character to represent it is stored. Based on what the operation actually is, it will require a number of operands, which are stored as unsigned integer locations inside a data buffer, followed by the location, where the result of the operation will be stored. In some operations a constant may be involved, this value is stored as is.

In the work of Ruge et al. (2014) the Modelica models were translated into C++ code instrumented with the ADOL-C headers and datatypes, using a mechanism similar to the generation of C code for the model in OpenModelica. The drawback was that compilation linking and one-time execution of this C++ code was quite slow due to the use of operator overloading, compared to the generated C code. It was suggested, that since OpenModelica was already analysing the model in great detail, could we not create the trace of the model directly instead of generating C++ code.

```
// define independent
{ op:assign_ind loc:0 }
{ op:assign_ind loc:1 }
// operations
{ op:mult_d_a loc:0 loc:4 val:-0.25 }
{ op:div_a_a loc:1 loc:0 loc:5 }
{ op:plus_a_a loc:4 loc:5 loc:6 }
{ op:plus_d_a loc:6 loc:3 val:3.0 }
{ op:log_op loc:0 loc:4 }
{ op:mult_d_a loc:4 loc:5 val:-3.0 }
{ op:plus_a_a loc:1 loc:5 loc:2 }
// define dependent
{ op:assign_dep loc:2 }
{ op:assign_dep loc:3 }
// death_not
{ op:death_not loc:0 loc:8 }
```

Figure 1. An example of a textual trace for ADOL-C

OpenModelica is able to generate textual information rather than binary. Therefore the first step required for creating a trace directly was to allow a textual representation of the trace and that ADOL-C understands such a textual representation. A simple ASCII representation of the operations, locations and constants was devised with some delimiters to make parsing easier. Each elementary operation supported by ADOL-C is given a textual name stored with the keyword "op:". The locations of all the operands inside the work buffer in decimal notation follow this and then the location of the result in the work buffer, each of these using the keyword "loc:". At the end any required constant for the particular operation is given in decimal floating point notation using the key-

word "val:". Braces separate one such record from another. This textual representation is a natural extension of the binary representation for ADOL-C traces, which has long been a part of the ADOL-C public API. An ADOL-C driver function can now be used to convert any traditional binary trace to this textual representation and store it in a file. This format will be made a part of the public API of ADOL-C in the next feature release. An example of a file containing such a textual trace is shown in Figure 1.

A driver was added to ADOL-C to be able to read and parse a text file in ASCII notation with the above information using regular expressions to match the format described above and convert it to the traditional binary notation at runtime. Anything not matching the defined regular expressions is considered a comment and ignored.

4 Generation of Operation Lists

In the first step of the compilation process in Modelica tool, a model is transformed by the front-end into a flat representation, consisting essentially of lists of variables, functions, equations and algorithms. In this phase, a basic structural analysis of the differential-algebraic equations (DAE) is performed to detect the states and discrete variables and eliminate alias variables. The basic step of a Modelica compiler is to causalize the DAE and transform into ordinary differential equations (ODE). Then the target code is generated from the optimized system in order to perform the simulation. For the simulation the generated code needs to be compiled by the target language compiler and linked with the simulation runtime library. The default target language of the OpenModelica Compiler (OMC) is C and the GNU C compiler is used as default C compiler. In order to generate operation lists by OMC, which are readable by ADOL-C, the code generation module of OMC has been extended by a new target, the ADOL-C target. Basically the operation lists are generated by traversing the equation expressions and for every mathematical operation creating the corresponding ADOL-C operation. This is straight forward for assignments thus the OMC has transformed all equations into assignments due to the causalization. The implicit equations of the strong connected components require special treatment (Griewank and Walther, 2008). Furthermore, the OpenModelica simulation runtime is linked with the ADOL-C library in order to enable the usage of the ADOL-C capability during the simulation process. At the current status of the implementation we evaluate the sparse Jacobian for the integration process. One main advantage over the former approach is that the compilation of the target language can be avoided by processing the operation lists directly.

5 First Results

The first results to test the performance of the approach presented here are based on benchmark models from the ScalableTestSuite library (Casella, 2015). In the current implementation status the sparse jacobian evaluation used

by the time integration method ida is used for evaluation. In the tables 1 and 2 all numbers are produced by using the model ScalableTestSuite.Elementary.SimpleODE.Models.CascadedFirstOrder.

Table 1. Evaluation time of the Jacobian. Compare OMC symbolic vs. ADOL-C

N	ADOL-C	OM Symbolic
100	0.000480442	0.000156783
200	0.000830835	0.000413299
400	0.00157551	0.000952923
800	0.00294508	0.00209405
1600	0.00676732	0.00536921
3200	0.0141433	0.012003
6400	0.0390204	0.0310391
12800	0.0771545	0.0756394
25600	0.1532143	0.1621433

In table 1 the time of one Jacobian evaluation is stated, calculated by $\frac{totalTime}{N}$, where *totalTime* is the time that is needed to evaluate the Jacobian over the entire simulation horizon and *N* is the number of evaluations done. One can see that the evaluation time for the given model is quite equal between the generated symbolic Jacobian by OMC and the evaluation by ADOL-C. Note that ADOL-C is performing additional work (e.g. memory allocation and colouring) in the first call.

Table 2. Generation performance of Jacobian. Compare OMC symbolic vs. ADOL-C

N	ADOL-C		OM Symbolic	
	generate	read	generate	compile
100	0.00046	0.01475	0.015	
200	0.00089	0.02879	0.032	
400	0.00178	0.05794	0.059	
800	0.00372	0.11320	0.119	0.03
1600	0.00860	0.22766	0.244	0.14
3200	0.01749	0.45620	0.523	0.38
6400	0.03702	0.91150	1.229	0.48
12800	0.07571	1.82352	2.569	1.01
25600	0.15910	3.60362	5.459	1.65

The performance of generating the appropriate Jacobian is stated in table 2. These timings are divided in two stages. For ADOL-C it is time for the generation of the operation list, and the time to read them at runtime. For the symbolic Jacobian generated by OMC it is the generation of directional derivative code and the additional time to compile the generated C code. This result shows the linear complexity of the new approach presented in this paper.

6 Conclusion and Future work

This paper presents a new approach to generate a model evaluation trace for algorithmic differentiation, where no

compilation of the model code is needed any more. The advantage of this approach is not only good performance, moreover it gives access to a feature-rich AD tool (e.g. higher-derivative, reverse mode). Furthermore, an extension of FMI involving adjoint derivatives is proposed and motivated by optimization-based applications, where such derivatives are mandatory. The implementation of this extension can be achieved by the approach described here. However, this requires some more implementation work, since the current implementation does not yet support all Modelica language features. The most important and challenging aspect is the treatment of implicit equations. In future the authors will continue working on supporting more language features with the approach described. Further, the here proposed FMI extension will be implemented and demonstrated with a complex example.

7 Acknowledgments

The presented work is part of the PARADOM project, that is funded by the Federal Ministry of Education and Research (BMBF) under the support code 01IH15002.

References

- F. Casella. Simulation of large-scale models in Modelica: State of the art and future perspectives. In P. Fritzson and H. Elmqvist, editors, *Proceedings 11th International Modelica Conference*, pages 459–468, Versailles, France, Sep 21–23 2015. The Modelica Association. doi:10.3384/ecp15118459.
- R. Franke, M. Walther, N. Worschech, W. Braun, and B. Bachmann. Model-based control with FMI and a C++ runtime for Modelica. In *Proceedings of the 11th International Modelica Conference*. Modelica Association, Paris, France, Sep. 2015.
- P. Fritzson, P. Aronsson, H. Lundvall, K. Nyström, A. Pop, L. Saldamli, and D. Broman. Openmodelica - a free open-source environment for system modeling, simulation, and teaching. In *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, pages 1588–1595, oct. 2006. doi:10.1109/CACSD-CCA-ISIC.2006.4776878.
- A. Griewank and A. Walther. *Principles and Techniques of Algorithmic Differentiation, Second Edition*. SIAM, 2008.
- A. Griewank, K. Kulshreshtha, and A. Walther. On the numerical stability of algorithmic differentiation. *Computing*, 94(2-4):125–149, 2012.
- V. Ruge, W. Braun, B. Bachmann, A. Walther, and K. Kulshreshtha. Efficient implementation of collocation methods for optimization using openmodelica and ADOL-C. In H. Tummescheit and K.-E. Årzén, editors, *Proceedings of the 10th Modelica Conference*, pages 1017–1025, Lund, Sweden, 2014. Modelica Association and Lund University Electronic Press. doi:10.3384/ECP140961017.
- A. Walther and A. Griewank. Getting started with ADOL-C. In U. Naumann and O. Schenk, editors, *Combinatorial Scientific Computing*, pages 181–202. Chapman-Hall, 2012.

PDEModelica and Breathing in an Avalanche

Jan Šilar^{*,+}, Filip Ježek[#], Jiří Kofránek⁺

⁺ Institute of Pathological physiology, First Faculty of Medicine, Charles University, Prague, Czech republic

[#] Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic

^{*} Corresponding Author Institute of Pathological physiology, U nemocnice 5, Praha 2 128 00, Czech Republic,
jansilar@jansilar.cz

Abstract

This paper presents an updated version of Modelica language extension for partial differential equations (PDE) called PDEModelica and implementation of its support in OpenModelica. This support is limited to 1-dimensional problems and the first and second partial derivatives. PDEModelica is introduced by a string equation model and later by a real life model of respiration during a snow burial. This model describes CO₂ advection and diffusion in snow described by advection-diffusion PDE.

PDEModelica, PDE, avalanche survival

Introduction

PDEModelica is a Modelica language extension for partial differential equations (PDE). It was designed by Levon Saldamli (Saldamli, 2006). This original extension is currently not supported by any tool. We focused on a subset of this extension for 1-dimensional models only and introduced several changes and enhancements. Support for the renewed extension has been implemented in OpenModelica. We present the extension using a simple string equation model at first and then a real-life problem of modelling respiration during a snow burial.

In the past four decades, avalanches were responsible for around 100 deaths annually in the European Alps only (Techel et al., 2016). When a victim is buried by an avalanche he or she repetitively inspires previously expired air as the motion of air in snow is restricted. The body metabolism consumes O₂ and produces CO₂ and thus the concentration of O₂ decreases and the concentration of CO₂ increases in the inspired and expired air. The concentrations of O₂ and CO₂ are partially restored by diffusion. But this process is not fast enough and if the victim is not rescued within approximately 15 minutes he or she may die of

asphyxiation, i.e. a lack of oxygen supply to the cells. Asphyxia could be caused by a variety of situations, including excess of CO₂. More than 75 % of deaths in an avalanche are caused by asphyxia (McIntosh et al., 2007). However the content of oxygen in snow should satisfy the body needs – Radwin (Radwin et al., 2001) proved, that volunteers buried in snow with the removal of the expired gas did not have any problems even after an hour long burial. In contrast, no removal resulted in serious hypercapnia (i.e. an excessive amount of carbon dioxide in blood) within 10 minutes. In this paper, we focus on modeling of CO₂ diffusion only, as the O₂ is then a very similar problem.

Modeling task

It is assumed, that a potential cavity around the mouth and the nose significantly increase the chance of survival. Roubík et al. carried out an experiment (Roubík et al., 2015) where the volunteers were breathing through a tube whose end opened into a cavity in snow of various volumes. They proved that the size of the cavity has a significant impact on the concentration of O₂ and CO₂ in the inspired and expired air. There are at least two possible mechanisms causing this effect. First, the small cavity has a small surface of the air-snow boundary and so the resistance for the air flux is high. This causes an increase in the work of breathing, an increase in the metabolism rate and thus an increase in O₂ consumption and CO₂ production. Second, the expired air is mixed with more fresh air in the cavity and then the inspired air is also more fresh. Both mechanisms probably take place in the process. The question is which one dominates. We were asked to help with the investigation using a model.

Methods

PDEModelica

Let us introduce all new language elements of PDEModelica on the advection equation model:

```
1 model advection "advection equation"
2   parameter Real pi =
      Modelica.Constants.pi;
3   parameter DomainLineSegment1D omega (L =
      1, N = 100);
4   field Real u( domain= omega);
5   initial equation
6     u = sin(2*pi *omega.x);
7   equation
8     der(u) + pder(u, x) = 0 indomain omega;
9     u = 0 indomain omega.left;
10    u = extrapolateField(u)
      indomain omega.right;
11end advection;
```

- The Domain `omega` represents the geometrical domain where the PDE holds. The domain is defined using the built-in record `DomainLineSegment1D` (line 3). This record contains among others `L` – the length of the domain, `N` – the number of grid points, `x` – the coordinate variable and the regions `left`, `right` and `interior`, representing the left and right boundaries and the interior of the domain.
- The field variable `u` is defined using a new keyword `field` (line 4). The `domain` is a mandatory attribute to specify the domain of the field.
- The `indomain` operator specifies where the equation containing the field variable holds. It is utilised in the initial conditions (IC) of the fields, in the PDE and in the boundary conditions (BC). The syntax is `equation indomain domain.region`. If the `.region` is omitted, `.interior` is the default.
- The IC of the field variable `u` is written using an expression containing the coordinate variable `omega.x`. (line 6).

- The PDE contains a partial space derivative written using the `pder` operator (line 8). Also the second derivative is allowed (not in this example), the syntax is e.g. `pder(u, x, x)`. It is not necessary to write e.g. `omega.x` in `pder`, even though `x` is a member of `omega`.
- The BC is on line 9. The current limitation is that BCs may be written only in terms of variables that are spatially differentiated.
- All fields that are spatially differentiated must have at each boundary either BC or extrapolation. This extrapolation should be done automatically by the compiler, but this has not been implemented yet. The current workaround is the usage of the `extrapolateField()` operator directly in the model.

Comparison to the original version of PDEModica

Our extension is restricted to 1-dimensional models only. This allows much simpler domain definition using the built-in `DomainLineSegment1D` record compared to the original extension which enables arbitrary geometry domain definition in multiple dimensions.

`pder()` is used instead of `der()` for partial derivatives. A shortcut to leave out the full qualification of the `x` coordinate is established. This was probably intended in the original extension also, but was not explicitly mentioned.

`indomain` is used instead of `in` as it is suggested in (Fritzson, 2015) because `in` is already utilized in `for` loops. `indomain` is mandatory not only in the BCs but also in the ICs and the PDEs here.

Field literals are written as expressions containing the coordinate variable `x` and thus the special syntax for the field literal constructor of the original extension was suppressed.

Solution process

The PDEs are solved using the method of lines (MOL) (Schiesser, 2012): during flattening of the model, the fields are replaced by arrays and the space derivatives

are replaced by finite differences (currently only the central difference is implemented)

$$\frac{\partial u}{\partial x} \rightarrow \frac{u[i+1] - u[i-1]}{2 \cdot dx},$$

$$\frac{\partial^2 u}{\partial x^2} \rightarrow \frac{u[i+1] - 2 \cdot u[i] + u[i-1]}{dx^2}$$

and thus the PDEs are converted into a system of ODEs. This resulting system may be written in standard Modelica and is solved by current OpenModelica solvers. The combination of the central space difference with an implicit Euler time solver results in a backward-time centered-space (BTCS) scheme which is a common finite difference method. The usage of the trapezoid time solver results in the Crank-Nicolson method (Strikwerda, 2004). Other time solvers may be also successful, even though the resulting methods were not investigated. A selection of a proper time solver is important.

It is also substantial to select a proper time step, so that the Courant–Friedrichs–Lewy (CFL) condition (Courant et al., 1967) is fulfilled.

To enable PDEModelica in OpenModelica, the compiler flag `--grammar=PDEModelica` must be set.

The features for the plotting fields (arrays) have not been implemented in OpenModelica yet. We use Octave to load the result file and plot the desired variables.

Model of breathing in snow

For the first stages of the research, the problem is simplified into a gas flow to and from a spherical snowball (see Figure 1). Due to small pressure differences, the gas is modeled as incompressible. The only significant pressure difference could occur at the boundary between the cavity and the snow, but Roubík et al. (Roubík et al., 2015) did experience only small pressure differences.

The snow is a porous material, formed by ice and air. The CO_2 could flow and diffuse across the snow through the air gaps. Given the ice density (916 kg/m³) and the density of snow (100 - 400 kg/m³) the snow consists of at least 55 % of air. Therefore, the air could penetrate through the snow and mix with the air captured within the snow. For simplification, we exclude the solubility in

ice and possible melted water. The gas has a volumetric concentration of CO_2 , the O_2 is omitted, but it follows the same principles. The gas transport in snow is modeled using the advection-diffusion equation.

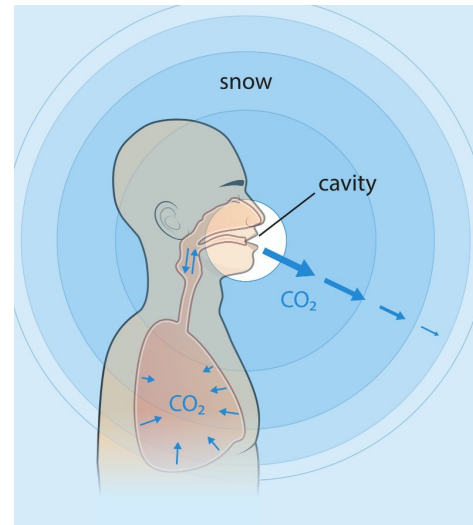


Figure 1 – Model schematics. The organism is producing CO_2 in a constant rate and it is concentrating in the lungs. The lungs expire to and inspire from a cavity, in which the air is ideally mixed. The air flux is given. The partial concentration of CO_2 in the cavity is drained by advection and diffusion through the snow. The dead volume in the airways is omitted.

Advection-diffusion equation and its formulation in PDEModelica

The advection-diffusion equation assuming the incompressible gas flow is

$$\frac{\partial c}{\partial t} + u \cdot \nabla c - D \Delta c = 0$$

where c is the concentration, u is the velocity of advection and D is a diffusion coefficient. We express this equation in the spherical coordinates. As our problem is spherically symmetrical, all derivatives except the derivatives in a radial direction are equal to zero. Then we obtain

$$\frac{\partial c}{\partial t} + \left(u - \frac{2D}{r} \right) \frac{\partial c}{\partial r} - D \frac{\partial^2 c}{\partial r^2} = 0,$$

$$u = \frac{q}{4\pi r^2}.$$

where r is the radius and q is a volumetric flow given by the lungs. This equation contains two partial derivatives. Using the principles described in the paragraphs above, the formulation of the advection-diffusion equation in PDEModelica is written in the Code listing 1. The full model of CO₂ breathing is available online¹.

```
model sb1m
  (...)
  Real C_CS "concentration on cavity-snow
  interface";
  DomainLineSegment1D omega(L = 0.5, N =
  100, x0 = R_C) "x is actually r, center on
  the left";
  field Real C_S(domain = omega)
  "concentration of CO2 in snow";
  (...)
  //Left BC during exhalation, extrapolation
  during inhalation
  C_S = if exhale then C_CS else
  extrapolateField(C_S) indomain omega.left;
  //The advection-diffusion equation
  der(C_S) + (q / (4 * pi * omega.x ^ 2)
  - 2 * D_S / omega.x) * pder(C_S, x)
  - D_S * pder(C_S, x, x) = 0
  indomain omega;
end sb1m;
```

Code listing 1: the advection-diffusion equation formulation in PDEModelica. New language elements are highlighted in purple.

Note, that the boundary conditions are switched with extrapolation every breathing cycle as the flux direction changes. This demonstrates the acausality of the proposed approach.

Results

In the presented model, we use the arbitrary parameter values to demonstrate the principles of CO₂ distribution. The exact identification of the values is a subject of additional research. However, the resulting trends are consistent with expectation and plausibility personally confirmed by the authors of (Roubik et al., 2015).

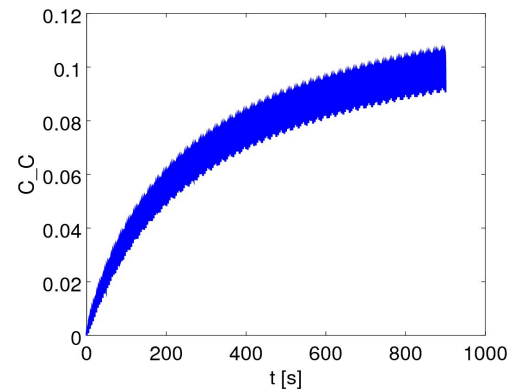


Figure 2 Concentration (fraction) in the cavity C_C (the Cavity volume 1 L) is changing between the inhale and the exhale.

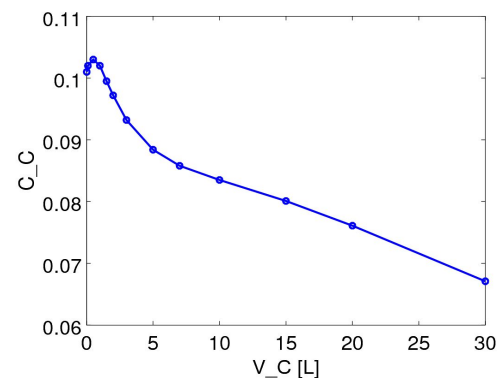


Figure 3: Average concentration C_C (average over 4 full breathing periods) in the snow cavity in time for various cavity sizes (V_C) at 15min.

The CO₂ concentration in the cavity rises with each expiration (Figure 2) and is rising towards an equilibrium. However, when the concentration of CO₂ is about 2 % the victim feels respiratory stimulation (here approx. 200s), at 6 % starts mental confusion (approx. 400s), followed by unconsciousness at 10% (approx. 800s) and later by death.

In Figure 3 we investigate the influence of the cavity size - the larger cavity, the longer the subject could survive (i.e. the lower cavity CO₂ concentration). If the volume of the cavity is smaller than 1L, CO₂ concentration does not change significantly. The size of the cavity has a huge impact from 1 to 5 L, but then the response becomes nearly linear. Unfortunately, the CO₂ concentration remains at unsatisfactory high levels.

¹ <https://github.com/jansilar/snowbreathing/>

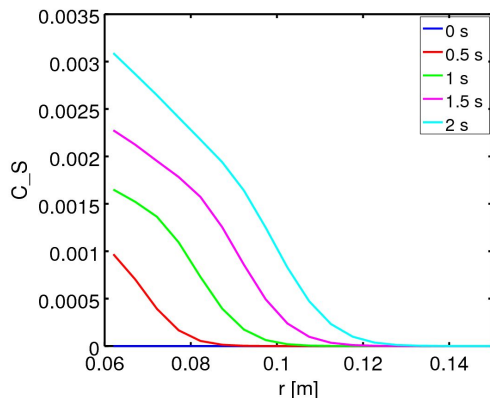


Figure 4 The concentration gradient dependable on the radius of the snowball during the first exhale (a half of breath period), the cavity volume 1 L.

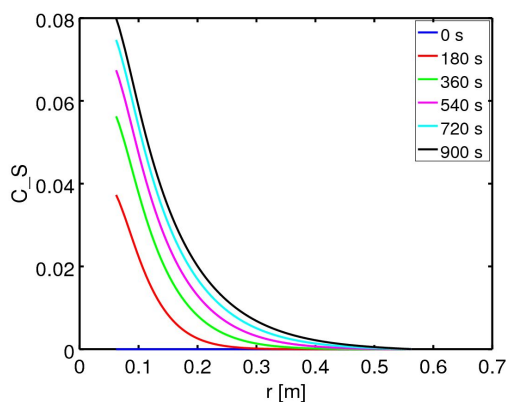


Figure 5 The concentration gradient dependable on the radius of the snowball during a longer periods, at the end of the breath period (i.e. after the exhale), the cavity volume 1 L.

We can see advection and diffusion of CO_2 into snow during the first breath out close to the cavity (Figure 4). Thanks to the r^2 attenuation of advection velocity, the CO_2 concentration is virtually zero within a few centimeters of the snow. Note that for long time periods the CO_2 proceeds further. Despite, the concentration is negligible in around 50 cm as the distribution volume grows rapidly with the radius (Figure 5).

Solution process performance

The model was translated and simulated several times with a different number of grid points. The trapezoid solver was used. The time step was chosen proportionally to the space step. The stop time (model time) was 5 minutes. The translation and simulation time and the size of the model binary file are in Table 1.

N	Step	Trans	Simul	Size
50	0.02	3,5	8,2	252
100	0.01	4,9	13,7	418
200	0.005	5,5	39,2	759
500	0.002	12,2	224,9	1843

Table 1 Performance comparison: N – number of grid points, step – the time step (s), trans – the time of translation (s), simul – the time of simulation (s), size – the size of the model binary (kB).

The simulation time increase substantially with increasing number of grid points. The results seem satisfying even for the simulation using 100 grid points (plotted). On the other hand this results were not verified by comparison with a different PDE simulation tool.

Discussion

The snow-breathing model

A mathematical model could help to study the countermeasures to avoid asphyxiation. This work supports the usage of devices for CO_2 removal and explains the underlying processes with the goal to contribute to their construction. Some CO_2 removal devices already exists, including a tube device to divert the CO_2 -rich exhale (Margid et al., 1998), but additional data are needed to prove their efficiency.

Changes in the human metabolism as a consequence of the increasing hypercapnia and hypoxia during the snow burial are not included in the model. Thus the presented model cannot describe the real process of breathing into snow. Development of the full model that includes the human physiology as well is the aim of the subsequent work. The current model has been greatly simplified by omitting oxygen, dead space in airways, solubility of CO_2 in other body compartments and in water contained in snow and also rising breath work, which produces more CO_2 . Thanks to Modelica implementation, it is planned to connect it directly with the most extensive open model of human physiology, the Physiome (Matejka and Kofránek, 2015).

The current parameters of the model are set arbitrarily and are not confirmed by the measurements. Therefore, the results may be taken as demonstrative only.

Alternative to PDEModelica

Instead of the presented solution, some other methods of using the PDEs in Modelica exists. One could make a usage of creating the PDEs in some other tool and then import them via FMI (Stavåker and Fritzson, 2014). For seamless Modelica integration, a dedicated library PDELib (Dshabarow et al., 2007) was developed. However, as it is not maintained, the examples are not working in the recent OpenModelica and Dymola versions.

We could have employed a manual PDE discretization and thus converting the PDEs into an ODE or DAE system. However using the manual discretization is in conflict with Modelica declarative philosophy. Utilising the language extension the modeller may focus on the model itself rather than its numerical solution. Any model written using the extension is more understandable and maintainable compared to using the manual discretization.

Conclusion

The presented model of breathing while buried in an avalanche has several limitations. The main purpose of this contribution was to demonstrate the ability of PDEModelica to solve PDE models. This enhancement is documented on the advection equation and then the advection-diffusion equation modelling breathing in snow after the avalanche burial. PDEModelica was able to successfully express these example models and the extended OpenModelica was able to solve them. Nevertheless the project is not finished and more work should be done. Automatic extrapolation on boundaries must be implemented. Both PDEModelica language extension and its implementation in OpenModelica have to be yet tested thoroughly on several different models and by comparison of results with reference PDE simulation tools.

References

- Courant, R., Friedrichs, K., Lewy, H., 1967. On the Partial Difference Equations of Mathematical Physics. IBM J. Res. Dev. 11, 215–234.
- Dshabarow, F., Cellier, F.E., Zimmer, D., Dshabarow, F., Com, C., Ch, I.E., 2007. Support for Dymola in the modeling and simulation of physical systems with distributed parameters. In: Proceedings of the 6th International Modelica Conference. pp. 683–690.
- Fritzson, P., 2015. Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical

- Approach. John Wiley & Sons.
- Margid, J., Beidleman, N., Harmston, C., 1998. O₂ and CO₂ levels with the Black Diamond AvaLung during human snow burials lasting up to one hour. Proceedings of the.
- Mateják, M., Kofránek, J., 2015. Physiomodel - an integrative physiology in Modelica. In: 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC).
ieeexplore.ieee.org, pp. 1464–1467.
- McIntosh, S.E., Grissom, C.K., Olivares, C.R., Kim, H.S., Tremper, B., 2007. Cause of death in avalanche fatalities. Wilderness Environ. Med. 18, 293–297.
- Radwin, M.I., Grissom, C.K., Scholand, M.B., Harmston, C.H., 2001. Normal oxygenation and ventilation during snow burial by the exclusion of exhaled carbon dioxide. Wilderness Environ. Med. 12, 256–262.
- Roubík, K., Sieger, L., Sykora, K., 2015. Work of Breathing into Snow in the Presence versus Absence of an Artificial Air Pocket Affects Hypoxia and Hypercapnia of a Victim Covered with Avalanche Snow: A Randomized Double Blind Crossover Study. PLoS One 10, e0144332.
- Saldamli, L., 2006. PDEModelica A High-Level Language for Modeling with Partial Differential Equations (PhD Thesis.). Linkopings universitet.
- Schiesser, W.E., 2012. The Numerical Method of Lines: Integration of Partial Differential Equations. Elsevier.
- Stavåker, K., Fritzson, P. et al, 2014. PDE Modeling With Modelica Via FMI Import Of Hiflow3 C++ Components With Parallel Multi-core Simulations. Proceedings of the 55th Scandinavian Conference on Simulation and Modeling.
- Strikwerda, J.C., 2004. Finite Difference Schemes and Partial Differential Equations. SIAM.
- Techel, F., Jarry, F., Kronthaler, G., Mitterer, S., Nairz, P., Pavšek, M., Valt, M., Darms, G., 2016. Avalanche fatalities in the European Alps: long-term trends and statistics. Geogr. Helv. 71, 147–159.

Multirotor Aerial Vehicle modeling in Modelica

Muhamed Kuric¹ Nedin Osmic¹ Adnan Tahirovic¹

¹Department of Automatic Control and Electronics

Faculty of Electrical Engineering

University of Sarajevo

Zmaja od Bosne bb, 71000 Sarajevo, Bosnia and Herzegovina

{muhamed.kuric, nedim.osmic, adnan.tahirovic}@etf.unsa.ba

Abstract

This paper presents a generalized Multirotor Aerial Vehicle (MAV) modeling framework which includes rigid body dynamics, gyroscopic effect and motor dynamics. We illustrate how this model can be used to derive any MAV platform constructed with an arbitrary number of rotors by using the quadrotor case as an example. Based on this result, we design the first Modelica-based MAV simulator. We validate the proposed design by using a simple altitude and attitude stabilization control system through a Modelica simulation setup.

Keywords: *Multirotor Aerial Vehicle, Modeling, Modelica*

1 Introduction

Technological advancements in recent years, including the miniaturization in battery, sensor and actuation technologies, as well as the availability of low cost high performance computing boards have enabled the genesis of intelligent autonomous flying machines. The most popular class of this machines are the so-called Multirotor Aerial Vehicles (MAVs) which represent motorized rotorcrafts that have favourable dynamical properties and can achieve small geometries. MAVs and especially the quadrotor configuration are now the de facto standard research platforms for aerial robotics with many potential applications including search and rescue in indoor and outdoor environments (Tomic et al., 2012), precision agriculture (Zhang and Kovacs, 2012), aerial construction (Lindsey et al., 2011; Willmann et al., 2012), inspection and maintenance (Mellinger et al., 2011; Jimenez-Cano et al., 2013), environmental monitoring (Alexis et al., 2009), exploration and mapping (Fraundorfer et al., 2012), aerial transportation (Michael et al., 2011; Mellinger et al., 2013) and swarming (Kushleyev et al., 2013).

Due to this growing interest, there have emerged multiple MAV simulation platforms mainly in MATLAB and ROS with notable examples being (Bresciani, 2008) and (Furrer et al., 2016), respectively. Both provide simulation for MAV dynamics (with the former covering only the quadrotor case) and sensors, and the latter having a less user-friendly interface via pure code and configuration. To the best of our knowledge, there are no existing MAV simulation platforms within the Modelica commu-

nity.

Our paper gives a simple way of deriving a proper dynamical model for a MAV constructed with an arbitrary number of rotors by using a generalized MAV model. Based on this paradigm, we also present a Modelica simulator that can be used for multirotor aerial vehicles. To the best of the authors' knowledge, this is the first Modelica-based MAV simulator available within the Modelica community.

The remainder of the paper is organized as follows. Section 2 describes how generalized MAV dynamics can be derived and how an appropriate dynamical model can be extracted for a quadrotor based MAV. In Section 3, we describe necessary classes to design the Modelica-based simulator for MAVs, while in Section 4, we validate the results throughout a simple altitude and attitude stabilization control system. Concluding remarks are presented in Section 5.

2 MAV dynamics

A large number of papers address MAV modeling putting the focus mostly on the quadrotor case. Noteworthy classical contributions include (Altug et al., 2002), (Hamel et al., 2002), (Pounds et al., 2002) and (Bouabdallah et al., 2004a). More recent examples of very detailed quadrotor and octotorotor modeling are presented in (Bangura and Mahony, 2012) and (Osmic et al., 2016), respectively. To the best of our knowledge, one of the most complete work regarding MAVs can be found in (Mahony et al., 2012), where the authors have derived MAV dynamics, included advanced state estimation, control and motion planning algorithms and therefore provided full system autonomy.

In this section, we will describe the dynamical model of the quadrotor, which is frequently considered to be the standard research platform for MAVs due to its simple construction and purposeful functionality. We use the results and nomenclature from (Osmic et al., 2016) and show that only minor changes are necessary to apply the final octocopter model presented in (Osmic et al., 2016) to any MAV, including also the quadrotor case.

2.1 MAV rigid body dynamics

In order to model the dynamics of any mobile robot it is common to define two frames of reference. A body fixed

frame $\{o\}$ is attached to the robots center of mass and all sensory data is measured with respect to this frame, while a ground fixed frame $\{g\}$ is used to define workspace goals in a intuitive and user-friendly manner. The body fixed and ground fixed frame represent right-handed Cartesian coordinate systems and are usually referred to as the local and global coordinate system, respectively.

Workspace goals can be defined in terms of global position coordinates x , y and z and orientation coordinates ϕ , θ and ψ (see Fig. 1), where positive directions of ϕ , θ and ψ are chosen according to the right-hand rule. Therefore, the position vector $\mathbf{x} = [x \ y \ z]^T$ and the orientation vector $\Psi = [\phi \ \theta \ \psi]^T$ can completely determine the vehicle's location in the workspace. As shown in Fig. 2, the local coordinates are described by the linear velocities u , v and w and the angular velocities P , Q , R . The positive directions of the angular velocities P , Q and R are also chosen according to the right-hand rule and therefore coincide with the positive directions of ϕ , θ and ψ . Both linear and angular velocity coordinates can also be expressed in compact vector form as $\mathbf{v} = [u \ v \ w]^T$ and $\mathbf{P} = [P \ Q \ R]^T$, respectively.

Forces and torques which act on a MAV are shown in Fig. 3. The thrust T is a force that acts towards the positive direction of the Z axis of the local coordinate system $\{o\}$, while the force G represents the gravitational force acting towards the negative direction of the Z_B axis of the global coordinate system $\{g\}$. τ_x , τ_y and τ_z represent the torques that move the vehicle around the X , Y and Z axes of the local coordinate system, respectively, and can be compactly denoted as $\boldsymbol{\tau} = [\tau_x \ \tau_y \ \tau_z]^T$. Their positive direction is also chosen to coincide with the positive directions of the angular velocities P , Q and R .

We can now describe the rigid body dynamics of any

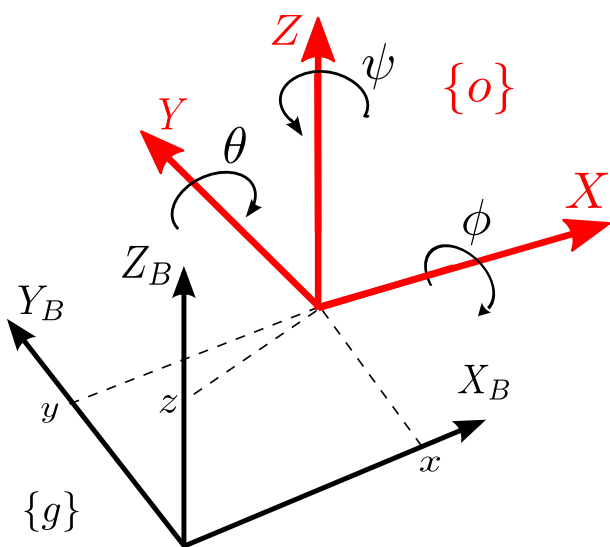


Figure 1. Global coordinates

MAV in accordance to the results presented in (Osmic et al., 2016). The kinematic model of the linear motion is given as

$$\dot{\mathbf{x}} = \mathbf{R}(\phi, \theta, \psi) \mathbf{v}, \quad (1)$$

where $\mathbf{R}(\phi, \theta, \psi)$ is the total rotation matrix which for the ZYX Euler convention has the form

$$\mathbf{R}(\phi, \theta, \psi) = \mathbf{R}(Z, \psi) \mathbf{R}(Y, \theta) \mathbf{R}(X, \phi) = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}, \quad (2)$$

and the elementary rotation matrices $\mathbf{R}(Z, \psi)$, $\mathbf{R}(Y, \theta)$ and $\mathbf{R}(X, \phi)$ are defined as

$$\mathbf{R}(X, \phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix}, \quad (3)$$

$$\mathbf{R}(Y, \theta) = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix}, \quad (4)$$

$$\mathbf{R}(Z, \psi) = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (5)$$

The kinematic model of the angular motion can be described by

$$\dot{\Psi} = \mathbf{R}_A^{-1}(\phi, \theta, \psi) \mathbf{P}, \quad (6)$$

where the matrix $\mathbf{R}_A^{-1}(\phi, \theta, \psi)$ for the ZYX Euler convention is

$$\mathbf{R}_A^{-1}(\phi, \theta, \psi) = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{bmatrix}. \quad (7)$$

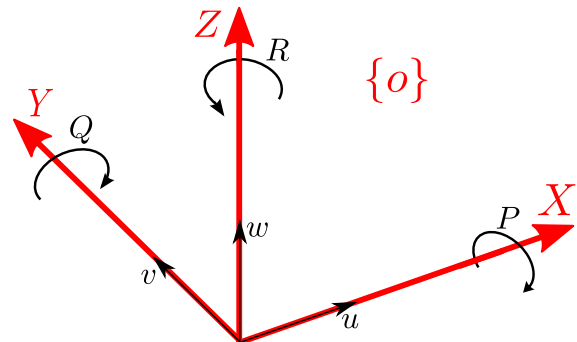


Figure 2. Local coordinates

The dynamic model of the linear motion can be represented by the following equation

$$\dot{\mathbf{v}} = \begin{bmatrix} 0 \\ 0 \\ \frac{T}{m_o} \end{bmatrix} + g \begin{bmatrix} s_\theta \\ -s_\phi c_\theta \\ -c_\phi c_\theta \end{bmatrix} - \mathbf{S}\mathbf{v}, \quad (8)$$

where m_o is the total mass of the MAV and the matrix \mathbf{S} is formed as

$$\mathbf{S} = \begin{bmatrix} 0 & -R & Q \\ R & 0 & -P \\ -Q & P & 0 \end{bmatrix}. \quad (9)$$

Finally, the dynamic model of the angular motion can be caught with

$$\dot{\mathbf{P}} = \mathbf{J}^{-1}(\boldsymbol{\tau} - \mathbf{S}\mathbf{J}\mathbf{P}), \quad (10)$$

where \mathbf{J} is a 3×3 matrix representing the inertia tensor of the MAV.

2.2 Quadrotor modeling

To tailor the previously derived MAV model to the quadrotor case we need to derive the inertia tensor \mathbf{J} , and define the thrust T and the torque vector $\boldsymbol{\tau}$. Since all of these quantities depend on the MAV's geometry, we consider a quadrotor case shown in Fig. 4 along with its simplified geometry illustrated in Fig. 5, where the length of the four arms is l , a hardware support plate is modeled as solid sphere of mass M having a radius r , and the four motors constructed with fixed pitch propellers are modelled as particles with mass m .

The axes of the local coordinate system, as shown in Fig. 4, represent principal axes of inertia, where the inertia

tensor matrix has the diagonal form

$$\mathbf{J} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}, \quad (11)$$

and I_{xx} , I_{yy} , I_{zz} being the moments of inertia around the X , Y and Z axes of the local coordinate system, respectively. These components can be derived via the Huygens-Steiner theorem (Morin, 2008) as

$$I_{xx} = I_{yy} = \frac{2Mr^2}{5} + 2ml^2 \quad (12)$$

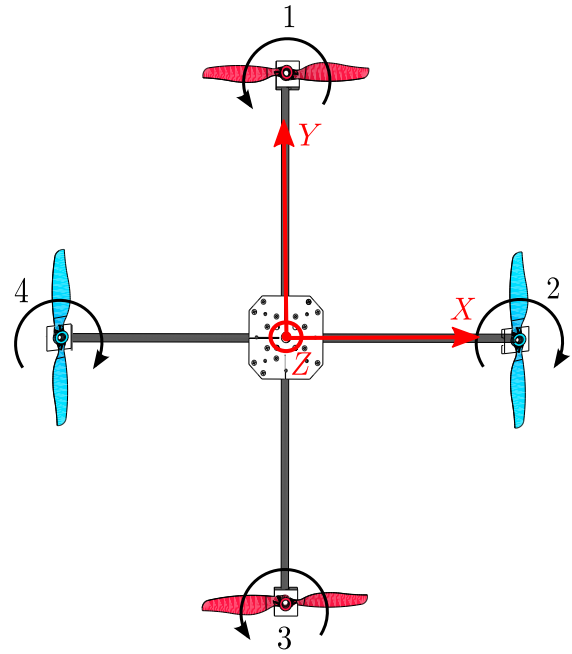


Figure 4. Quadrotor geometry

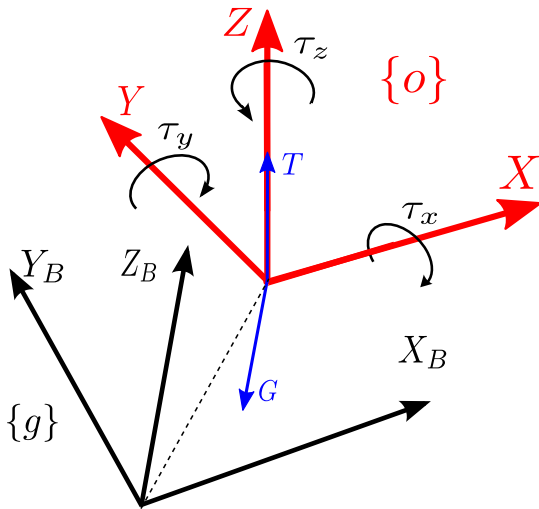


Figure 3. Forces and torques acting on the system

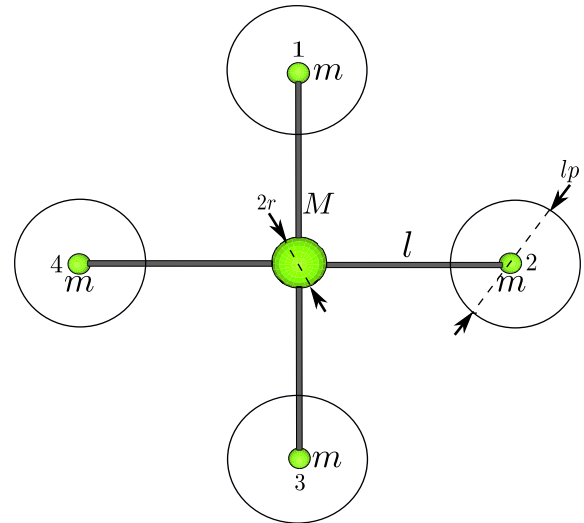


Figure 5. Quadrotor simplified geometry

and

$$I_{zz} = \frac{2Mr^2}{5} + 4ml^2. \quad (13)$$

In order to derive the thrust T and the τ_x and τ_y components of the torque vector $\boldsymbol{\tau}$, we will consider the rotor forces acting on the quadrotor system as depicted in Fig. 6. Thus T , τ_x and τ_y are given as follows

$$T = F_1 + F_2 + F_3 + F_4, \quad (14)$$

$$\tau_x = l(F_1 - F_3), \quad (15)$$

$$\tau_y = l(F_4 - F_2). \quad (16)$$

In accordance to the work presented in (Mahony et al., 2012), the rotor forces F_i ($i = \overline{1..4}$) can be approximated as

$$F_i = b\Omega_i^2 \quad (i = \overline{1..4}), \quad (17)$$

where $b \left[\frac{Ns^2}{rad^2} \right]$ is the rotor thrust constant and $\Omega_i \left[\frac{rad}{s} \right]$ is the angular velocity of the i -th rotor. Combining eqs. (14), (15), (16) and (17) yields

$$T = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2), \quad (18)$$

$$\tau_x = bl(\Omega_1^2 - \Omega_3^2) \quad (19)$$

and

$$\tau_y = bl(\Omega_4^2 - \Omega_2^2). \quad (20)$$

The torque τ_z is a consequence of Newton's third law and can be formed as

$$\tau_z = -M_1 + M_2 - M_3 + M_4, \quad (21)$$

where M_i ($i = \overline{1..4}$) is the counter induced torque of the i -th rotor. According to (Mahony et al., 2012) the counter torque can approximated as

$$M_i = d\Omega_i^2 \quad (i = \overline{1..4}), \quad (22)$$

where $d \left[\frac{Nms^2}{rad^2} \right]$ is the rotor drag constant. Combining equations (21) and (22) yields

$$\tau_z = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2). \quad (23)$$

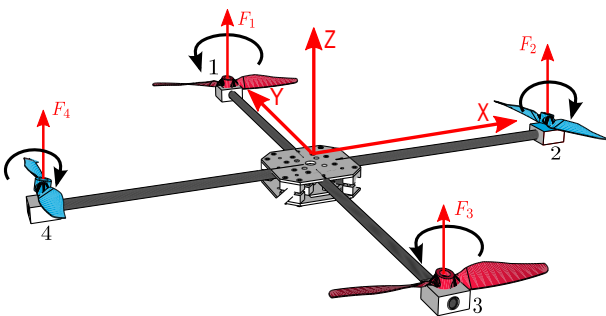


Figure 6. Rotor forces acting on the quadrotor system

Finally, we can represent the system actuation via matrix equation

$$\begin{bmatrix} T \\ \boldsymbol{\tau} \end{bmatrix} = \mathbf{A}\boldsymbol{\Omega}_s, \quad (24)$$

where \mathbf{A} is the actuation matrix

$$\mathbf{A} = \begin{bmatrix} b & b & b & b \\ bl & 0 & -bl & 0 \\ 0 & -bl & 0 & bl \\ -d & d & -d & d \end{bmatrix}, \quad (25)$$

and $\boldsymbol{\Omega}_s$ is the squared rotor velocity vector defined as

$$\boldsymbol{\Omega}_s = [\Omega_1^2 \quad \Omega_2^2 \quad \Omega_3^2 \quad \Omega_4^2]^T. \quad (26)$$

It is evident from this result that any MAV can be modelled by choosing the appropriate inertia tensor \mathbf{J} and actuation matrix \mathbf{A} as parameters, and picking the squared rotor velocity vector $\boldsymbol{\Omega}_s$ of the right size as a system input. For any MAV constructed with $n \geq 4$ rotors, the actuation matrix has the dimension $4 \times n$ and the squared rotor velocity vector $\boldsymbol{\Omega}_s$ has the length n .

Moreover, we can include the gyroscopic effect in the dynamic model of the angular motion given by eq. (10) as

$$\dot{\mathbf{P}} = \mathbf{J}^{-1} \left(\boldsymbol{\tau} - \mathbf{S}\mathbf{J}\mathbf{P} - \mathbf{S} \begin{bmatrix} 0 \\ 0 \\ I_{zzm}W_g \end{bmatrix} \right), \quad (27)$$

where I_{zzm} is the rotor moment of inertia and W_g is the gyroscopic term given as

$$W_g = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \quad (28)$$

for the quadrotor case. In order to generalize the gyroscopic term for any MAV configuration, it is more appropriate to choose the rotor velocity vector $\boldsymbol{\Omega}$

$$\boldsymbol{\Omega} = [\Omega_1 \quad \Omega_2 \quad \Omega_3 \quad \Omega_4]^T \quad (29)$$

as system input and express the gyroscopic term as

$$W_g = \text{sign}(\mathbf{A}_z)\boldsymbol{\Omega}, \quad (30)$$

where \mathbf{A}_z is the fourth row of the actuation matrix \mathbf{A} , while the squared rotor velocity vector $\boldsymbol{\Omega}_s$ can easily be computed by calculating the element-wise square of the vector $\boldsymbol{\Omega}$.

2.3 Motor dynamics

Each rotor of a quadrotor MAV is driven by a DC motor. Therefore, in order to obtain a precise MAV model, it is inevitable to include the motor dynamics to address effects like motor response time, saturation and power consumption. In accordance to the work presented in (Osmic et al., 2016), a simplified model can be used for this purpose which is given by

$$I_{zzm}\dot{\Omega}_i + \frac{K_m K_e}{R}\Omega_i = \frac{K_m}{R}v_i - \tau_{li}, \quad i = \overline{1..4}, \quad (31)$$

where $K_m \left[\frac{Nm}{A} \right]$ is the mechanical motor constant, $K_e \left[\frac{Vs}{rad} \right]$ being the electrical motor constant, R denotes the armature resistance, v_i is the armature voltage, with τ_{li} being the load torque of the i -th motor. The load torque is the aerodynamic drag which can be computed as

$$\tau_{li} = d\Omega_i^2, \quad i = \overline{1..4}. \quad (32)$$

The input voltage of each motor is saturated by the following box constraint

$$0 \leq v_i \leq v_{\max}, \quad i = \overline{1..4} \quad (33)$$

where v_{\max} is the maximum armature voltage, and consequently the angular velocity of each rotor is also box constrained by

$$0 \leq \Omega_i \leq \Omega_{\max}, \quad i = \overline{1..4}, \quad (34)$$

where Ω_{\max} is the maximum angular velocity which can be easily computed from the stationary state of the motor dynamic model given by (31).

Finally, the rotor moment of inertia I_{zzm} can be approximately calculated as

$$I_{zzm} = \frac{m_p l_p^2}{12}, \quad (35)$$

where m_p is the mass and l_p being the length of the rotor.

3 Modelica design

In order to provide a greater end-user utilization, we designed the following Modelica blocks / classes:

- MavBase
- MavSimple
- MavFull

The MavBase block, as shown in Fig. 7, is the simplest and it models the rigid body dynamics including the gyroscopic effect covered with eqs. (1), (6), (8) and (27). Its inputs are the generalized forces acting on the system and the outputs are the global coordinates of the system and its derivations.

The MavSimple block, as shown in Fig. 8, extends the MavBase block with the actuation model given by eq. (24)

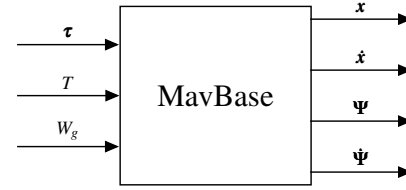


Figure 7. MavBase block

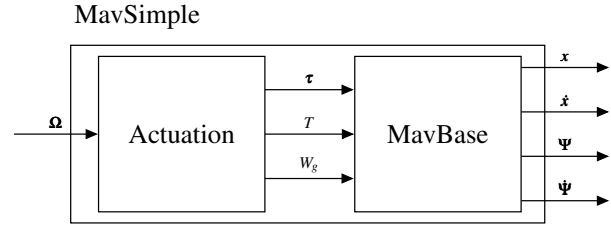


Figure 8. MavSimple block

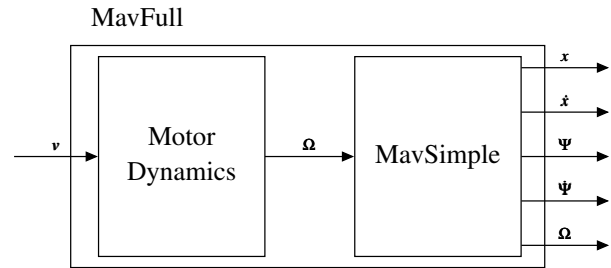


Figure 9. MavFull block

with the input being the angular velocity vector Ω and the outputs being the global coordinates of the system and its derivations.

Finally, the MavFull block, as shown in figure 9, provides the greatest level of detail. It extends the MavSimple block and adds the motor dynamics (31) to the model. The block input is the motor voltage vector \mathbf{v} with the outputs being the global coordinates of the system and its derivations, as well as the angular velocity vector Ω . The angular velocity vector as system output is necessary to provide motor level control possibilities.

The parameters of the blocks are given in Table 1, 2 and 3, and their default values match the AscTec Pelican quadrotor (AscTec, 2016).

Table 1. MavFull block parameters

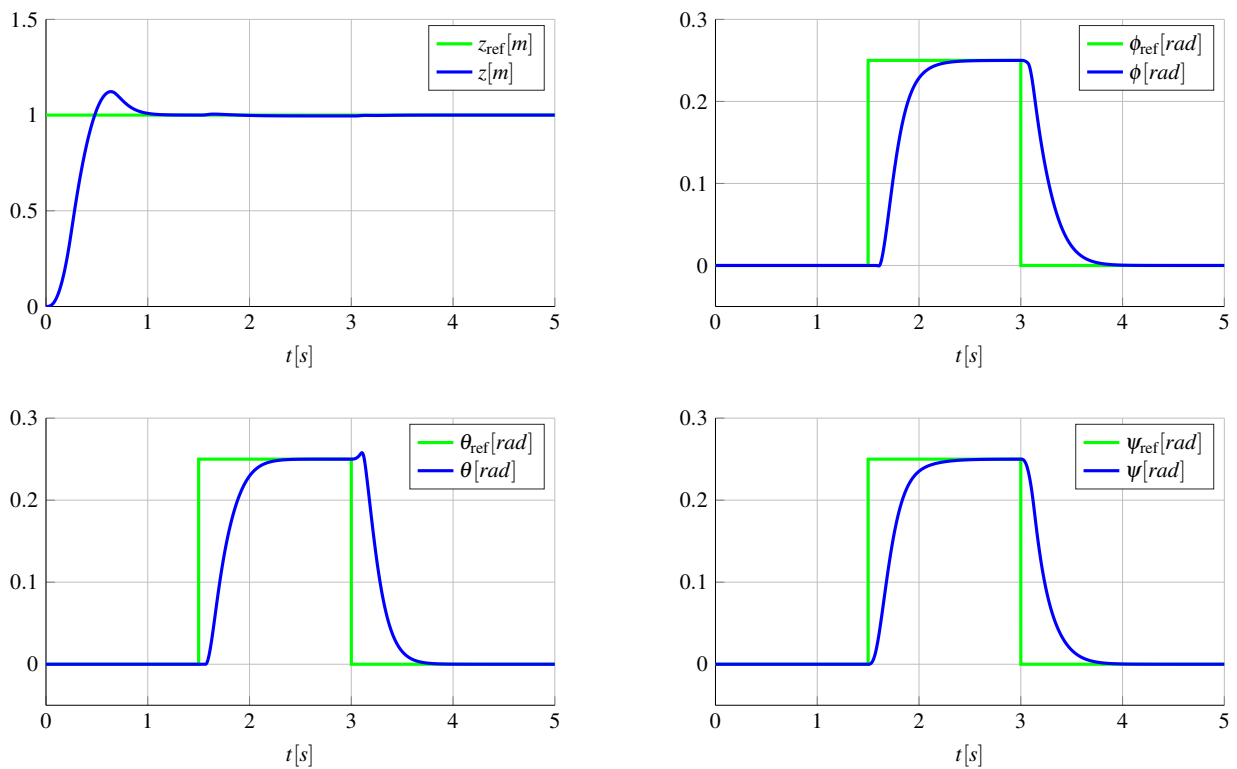
Parameter	Value	Unit	Description
R	0.1107	Ω	Resistance
K_m	0.01	$\frac{Nm}{A}$	Motor size constant
K_e	0.01	$\frac{Vs}{rad}$	Motor velocity constant
v_{\max}	11.1	V	Maximum voltage
Ω_0	569.3572	$\frac{rad}{sec}$	Initial angular velocity

Table 2. MavBase block parameters

Parameter	Value	Unit	Description
m_o	1.32	kg	MAV total mass
J	$\begin{bmatrix} 0.0128 & 0 & 0 \\ 0 & 0.0128 & 0 \\ 0 & 0 & 0.0239 \end{bmatrix}$	kgm^2	Inertia tensor
I_{zzm}	$4.3011 \cdot 10^{-5}$	kgm^2	Rotor moment of inertia

Table 3. MavSimple block parameters

Parameter	Value	Unit	Description
n	4		Input size
b	$9.9865 \cdot 10^{-6}$	$\frac{Ns^2}{rad^2}$	Aerodynamic thrust constant
d	$1.5978 \cdot 10^{-7}$	$\frac{Nms^2}{rad^2}$	Aerodynamic drag constant
A	$\begin{bmatrix} b & b & b & b \\ 0.211 \cdot b & 0 & -0.211 \cdot b & 0 \\ 0 & -0.211 \cdot b & 0 & 0.211 \cdot b \\ -d & d & -d & d \end{bmatrix}$		Actuation matrix


Figure 10. Altitude and attitude control simulation results

4 Simulation results

A simple altitude and attitude control system was designed in order to validate the designed classes. Altitude and attitude control simulation results are presented in Fig. 10. We notice that the system states have been stabilized within 1 second and that only very minor overshoots are present in the altitude z and the pitch θ .

The simulation example shows that the control results are very satisfactory, in particular the system suffers only a minor loss in altitude during the challenging reference orientation maneuver, which can be considered excellent control behaviour. Additionally, the simulation results are very similar to those obtained in (Bouabdallah et al., 2004b) and (Osmic et al., 2016) which suggests that the model derivation in this paper is correct.

5 Conclusion

This paper described how a generalized MAV modeling framework can be used to obtain any MAV model. A quadrotor based MAV was presented as an example, and the final model was formed by using its rigid body dynamics, the gyroscopic effect that influences the vehicles motion, and appropriate motor dynamics. To model the dynamics of any given MAV platform, it was shown that is only required to choose adequate parameter values, which correspond to the vehicle of interest, and inject them into the generalized MAV model.

Based on the presented generalized MAV model derivation, we have designed the following Modelica classes: MavBase, MavSimple and MavFull. MavBase represents the rigid body dynamics of the MAV including the gyroscopic effect. MavSimple extends the MavBase class and adds system actuation, while MavFull extends MavSimple with motor dynamics. These classes can be used to simulate the dynamic behaviour of any MAV within Modelica to any required level of detail, and thus providing similar functionalities as the Gazebo simulator RotorS (Furrer et al., 2016) which is frequently used for this purposes, but with a more user friendly interface.

Finally, we have validated the designed Modelica simulator through a simple altitude and attitude stabilization control system. Namely, we have obtained very similar control results like those currently present in the state of the art, which suggests that the generalized model derived and the MAV simulator designed in this paper are correct.

References

- K Alexis, G Nikolakopoulos, A Tzes, and L Dritsas. Coordination of helicopter UAVs for aerial forest-fire surveillance. In *Applications of intelligent control to engineering systems*, pages 169–193. Springer, 2009.
- Erdinc Altug, James P Ostrowski, and Robert Mahony. Control of a quadrotor helicopter using visual feedback. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 72–77. IEEE, 2002. doi:10.1109/ROBOT.2002.1013341.
- AscTec. Ascending technologies, gmbh, 2016. URL <http://www.asctec.de/>.
- Moses Bangura and Robert Mahony. Nonlinear dynamic modeling for high performance control of a quadrotor. In *Australasian conference on robotics and automation*, pages 1–10, 2012.
- Samir Bouabdallah, Pierpaolo Murrieri, and Roland Siegwart. Design and control of an indoor micro quadrotor. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pages 4393–4398. IEEE, 2004a. doi:10.1109/ROBOT.2004.1302409.
- Samir Bouabdallah, Andre Noth, and Roland Siegwart. PID vs LQ control techniques applied to an indoor micro quadrotor. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2451–2456. IEEE, 2004b.
- Tammaso Bresciani. Modelling, identification and control of a quadrotor helicopter. *MSc Theses*, 2008.
- Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor MAV. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4557–4564. IEEE, 2012.
- Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. RotorS-A Modular Gazebo MAV Simulator Framework. In *Robot Operating System (ROS)*, pages 595–625. Springer, 2016.
- Tarek Hamel, Robert Mahony, Rogelio Lozano, and James Ostrowski. Dynamic modelling and configuration stabilization for an x4-flyer. *IFAC Proceedings Volumes*, 35(1):217–222, 2002. doi:10.3182/20020721-6-ES-1901.00848.
- AE Jimenez-Cano, Jesús Martin, Guillermo Heredia, Aníbal Ollero, and R Cano. Control of an aerial robot with multi-link arm for assembly tasks. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4916–4921. IEEE, 2013.
- Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, 35(4):287–300, 2013.
- Quentin Lindsey, Daniel Mellinger, and Vijay Kumar. Construction of cubic structures with quadrotor teams. *Proc. Robotics: Science & Systems VII*, 2011.
- Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE robotics & automation magazine*, 19(3):20–32, 2012. doi:10.1109/MRA.2012.2206474.
- Daniel Mellinger, Quentin Lindsey, Michael Shomin, and Vijay Kumar. Design, modeling, estimation and control for aerial grasping and manipulation. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2668–2673. IEEE, 2011.

- Daniel Mellinger, Michael Shomin, Nathan Michael, and Vijay Kumar. Cooperative grasping and transport using multiple quadrotors. In *Distributed autonomous robotic systems*, pages 545–558. Springer, 2013.
- Nathan Michael, Jonathan Fink, and Vijay Kumar. Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, 30(1):73–86, 2011.
- David Morin. *Introduction to classical mechanics: with problems and solutions*. Cambridge University Press, 2008.
- Nedim Osmic, Muhamed Kuric, and Ivan Petrovic. Detailed octocopter modeling and PD control. In *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*, pages 2182–2189, 2016.
- Paul Pounds, Robert Mahony, Peter Hynes, and Jonathan M Roberts. Design of a four-rotor aerial robot. In *Proceedings of the 2002 Australasian Conference on Robotics and Automation (ACRA 2002)*, pages 145–150. Australian Robotics & Automation Association, 2002.
- Teodor Tomic, Korbinian Schmid, Philipp Lutz, Andreas Domel, Michael Kassecker, Elmar Mair, Iris Lynne Grix, Felix Ruess, Michael Suppa, and Darius Burschka. Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. *IEEE robotics & automation magazine*, 19(3):46–56, 2012.
- Jan Willmann, Federico Augugliaro, Thomas Cadalbert, Raffaello D’Andrea, Fabio Gramazio, and Matthias Kohler. Aerial robotic construction towards a new field of architectural research. *International journal of architectural computing*, 10(3):439–459, 2012.
- Chunhua Zhang and John M Kovacs. The application of small unmanned aerial systems for precision agriculture: a review. *Precision agriculture*, 13(6):693–712, 2012.

Rotating Machinery Library for Diagnosis

Tatsuro Ishibashi¹ Bing Han² Tadao Kawai³

¹Meidensha Corporation, Japan, ishibashi-tat@mb.meidensha.co.jp

^{2,3}Department of Mechanical & Physical Engineering, Osaka City University, Japan,
{han,kawai}@mech.eng.osaka-cu.ac.jp

Abstract

This paper presents our new rotating machinery library. Diagnosing the complex system accurately based on stochastic method requires an enormous amount of data, both with and without faults. Acquiring operation data with all kinds of faults for each components is very hard and costly. To generate data for rotating machinery diagnosis, we developed rotating machinery library using Modelica. It provides the basic components such as rotor, shaft, bearing, coupling, housing and support. Its component models are implemented on basis of rotor dynamics theory. This library makes it possible accessing rotating machinery operation data with various faults such as unbalanced rotor, shaft bending and ball bearing faults. To validate our models, we compared both Modelica simulation and experiment with a rotor kit as a test case.

Keywords: *Rotating Machinery, Vibration, Diagnosis*

1 Introduction

Preventive maintenance has been the main stay of industry for a long time. Recently, IoT (Internet of Things) and Industry4.0 have become very popular to manage and control a system such as manufacturing system. These have naturally increased the focus on Condition-Based Maintenance (CBM). IoT makes it possible monitoring the system state on time and accumulating a large amount of data. So many sensors and sensor network are attached to each component of a system for the purpose of data acquisition. By collecting and analyzing these acquired data, it makes feasible to manage a system efficiently or detect problem in a system with high accuracy at early stage.

Although this concept is very important, there are many difficulties in measurement. It is not always possible that we attach sensors where we would like to measure. Neither is it possible that we measure significant features of a system due to the lack of sensors.

By stochastic approach such as machine learning, fault detection of complex system accurately requires an enormous amount of data, both with and without faults to determine the threshold of signal amplitude. As a method of grasping the operating states with faults, it is conceivable to collect operation data by embedding the

damaged part into the actual machine or continuing to operate it until it gets damaged. If the equipment is large, the lifetime is long, or the equipment is expensive, it is difficult to accumulate data. Rotating machinery equipment apply to the above. IoT will help us collect and acquire data. Still, accumulating data with all kinds of faults for each components is very hard and costly. It is desirable to analyze the state when faults occur in the rotating machinery. Some simple fault cases are schematized and modeled. It is possible to generate data of rotating machinery equipment with faults such as unbalanced rotor, shaft bending, ball bearing faults and misalignment of coupling by simulation.

Modelica is an object-oriented, declarative, multi-domain modeling language for component-oriented modeling of complex systems. It is a powerful tool which simulates a complex physical system. Almost all design parameters such as shape and rigidity are easily set to a model and a variable behavior of a certain component in a system is easily obtained. By building Modelica cyber system, we can also obtain physical quantities and features of components which are inaccessible in a real physical system. A model-based diagnosis and design approach including fault mode with Modelica has recently reported (Klenk *et al*, 2014; Minhas *et al*, 2014).

Hence, we focus on developing rotating machinery library based on well-established rotor dynamics theory using Modelica. It provides the basic components such as rotor, shaft, bearing, coupling, housing and support. Our final goal is generation of training data including unmeasurable quantities for diagnosis by statistical classification algorithms.

To validate our models, we compared both Modelica simulation and experiment with a rotor kit as a test case. We calibrated the fault related parameter embedded in Modelica model so that simulation results were in good agreement with the experiment. Data acquisition from physical system was done through COMEDI (CONtrol and MEasurement Device Interface for Linux / RTAI).

The followings show our new rotating machinery library, the example of generated data from this library and validation of this library with a rotor kit as a test case.

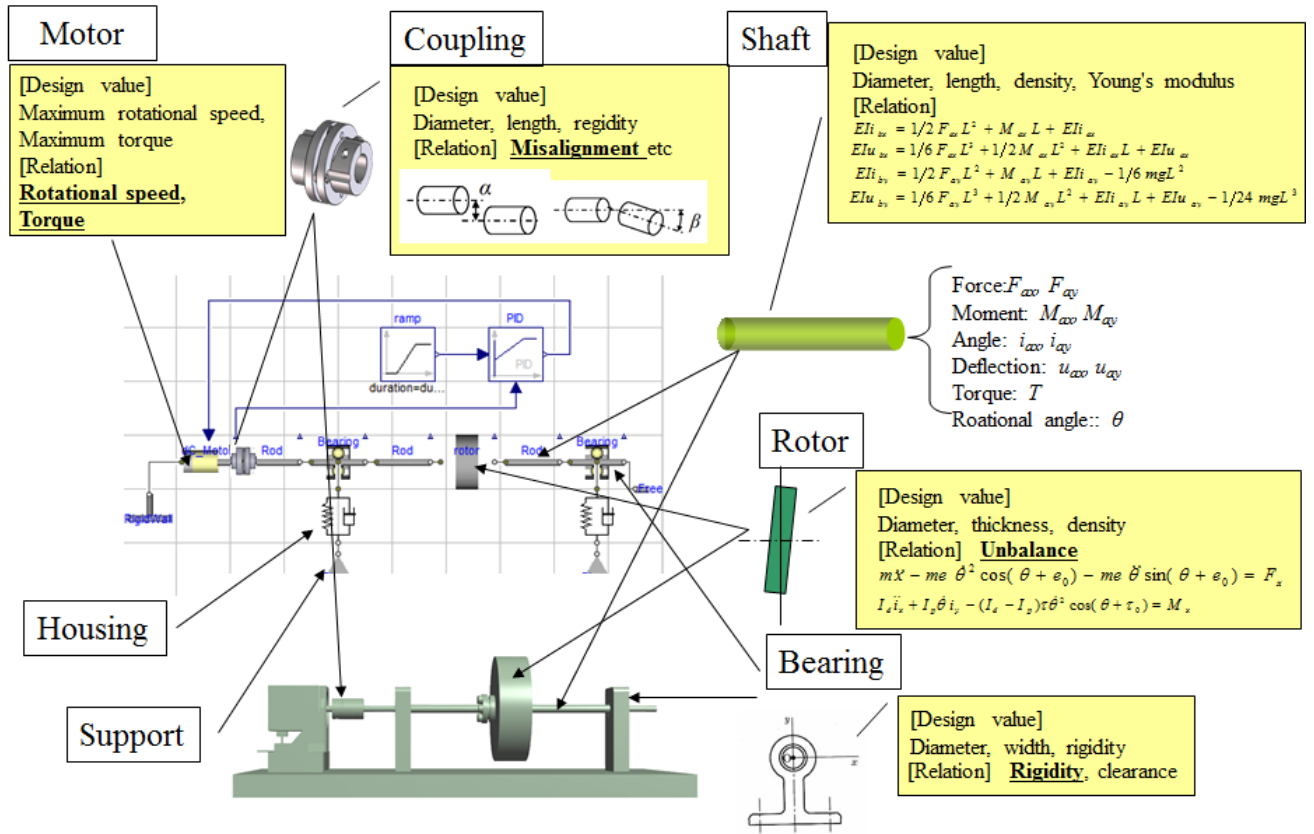


Figure 1. An example of a rotating machinery system. The upper figure describes our new library components of Modelica cyber system. The lower figure is corresponding physical system (The image is designed by CAD tool).

2 Rotating Machinery Library

Generally, a rotating machinery system has many problems. Static and dynamic unbalance of a rotor, a bend of a shaft, rigidity of housing, damaged ball bearing. These failures are included in our rotating machinery library for the diagnosis. Specifications of a motor, a coupling, a shaft, a rotor and bearings are decided at design process and set as parameters of the model. Each component shown in Figure 1 is built based on well-established rotor dynamics theory (Ishida Yamamoto, 2012; Matsushita *et al*, 2017).

The basic flange has 5 DOF (degree of freedom), consisting of 4 DOF (two dimensional deflection and slope) for transverse vibration of the rotor system and 1 DOF (angle) for torsional vibration, neglecting axial vibration. Figure 2 describes the flange. The followings are 10 flange variables. 5 potential variables are

u_x : Deflection of horizontal direction,
 u_y : Deflection of vertical direction,
 i_x : Deflection angle of horizontal direction,
 i_y : Deflection angle of vertical direction,
 θ : Rotational angle.

Corresponding flow variables are

F_x : Force of horizontal direction,
 F_y : Force of vertical direction,
 M_x : Moment of horizontal direction,
 M_y : Moment of vertical direction,
 T : Rotational Torque.

2.1 Rotor

In our library, the rotor is considered as a single mass in the form of a point mass, a rigid disc or a long rigid shaft. The rotor model is followed as 4 DOF Jeffcott rotor model. The forces and moments are given from next component through connector. The static unbalance model equations are following.

$$m \ddot{u}_x - m e \ddot{\theta}^2 \cos(\theta + e_0) - m e \ddot{\theta} \sin(\theta + e_0) = F_x \quad (1)$$

$$m \ddot{u}_y - m e \ddot{\theta}^2 \sin(\theta + e_0) + m e \ddot{\theta} \cos(\theta + e_0) + m g = F_y \quad (2)$$

The dynamic unbalance model equations are following.

$$I_d \ddot{i}_x + I_p \ddot{\theta} i_y - (I_d - I_p) \tau \ddot{\theta}^2 \cos(\theta + \tau_0) = M_x \quad (3)$$

$$I_d \ddot{i}_y - I_p \ddot{\theta} i_x - (I_d - I_p) \tau \ddot{\theta}^2 \sin(\theta + \tau_0) = M_y \quad (4)$$

Also, dynamic load torque effect is considered.

$$I_p \ddot{\theta} = F_x e \sin(\theta + e_0) - F_y e \cos(\theta + e_0) + T \quad (5)$$

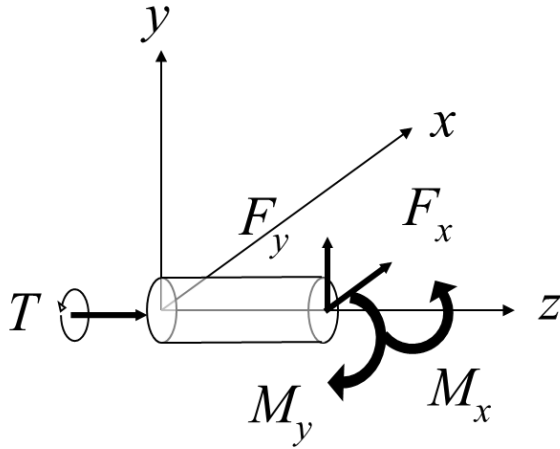


Figure 2. Flow variables of flange.

Here,

$\dot{}$: Time-derivative operator (i.e. $\frac{d}{dt}$),

m : Rotor mass,

e : Eccentricity,

e_0 : Initial phase of eccentricity,

τ : Slope of dynamic unbalance,

τ_0 : Initial phase of dynamic unbalance,

I_d : Diametral moment of inertia,

I_p : Polar moment of inertia.

2.2 Shaft

The shaft is considered as flexible (elastic) shaft. It is a linear elastic beam, uniform loading. The force and deflection relationships between two flanges follow.

$$EI \begin{pmatrix} i_{bx} \\ i_{by} \end{pmatrix} = \frac{L^2}{2} \begin{pmatrix} F_{ax} \\ F_{ay} \end{pmatrix} + L \begin{pmatrix} M_{ax} \\ M_{ay} \end{pmatrix} + EI \begin{pmatrix} i_{ax} \\ i_{ay} \end{pmatrix} + \begin{pmatrix} 0 \\ -\frac{mgL^2}{6} \end{pmatrix} \quad (6)$$

$$EI \begin{pmatrix} u_{bx} \\ u_{by} \end{pmatrix} = \frac{L^2}{6} \begin{pmatrix} F_{ax} \\ F_{ay} \end{pmatrix} + \frac{L}{2} \begin{pmatrix} M_{ax} \\ M_{ay} \end{pmatrix} + EIL \begin{pmatrix} i_{ax} \\ i_{ay} \end{pmatrix} + EI \begin{pmatrix} u_{ax} \\ u_{ay} \end{pmatrix} + \begin{pmatrix} 0 \\ -\frac{mgL^3}{24} \end{pmatrix} \quad (7)$$

Here,

m : Shaft mass,

L : Shaft length,

E : Young's modulus,

I : Second moment of area,

g : Constant of gravitation,

Subscript a is left flange, and b is right flange.

2.3 Bearing

Bearing models have deflection rigidities in series and deflection angle rigidities in parallel. The bearing

models also have damping factor. In addition to the above model, a bearing damage models on inner and outer ring are modeled by applying impulsive force. Bearing force relationships is following.

$$\begin{pmatrix} F_x \\ F_y \end{pmatrix} = k \begin{pmatrix} \Delta u_x \\ \Delta u_y \end{pmatrix} + c \begin{pmatrix} \frac{d\Delta u_x}{dt} \\ \frac{d\Delta u_y}{dt} \end{pmatrix} + F_n P \begin{pmatrix} \cos\varphi \\ \sin\varphi \end{pmatrix} \quad (8)$$

Here,

k : Spring constant of bearing force,

c : Damping constant of bearing force,

$\Delta u = u_c - (u_a + u_b)/2$: Difference of deflection,

F_n : Quantity of impulsive force due to ball and ring collision,

P : If collision occurs, $P = 1$, otherwise 0,

φ : Impulsive force angle.

Subscript c is housing or support flange.

We calculate the quantity of collision force using Hertz contact theory. Collision event is written by when statement.

Following is outer ring case.

$$F_n = 1.143m^{0.6}K^{0.4}v^{1.2} \quad (9)$$

$$v = \frac{D+d}{2} \omega_1 \sin\theta_d \quad (10)$$

$$t = 3.128m^{0.4}K^{-0.4}v^{-0.2} \quad (11)$$

Here,

D : Diameter of inner ring,

d : Diameter of ball,

v : Ball collision speed,

Z : Number of balls,

ω : Rotational speed.

ω_1 : Ball orbital motion rotational speed,

t : Contact time,

θ_d : Collision angle.

K : Proportional constant of force.

K is determined by elastic moduli and Poisson's ratio.

$$\omega_1 = \frac{D}{2(D+d)} \omega \quad (12)$$

From relationships above, characteristic frequency of outer and inner ring collision events are described by ball orbital motion rotational speed.

$$f_{outer} = \frac{Z\omega_1}{2\pi} \quad (13)$$

$$f_{inner} = \frac{Z(\omega - \omega_1)}{2\pi} \quad (14)$$

2.4 Coupling

Coupling models have both offset and angular misalignment. The deflections relationships between two flanges follow.

$$u_b - u_a = e(\cos(\theta + e_0) \sin(\theta + e_0)) \quad (15)$$

$$i_b - i_a = a(\cos(\theta + a_0) \sin(\theta + a_0)) \quad (16)$$

Here,

e : Offset misalignment length,

e_0 : Initial phase of offset misalignment,
 a : Angular misalignment length,
 a_0 : Initial phase of angular misalignment.
 Coupling models also have deflection rigidities.

2.5 Housing

Housing model is spring damper against deflection. It doesn't have deflection angle rigidities.

2.6 Support

Support models have simple support, rigid support and free end. Simple support follows the condition that deflections and moments are zero. Rigid support follows the condition that deflections and deflection angles are zero. Free end follows the condition that forces and moments are zero.

2.7 Motor

Motor models are extended from `Modelica.Electrical.Machines.BasicMachines` to be compatible with our library flange.

3 Generation of data with fault

Ball bearing in rotating machinery is very frequently damaged during operation. Inner and outer rings in bearing can be damaged due to over load, corrosion, improper lubrication and installation. Over load or weak lubrication causes friction. As a result of friction, temperature increases, so that oil lost its properties. Also, current flows on bearings, because of voltage difference between stator and rotor does. Oil acts as a dielectric material in condenser. These faults produce small particles in bearing.

Using our library, we generated outer and inner ring damaged ball bearing vibration case data in addition to static unbalance respectively. Figure 3 and 4 are simulation results respectively.

By building Modelica cyber system with the other faults referred in previous section, we can obtain time series data with some faults from Dymola simulation. Using these simulation generated data for training dataset after analysing and signal processing experimental measured data, it is possible making system fault detection algorithm based on machine learning theory (Figure 5). In case of the damaged ball bearing, envelope processing of bearing eigen frequency is required. Our final goal is generation of training data including unmeasurable quantities for diagnosis by statistical classification algorithms.

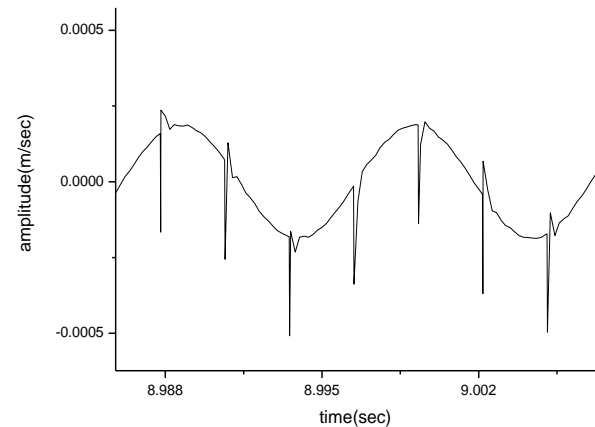


Figure 3. Outer ring fault with static unbalance.

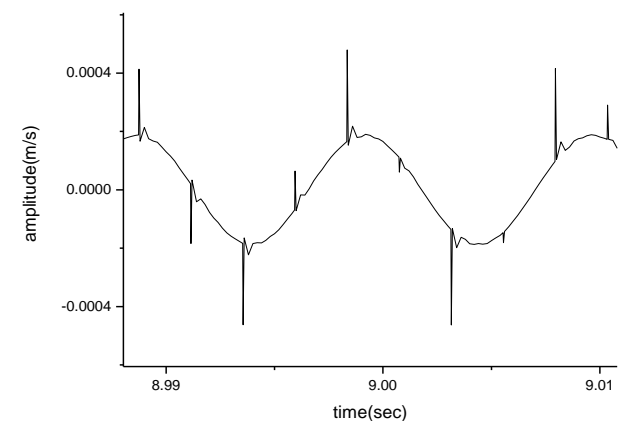


Figure 4. Inner ring fault with static unbalance.

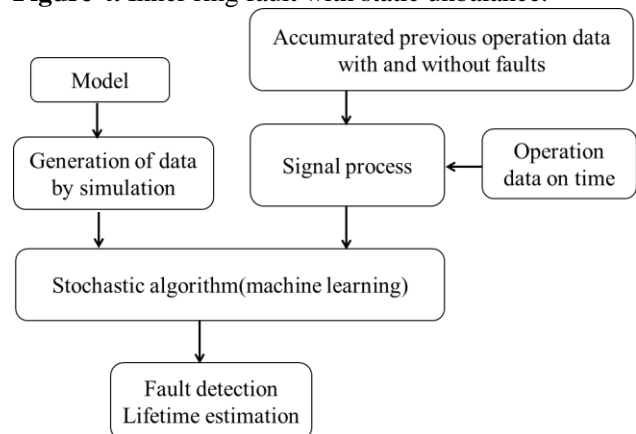


Figure 5. Diagnosis method using simulation data as training dataset.

4 Model Validation

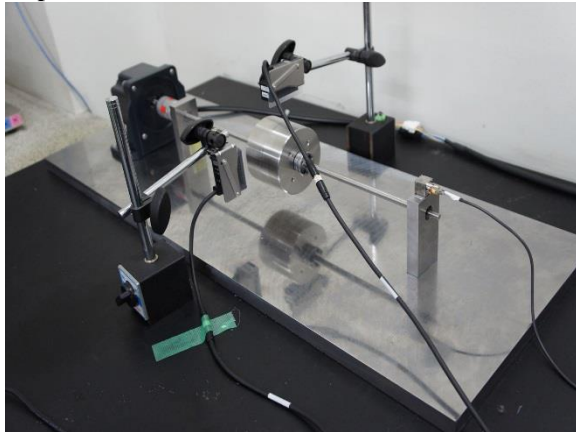
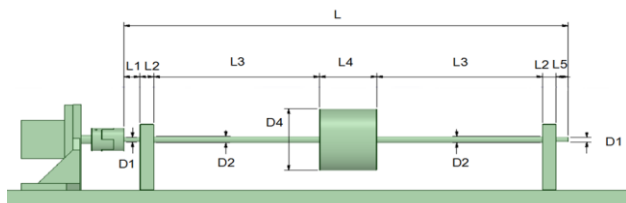
4.1 Experimental setup

To validate our models, we built a rotor kit physical system shown in Figure 6. Vertical and horizontal deflection of a rotor were measured by two laser

Table 1. Parameters of Modelica cyber system.

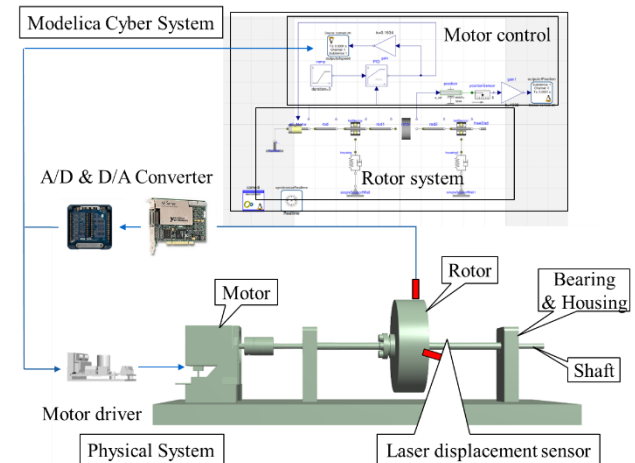
Shaft		Bearing	
Length L [mm]	500	Width L2 [mm]	15
Shaft diameter D1 [mm]	6	Damping constant [N. m. s/rad]	0.053
length L1 [mm]	50	Rigidity [N. m/rad]	23
Shaft diameter D2 [mm]	8	Coupling	
Length L3 [mm]	180	Rigidity [N/m]	650
Length L5 [mm]	50	Motor	
Young's modulus [Pa]	2.06×10^9	Torque [N. m]	0.4
Density [kg/m ³]	8000	Voltage [V]	24
Rotor		Rotating speed [rpm]	2500
Diameter D4 [mm]	80		
Width L4 [mm]	62		
Young's modulus [Pa]	2.06×10^9		
Density [kg/m ³]	8000		

displacement sensors (IL-030 KEYENCE).

**Figure 6.** Rotor kit physical system.**Figure 7.** Parameters of Modelica cyber system.

We built Modelica cyber system based on our new library in Section 2 (Figure 1 and Figure 7). The parameters of Modelica cyber system are shown in Table 1. The parameters were determined by specifications of each component except bearing. The bearing support was neither simple support nor rigid support. It had some rigidity against the bending of the shaft. We determined the bearing model parameters of the deflection angle rigidity and damping from the preliminary impulse experiment.

To acquire data from the physical system to Dymola, we used National Instruments A/D and D/A Converter, Modelica_DeviceDrivers, Modelica_Synchronous and COMEDI (CONTROL and MEasurement Device Interface for Linux / RTAI) (Ferretti *et al*, 2005). The OS was openSUSE Leap 42.1. The motor was AC Servomotor (NX410AA-1 Oriental motor). The input signal voltage driving motor was controlled through COMEDI. Figure 8 shows the rotor kit measurement and control system we built.

**Figure 8.** Rotor kit measurement and control system.

We tested the response of this system by applying step input to both physical system and Modelica cyber system. Figure 9 shows the response of motor speed and the rotor horizontal displacement vibration against the input voltage. A bit difference between experiment (physical system) and simulation (Modelica cyber system) was observed due to input signal overshooting in physical system.

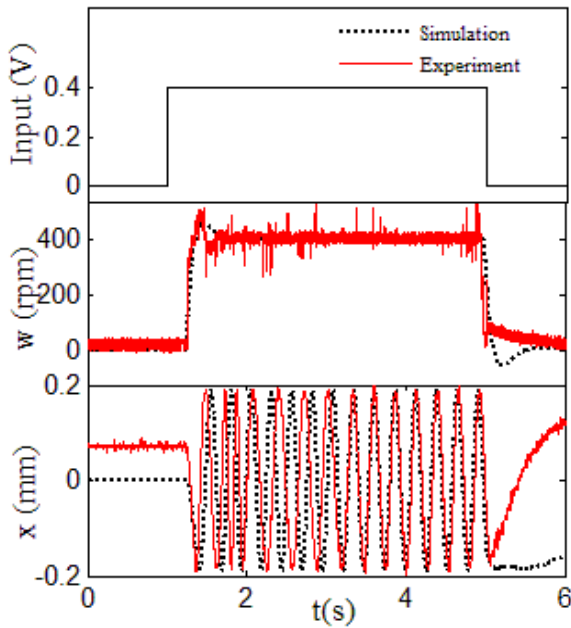


Figure 9. The step response of simulation and experiment. Input voltage, rotational speed and deflection versus time.

We also tested the ramp response of both systems. We rose the rotation speed up to 2500 rpm over the first critical speed 1600 rpm. Figure 10 shows the result. The deflection beating over 1600 rpm is due to equation (5) dynamic load torque effect of unbalance. By varying parameters of both eccentricity and bend of shaft, the responses of motor speed and horizontal vibration between simulation and experiment had little differences. By varying the bend of shaft value, we found the simulation became well-consistent with experiment.

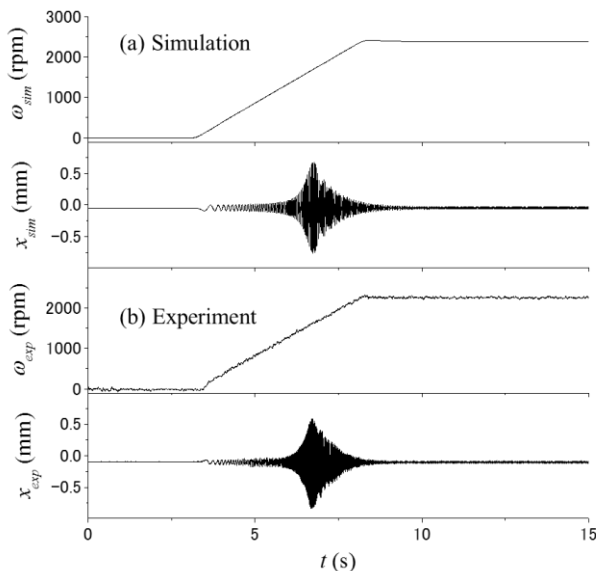


Figure 10. The ramp response of both system. Rotational speed and horizontal deflection versus time. (a) Simulation (Modelica cyber system). (b) Experiment (physical system).

4.2 Correlation to Physical Test

We calibrated the fault related parameter, as we found the bend of shaft had strongly correlated with the physical system. We preprocessed both simulation and experiment data by high pass filter at cutoff frequency $f_c = 0.4$ Hz, envelope signal processing and low pass filter at cutoff frequency $f_c = 10$ Hz. High pass filter was used to remove offset. Low pass filter was used to smooth the envelope curve. We optimized the bend of the shaft parameter by using Optimization Library (DLR, 2016). We used Integrated Squared Deviation as Criteria function and Pure Random Search algorithm for optimization.

$$E = \int \{u_1(t) - u_2(t)\}^2 dt \quad (17)$$

Here,

$u_1(t)$: Envelope processed simulation data.

$u_2(t)$: Envelope processed experiment data.

Figure 11-13 show the result of the bend of the shaft optimization. The evaluation function decreased during iteration (Figure 11). The envelope curve had changed after optimization (Figure 12). The bend of shaft value became 0.01 mm, as initial value was zero. By setting this value as parameter for simulation model of Modelica, the simulation envelope curve became well consistent with experiment (Figure 13).

To validate the static unbalance model, we added a screw on the rotor and made static unbalance at the eccentricity $e = 0.05$ mm. By setting the bend of the shaft parameter 0.01 mm and optimizing the eccentricity as above, the eccentricity parameter became $e = 4.87 \times 10^{-2}$ mm. We were able to estimate the eccentricity value from our library and Optimization library. Figure 14 shows the envelope curves of both the eccentricity parameter optimized simulation and experiment. We were able to generate the unbalanced rotor vibration data consistent with the physical system. Using Modelica cyber system, we can access the unmeasurable physical quantities such as bearing load under operation. Simulating this kind of complex physical system is very hard by coding with Simulink. The more detailed simulation by the FEM (Finite Element Method) takes much time. Using Modelica, we can simulate and generate data very easily in a reasonable amount of calculation time

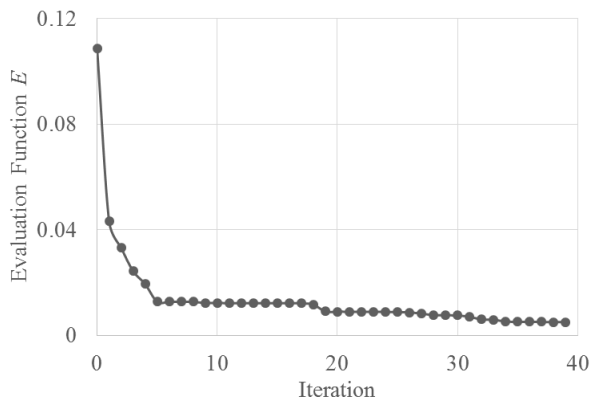


Figure 11. Evaluation function against optimization iteration.

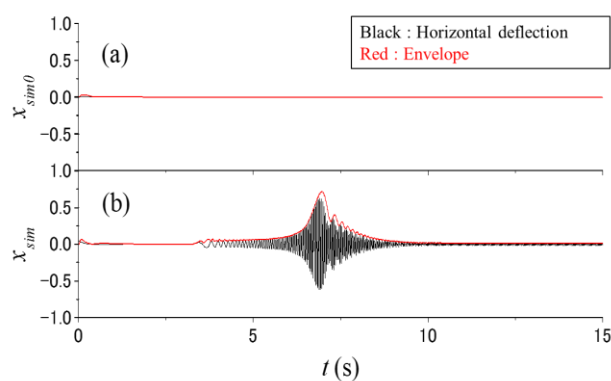


Figure 12. Envelope curve change by optimization. (a) Before optimization. (b) After optimization. Red line shows envelope of deflection signal.

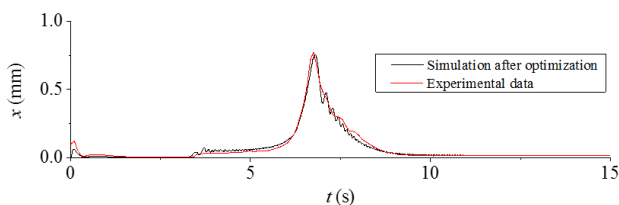


Figure 13. Comparison between simulation after the bend of the shaft optimization and experiment.

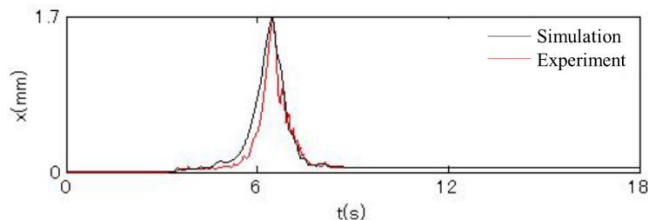


Figure 14. Comparison between the eccentricity optimized simulation (setting the optimized bend of the shaft parameter) and experiment with static unbalance.

5 Conclusions

This paper described Rotating Machinery Library we had developed based on rotor dynamics theory. This library generates rotating machinery system operation data with some faults. We validated our models by comparing both Modelica simulation and experiment with a rotor kit as a test case.

Further developments will focus on not only mechanical components, but also electrical components failures in rotating machinery system. Also, we will develop diagnosis methodology for identifying faults by stochastic and physical model based approach.

Acknowledgements

We gratefully thank Dr. Gao at Modelon KK, Japan for technical advices of developing library.

References

- COMEDI Control and Measurement Interface
<http://www.comedi.org/>
- DLR, Optimization Library for Dymola Version 2.2.2 Tutorial, 2016.
- Gianni Ferretti, Marco Gritti, Gianantonio Magnani, Gianpaolo Rizzi and Paolo Rocco. Real-Time Simulation of Modelica Models under Linux / RTAI. *Proceedings of the 4th Modelica Conference 2005*, pp. 359-365 Hamburg, Germany.
- Yukio Ishida and Toshio Yamamoto, *Linear and Nonlinear Rotordynamics: A Modern Treatment with Applications*, 2nd Edition, Wiley-VCH, 2012.
- Matthew Klenk, Johan de Kleer, Daniel G. Bobrow and Bill Janssen Using Modelica Models for Qualitative Reasoning. *Proceedings of the 10th International Modelica Conference pp. 205-211*, Lund, Sweden. doi: 10.3384/ecp14096205
- Raj Minhas, Johan de Kleer, Ion Matei, Bhaskar Saha, Bill Janssen, Daniel G. Bobrow and Tolga Kurtoglu. Using Fault Augmented Modelica Models for Diagnostics. *Proceedings of the 10th International Modelica Conference 2014*, pp. 437-445, Lund, Sweden. doi: 10.3384/ecp14096437
- Osami Matsushita, Masato Tanaka, Hiroshi Kanki, Masao Kobayashi and Patrick Keogh *Vibrations of Rotating Machinery* Springer Japan, 2017. doi: 10.1007/978-4-431-55456-1

Modelling and Simulation of the passive Structure of a 5-Axis-Milling Machine with rigid and flexible bodies for evaluating the static and dynamic behaviour

Michael Schneider, B.Eng.¹ Prof. Anton Haumer¹ Dipl.-Ing. Rupert Köckeis²

¹Faculty of Electrical Engineering and Information Technology, OTH Regensburg, 93053 Regensburg, michael.schneider@st.oth-regensburg.de, anton.haumer@oth-regensburg.de

²Department of Electrical Engineering, MAX STREICHER GmbH & Co. KG aA, 94469 Deggendorf, rupert.koeckeis@streicher.de

Abstract

Most of the mechanical simulations for industrial usage are done by finite element (FE-) analysis. Milling machines are mechatronic systems, combining electrical, mechanical and control components for machining certain materials. Modelica provides a powerful and strong tool to simulate different physical areas in one model. For this usage a mechanical model of a 5-Axis-Milling Machine is implemented with rigid and flexible bodies. Specific attention will be paid to which components can be modelled as rigid bodies without significant deviation in accordance to the real behaviour of the machine. Two classes of implementing flexible bodies in multi body systems are given by the Flexible Bodies Library, advantages and disadvantages of both classes will be evaluated. At the end a comparison of the static and dynamic behaviour of the passive structure of the model in contrast to a FE-analysis is given.

Keywords: milling machine, flexible body, multibody system, clay modelling

1 Introduction

High Speed Cutting (HSC) Machines are present in different technical areas today. The automotive sector uses HSC-Machines for editing clay models of vehicles to improve the design and the aerodynamic behaviour. For this usage high performance and very high precision is required. The validation of mechanical improvements on existing machines is expensive and time consuming. On the other hand FE-Models can not describe the whole mechatronic system, because the exact influence of the electrical drive train in mechanic models is described insufficiently. For this reason a multiphysical model with an electrical, a mechanical and a control system model has to be created. The mechanical model should describe the behaviour of the passive structure in a good approximation. A time efficient model of the machine consisting of rigid and flexible parts without major deviations is developed. During the modelling process several modelling issues have to be solved with particular models.

2 Structure of the Machine

Figure 1 shows a front view of the milling machine with its axis designation and its initial frame in the top right corner. Machine constructions where all linear movement

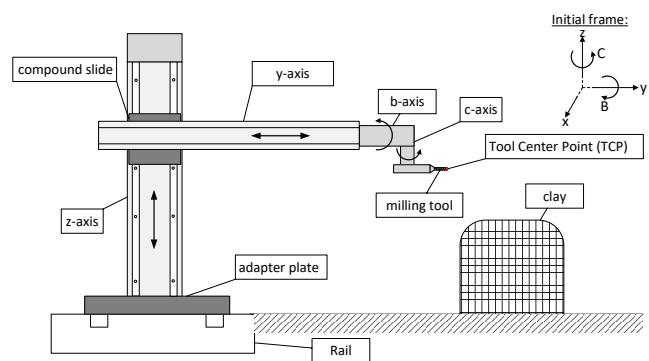


Figure 1. Structure of the milling Machine

axes are facing the milling tool side are called travelling column machines. The considered machine has a horizontal tool spindle, the end of the milling tool is called "Tool Center Point" (TCP). The three main axes (x-, y- and z-axis) realize linear movements in the cartesian space, two minor axis (b- and c-axis) enable rocking and rolling motions of the TCP. All of the three linear movements are equipped with linear guides for stiff transition between the movable and the fixed part of the machine. A compound slide enables movements in y- and z-direction. At the end of the y-axis a milling head is mounted. The linear axes are driven by short stator linear motors, the rocking and the rolling movement is achieved by synchronous motors. An additional synchronous motor drives the milling tool with a very high speed to reduce the cutting force at the TCP. A classification of the different mechanical machine parts is useful. Components with small measurements and high rigidity compared to large machine parts are called "secondary machine parts", large components with low rigidity are called "main machine parts".

3 Modelling of the secondary machine parts

3.1 Linear guidances at machine tools

Linear guidances are used to achieve relative movements between fixed and movable parts of machine tools and expose high static, dynamic and thermal stiffness. Furthermore they should show very high dynamical accuracy and low wear running. A classification in two components is usual, a guide carriage mounted at the movable part of the machine and a guide rail at the fixed machine part. The whole carriage-rail system has flexibility in three directions and could therefore deform as a result of a pulling, compressive or a shear force. Modelling this component with flexible elements would terminate in long simulation times due to the incidence of this component in the whole structure. For the purpose of simplification this component could be modeled with rigid bodies if the flexible transition between carriage and rail is modeled in an other more handable way (Queins, 2005, p. 50). But to represent the flexibility of the whole carriage-rail system, that has a significant influence on the static and dynamic behaviour of the structure, a possible replacement for a flexible body model of the carriage-rail system is given at Figure 2. The

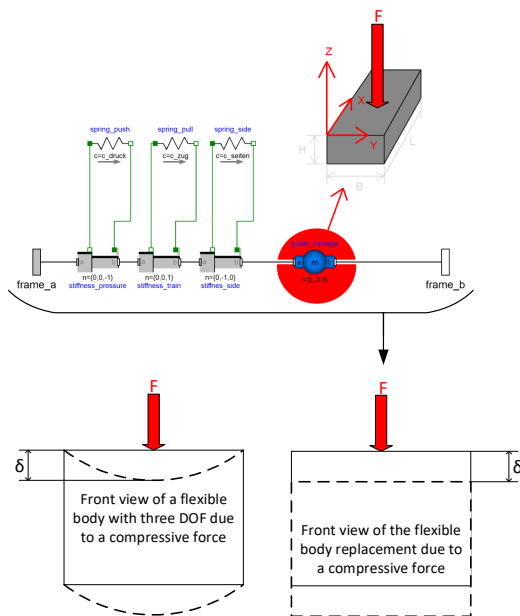


Figure 2. Replacement for a flexible body system by a rigid body system for a carriage-rail component

values of the spring stiffnesses can be calculated through a linearization of the spring characteristic curve of the carriage-rail system. This curves show a progressive or a degressive characteristic of the carriage-rail system but in a good approximation it can be linearized at the operating point (Queins, 2005, p. 50). The spring characteristic can be assumed as linear subsequently to the large stiffness of the carriage-rail system compared to other more flexible machine components.

3.2 Model of the adapter plate and the compound slide

The adapter plate connects the moving part of the machine with the machine base also called "rail". This connection is obtained by a linear recirculating roller bearing, which is especially used for longitudinal guides that provides high stiffness of guidance systems. The adapter plate consists of two linear guidance systems with four guide carriages that are mounted at a mechanical cast component. Two guide rails are mounted on the rail, together these components build the adapter plate. To reduce the calculation effort the mechanical cast component is modeled with a rigid body due to the larger flexibility of the carriage-rail system. Values of the spring stiffness of a whole carriage-rail system is extracted out of Data Sheets of the linear guidance manufacturer. Figure 3 depicts the model of the adapter plate with the four carriage-rail transitions. The usage of two spatial separated guide units

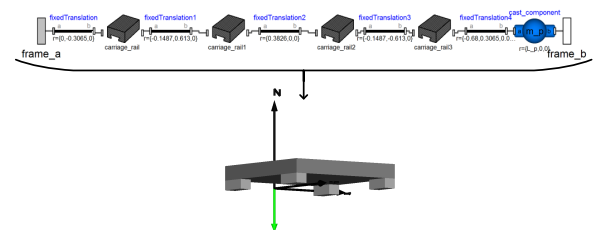


Figure 3. Model and animation of the adapter plate with four carriage-rail transitions

causes a distribution of the mechanical load and therefore an improved damping behaviour. The geometrical dimensions of the guide carriage and the spring stiffness will be presented to the carriage-rail transition models. Therefore this model especially characterises the transition between the fixed and the movable part of the machine, deformations of the components realizing this transition are neglected (Hoffmann, 2008). With the help of a compound slide, travel motions in y- and z-direction can be realised. Each of the two axis motions is equipped with two four-row linear recirculating ball bearings, where any of them consists of two carriage guides and one guide rail. The carriage rail system, which realises the z-motion is mounted at the back side of a steel plate in the same way as the adapter plate. A reverse arrangement, where the carriage-guides are mounted at the fixed part of the machine, which is the compound slide, provides movements in the y-direction. The carriage rails are fixed at the back side of the y-cantilever over the whole dimension of the component. In the same way as the adapter plate is modeled, the compound slide will be modeled. Based on the assumption that the shift between the fixed and the movable parts is more flexible than the self deformation of the steel plate, only the carriage-rail systems are modeled flexibly. Figure 4 depicts the structure of the compound slide

model with the eight carriage-rail transitions splitted into the different axis movements and the related 3D-model of the component. The animation of the component depicts

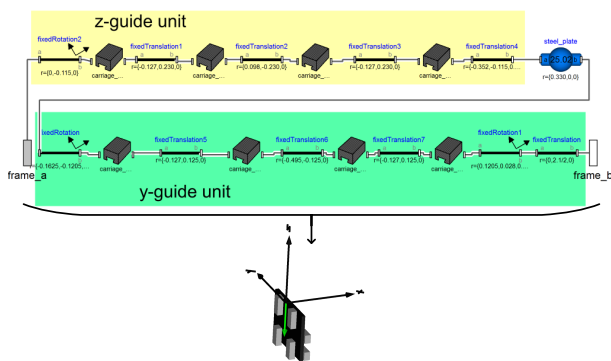


Figure 4. Model and animation of the compound slide with eight carriage-rail transitions splitted into the different axis movements

the reversal arrangement of the different linear guide systems. It makes no difference whether the carriage-rail shift between the y-cantilever and the compound slide is modeled at the slide itself or at the horizontal cantilever. In order to the same reasons as discussed at the adapter plate, only the flexibility of the transitions is modeled, the steel plate is assumed to be rigid.

3.3 Modelling of the Milling Head

The milling head is mounted at the end of the y-cantilever, it consists of eight parts. All of these parts are used to achieve rotary movements of the milling tool, which is mounted at the end of the component. Different connection flanges combine the rotary axes mechanically with each other. In comparison with the secondary and especially with the main machine parts the dimensions of the structural elements of the milling head are small. As a result of this assumption and as the milling head only acts as a load at the end of the y-cantilever the components can be modeled rigidly. In Figure 5 the model and the animation of the milling head is illustrated. The rotary motions of the different axes are implemented by components of the `Modelica.Mechanics.Multibody.Joints` li-

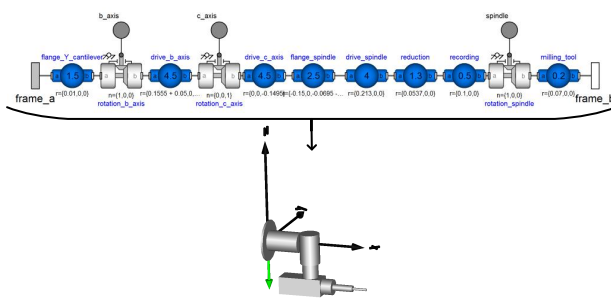


Figure 5. Model and animation of the milling head with rigid bodies

brary. Different parameters of the rigid components, such as the mass or geometric dimensions, are defined by design data of the milling head.

4 Modeling of the main machine parts

4.1 Modeling of the Y-Cantilever

The initial situation at the y-cantilever can be described as follows, the cantilever is connected to the compound slide by the linear guides as described in section 3.2. It performs movements in the YZ-plane due to external forces caused by a linear direct drive. These forces, the connected milling head together with gravitational forces are leading to deformations of the cantilever and therefore to relative movements of the TCP with respect to the initial frame. For this reasons the y-cantilever has to be modeled with flexible bodies, and therefore the usage of the `FlexibleBodies` library is necessary. Choosing boundary conditions with respect to this situation would lead to the following conditions. In Figure 6 above, the initial situation with three chosen boundary conditions (red numbers) is depicted. The boundary conditions can be chosen

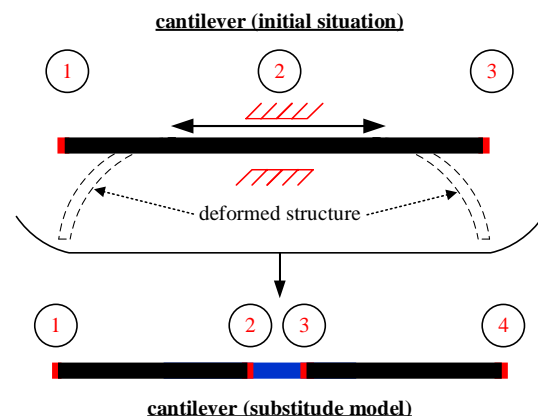


Figure 6. Initial situation of the y-cantilever and substituted model

free at point 1 and 3 and as movable support at number 2 because of the parallel guidance at the compound slide. The base class of a beam from the `FlexibleBodies` library enforces the user to choose boundary conditions for each deformation type at the endings of the beam to get correct results (Dr.-Ing. Andreas Heckmann et al., 2016). It is not possible to choose boundary conditions at certain points within the length of the beam. Choosing the base class of a beam model with two boundary conditions for each deformation type would therefore lead to large errors due to neglecting the third boundary condition. Furthermore the point, where the beam is movably supported changes during the milling process. In order to minimize the relative error of the y-cantilever model a substitute model (Figure 6 bottom) has to be created. The model is splitted into three parts, where two parts consists of flexible beams (Figure 6 black) and one part is a rigid

body (Figure 6 blue). The whole y-cantilever is divided at the center in two flexible beams of equal length. A rigid body, which describes the rigid clamping of the beam at the compound slide, is inserted between the flexible bodies. The division of the flexible structure and the insertion of a rigid body leads to a system where four boundary conditions have to be chosen but the conditions where the flexible beams are connected to the rigid body have to be clearly chosen as clamped. In order to align boundary conditions with the degrees of freedom of the joints, to which the beam is attached no condition should bound the elastic y-motion of the node attached at the attachment frame(Heckmann, 2010). Because of the joint that realizes the y-motion of the beam, at point 1 no boundary condition is chosen for any type of deformation and the condition for point 4 is free. The restriction of this model is, that only small movements of the y-cantilever can be considered otherwise this model leads to massive errors. However the position of the y-cantilever can be assumed to be constant during the milling process of one part of a vehicle such as the side or the front surface. In addition to this problem parameters such as the cross section, the modulus of elasticity or the density can not be chosen easily. Taking a closer look at the cross section in Figure 7 of the flexible structures in the substituted model illustrates this fact. The entire beam consists of two parts,

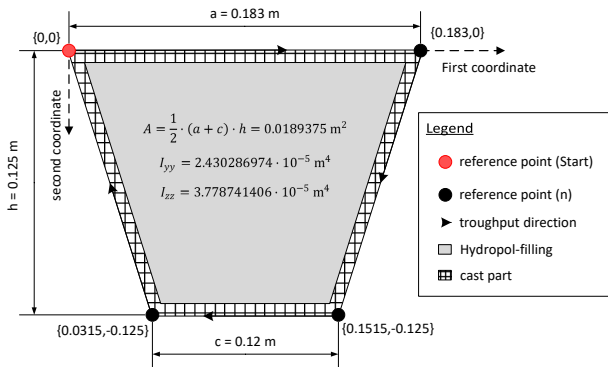


Figure 7. Cross section of the y-cantilever with geometrical moments of inertia

an empty cast part and a special vibration reducing material which is inserted there. The materials have different modulus of elasticity E and different densities ρ , for this reason a fictitious material has to be developed, considering the shares V_f of both materials. The material constants of the fictitious material can be calculated in the following way(Gross et al., 2014, p. 279-286).

$$V_f = \frac{A_{cast}}{A_{fill}} \quad (1)$$

$$\rho_{beam} = V_f \cdot \rho_{cast} + (1 - V_f) \cdot \rho_{fill} \quad (2)$$

$$E_{beam} = V_f \cdot E_{cast} + (1 - V_f) \cdot E_{fill} \quad (3)$$

Small movements of the y-cantilever are permissible but to consider the additional extension of the second beam, if it is moving towards the milling head, an additional line force has to be applied. This additive force has to be applied over the whole second beam and is depending on the covered distance towards the milling tool. The whole model of the y-cantilever with the additional line force acting as a point load and the animation of the component is depicted in Figure 8. In order to reduce dynamic flexi-

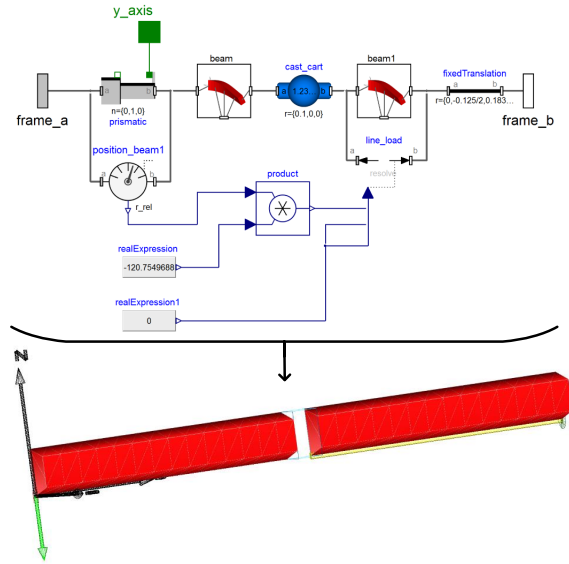


Figure 8. Model and animation of the y-cantilever with an additional line load

bility, steel ribs are welded over the whole length of the beam. This mechanical design detail could not be considered by the base class of a flexible beam without a subdivision of the y-cantilever model in several subsections and therefore a higher computational effort. Because of the new development of different machine parts, no real measurements are done on this components till yet. For this reason an independent validation of the y-cantilever model with the aid of real measurements is not possible.

4.2 Modelling of the Z-Tower

The second main machine part that shows high static and dynamic flexible behaviour is the z-tower. Due to the special geometry of this part, a simple beam model would not describe the behaviour. Therefore the second base class of the FlexibleBodies library, the ModalBody class is used to get more accurate results. A finite element model of the z-tower is reduced in two steps, in order to achieve a reduction of the number of degrees of freedom. The result of this reduction is a modal representation of the component with 221 degrees of freedom. This modal representation is stored in a standard input data file, that is the input for the ModalBody class either than for other multi-body simulation programs like SIMPACK. The second input file for the ModalBody provides informations of the geometri-

cal shape of the considered body by a wavefront file. This file offers the user a 3D-view of the considered eg. the deformed component (Andreas Heckmann et al., p.88-94). In order to combine the movement of the compound slide and therefore the y-cantilever over the whole length of the tower the usage of the extended base class `MovingLoad` is suitable. It provides a connector, which allows it to attach moving forces/bodies/systems to the flexible structure. It is also possible to control this movement by a flange but additional parameter need to define this movement (Dr.-Ing. Andreas Heckmann et al., 2016). Shape functions are used for defining deformations and calculating forces acting on the `ModalBody`, this shape functions are only known at certain points, specified by the `Nodes` parameter vector. This makes interpolation necessary and one requirement for the `ModalBody` especially it needs enough points in the vector to make this interpolation stable. The load is only connected to the right guide rail that is close to the milling head and the TCP. Six Simulation nodes out of 36 on this guide rail provide a stable interpolation. Another problem is to keep small the wavefront file, that provides a 3D-view of the z-tower otherwise the internal animation of Dymola will crash. The initial object file of the z-tower provided a mesh with 30095 elements. After applying a quadratic edge collapse decimation filter a reduction to 1700 elements provides a stable animation. Figure 9 depicts the reduced animation file of the z-tower with the six simulation nodes.

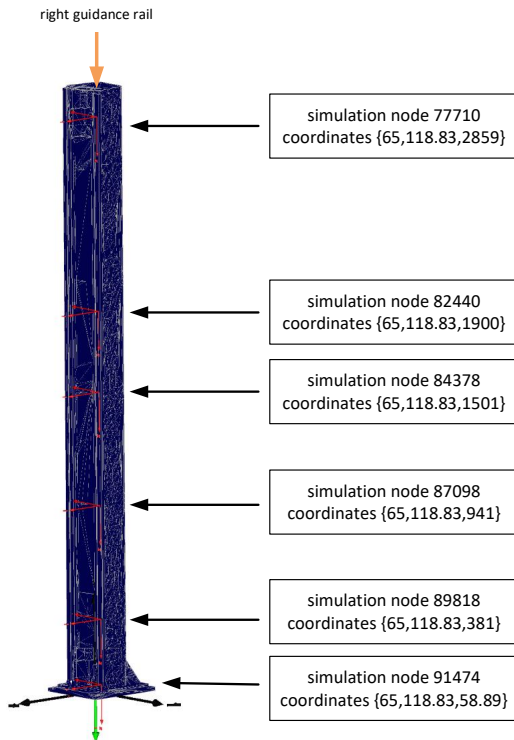


Figure 9. Reduced animation of the z-tower with used simulation nodes and their coordinates

5 Static and dynamic investigations

The basis of the static and dynamic investigations are the presented secondary machine parts in section 3 and the main machine parts in section 4. Together they build a multibody model with flexible and rigid bodies of the milling machine.

5.1 Static investigations

The static behaviour of the multibody model will be compared with the results of a finite element simulation in order to evaluate the multibody model in contrast to a mechanic specific simulation tool. The results of the finite element simulation are also not validated but they will match the real static behaviour of the machine close due to the integration of specific construction details out of CAD-constructions. Both simulations are done without any load at the end of the y-cantilever. To evaluate the static behaviour of the model a constant force in one direction is applied at the end of the y-cantilever to get static shifts in all three translational directions. This constant force will be achieved through a step function with a short delay time to provide that the whole structure is in a steady state. The y-cantilever will therefore be in an unstable position at the highest point at the z-tower and fully extended. The area in which the substitute model of the y-cantilever is valid is left and only the magnitudes of the static shifts can be rated. After five seconds to get the structure in a steady state, a constant force of $F = 10 \text{ N}$ is applied at the end of the y-cantilever and after 0,35 s the structure comes already to a steady state. If the relative movement at the

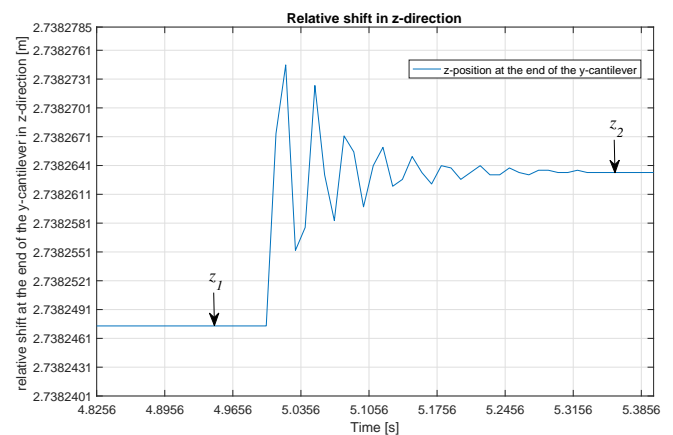


Figure 10. Relative shift at the end of the y-cantilever in z-direction

end of the y-cantilever is divided by the applied force the relative shift V_z in z-direction can be calculated.

$$V_z = \frac{z_2 - z_1}{F} = \frac{1.5974 \cdot 10^{-5} \text{ m}}{10 \text{ N}} = 1.5974 \cdot 10^{-3} \frac{\text{mm}}{\text{N}} \quad (4)$$

In the same way as shown above the relative shifts in y- and x-direction can be calculated.

$$V_y = \frac{y_2 - y_1}{F} = \frac{1.0729 \cdot 10^{-5} \text{ m}}{10 \text{ N}} = 1.0729 \cdot 10^{-3} \frac{\text{mm}}{\text{N}} \quad (5)$$

$$V_x = \frac{x_2 - x_1}{F} = \frac{5.9605 \cdot 10^{-6} \text{ m}}{10 \text{ N}} = 5.9605 \cdot 10^{-4} \frac{\text{mm}}{\text{N}} \quad (6)$$

Comparing the results of the multibody simulation to the finite element solutions it can be stated that in a good approximation they are convergent. Deviations can be justified by the neglect of the flexibility of the guidance system, the disregard of the ribs in the y-cantilever and the desertion of the scope of the y-cantilever. Regarding the magnitudes of the static shifts it can be said that they are absolutely convergent to the finite element solution.

5.2 Dynamic investigations

Positioning movements will encourage oscillations of the whole structure and therefore also vibrations of the TCP. This oscillations will also lead to positioning errors during the milling process. To have a closer look at a positioning movement the y-cantilever will be in a stable position in the centre of the traverse range of the z-tower. A simple ramp function will increase the x-position of the structure till reaching a final value. The substitute model of the y-cantilever is therefore in position that is within the validity range. The ramp starts at position $x = 0 \text{ m}$ at 0.5 s and increases this position till reaching the final value of $x = 1 \text{ m}$ at 1.5 s . Figure 11 depicts the dynamical behaviour during the positioning process. The simula-

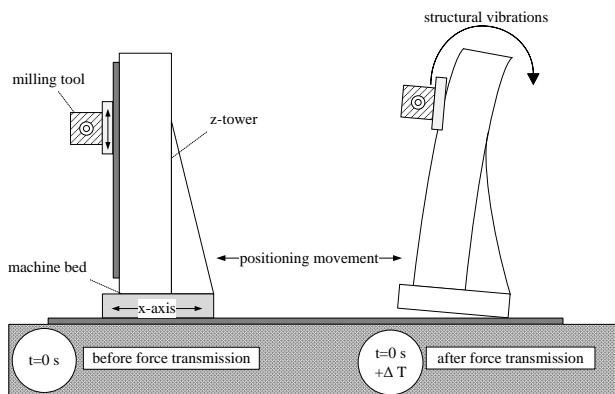


Figure 11. Dynamical behaviour during the positioning movements

tion results in Figure 12 are showing absolutely the same behaviour of the structure. After reaching the final position an overshoot relative to the desired position could be determined. The envelope of this overshoot is an exponential function which indicates a strong viscose damping behaviour of the structure(Hoffmann, 2008).

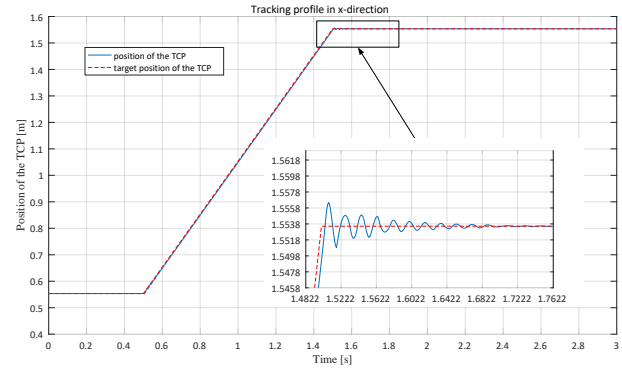


Figure 12. Tracking profile in x-direction

6 Conclusion and outlook

The mechanical model of the 5-axis-milling machine depicts the static and dynamic behaviour very accurate. The simplifications made during the modeling process have only marginal influences on the result. For modeling flexible bodies within multibody structures the FlexibleBodies library is a very accurate tool. The base classes Beam and ModalBody offers the user different methods of implementing those flexible structures. The beam class is only useful for describing simple beam models, complex geometries or designs are difficult to model with this class. An extension of the beam base class, where the user is able to define more than two boundary conditions and make the position of these boundary conditions time dependent would therefore be a relief. The amount of usable boundary conditions is limited to three, implementing other boundary conditions such as "movable clamped" would make the modeling of problems, such as the movable cantilever, easier without FE-pre-processing. In order to model such beam structures and defining input datas such as boundary conditions or the number of the required eigenmodes for getting correct results, a very deep theoretical background is needed. If the user only wants to implement existing models derived out of a finite element analysis the ModalBody class offers a very powerful tool for implementing those structures. In the next step, models of the drive trains that move the different axes will be developed. In combination with this drive models the whole behaviour of the mechatronic system can be observed and compared to the real system. After this validation of the model the trajectory guidance will be improved by changing the control strategy of the drive trains.

References

- Andreas Heckmann, Martin Otter, Stefan Dietz, and José Díaz López. The DLR FlexibleBodies library to model large motions of beams and of flexible bodies exported from finite element programs. In *Proceedings of the 5th International Modelica Conference*, pages 85–95. URL <https://www.modelica.org/events/>

modelica2006/Proceedings/proceedings/
Proceedings2006_Voll.pdf.

Dr.-Ing. Andreas Heckmann, Prof. Dr.-Ing. Martin Otter, Martin Leitner, Jakub Tobolar, and Stefan Hartweg. FlexibleBodies: Library to model large motions of flexible beams, anular plates and of flexible bodies exported from finite element programs, 2016. Version 2.2.

Dietmar Gross, Werner Hauger, Jörg Schröder, and Wolfgang A. Wall. *Technische Mechanik 2: Elastostatik*. Springer-Lehrbuch. Springer Vieweg, Berlin, 12., aktual. Aufl. edition, 2014. ISBN 978-3-642-40965-3. doi:10.1007/978-3-642-40966-0. URL <http://dx.doi.org/10.1007/978-3-642-40966-0>.

Andreas Heckmann. On the choice of boundary conditions for mode shapes in flexible multibody systems. *Multibody System Dynamics*, 23(2):141–163, 2010. ISSN 1384-5640. doi:10.1007/s11044-009-9177-z.

Frank Hoffmann. *Optimierung der dynamischen Bahnge-
nauigkeit von Werkzeugmaschinen mit der Mehrkörpersimu-
lation: Zugl.: Aachen, Techn. Hochsch., Diss., 2008*, volume
2008,8 of *Ergebnisse aus der Produktionstechnik Werkzeug-
maschinen*. Apprimus-Verl., Aachen, 2008. ISBN 978-3-
940565-12-9.

Marcus Queins. *Simulation des dynamischen Verhaltens von
Werkzeugmaschinen mit Hilfe flexibler Mehrkörpermodelle:
Zugl.: Aachen, Techn. Hochsch., Diss., 2005*, volume
2005,12 of *Berichte aus der Produktionstechnik*. Shaker,
Aachen, 2005. ISBN 3-8322-4224-4.

Modeling and Simulation on Environmental and Thermal Control System of Manned Spacecraft

Sun Lefeng¹ Jin Jian¹ Chen Liping² Liu Wei² Huang Lei² Zhou Fanli² Liu Qi²

¹China Academy of Space Technology, Beijing, China,

sunlefeng@hotmail.com, jinjian0331@126.com

²Suzhou Tongyuan Software & Control Technology Co., Suzhou, China,

{chenlp, liuw,huangl, zhoufl, [liuq](mailto:liuq}@tongyuan.cc)}@tongyuan.cc

Abstract

In order to support crew resides, key air environment parameters of manned spacecraft should be controlled within index range by environmental and thermal control system. In this paper a model of manned spacecraft environmental and thermal control system in Modelica language is developed. Using this simulation model, we analyze air environment parameters varying trend as the crew metabolic level variation. The results show that crew metabolic level could influence air environment parameters dramatically. Furthermore, air environment parameters should be analyzed comprehensively due to important affection of air temperature to oxygen partial pressure, carbon dioxide partial pressure and relative humidity. The work in this paper is helpful to provide a new method for analysis of environmental and thermal control system of manned spacecraft.

Keywords: *manned spacecraft, Modelica, MWorks; temperature/humidity control, carbon dioxide removal, oxygen pressure control*

1 Introduction

Environmental and thermal control system is a system to guarantee a good life and thermal environment and the key technology to realize manned spaceflight [1].

The current commonly used analysis method is to establish a pressurized cabin simulation model using CFD (computational fluid dynamics) [2-6]. The method is used to analyze the temperature and humidity, partial pressure of oxygen, etc. Using this design method has the following shortcomings:

1) In the program design, the designer is concerned with the system level indicators. CFD software is good at equipment level analysis, not suitable for system level analysis;

2) CFD software is not suitable for system level analysis so that it is difficult to analyze the interrelationship between each parameter of the system;

On the contrary, Modelica is an object-oriented modeling and simulation language. Modelica is good at system level analysis. With Modelica, we can establish

mathematical models of each component and an integral model of the entire environmental and thermal control system. The paper has the following objectives:

- 1) Establish a model of manned spacecraft environmental and thermal control system in Modelica language;
- 2) Analyze the interrelationship of key parameters of manned spacecraft environmental and thermal control system.
- 3) Analyze the influence of the change of key parameters on the system.

2 System Descriptions

In this paper, the cabin environment is assumed to be insulated from the outside. Schematic diagram of environmental and thermal control system is shown below.

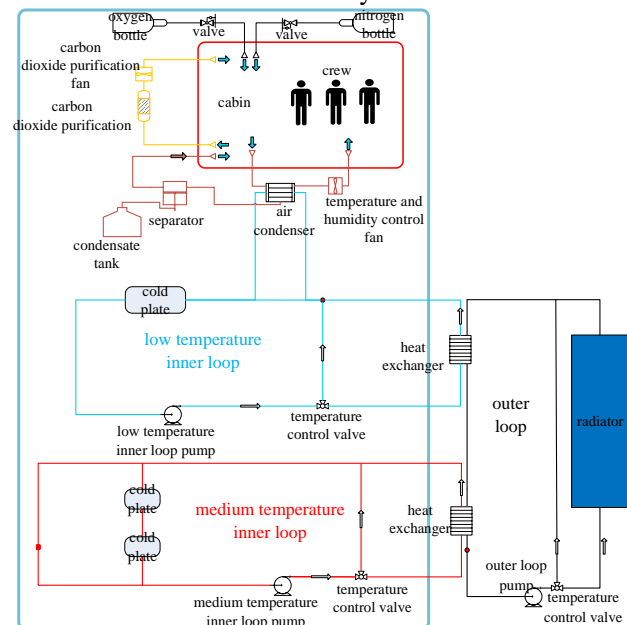


Figure 1 Environment and thermal control system

2.1 Constituent of Environmental and Thermal Control System

Environmental and thermal control system includes:

1) Cabin pressure control system: cabin is equipped with high pressure oxygen bottle. When partial pressure of oxygen bellows the lower limit, oxygen bottle begins to supply oxygen until partial pressure of oxygen reaches the higher limit.

2) Carbon dioxide purification system: cabin is equipped with non-regenerative cabin dioxide purification tank and fan. Fan extracts air from the cabin to purification tank. When carbon dioxide partial pressure reaches the higher limit, a new purification tank will be automatically replaced.

3) Temperature and humidity control system: cabin is equipped with air condenser, moisture separator. Air condenser provides the cold source for the temperature control loop; Fan extracts air from the cabin to air condenser; Water and gas mixture which is collected by the air condenser enters moisture separator to separate. The separated water enters water tank and the separated air returns to the cabin.

4) Low temperature inner loop control system: low temperature inner loop is equipped with pump, heat exchanger and temperature control valve. The speed of pump is a parameter which is set before simulation. Temperature control valve opening is controlled by the PID controller which is set a temperature control point.

5) Medium temperature inner loop control system: medium temperature inner loop is equipped with pump, heat exchanger and temperature control valve.

6) Outer loop control system: heat collected by the low temperature inner loop control system and medium temperature inner loop control system is transferred to the outer loop through heat exchanger. Outer loop collects heat load of the cabin and equipment and exhausts to the space through radiator.

2.2 Metabolic Level of Astronaut

The metabolic level of astronaut changes with the different forms of activities. Referring to the international space station, this paper takes into account four metabolic levels:

1) Sleeping: Metabolic heat production is 80W. The rate of oxygen consumption is 0.0202kg/h. Carbon dioxide output rate is 0.023kg/h;

2) Resting: Metabolic heat production is 100W. The rate of oxygen consumption is 0.0252kg/h. Carbon dioxide output rate is 0.029kg/h;

3) Mild activity: Metabolic heat production is 170W. The rate of oxygen consumption is 0.0432kg/h. Carbon dioxide output rate is 0.049kg/h;

4) Moderate activity: Metabolic heat production is 240W. The rate of oxygen consumption is 0.0606kg/h. Carbon dioxide output rate is 0.069kg/h;

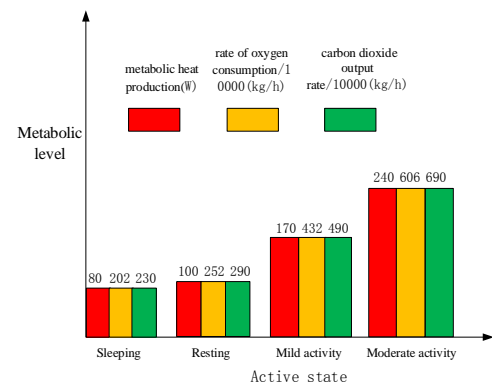


Figure 2 Metabolic level of astronaut at different active state

2.3 Astronaut Schedule

This paper assumes there are three astronauts in the cabin and they are always at the same level of metabolism. Astronaut schedule in a day is arranged as follows: Sleeping is 7 hours. Resting is 4 hours. Moderate activity is 2 hours. Mild activity is 11 hours. Schedule is in accordance with the above order.

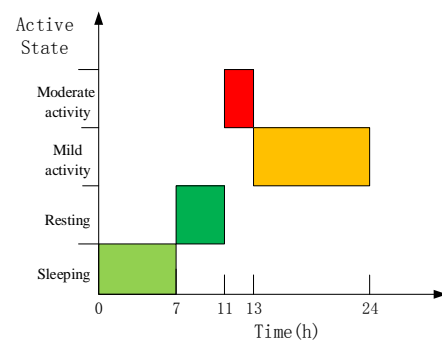


Figure 3 Crew's active state diagram in a day

2.4 Indicator of Air Environment

Indicators of air environment refer to International Space Station [7]. We can see it as follows:

Table 1 The goal of each air environment indicator

Goal	Request
temperature	20~26 °C
relative humidity	30%~70%
partial pressure of oxygen	19000 ~ 22000 Pa
partial pressure of carbon dioxide	≤700 Pa

3 Models

We use MWorks as a basic platform for modeling and simulating the environmental and thermal control system of manned spacecraft [8].

3.1 Interface Design

Interfaces for physical component models must be physically able to connect components. Environmental and thermal control system involves two areas of heat and fluid so that its interfaces are heat and fluid interfaces which are shown in table 2. There are two basic types of variables for Modelica interfaces, which are flow variables and potential variables. Interfaces connection comply with the general theory of Kirchhoff's law, namely the sum of flow variables is zero and the potential variables are equal when interfaces connect to each other. In order to meet the needs of thermal fluid system modeling, Modelica has added a new interface variable which is stream variable.

Table 2 Interface types and variables in interfaces

Interface type	Variable	Variable type
fluid interface	pressure	potential variable
	mass flow rate	flow variable
	mass ratio	stream variable
	specific enthalpy	stream variable
thermal interface	temperature	potential variable
	heat flow	flow variable

3.2 Medium Models

Medium models are obtained by expansion and specialization from standard medium of Modelica standard library. Medium models and component models can be decoupled when independent medium models are designed. Medium models are defined as replaceable models. Component models can select medium model by redeclaration. A medium model is a model package which is made up of four parts:

- 1) Constants, which contain the name of medium and molar mass, etc.
- 2) Attribute models, which mainly include state equation and other thermodynamic equation.
- 3) Functions, which calculate property parameters in different states.
- 4) Types, which apply to the thermodynamic variables.

3.3 Component Models

A component model is a restricted category of Modelica which can include parameters, variables, nested classes, equations and algorithms. A component model is established using a bottom-up approach, inherits base class model, declare subcomponent model and interface and add variables and equations. Component models correspond to the system basic physical components such as pipe, valve, fan, heat exchanger, etc. Component models are fully functional models which can be directly instantiated and used. Main equations of the major components are described as follows:

3.3.1 Cabin

The cabin interacts with the outside world can be abstracted thermal and fluid interfaces. The main equations of a cabin is shown as follows:

- 1) Mass conservation is calculated by the using the following equation:

$$\frac{dm_j}{dt} = w_{in}x_{in,j} - w_{out}x_{out,j} + w_{lf,j} \quad (1)$$

Where m_j denotes the mass of the J kind ingredient; w_{in} denotes air mass which flow in the cabin; $x_{in,j}$ denotes mass percentage of the J kind ingredient which flow in the cabin; w_{out} denotes air mass which flow out the cabin; $x_{out,j}$ denotes mass percentage of the J kind ingredient which flow out the cabin; $w_{lf,j}$ denotes mass percentage of the J kind ingredient metabolized by Astronaut.

- 2) Energy passed to the bulkhead is calculated by using the following equation:

$$\frac{dU_{wall}}{dt} = q_{wall} \quad (2)$$

Where U_{wall} denotes internal energy of bulkhead; q_{wall} denotes total heat passed to the bulkhead.

- 3) Energy passed to air of the cabin is calculated by using the following equation:

$$\frac{dU_{air}}{dt} = w_{in}h_{in} - w_{out}h_{out} + q_{air} \quad (3)$$

Where U_{air} denotes internal energy of air in the cabin; h_{in} denotes enthalpy of air which flow in the cabin; h_{out} denotes enthalpy of air which flow out the cabin; q_{air} denotes total heat added to air.

3.3.2 Crew

The crew interacts with the outside world can be abstracted thermal and fluid interfaces. The main equations of a crew is shown as follows:

- 1) Metabolism is calculated by the using the following equation:

$$W = \mu(Q_{act} - Q_{bas}) \quad (4)$$

$$Q = Q_{act} + Q_{shiv} \quad (5)$$

Where w denotes mechanical force of every crew; μ denotes mechanical efficiency; Q_{act} denotes metabolic activity of crew; Q_{bas} denotes basal metabolic activity of crew; Q_{shiv} denotes heat generated by muscle tremors.

- 2) Breathing is calculated by using the following equation:

$$m_{O_2} = \frac{Q}{60000 \cdot 247.35 \cdot (0.23 \cdot RQ + 0.77)} \quad (6)$$

$$m_{CO_2} = RQ \cdot m_{O_2} \cdot \frac{MW_{CO_2}}{MW_{O_2}} \quad (7)$$

Where RQ denotes respiratory coefficient; MW_{CO_2} is molar mass of carbon dioxide; MW_{O_2} denotes molar mass of oxygen.

Through inheriting interfaces and adding parameters, variables and equations, we can establish the crew Modelica model.

3.3.3 Carbon Dioxide Purification

The carbon dioxide purification interacts with the outside world can be abstracted thermal and fluid interfaces. The main equations of a carbon dioxide purification is shown as follows:

- 1) Total amount of carbon dioxide purification control of a purification tank is calculated by using the following equation:

$$M_{CO_2} = x_{load} \cdot m_{LiOH,0} \quad (8)$$

Where x_{load} denotes mass of carbon dioxide purified per kg in the initial state; $m_{LiOH,0}$ denotes mass per purification tank.

- 2) Carbon dioxide purification rate control is calculated by using the following equation:

$$w_{CO_2} = 0.9185ar \left[1 - \frac{x_{load}}{0.9185a} \right] m_{LiOH,0} \quad (9)$$

Where r denotes chemical reaction rate of carbon dioxide.

- 3) Water production rate control is calculated by using the following equation:

$$w_{H_2O} = 0.9185ar \left[1 - \frac{x_{load}}{0.9185a} \right] m_{LiOH,0} \frac{MW_{H_2O}}{MW_{CO_2}} \quad (10)$$

Where MW_{H_2O} denotes molecular weight of water; MW_{CO_2} denotes molecular weight of carbon dioxide.

- 4) Mass conservation is calculated by using the following equation:

$$x_{out,CO_2} = \frac{w_{in,Z} x_{in,CO_2} - w_{CO_2}}{w_{out,Z}} \quad (11)$$

$$x_{out,H_2O} = \frac{w_{in,Z} x_{in,H_2O} + w_{H_2O}}{w_{out,Z}} \quad (12)$$

Where x_{out,CO_2} denotes mass percentage of carbon dioxide in the air which flows out purification tank; x_{out,H_2O} denotes mass percentage of water in the air which flows out purification tank; $w_{in,Z}$ denotes mass flow rate of air which flows in purification tank; $w_{out,Z}$ denotes mass flow rate of air which flows out purification tank; x_{in,CO_2} denotes mass percentage of carbon dioxide in the air which flows in purification tank; x_{in,H_2O} denotes mass percentage of water in the air which flows in purification tank.

- 5) Momentum conservation is calculated by using the following equation:

$$\Delta p = \Delta p_{ref} \left[\frac{w_{in,Z}}{w_{ref}} \right] \times \left| \frac{w_{in,Z}}{w_{ref}} \right| \times \frac{\rho_{ref}}{\rho_{in,Z}} \quad (13)$$

Where Δp denotes pressure difference of air which flows through purification tank; Δp_{ref} denotes reference pressure difference of air which flows through purification tank; w_{ref} denotes reference mass flow rate of air which flows in purification tank; $\rho_{in,Z}$ denotes density of air which flows in purification tank; ρ_{ref} denotes reference density of air which flows in purification tank.

- 6) Energy conservation is calculated by using the following equation:

$$\frac{dT_{bed}}{dt} = \frac{w_{in,Z} h_{in,Z} - w_{out,Z} h_{out,Z} + q_{reac}}{M_{bed} C_{p_{bed}}} \quad (14)$$

Where T_{bed} denotes temperature of purification tank; $h_{in,Z}$ denotes enthalpy of air which flows in purification tank; $h_{out,Z}$ denotes enthalpy of air which flows out purification tank; q_{reac} denotes heat generated by the chemical reaction; M_{bed} denotes total mass of purification tank; Cp_{bed} denotes specific heat of purification tank.

3.3.4 Air Condenser

The air condenser interacts with the outside world can be abstracted thermal and fluid interfaces. The main equations of an air condenser is shown as follows:

1) Mass conservation of air side is calculated by using the following equation:

$$w_{air,in} = w_{air,out} + w_{drain} \quad (15)$$

Where $w_{air,in}$ denotes mass flow rate of air which flows in air condenser; $w_{air,out}$ denotes mass flow rate of air which flows out air condenser; w_{drain} denotes mass flow rate of gas and liquid mixture which flows in moisture separator.

2) Liquid flow control is calculated by using the following equation:

$$w_{drain,liquid} = \varepsilon_{slurper} x_{slurper,liq} w_{air,in} \quad (16)$$

Where $w_{drain,liquid}$ denotes mass flow rate of liquid which flows in moisture separator; $\varepsilon_{slurper}$ denotes separation efficiency; $x_{slurper,liq}$ denotes mass ratio of liquid in mixture.

3) Heat exchange control is calculated by using the following equation:

$$q_{ex} = e_{hex} \min(C_{air}, C_{cold}) (T_{air,in} - T_{cold,in}) \quad (17)$$

Where q_{ex} denotes total heat exchange; e_{hex} denotes heat exchange efficiency; C_{air} denotes heat capacity in the gas side; C_{cold} denotes heat capacity in the liquid side; $T_{air,in}$ denotes inlet temperature in the gas side; $T_{cold,in}$ denotes inlet temperature in the liquid side.

3.3.5 Pump

The pump interacts with the outside world can be abstracted thermal and fluid interfaces. The main equations of a pump is shown as follows:

1) Energy is calculated by using the following equation:

$$\frac{dT_{out}}{dt} = \frac{w_{in}(h_{in} - h_{out}) + q}{Cp_{fluid} \cdot \rho \cdot V + M_{dry} Cp_{dry}} \quad (18)$$

Where T_{out} denotes outlet temperature of working fluid; h_{in} denotes inlet enthalpy of working fluid; h_{out} denotes outlet enthalpy of working fluid; q denotes power delivered to fluid; Cp_{fluid} denotes specific heat capacity of the fluid; ρ denotes density of working fluid; V denotes volume of fluid; M_{dry} denotes mass of solid wall; Cp_{dry} denotes specific heat capacity of solid wall.

2) Hydraulic efficiency is calculated by using the following equation:

$$\varepsilon = \frac{\rho \cdot g \cdot TDH \cdot Q}{W} \quad (19)$$

Where ε denotes hydraulic efficiency of pump; TDH denotes total dynamic head of pump; Q denotes volume flow rate; W denotes braking power.

3.3.6 Fan

The fan interacts with the outside world can be abstracted thermal and fluid interfaces. The main equations of a fan is shown as follows:

1) Energy is calculated by using the following equation:

$$\frac{dT_{out}}{dt} = \frac{w_{in}(h_{in} - h_{out}) + q}{M_{dry} Cp_{dry}} \quad (20)$$

Where T_{out} denotes outlet temperature of working fluid; h_{in} denotes inlet enthalpy of working fluid; h_{out} denotes outlet enthalpy of working fluid; q denotes power delivered to fluid; M_{dry} denotes mass of solid wall; Cp_{dry} denotes specific heat capacity of solid wall.

2) Hydraulic efficiency is calculated by using the following equation:

$$\varepsilon = \frac{\rho \cdot g \cdot TDH \cdot Q}{W} \quad (21)$$

Where ε denotes hydraulic efficiency of fan; TDH denotes total dynamic head of fan; Q denotes volume flow rate; W denotes braking power.

3.4 Subsystem Models

Through inheriting component models, we establish cabin pressure control system, carbon dioxide purification system, temperature and humidity control system, fluid loop system which includes low temperature inner

loop, medium temperature inner loop and outer loop. These subsystem models are shown as follows:

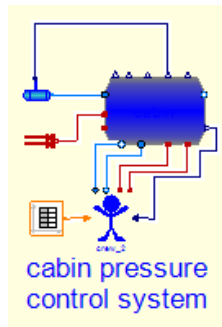


Figure 4 Cabin pressure control system

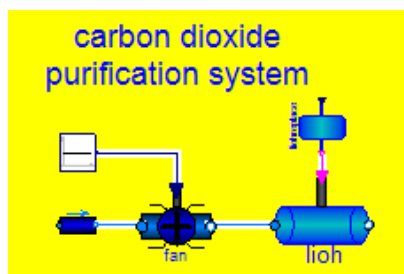


Figure 5 Carbon dioxide purification system

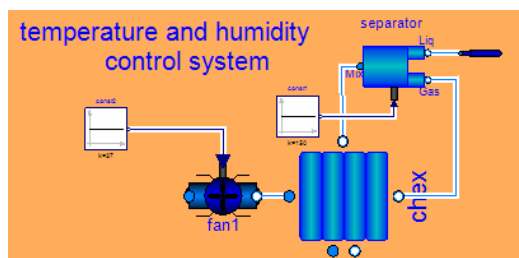


Figure 6 Temperature and humidity control system

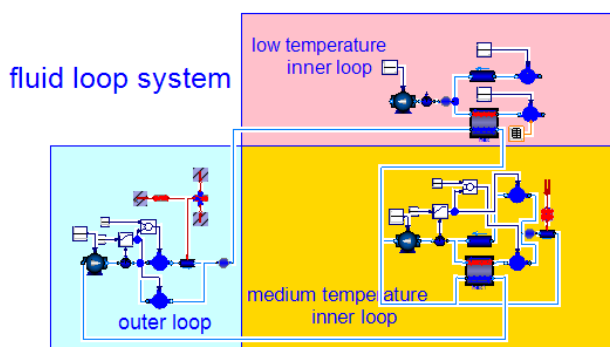


Figure 7 Fluid loop system

3.5 System Model

Through inheriting subsystem models, the model of environmental and thermal control system is established as follows:

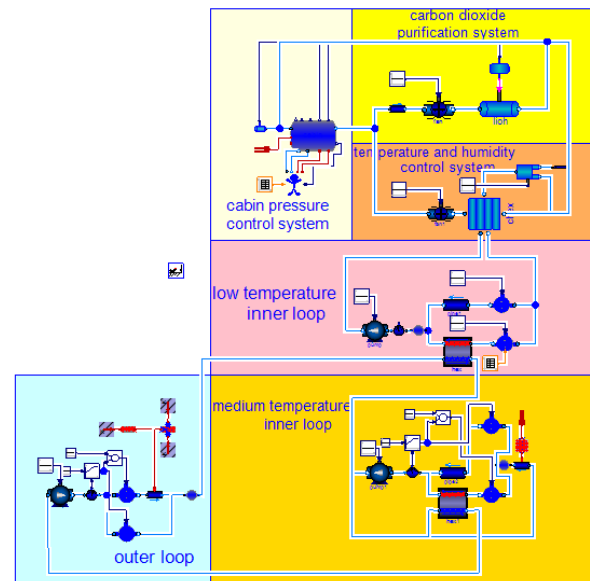


Figure 8 Model of environmental and thermal control system based on Modelica

4 Simulations and Analysis

This paper analyzes the key parameters of the air environment of the cabin in a day. The simulation parameter settings are shown in table 3. The results are shown in figure 9 to figure 16.

Table 3 Simulation parameter settings

Component name	Parameter name	Values
cabin	air volume in cabin	100 m ³
	cabin inner wall area	600 m ²
carbon dioxide purification	Mass of carbon dioxide purifying agent per box	2kg
carbon dioxide purification fan	speed	150rad/s
air condenser	thermal conductivity	350W/K
separator	speed	150rad/s
temperature and humidity control fan	speed	32rad/s
low temperature inner loop pump	speed	150rad/s
medium temperature inner loop pump	speed	50rad/s

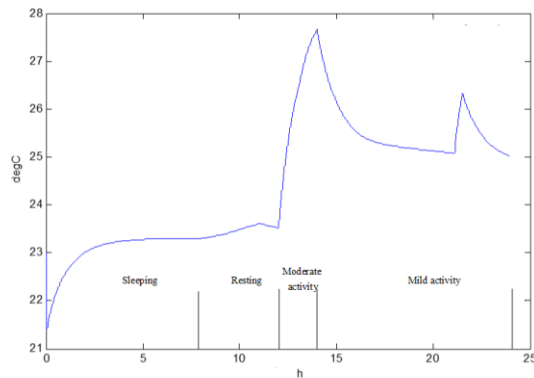


Figure 9 Air temperature

Air temperature in the cabin changes with the variation of crew metabolism (Figure 9). The abscissa is time and its unit is hour. The ordinate is temperature and its unit is centigrade. When crews are sleeping, air temperature is about 21.5 centigrade; From fourth to twelfth hours, air temperature increases slightly; At twelfth hours, when crews are in moderate activity, air temperature increases rapidly and reaches 27.7 centigrade at the highest point, beyond the upper limit of the index; At fourteenth hours, when crews are in mild activity, air temperature decreases rapidly; At twenty-first hours, oxygen partial pressure reaches the lower limit, cabin begins to fill oxygen so that air temperature increases; At twenty-second hours, because the oxygen partial pressure reaches the higher limit, cabin stops filling oxygen so that air temperature decreases.

In summary, during a day, when the crew is in moderate activity, the air temperature is outside the normal range.

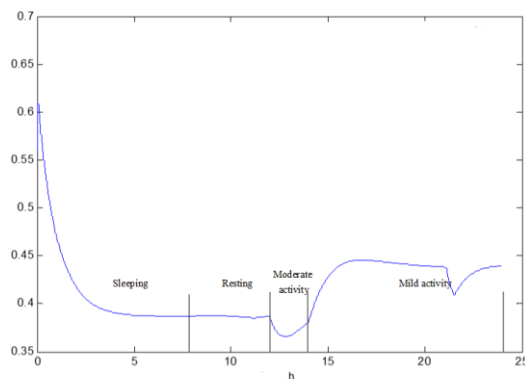


Figure 10 Relative humidity

Relative humidity of air is directly related to metabolic level of crew and air temperature (Figure 10). The abscissa is time and its unit is hour. The ordinate is relative humidity. When crews are sleeping, relative humidity of air remains at around 38%; At fourteenth hours, when crews are in moderate activity, although the crew metabolic wet increases, temperature increases rapidly so that relative humidity of air decreases; At twenty-first hours, because the oxygen partial pressure

reaches the lower limit, cabin begins to fill oxygen so that air temperature increases. Relative humidity of air decreases; At twenty-second hours, because the oxygen partial pressure reaches the higher limit, cabin stops to fill oxygen so that air temperature decreases. Relative humidity of air increases.

In summary, air temperature has a greater impact to relative humidity than metabolic level of crew. Relative humidity of air is in the normal range in a day.

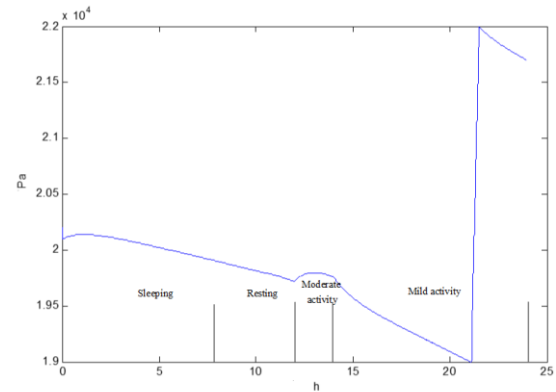


Figure 11 Oxygen partial pressure

The partial pressure of oxygen has a relationship to crew oxygen consumption rate, cycle of supplying gas, air temperature, etc (Figure 11). The abscissa is time and its unit is hour. The ordinate is the partial pressure of oxygen and its unit is Pa. When crews are sleeping and resting, oxygen partial pressure decreases slowly; At twelfth hours, when crews are in moderate activity, although the crew metabolism strengthen and oxygen consumption increases, temperature increases rapidly so that oxygen partial pressure increases; At fourteenth hours, when crews are in mild activity, although crew metabolism decline and oxygen consumption decreases, air temperature decreases rapidly so that oxygen partial pressure decreases; At twenty-first hours, because the oxygen partial pressure reaches the lower limit, cabin begins to fill oxygen so that oxygen partial pressure increases; At twenty-second hours, because the oxygen partial pressure reaches the higher limit, cabin stops to fill oxygen so that oxygen partial pressure decreases.

In summary, oxygen partial pressure is in the normal range in a day.

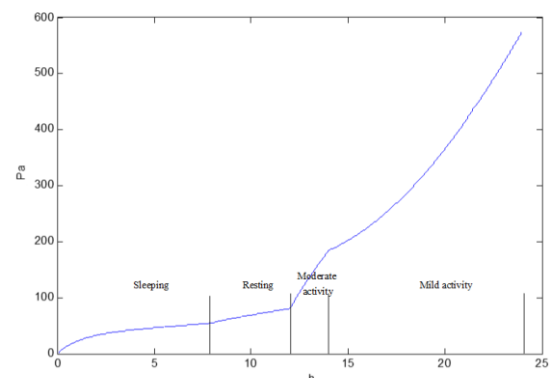
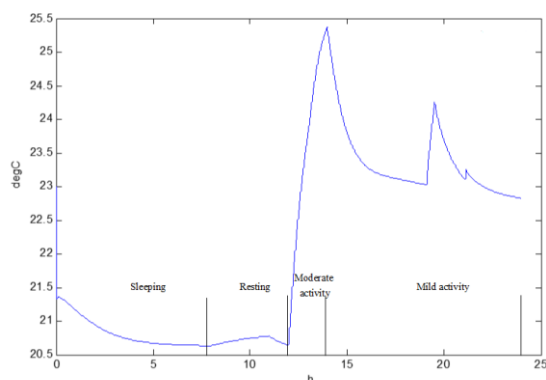
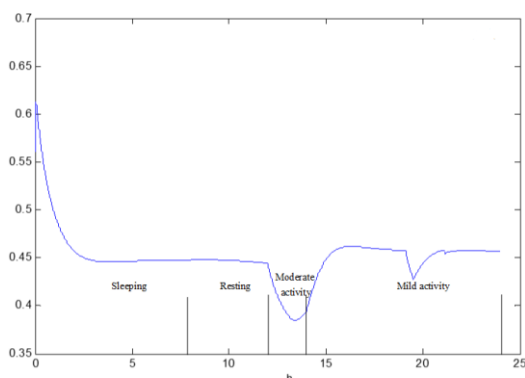
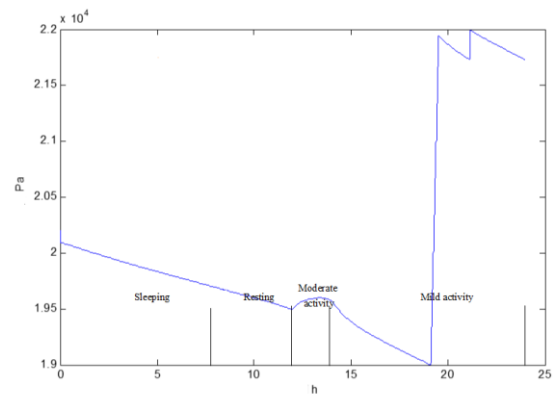
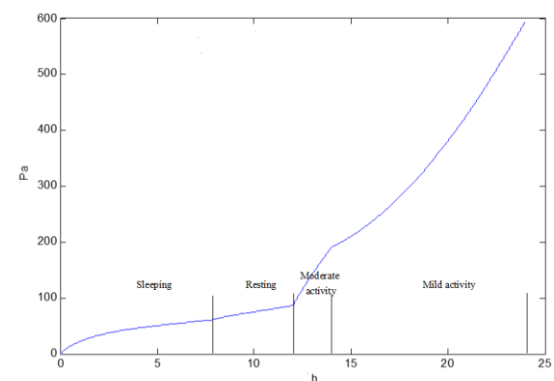


Figure 12 Carbon dioxide partial pressure

Carbon dioxide partial pressure changes with the change of crew metabolism (Figure 12). The abscissa is time and its unit is hour. The ordinate is the partial pressure of oxygen and its unit is Pa. We suppose the initial carbon dioxide partial pressure is 0. When crews are sleeping or resting, carbon dioxide partial pressure increases; At twelfth hours, when crews are in moderate activity, crew metabolism strengthen and more carbon dioxide is generated, carbon dioxide partial pressure increases fast; At fourteenth hours, when crews are in mild activity, crew metabolism decline and less carbon dioxide is generated, carbon dioxide partial pressure increases slowly.

In summary, carbon dioxide partial pressure is in the normal range in a day.

In the case of the above model parameters, the air temperature beyond the normal range in a day. The air temperature is controlled by the condensing dryer, and the fluid flow into the condensing dryer is controlled by the temperature and humidity control fan. In the case of other parameters unchanged, we increase the temperature and humidity control fan speed to 40 rad/s and observe the change of air environmental parameters.

**Figure 13** Air temperature at 40 rad/s of fan speed**Figure 14** Relative humidity at 40 rad/s of fan speed**Figure 15** Oxygen partial pressure at 40 rad/s of fan speed**Figure 16** Carbon dioxide partial pressure at 40 rad/s of fan speed

In summary, in a day, air temperature, air relative humidity, oxygen and carbon dioxide partial pressure are in the range of indicators.

5 Conclusions

In this paper a model of manned spacecraft environmental and thermal control system in Modelica language is developed based on the professional knowledge. Using this simulation model, air environment parameters varying trend as the crew metabolic level variation has been analyzed. Draw the conclusion as follows:

- 1) Crew metabolic level could influence air environment parameters dramatically.
- 2) Air environment parameters should be analyzed comprehensively due to important affection of air temperature to oxygen partial pressure, carbon dioxide partial pressure and relative humidity.
- 3) The simulation of the environmental and thermal control system can be carried out by modifying the key parameters of the components, which greatly reduces the workload of the test and the working time of the engineer.

References

- [1] Lin Guiping, Wang Puxiu. Manned space life support technology [M]. Beijing: Beijing University of Aeronautics and Astronautics press, 2006:37-147.
- [2] Cheng Wenlong, Zhao Rui, Huang Jiarong, et al. Numerical simulation of flow heat transfer and humidity distribution in pressured cabins of an independent flight manned spacecraft[J]. Journal of Astronautics, 2009, 30(6): 2410-2416.
- [3] Fu Shiming, Xu Xiaoping, Li Jindong, et al. Carbon dioxide accumulation of space station crew quarters [M]. Journal of Beijing University of Aeronautics and Astronautics, 2007, 33(5):523-526(in Chinese).
- [4] Ji Chaoyue, Liang Xingang, Ren Jianxun. Numerical study of crew carbon dioxide discharge in pressurized cabin of space station[C]. The fifth space thermal physics conference, Chinese Astronautical Society, 2000.9. Huangshan, 147-150.
- [5] Zhong Qi, Liu Qiang, etc. A numerical investigation on heat transfer and flow in a pressurized cabin of spacecraft [J]. Journal of Astronautics, 2002, 23(5):44-48(in Chinese).
- [6] Huang Jiarong, Fan Hanlin. Steady numerical simulation for the humidity distribution in manned spacecraft habitation cabin [J]. Journal of Astronautics, 2005, 26(3):349-353(in Chinese).
- [7] Wieland P O. Living together in space: The design and operation of the life support systems on the International Space Station. NASA/TM1998-206956[R].
- [8] Yu Tao, Zeng Qingliang. Multi_domain simulation based on the modeling language Modelica [J]. Journal of Shandong University of Science and Technology, 2005, 24 (4):13-16.

Modeling and simulation of a complex ThermoSysPro model with OpenModelica

Dynamic Modeling of a combined cycle power plant

El Hefni Baligh¹ Bouskela Daniel¹

¹EDF R&D STEP, Electricité de France, {baligh.el-hefni, daniel.bouskela}@edf.fr

Abstract

ThermoSysPro (TSP) is a generic library for the modeling and simulation of power plants and other kinds of energy systems. TSP library is developed by EDF and released under open source license. The library features multi-domain modeling such as thermal-hydraulics, neutronics, combustion, solar radiation, instrumentation and control.

Numerous organizations and individuals worldwide now use TSP. Until recently, the TSP library could be used only under Dymola for the modeling and simulation of complex power plants. But now, with the latest version of OpenModelica (OM), we can simulate complex models of power plants with complex scenarios.

To be able to use TSP under OM, some adaptations have been applied to our models, essentially the method used to make inverse computation.

The objective of this work is to evaluate the potentiality, capability and efficiency of using OpenModelica tools to perform dynamic studies of power plants. A combined cycle power plant has been chosen as a representative test case of the complexity of this type of study.

The paper describes the dynamic model of a combined cycle power plant, whose objective is to study a step variation load from 100% to 50% and a full gas turbine trip, using OM software. Also, the structure of the model, the parameterization data, the results of simulation runs, the difficulties encountered using OM and the comparison between Dymola and OM are presented.

Keywords: Modelica; OpenModelica; ThermoSysPro; thermal-hydraulics; combined cycle power plant; dynamic modeling; inverse problems.

1 Introduction

Modeling and simulation play a key role in the design phase and performance optimization of complex energy processes. They also play a significant role in the future for power plant maintenance and operation.

ThermoSysPro is a generic library for the modeling and simulation of power plants and other kinds of energy systems. ThermoSysPro library is developed by EDF and released under open source license.

The foundations of the library are based on first physical principles: mass, energy, and momentum conservation equations, up-to-date pressure losses and heat exchange correlations, and validated fluid properties functions. The correlations account for the non-linear behavior of the phenomena of interest. They cover all water/steam phases, oil, molten salt and all flue gas compositions. The granularity of the modeling may be freely chosen. Some correlations are given by default since they correspond to the most frequent use-cases, but they can be freely modified by the user if needed. This includes the choice of the pressure drop or heat transfer correlations. Special attention is given to the handling of two-phase flow, as two-phase flow is a common phenomenon in power plants.

The library components are written in such a way that there are no hidden or unphysical equations, that components are independent from each other and to ensure as much as possible upward and downward compatibility across tools and library versions. This is particularly important in order to control the impact of component, library or tool modifications on the existing models.

This library is aimed at providing the most frequently used model components for the 0D-1D static and dynamic modeling of thermodynamic systems, mainly for power plants, but also for other types of energy systems such as industrial processes, energy conversion systems, buildings etc. It involves disciplines such as thermal-hydraulics, combustion, neutronics and solar radiation.

The ambition of the library is to cover all the phases of the plant lifecycle, from basic design to plant operation. This includes for instance system sizing, verification and validation of the instrumentation and control system, system diagnostics and plant monitoring. To that end, the library will be linked in the future to systems engineering via the modeling of systems properties, and to the process measurements via data reconciliation and data assimilation.

The library may be downloaded freely together with the OpenModelica software from <https://openmodelica.org/download/download-windows>.

Several test-cases were developed to validate the library in order to cover the full spectrum of use-cases for power plant modeling:

- Dynamic model of a 1300 MWe nuclear power plant covering the primary and secondary loops,
- Dynamic model of steam generators for sodium fast reactor (David F., Souyri A. and Marchais G, 2009)
- Static and dynamic models of a biomass plant (El Hefni B. and Péchiné B, 2009),
- Physics/neutronics model in Modelica for a tool, to assist the operator, to control the power plant for infrequent transients and to establish a strategy of optimal operating procedure (El Hefni B., 2011),
- Dynamic model of a concentrated solar power plant (El Hefni B., 2013),
- Dynamic multi-configuration model of a 145 MWe concentrated solar power plant with the ThermoSysPro library (tower receiver, molten salt storage and steam generator)', (El Hefni B., Soler R., 2014),
- Dynamic simulation of a 1MWe CSP tower plant with two-level thermal storage implemented with control system (S.J. Liua et al., 2014),
- Dynamic simulation and experimental validation of Open Air Receiver and Thermal Energy Storage systems in solar thermal power plant (Qing Li et al., 2015).

The objective of this paper is to show the potentiality, capability and efficiency of OpenModelica tools to perform dynamic studies using complex models such as the combined cycle power plant model.

In order to challenge the dynamic simulation capabilities of the library, a step load variation from 100% to 50% and a turbine trip (sudden stopping of the gas turbine) were simulated.

2 How to use OpenModelica for inverse problems (model inversion)

As the initial state of the simulation is in general defined by the observable outputs of the system (e.g. the nominal power output, the pressure inside the boiler, etc.), it is necessary to solve an inverse problem to compute the initial state. Moreover, it is necessary to start the simulation from a stationary (or steady) state in order to avoid the numerical difficulties which arise when the system is started out of equilibrium (oscillations, stiffness ...).

The inverse problem basically consists in setting (fixing) a state variable of the model to a known measurement value to compute by model inversion the value of a parameter or a boundary condition.

Modelica allows to express inverse problems, which is a powerful feature for computing operation points, which are defined by their observable outputs, and for system sizing, to compute parameterised characteristics.

To implement the inverse problem under Dymola, it is enough simply to fix the value of the state variable and declared it to (fixed = true) and released the parameter to be computed and declared it as (fixed = false). However, this method is incompatible with OpenModelica (no standard Modelica language).

Here is a simple example to illustrate the deference between Dymola and OM for the implementation of the reverse problem (*standard Modelica language*). Furthermore, for the demonstration we use a simple model to calculate the pressure drop in a tube, so:

$$P_i - P_o = K \cdot \frac{Q \cdot |Q|}{\rho}$$

P_i is the fluid pressure at the singularity inlet (Pa), P_o is the fluid pressure at the singularity outlet (Pa), Q is the fluid mass flow rate (kg/s), K is the friction pressure loss coefficient (m^{-4}) and ρ is the average density of the fluid (kg/m^3).

As the above equation is implemented in a TSP component model called **SingularPressureLoss**, this model component is used to illustrate inverse calculation. The model uses the following component models (see Figure 1):

- SingularPressureLoss model,
- SourceP model,
- SinkP model.



Figure 1. TestSingularPressureLoss model (test-case).

The equation above makes it possible to calculate the flow rate of the fluid through the tube, provided that the pressure drop in the tube, the coefficient of the pressure drop and the fluid density are known (parameters).

The model inversion (calibration) consists in setting the mass flow rate of the fluid through the tube (Q) and the friction pressure loss coefficient of the pipe (K) can be computed. To express this inverse problem with Dymola, it suffices to write: [Q (fixed = true, start = 500)] and [K (fixed=false, start=100)] in the parameters

windows of the *SingularPressureLoss* model (see Figure 2).

Figure 2. Data for the *SingularPressureLoss* model.

On the other hand, to express this inverse problem with OpenModelica (also valid for Dymola), it is necessary to write: `[Q (fixed = true, start = 500)]` and `K = K_pipe` in the parameters windows of the *SingularPressureLoss* model (see Figure 3).

Also, the parameter `K_pipe` (new intermediary parameter), must be created (declared) in the main model, with `[K_pipe (fixed=false, start =1.e2)]`, see Table 1.

Figure 3. Data for the *SingularPressureLoss* model.

Table 1. The declaration of the `K_pipe` in the main model.

```
model TestSingularPressureLoss
  parameter Real K_pipe (fixed=false,start=1.e2)
  "Pressure loss coefficient";
  equation
```

3 Combined cycle power plant model

The power plant model is a complete model of a real combined cycle power plant:

Gas Turbine (GT): Nominal power: 2*226 MW,

Steam Generator (HRSG): Thermal power: 2*360 MW,

Steam Turbine: Nominal power: 277 MW,

Condenser:

Thermal power: 428 MW.

Outlet water temperature: 305 K

Vacuum pressure: 6100 Pa.

3.1 Model description

Currently, two models are used: one to simulate the power generator step reduction load (see Figure 4), the other to simulate the full GT trip (see Figure 5). In the model used to simulate the GT trip, the gas turbine is replaced by a boundary condition.

The model contains two main parts: the water/steam cycle and the flue gases subsystem. Only one train is modelled, so identical behavior is assumed for each HRSG and for each gas turbine.

HRSG model:

The model consists of 16 heat exchangers (3 evaporators, 6 economizers, 7 super-heaters), 3 evaporating loops (low, intermediate and high pressure), 3 drums, 3 steam turbine stages (HP, IP and LP), 3 pumps, 9 valves, several pressure drops, several mixers, several collectors, 1 condenser, 1 generator, several sensors, sources, sinks and the control system limited to the drums level control.

An important feature of this model is that the thermodynamic cycle is completely closed through the condenser. This is something difficult to achieve, because of the difficulty of finding the numerical balance of large closed loops.

The list of components used for the development of the HRSG model is given in Table 2.

Table 2. Library components used in the HRSG model.

Type	Model name in the library
Condenser	DynamicCondenser
Drum	DynamicDrum
Generator	Generator
Heat exchanger	DynamicExchangerWaterSteamFlue Gases = DynamicTwoPhaseFlowPipe ExchangerFlueGasesMetal HeatExchangerWall
Pipe	LumpedStraightPipe
Pump	StaticCentrifugalPump
Steam turbine	StodolaTurbine
Valve	ControlValve

Water mixer	VolumeB, VolumeC
Water splitter	VolumeA, VolumeD

Heat Exchanger:

Based on first principles mass, momentum and energy balance equations, the following phenomena are represented:

- Transverse heat transfer,
- Mass accumulation,
- Thermal inertia,
- Gravity,
- Pressure drop within local flow rate.

Drum and Condenser:

Based on first principles mass and energy balance equations for water and steam, the following phenomena are represented:

- Drum level and swell and shrink phenomenon,
- Heat exchange between the steam/water and the wall,
- Heat exchange between the outside wall and the external medium.

Steam turbine:

Based on an ellipse law and an isentropic efficiency.

Pump:

Based on the characteristics curves.

Pressure drop in pipes:

Proportional to the dynamic pressure \pm the static pressure.

Mixer/splitter:

Based on the mass and energy balances for the fluid.

GT model:

The model consists of 1 compressor, 1 gas turbine, 1 combustion chamber, sources, sinks and 1 air humidity model.

The list of component models used for the development of the GT model is given in Table 3.

Table 3. Library components used in the GT model.

Type	Model name in the library
Air humidity	AirHumidity
Compressor	GTCompressor
Gas turbine	CombustionTurbine
Combustion chamber	GTCombustionChamber

Gas turbine:

Based on correlations for the characteristic.

Compressor:

Based on correlations for the characteristic.

Combustion chamber:

Based on first principles mass, momentum and energy balance equations. The pressure loss in the combustion chamber is taken into account.

The “*CombinedCyclePowerPlant*” model contains 673 component models, generating 10802 variables, 257 differentiated variables, 2752 equations and 1855 nontrivial equations.

3.2 Data implemented in the model

All geometrical data were provided to the model (pipes and exchangers lengths and diameters, heat transfer surfaces of exchangers, volumes ...).

The plant characteristics are given below.

Gas Turbine (GT)

Compressor compression rate: 14

Steam Generator (HRSG)

HRSG with 3 levels of pressure.

High pressure circuit at nominal power: 127 bar

Intermediate pressure circuit at nominal power: 27 bar

Low pressure circuit at nominal power: 5.0 bar

Steam Turbine

High pressure at nominal power: 124.5 bar, 815 K

Intermediate pressure at nominal power: 25.5 bar, 801 K

Low pressure at nominal power: 4.8 bar, 532 K

Condenser

Steam flow rate: 194 kg/s

Water temperature at the inlet: 306 K

3.3 Model calibration

The calibration phase consists in setting the maximum number of thermodynamic variables to known measurement values taken from on-site sensors for 100% load. This method ensures that all needed performance parameters and size characteristics can be computed. The variables imposed in the model are:

- Pressure at the outlet of the pumps,
- Pressure at the inlet of the steam turbines,
- Specific enthalpy at the inlet of the steam turbines,
- Liquid level in drums and in condenser,
- Overall heat exchangers coefficients,
- Isentropic efficiency of the compressor,
- Exhaust temperature of the gas turbine,
- ...

The main computed performance parameters are:

- The characteristics of the pumps,
- The ellipse law coefficients of the steam turbines,
- The isentropic efficiencies of the turbines,
- The CV of the valves and the valves positions (openings).

- Heat exchangers fouling coefficients,
- Nominal isentropic efficiency of the compressor,
- Nominal isentropic efficiency of the gas turbine,
- ...

3.4 Simulation scenarios

For simulation runs, two scenarios were selected. The first scenario is a power generator step reduction from 100 to 50% load:

- Initial state (combined cycle): 100 % load
- Final state (combined cycle): 50% load (800 s slope).

The second scenario is a full GT trip (sudden stopping of the gas turbine):

- Initial state (GT exhaust): 894 K, 607 kg/s
- Final state (GT exhaust): 423 K, 50 kg/s (600 s slope).

3.5 Simulation

Simulation runs were done using Dymola 2017 and OpenModelica 1.11.0. The simulation of the scenarios were mostly successful. However, some difficulties were encountered when simulating large transients, mainly stemming from the large size of the model:

- Poor debugging facility,
- Large number of values to be manually provided by the user for the iteration variables, *for the two tools*.

In particular, it has been observed that, *the two tools* cannot calculate the initial states, when all iterations variables are not set close to their solution values.

3.6 Simulation results

The model is able to compute:

- The air excess,
- The distribution of water and steam mass flow rates,
- The thermal power of heat exchangers,
- The electrical power provided by the generator,
- The pressure temperature and specific enthalpy distribution across the network,
- The drums levels and the condenser level,
- The performance parameters of all the equipments,
- The global efficiencies of the water/steam cycle and gas turbine.

The results of the simulation runs are given in Figure 6 and Figure 7. They are consistent with the engineer's expertise. **The comparison between simulation results and GE (General Electric) results (FOUQUET L, 2004) for 100 % load and 50 % load, have shown that the Simulation results are very close to the GE values (Design results).**

The computational time is faster than real time.

3.7 Comparison between Dymola and OpenModelica

The simulation results of OM are very close to the simulation results of Dymola. The simulation time with OM is between 15% and 60% times slower than the simulation time with Dymola, depending on the scenario and the tolerance (see Table 4).

Table 4. Simulation time.

Simulation time (s)				
	Dymola 2017		OpenModelica1.11.0	
	Tolérance=0,001	Tolérance=0,0001	Tolérance=0,001	Tolérance=0,0001
Variation de charge (simulation 2500 s)	58	73	75	84
Trip TAC (simulation 10000 s)	174	310	240	492

However, OM is still 20 times faster than real time in the worst case.

4 Conclusion

A dynamic and rather large model of a combined cycle power plant has been developed to validate the ThermoSysPro library. This model comprises the flue gas side and the full thermo-dynamic water/steam cycle closed through the condenser. Two difficult transients were simulated with Dymola 2017 and OpenModelica 1.11.0: a step reduction load of the power generator and a full gas turbine trip. The results are mostly consistent with the engineer's expertise.

Despite of some simulation difficulties because of the lack of debugging tools for Modelica models, this work shows that the library is complete and robust enough for the modelling and simulation of complex power plants with the two software.

The simulation results of OM are very close to the simulation results of Dymola. The simulation time with OM is slower than simulation time with Dymola, but still 20 times faster than real time.

This work shows that OpenModelica software is very satisfying for thermo-hydraulic modelling and simulation.

Acknowledgements

This work was partially supported by the OPENCPS project.

References

David F., Souyri A. and Marchais G., 'Modeling Steam Generators for Sodium Fast Reactors with Modelica', *Modelica 2009 conference proceedings*.

El Hefni B. and Péchiné B., 'Model driven optimization of biomass CHP plant design', *Mathmod conference 2009, Vienna, Austria*.

El Hefni B. 'Dynamic modeling of concentrated solar power plants with the ThermoSysPro Library (Parabolic Trough collectors, Fresnel reflector and Solar-Hybrid)', *SolarPaces 2013, Elsevier's Energy Procedia*,

El Hefni B., 'Modèle physique/neutronique en Modelica d'un outil d'aide au pilotage du transitoire sensible de montée en puissance à 3%Pn/h après rechargement. Maquettage d'un outil d'aide au pilotage sous Excel/VB', *LMCS 2011*.

El Hefni B. and Soler R. 'Dynamic multi-configuration model of a 145 MWe concentrated solar power plant with the ThermoSysPro library (tower receiver, molten salt storage and steam generator)', *SolarPaces 2014, Elsevier's Energy Procedia*.

Qing Li, Fengwu Bai, Bei Yang, Baligh El Hefni and Sijie Liu, 'Dynamic simulation and experimental validation of Open Air Receiver and Thermal Energy Storage systems in solar thermal power plant', *SWC 2015 en Koré 2015*.

Liua S.J., Faille D., Fouquet M., El Hefni B., Wangc Y., Zhang J.B., Wang Z.F., Chen G.F and Soler R., 'Dynamic simulation of a 1 MWe CSP tower plant with two-level thermal storage implemented with control system', *SolarPaces 2014, Elsevier's Energy Procedia*.

El Hefni B., Bouskela D. Lebreton G., 'Dynamic modelling of a combined cycle power plant with ThermoSysPro', *MODELICA 2011 conference*.

FOUQUET L, EDF Pôle Industrie, CNET: Y.PM.X.000.PPPP.00.X.0872: PhuMy2.2: Overall plant operation description, 2004.

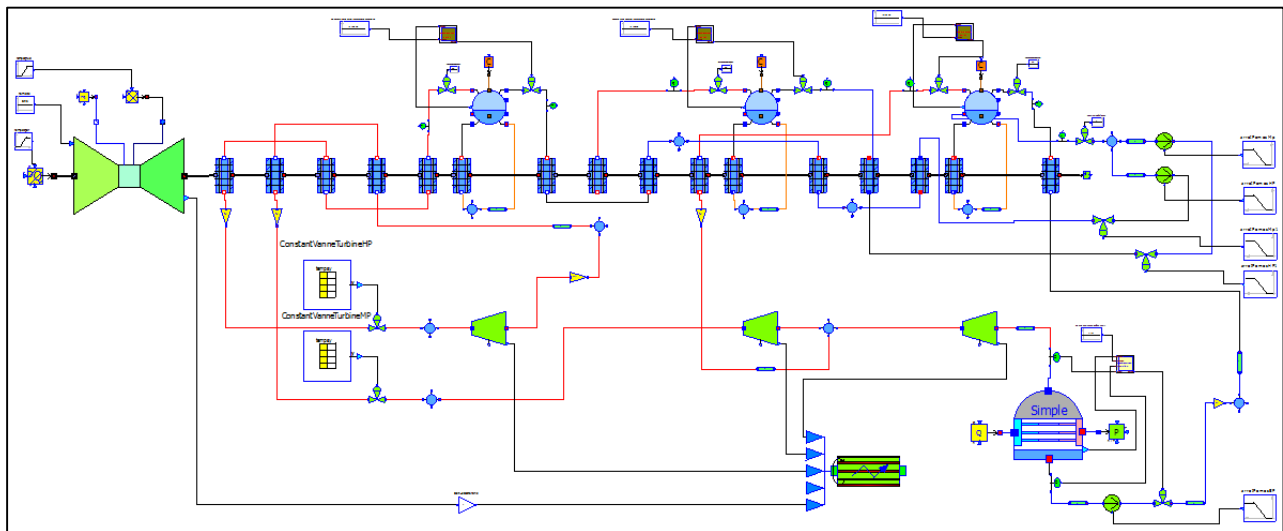


Figure 4. Model of the combined cycle power plant used for the power generator step reduction load.

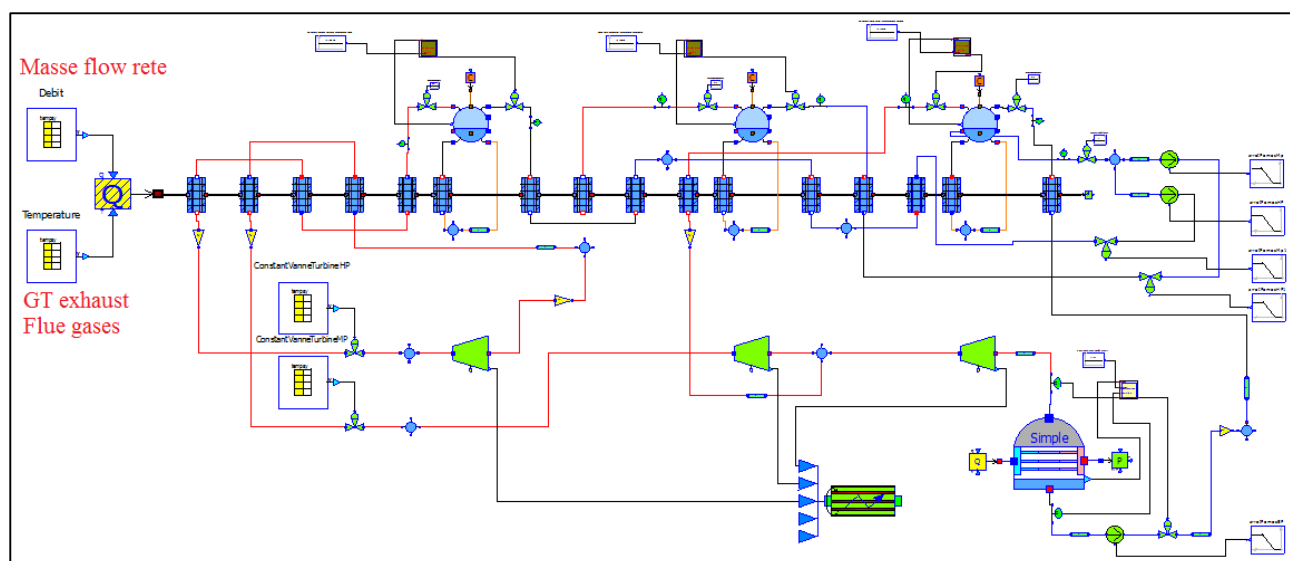


Figure 5. Model of the combined cycle power plant used for the full GT trip.

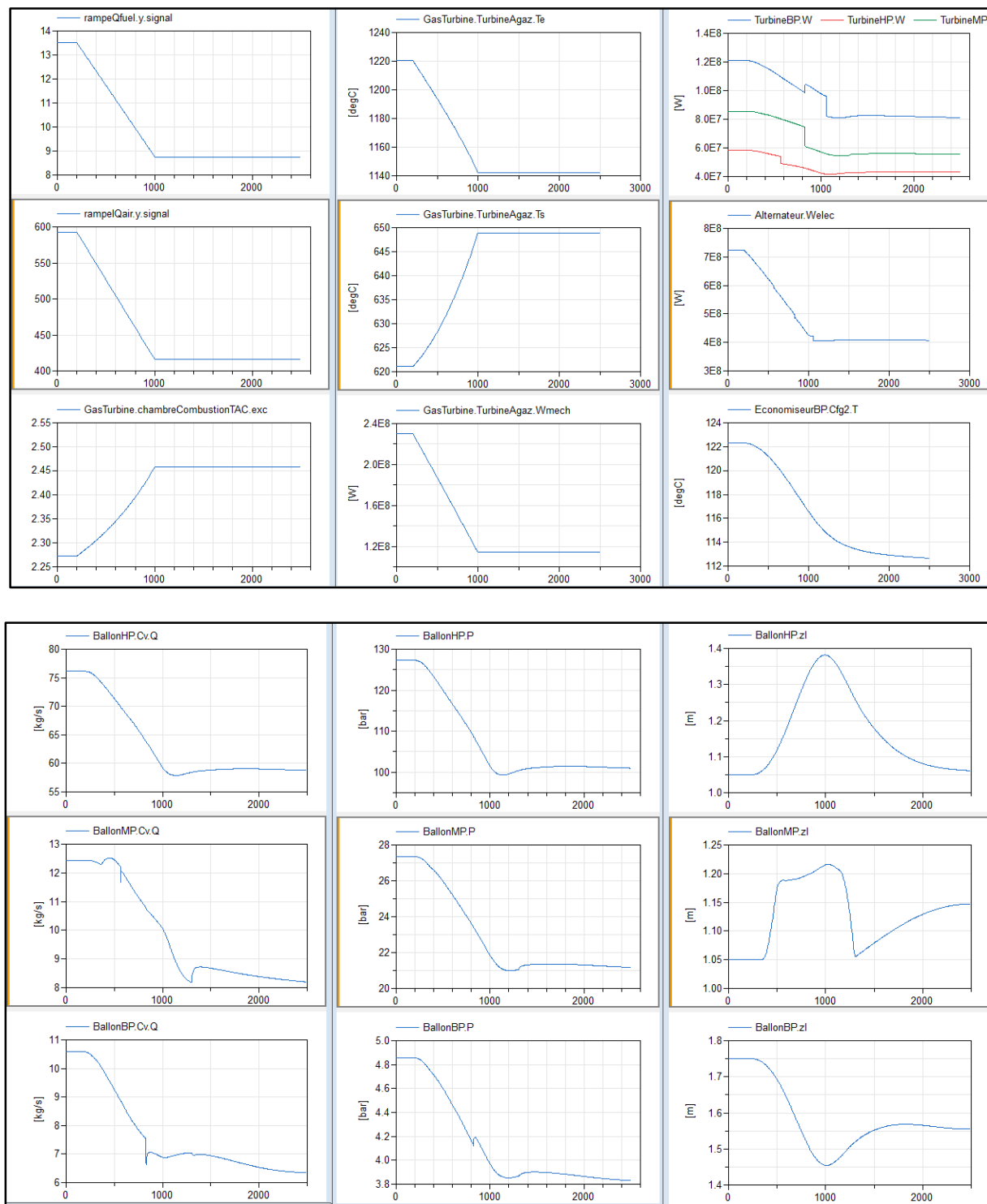


Figure 6. Power generator step reduction simulation (-50%): natural gas mass flow rate, air mass flow rate, excess air temperature at the inlet of the combustion turbine, exhaust temperature (gas turbine), mechanical power of the combustion turbine, mechanical power produced by each steam turbine, generator power, HRSG temperature at the outlet, steam mass flow rate produced in each drum, the drums pressure, and the drums level.

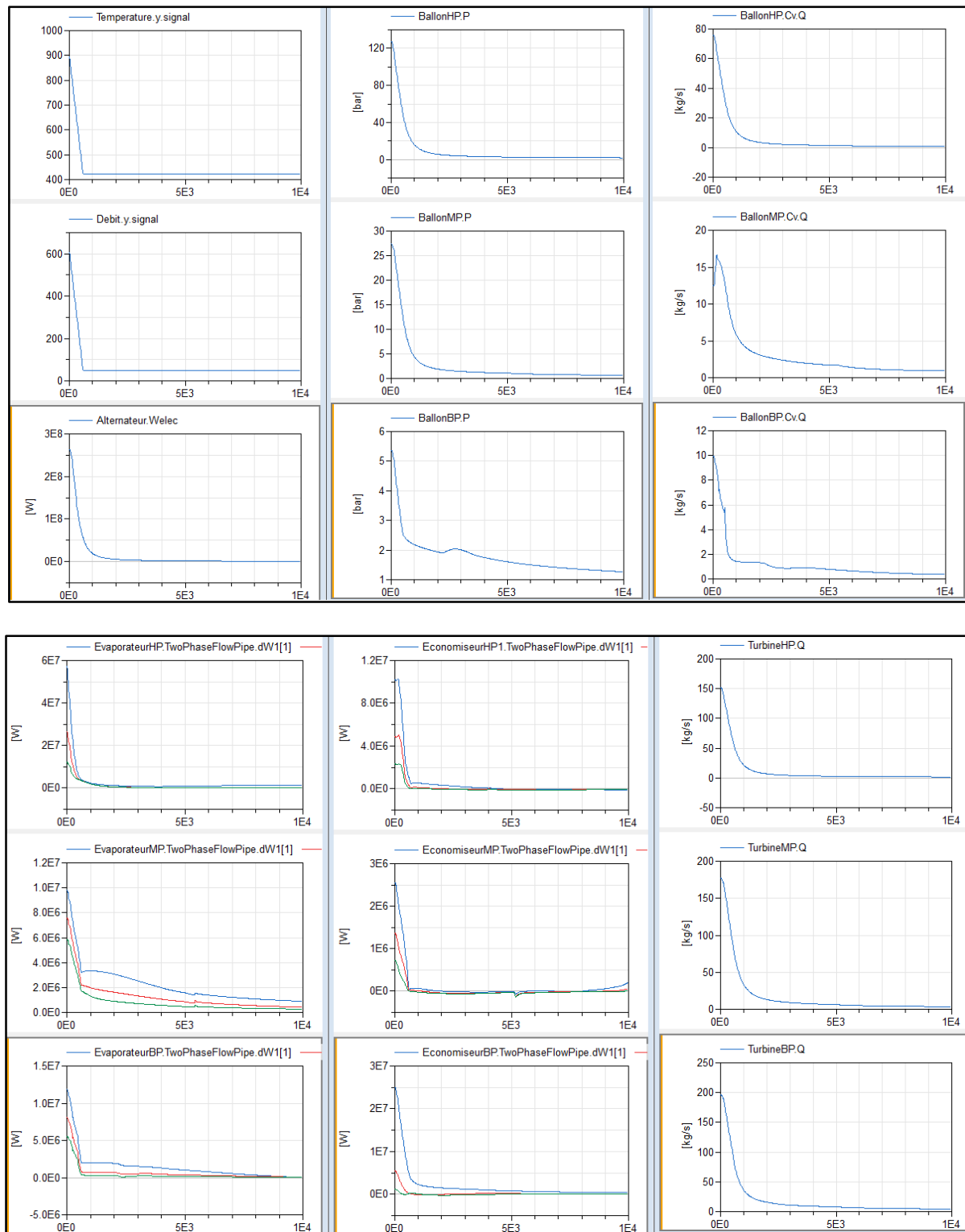


Figure 7. Gas turbine trip simulation: flue gases temperature at the inlet of the HRSG, flue gases mass flow rate at the inlet of the HRSG, generator power, the drums pressure, steam mass flow rate produced in each drum, thermal power produced in each heat exchanger (Evaporators HP, IP, LP and economizers HP, IP, LP), and steam mass flow rate in each steam turbine.

A Power-Based Model of a Heating Station for District Heating (DH) System Applications

Abdulrahman Dahash Annette Steingrube Mehmet Elci

Fraunhofer-Institute for Solar Energy Systems, Heidenhofstraße 2, 79110 Freiburg im Breisgau, Germany
{adahash, asteingr, melci}@ise.fraunhofer.de

Abstract

District Heating (DH) systems are often seen as a good practical approach to meet the local heat demand of the districts due to its ability to provide affordable and low carbon energy to the consumers. Yet, under today's regulations to renovate the buildings into more energy-efficient ones, the local heat demand is decreasing. Therefore, the operation of DH systems is also affected by the changing heat demand profile, which might lead to less profit for the operators of DH systems. Thus, the operators of DH systems strive for an optimal operation at which the heat demand is met and the profits are maximized. Due to the fact that these systems are complex-physical systems, therefore it is difficult to conduct any experimental investigation on them in order to examine the optimal operation. Accordingly, it is crucial to create fundamental models to investigate the optimal operation of such systems. In this paper, a power-based model is built to represent the heating station as part of a DH system. Then, the model is validated using real data from an existing heating station in Freiburg, Germany. The validation results reveal that the goodness-of-fit for the model is held to be good enough to test it for operational optimization cases.

Keywords: *Modelica, Dymola, Dynamic Modeling, Heating Station, District Heating System, Power-Based Model, Optimization.*

1 Introduction

District heating (DH) is considered a promising technology to improve the energy efficiency of the space heating systems in buildings (i.e. residential, commercial and industrial) (Olsthoorn et al., 2016). Thus, a greater interest in installing DH systems has arisen in many countries such as European countries, China and Russia. (Jie et al., 2015). DH systems have many advantages, for instance, an optimum use of fuel and thereby limitation of pollution (Benonysson et al., 1995).

Generally, a DH system is represented by transmission networks employed to supply heat from supply side (i.e. generation site) directly and/or indirectly to demand side (i.e. end users) to meet their

space heating and domestic hot water (DHW) demand as shown in Figure 1.

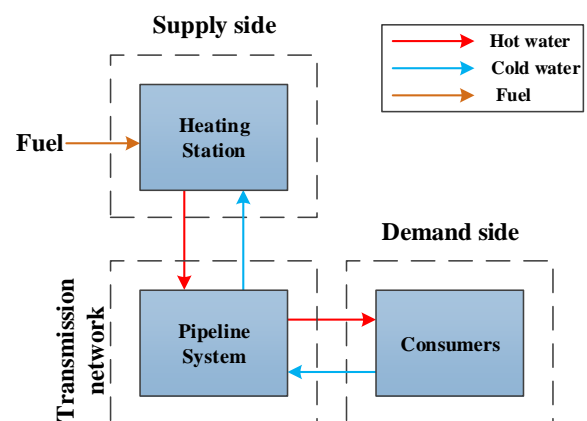


Figure 1: Generic block diagram of a district heating system

Often, DH can be coupled either with centralized heating stations and/or distributed heating units. Thus, numerous kinds of heat generation technologies (boilers, cogeneration plants, heat pumps, etc.) and energy sources (fossil fuels, renewable energies, etc.) can be adopted (Joelsson et al., 2008).

In Germany, combined heat and power (CHP) based DH systems are often seen as a key solution to meet the local heat demand in buildings and, therefore, these CHP units are frequently heat-driven (Elci et al., 2015). This operation mode compared to electricity-driven mode is often seen as the most economical and ecological option and therefore preferred by most operators of distributed energy systems due to the utilization of produced heat to meet the local heat demand and, therefore, no heat is wasted (Shipley et al., 2008) (Bracco et al., 2013). While the produced heat is utilized, the generated electricity is fed into the national power grid either at a fixed tariff, or at a variable tariff that is depending on the electricity price at the European Energy Exchange (EEX). However, because of the refurbishment of buildings to be more energy-efficient, there is a significant change in the heat demand profile of the buildings. Accordingly, this changing profile of the heat demand has a major impact on the operation of CHP units in DH system,

creating the first challenge in this field by disrupting the viable heat-driven regime mode.

Moreover, a CHP-based DH system can help the power grid to work smoothly by generating electricity when the renewables share in the grid is low due to calm-dark weather (e.g. low solar irradiation) (Kelly et al., 2009). Thus, another challenge has arisen in the field of DH systems, especially in cases when the heat demand is low in the buildings, the storage system is full and it is necessary to operate the heating station to overcome the fluctuations from renewables.

Considering these challenges (continuous changing heat demand and renewables fluctuations), the performance of DH systems (CHP-based) in different operation regimes has to be examined, in order to achieve optimal operation in which the system responds quickly to deviations in electricity prices and distributes the load among the heat sources in the system, so that the highest possible financial gain is achieved while simultaneously the heat demand is fully met. It is surely challenging to achieve this optimization unless fundamental models are built to help in investigating the performance of energy systems under different circumstances.

In this paper, the authors present an approach for modeling of heating stations for DH system applications. The presented model is a power-based model and, therefore, it shows the amount of energy flow between the different generation technologies in the heating station (supply side). The advantages of this modeling approach are less simulation time, better understanding of the energy flow influence on the heating station's operation and assistance in developing power-based control strategies for achieving optimal operation. Whereas the limitation is that the thermo-hydraulic aspects (e.g. pressure, flow rates) are neglected.

2 Methodology

2.1 Case Study

As case study, a district in the city of Freiburg in the south of Germany was used. This district is called (Weingarten) and was built in the 1960s and has 9,000 inhabitants. The western part of Weingarten has a population of 5,800 and an area of 0.3 km², with a gross floor area of about 271,240 m². The gross floor area comprises: 16-floor residential tower block buildings, 8-floor and 4-floor blocks of flats and non-residential buildings. Under current regulations regarding comfort, energy efficiency and modern building technology, the buildings in the western part have to be renovated to match current requirements.

This refurbishment works contains modernizing the district's buildings, renew Weingarten's energy supply system and operate it optimally. Figure 2 shows a site plan about the refurbishment area in Weingarten district in which the red colored buildings are the targeted buildings for refurbishment.

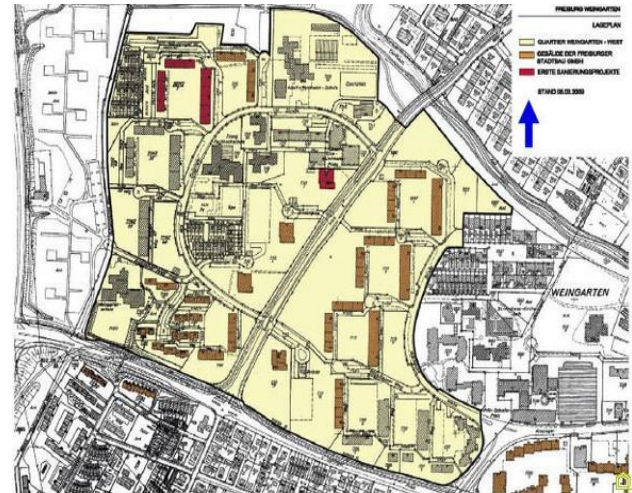


Figure 2: Site plan of the refurbishment area: the planned to be renovated buildings are colored red (Foschung für die Energieeffiziente Stadt, 2016)

The heat supply is delivered by a central heating station that supplies heat to two districts (i.e. Rieselfeld, Weingarten) via a DH network as shown in Figure 3 below. In the heating station, the annual heat generation is 67,400 MWh/a, and maximum heat output is 26,000 kW and, therefore, 6 gas-fired CHP units are installed and the operation of them is mainly heat-driven. Consequently, two CHP units are operating almost continuously year-round to meet the baseload. The six CHP units produce a total electrical power of 7,200 kW_{el} and a heat output of approximately 9,600 kW_{th}. The CHP modules attain an average of 5,650 full load hours yearly. Hence, over 75 % of the annual amount of heat produced comes from CHP units, while the remainder is generated by peak boilers. Also, in order to achieve smooth operation of the CHP units, there are two heat storage systems with a total capacity of 360 m³. They help in meeting the demand over short periods. Additionally, three gas-fired boilers each with 9.3 MW are employed for peak loads.

Due to the fact that CHP units can produce both heat and electricity, the electricity from the six CHP units is fed into the power grid while the heat produced is used to cover the heat demand.

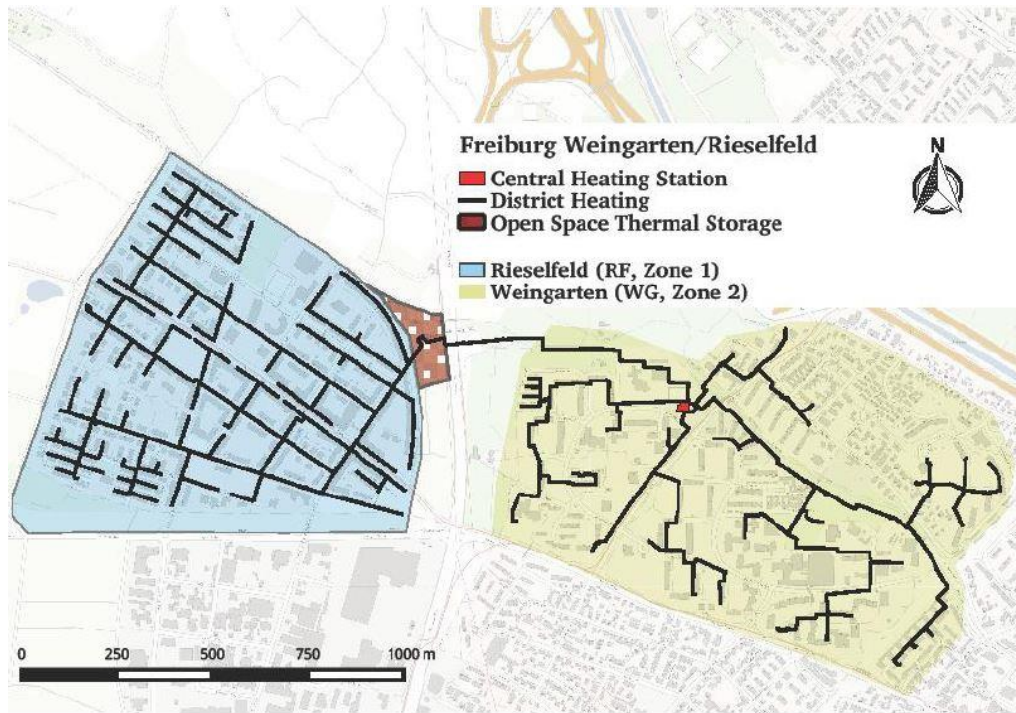


Figure 3: Top view of the Weingarten and Rieselfeld district with the central heating station and DH network (Bachmaier et al., 2015)

2.2 Modeling of the Heating Station

The model represents the real heating station in which an equivalent boiler component is used to represent the three peak boilers. Also, the CHP units are modeled to show the amount of energy flow while the storage system is modeled as stratification water storage. The modeling allows the different temperature segments to be shown within the storage system. In this work, the Modelica standard library (MSL) for basic components (e.g. prescribed heat flow, sensors and etc.), while buildings library is used for thermo-hydraulic components (e.g. flow sources/sinks, storage etc.).

2.2.1 Consumer

This component represents the demand side to which the heat shall be supplied. Therefore, it has two ports, one of which is an output signal for heat demand and the other an input signal for heat supply. The Consumer component is afterwards connected with the first controller in the heating station (*1st CHP controller*) and is backwards connected with the heat supply collectors using the different technologies. Also, the heat demand profile is read from a text file that is implemented in component (*Heat_Demand_Profile*) as seen in the left part of Figure 4.

This component plays a major role in the instant energy balance. For example, if the heat demand is higher than the heat supply it signals negative energy flow as seen in the gain component in Figure 4. Then

the signal is translated in order to operate the heating station and therefore:

$$\dot{Q}_{\text{demand}} = \dot{Q}_{\text{supply}} \quad (1)$$

Also, Figure 4 shows that there is a water source (a pump) implemented in the consumer model, the function of this component is to provide water with a predefined mass flowrate and temperature. Then the supplied water gets a signal of the required heat demand with a negative sign and a signal of the supplied heat simultaneously in order to inspect the energy balance and to fulfill it.

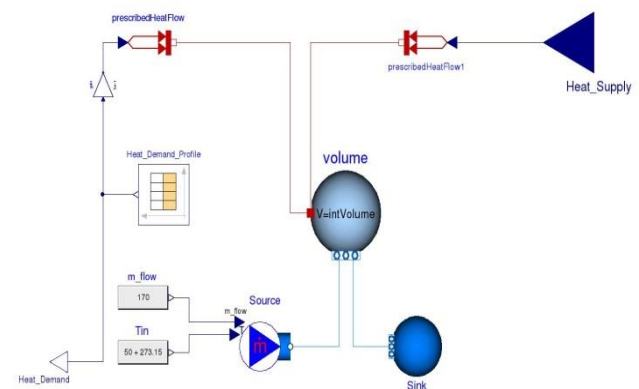


Figure 4: Structure of consumer component

2.2.2 Heat Sources

2.2.2.1 CHP unit

The CHP unit has a Boolean input that works as an on/off button, and an output that gives the amount of heat produced by this unit.

For the sake of simplicity in the modeling of CHP units, the following assumptions are made:

1. The response time of the components is included in the model. In reality, all the mechanical or electrical equipment has a certain response time (Smit, 2006). However, in this model, the response time is approximated to 1800 seconds for the entire CHP and it is given in the delay component that receives and sends Boolean signals as shown in Figure 5.
2. CHP units do not run at partial loads, they run only at full loads. When the CHP units run at partial loads, the thermal and electrical efficiency of the CHP units are different than the nominal values. Hence, only full load operation is considered as it is also valid in the actual system, Weingarten heating station.

$$\dot{Q}_{\text{CHP}} = \begin{cases} 1.5 \text{ MW} \\ 0 \end{cases} \quad (2)$$

3. The system is not modeled as a closed loop, meaning that the supply and return temperature of the water or steam in the system is not controlled. This assumption is made to reduce the run time of the simulation.

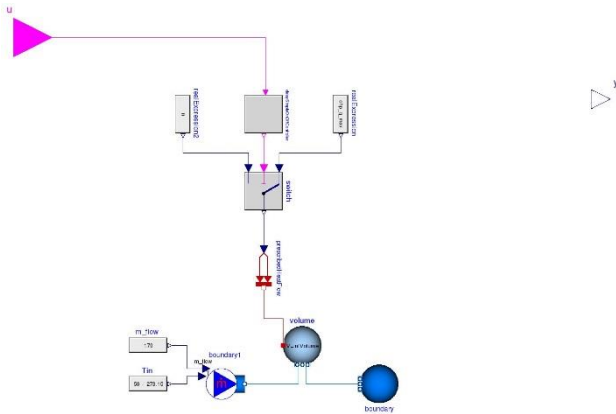


Figure 5: Structure of CHP unit component

2.2.2.2 Hot Water Storage

The main storage component has 2 ports which are input signals; one represents the amount of heat charging while the other is the amount of heat discharged.

Figure 6 shows the structure of the storage component. Obviously, the charging and discharging ports are connected to the water tank. As the heat flow direction is crucial to the discharging process, therefore the discharge value is provided as a negative value.

The water tank component representing the storage tank itself was largely built and developed by Lawrence Berkeley National Laboratory (LBNL) and can be found in the buildings library (Wetter, 2016).

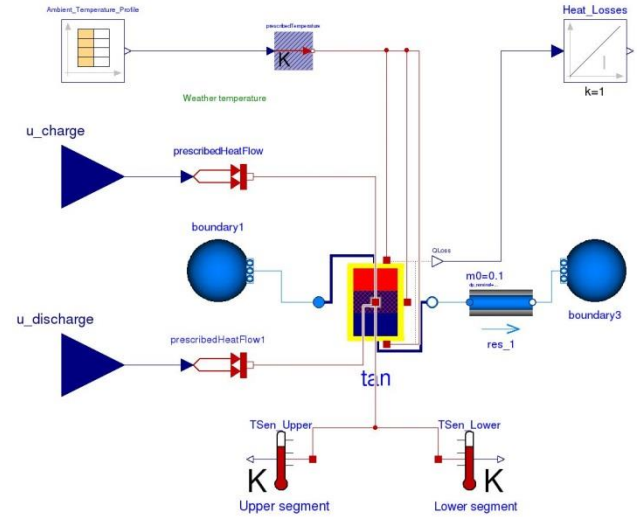


Figure 6: Structure of storage model

The water tank, which is cylindrical in shape, loses heat to the environment due to heat transfer mechanisms arising through the walls of the tank because of the different temperatures. Thus, it is essential to obtain the optimal storage volume by reducing the Surface Area (SA) to its acceptable minimum value and increase the storage volume to the maximum value. Therefore, it is assumed that the tank height is twice its radius, to achieve the minimum SA and maximum volume (Dearling et al., 2006):

$$V = 2\pi \cdot r^2 \cdot h \quad (3)$$

$$SA = 2\pi \cdot r^2 + 2\pi \cdot r \cdot h \quad (4)$$

$$h = 2r \rightarrow \max V \text{ and } \min SA \quad (5)$$

Due to technical restrictions regarding storage, it is decided to set the minimum temperature in the storage system (maximum temperature at last segment) to 70°C and the maximum supply temperature from the storage is set to 100°C. According to which the thermal storage capacity can be obtained by the following equation:

$$E = m \int_{T_1}^{T_2} c_p(T) \cdot dT = m \cdot \bar{c}_p(T) \cdot \Delta T \quad (6)$$

This restriction plays a key role in reducing heat losses from the tank. Heat losses are calculated within the model, taking into account the ambient temperature as Figure 6 shows. Ambient temperatures are given as a measurement and implemented in the system in order to show the real behavior of the storage system. Moreover, "T_Sen_Lower" component is the temperature sensor for indicating the temperature at lower segment of the tank while "T_Sen_Upper" is used as an indicator for CHP units to decide whether storage can be discharged or not.

The thermal conductance of the tank is important as it influences the heat losses from the tank. Thus, it is

calculated in such a way that the influence of the storage medium is excluded. This assumption is true because the impact of the storage medium on the thermal conductivity of the tank is very small. Therefore, the conductance is calculated as below:

$$G = \frac{k}{L} \cdot SA \quad (7)$$

The insulation layers of the tank are made of polystyrene which has a thermal conductivity of 0.03 W/m.K with a thickness of 0.1 m (Terry et al., 2012). Hence, the conductance is automatically calculated as a function of the storage surface area for any given storage volume.

2.2.2.3 Boiler

The boiler component has only two ports, one an input and the other an output. These ports determine the required demand and the supply from the boiler.

Figure 7 shows the configuration of the boiler in Dymola. The boiler input is clearly seen by the port “u” which is the remaining demand, and “y” represents the boiler output. Also, the boiler input is the remaining heat demand after all CHP units run and the storage capacity is discharged.

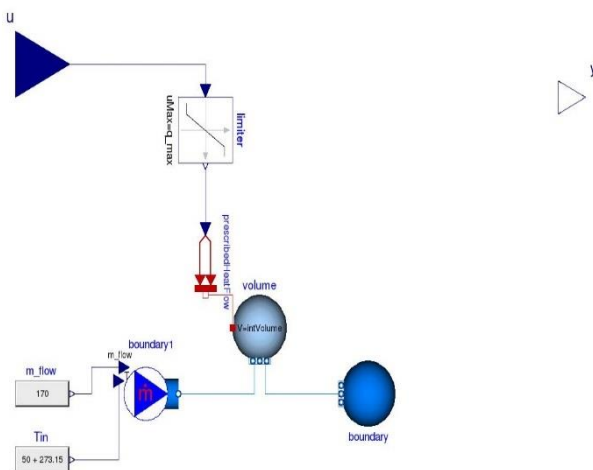


Figure 7: Structure of boiler component

2.2.3 Central Controlling

2.2.3.1 CHP Controller

Similar to the real heating station, the bottom segment temperature for storage is set to 70°C and the upper one is set to 100°C. Moreover, for each CHP unit, an individual CHP controller is installed in the system. In this controller, the heat demand and the storage temperatures (upper and bottom) are simultaneously checked. From Figure 8, it can be

clearly seen that there are 3 cases to run the CHP unit, which are:

1. **Power case (a):** if the heat demand is higher than the nominal CHP's heat output and the temperature of the bottom segment is higher than 70°C, then the CHP unit runs.
2. **Power case (b):** if the heat demand is higher than nominal CHP's heat output and the temperature of the bottom segment is lower than 70°C, then the CHP unit runs.
3. **Power case (c):** the CHP unit runs, when the following conditions are all true:
 - i. The heat demand is lower than nominal CHP's heat output, and
 - ii. The heat demand is higher than 95 % of the CHP's heat output (equals 1.425 MW), and
 - iii. The upper storage temperature is lower than 95°C.

Regarding power case (a), as the storage temperature is equal to or higher than 70°C, this means the storage can be discharged. On the contrary, if the storage temperature is less than the set bottom temperature (70°C), this means the energy stored in the storage system cannot be used and, therefore, power case (b) is activated to supply the heat directly to the consumers. While power case (c) is activated in order to cover the heat demand that is higher than 1.425 MW and the remaining of the heat output charges the storage.

Moreover, if the heat demand (or the remaining heat demand for CHP 2-6) is less than 1.425 MW or the upper storage temperature is higher than 95°C, then the corresponding CHP unit turns off. Moreover, if the bottom storage temperature is set to a constant value (i.e. 70°C), then a strange behavior for CHP units is seen because the CHP unit starts ramping up and down between 0 and the maximum heat output in order to maintain the storage temperature at the exact-desired level. This results in some problems with the modeling. This problem is seen in winter season because the heat demand is high, therefore the storage cannot be charged, and so the temperature cannot be kept above the minimum level. However, keeping the temperature right at a specific temperature is not necessary for the model. Nevertheless, obtaining accurate results for validation is of importance. In order to avoid such problems in modeling, the minimum temperature shall be set in a specific range, so instead of taking a fixed minimum temperature of 70°C, it is taken between 68°C and 70°C. For this reason, a “hysteresis” component from Modelica standard library (MSL) itself is implemented to solve the above mentioned problem.

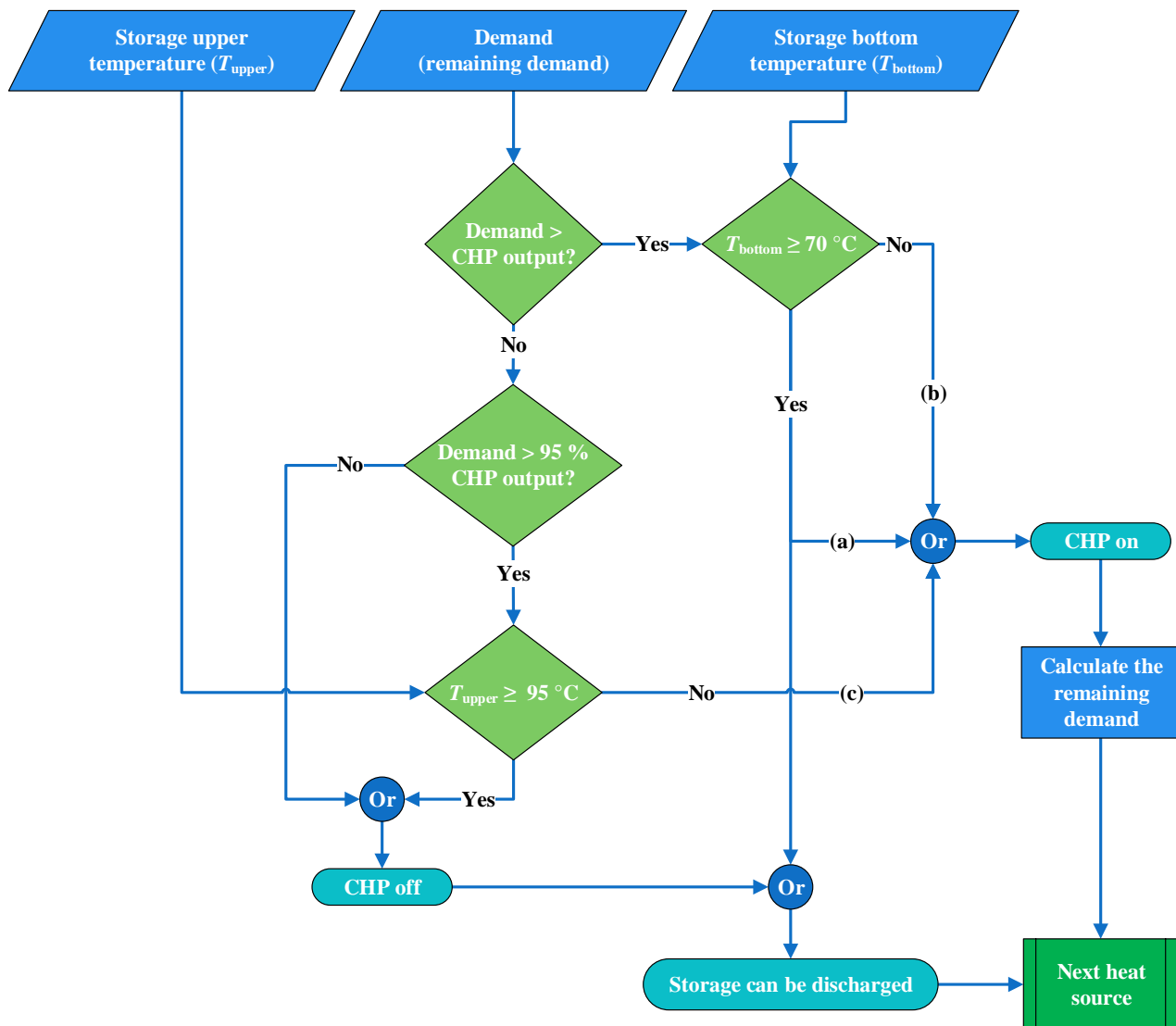


Figure 8: CHP controller flowchart implemented for each chp unit

2.2.3.2 Storage Controller

This controller plays a secondary role in the energy balance of the entire heating station next to the consumer component, since it gives a signal to discharge or charge the storage system. It has 3 input signals and a single output signal. One of the input signals is the storage system temperature at the bottom of the storage tank. Based on the temperature, a decision is made as to whether the storage system can be discharged.

However, if the temperature of storage's bottom is higher than 70°C, this sends a true signal to the switch component to discharge the storage system to cover the remaining demand. Otherwise the output “y” equals zero when the temperature is less than 70°C. The remaining demand is given by the following equation:

$$\dot{Q}_{\text{storage}} = \dot{Q}_{\text{demand}} - \sum_{i=1}^6 \dot{Q}_{\text{CHP},i} \quad (8)$$

Occasionally, the storage system cannot be discharged because the last segment temperature is less than that allowed for discharging, and therefore the remaining heat demand proceeds to the next controller, which is the boiler controller that runs the boiler in order to meet the required amount of heat.

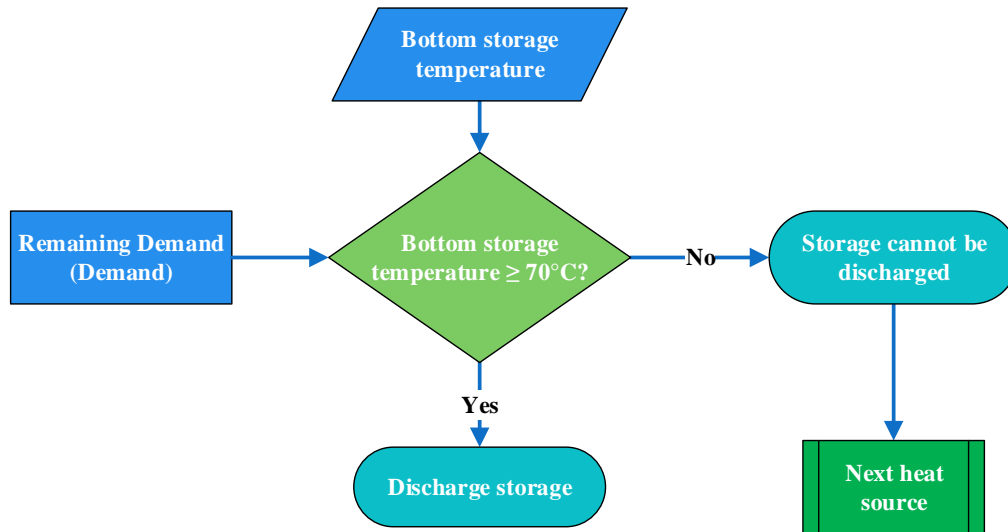


Figure 9: Storage controller flowchart

2.2.3.3 Boiler Controller

The boiler controller is a simple unit which computes how much heat demand remains after the total output of the CHP units and the discharged capacity of the storage system as Figure 10 shows. Next, it gives an output signal to run the boiler in a partial mode to meet the remaining heat demand, thus:

$$0 \leq \dot{Q}_{\text{boiler}} \leq 27.9 \text{ MW} \quad (9)$$

Here, the remaining demand is computed as below:

$$\dot{Q}_{\text{boiler}} = \dot{Q}_{\text{demand}} - \sum_{i=1}^6 \dot{Q}_{\text{CHP},i} - \dot{Q}_{\text{storage}} \quad (10)$$

The term \dot{Q}_{storage} refers to the usable heat in the storage system. Therefore, the usable temperature lies between 70°C and 100°C.

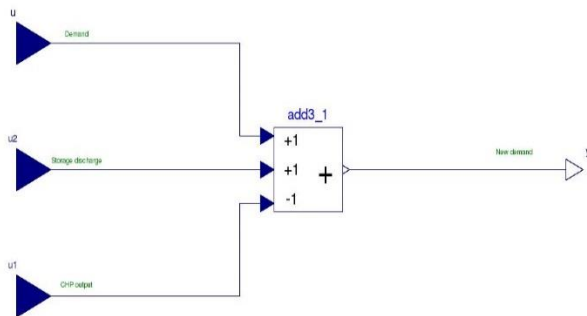


Figure 10: Structure of boiler controller

3 Validation

It is held that models with coefficient of determination, $R^2 \geq 0.7$ and coefficient of variation of root mean square error (CV-RMSE) $\leq 7\%$ are arbitrarily deemed to be “good” models (Reddy et al.,

1997). Whilst models with CV-RMSE less than 5% can be considered excellent models, those with less than 10% can be considered good models, those with less than 20% can be taken as mediocre models, and those greater than 20% are considered poor models (Balci, 1998). However, the constraints that are set in this article for the evaluation of the goodness-of-fit for the model are: $R^2 \geq 0.7$ and $\text{CV-RMSE} \leq 15\%$. Then it can be said that the model is held to be good.

In validation process, the data sets of the CHP units (*both simulated and monitored*) are required for validation purposes. This is because the variation between them is important as they are the first heat source that runs in order to meet the heat demand, and they are therefore the most influential parameters, with any disruption in their output having an impact on the other energy systems.

First, the model is visually validated for 9 days of January as a representation of winter season (high heat demand period) as Figure 11 shows.

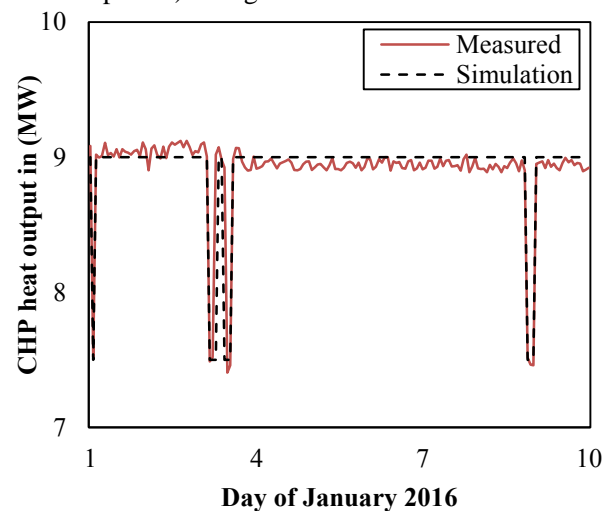


Figure 11: CHP heat output for 9 days of January 2016

Then numerical validation is performed and the results are as below:

$$R^2 = 0.84$$

$$CV - RMSE = 2 \%$$

The result of $CV - RMSE = 2 \%$ indicates that only this percentage of the real data is not explained by the model.

As the measured data are available for other periods, it is more obvious to validate another period of winter season and, therefore; another 6 days of February (from 5th to 11th of February) are taken to examine the creditability of the model for winter season as shown in Figure 12.

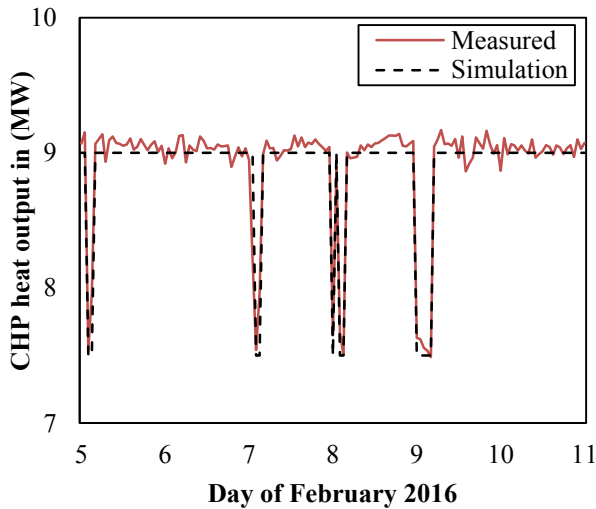


Figure 12: CHP heat output for 6 days of February 2016

The numerical validation results are:

$$R^2 = 0.85$$

$$CV - RMSE = 2 \%$$

These results confirm again that the model has a goodness-of-fit in the representation of the actual system and can be used to develop and test control strategies for the heating station. Nevertheless, an uncertainty analysis is crucial after the development in order to investigate the uncertainty percentage in the model for the developments.

Regarding summer season (relatively low heat demand compared to winter), a time series of 7 days out of April 2016 is chosen to evaluate the goodness-of-fit for the model and it is shown in Figure 13. The numerical results are as follows:

$$R^2 = 0.87$$

$$CV - RMSE = 7 \%$$

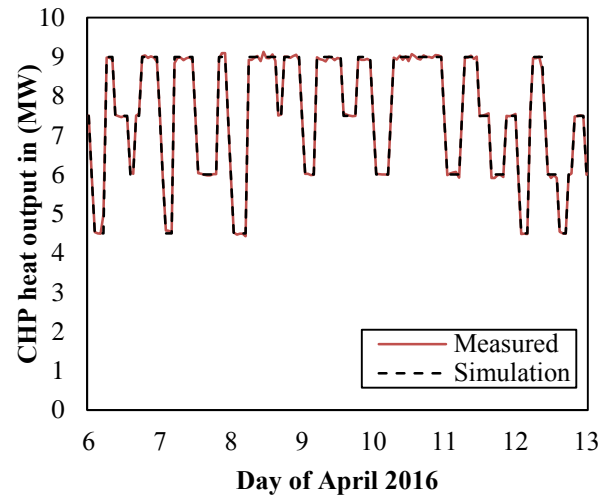


Figure 13: CHP heat output for 7 days of April 2016

For this time period, the R^2 value of 0.87 ($1 \geq 0.87 \geq 0.7$) indicates that the model can represent the real heating station with a good approximation of its real behavior.

As spoken earlier, due to the availability of measured data from the heating station for other periods, it is worthwhile to validate another time series from summer season. Thus, a time series of 5 days is taken from 9th to 14th May 2016 as Figure 14 shows.

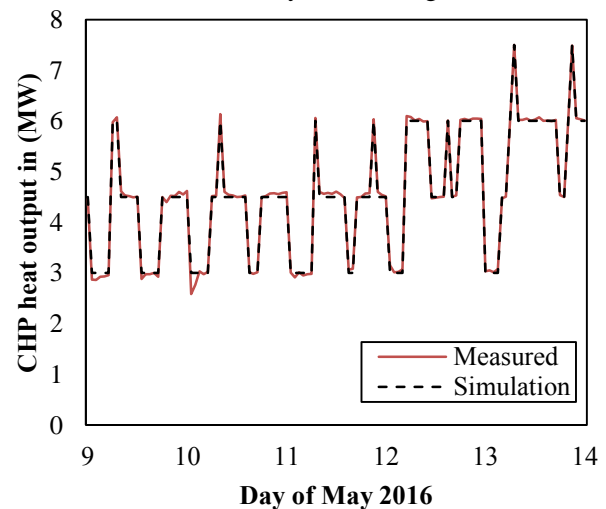


Figure 14: CHP heat output for 5 days of May 2016

Visually, the matching between both series is held to be “good” enough, thus proceeding to numerical validation:

$$R^2 = 0.84$$

$$CV - RMSE = 7 \%$$

The result of R^2 confirms again the goodness-of-fit for the model. While a $CV - RMSE$ value of 7 % indicates that the dispersion of the simulated and real data around the mean of the real data is quite low, and it is therefore clearer that the mathematical model fits the real heating station to a high degree.

4 Discussion and Remarks

The model constructed represents a particularly complex energy-supply system that comprises different energy sources and therefore there is a challenge in terms of energy system modelling and accurate prediction. Any given energy system is characterized by multiple parameters including material properties, casing temperatures and mechanical efficiency of the corresponding energy sources. In addition, there are equipment maintenance schedules, mechanical damage, HVAC and plant operation, real climate and many other parameters to consider. All together, these represent diverse sources for the uncertainty in the model. However, this does not mean that the model cannot fit the actual physical systems to an acceptable degree, but it does lead to a basic requirement to point out the sources of uncertainty.

The various sources of uncertainty in the model can be classified as follows:

1. **Specification uncertainty:** this kind of uncertainty refers to the physical errors that can arise from incomplete or inaccurate specifications for the complex physical model or process. It may also involve excluding some physical equations or properties, such as the geometry and material properties of the CHP units, boiler and storage system, and the fluctuating efficiency of the CHP units which is taken as constant in the model.
2. **Modelling uncertainty:** this arises due to the simplifications and assumptions about the complex physical state. It may also involve the exclusion of some energy systems due to their small effect on the model compared to the effort that is required in order to implement them in the model. An example is the exclusion of the CHP casing temperature which has an impact on total CHP efficiency. Moreover, due to the fact that the simulation results are discrete values while the real data are continuous as shown in Figures 11, 12, 13 and 14, this also has an impact on the creditability of the model.
3. **Operation uncertainty:** this involves external conditions that cannot be integrated into the model constructed because they are unexpected. This is mainly seen in case of damage or other unforeseen effects on the energy conversion chain or system. For instance, the mechanical damage that can occur in the pump or turbocharger of each CHP unit is always unexpected and cannot be predicted.

5 Conclusion

This paper presents the modeling process of heating stations for DH system applications using Modelica/Dymola to build a power-based model and then validate it with real data from an existing heating station (Weingarten). Validation results reveal that the

goodness-of-fit for the model is considered to be good enough, which permits employing this model for further research work to perform investigations for operational optimization. Furthermore, in (Dahash, 2016), the model is tested for some operational optimization methods and it shows good applicability to be used for power-based optimization methods.

Also, it is worthwhile to clarify that this paper (mainly validation results) does not confirm the applicability of the model with the shown controllers for any existing heating station. It simply reveals that the representation of a specific heating station is held to be good and then it states the sources of uncertainties in the model. Moreover, in order to look for other heating stations, their control strategies should be implemented and adjusted accordingly in the model.

Acknowledgements

This work is part of the project *Weingarten 2020 Monitoring* funded by the BMWi (Federal Ministry for Economic Affairs and Energy, Project No.: O3ET2364A). Our thanks go to the operator of Weingarten heating station, Badanova WärmePlus, for the cooperation in the project.

The model described in this article is built as a part of a master thesis supervised by Prof. Dr-Ing. Peter Treffinger and, therefore, the authors wish to thank him for his continuous support.

Nomenclature

Symbol	Description	Unit
CV	Coefficient of variation for	[-]
$-RMSE$	root mean square error	
G	Conductance	[W/K]
h	Height of the storage tank	[m]
k	Thermal conductivity	[W/m.K]
L	Thickness of the insulation	[m]
\dot{Q}	Heat flow rate	[kW]
r	Radius of the storage tank	[m]
R^2	Coefficient of determination	[-]
SA	Surface area	[m ²]
V	Volume of the storage tank	[m ³]

References

Bachmaier, A., Narmsara, S., Eggers, J. Bleicke and Herkel, S., 2015. Spatial Distribution of Thermal Energy Storage Systems in Urban Areas Connected to District Heating for Grid Balancing. *Energy Procedia*, Issue 73, pp. 3-11.

- Balci, O., 1998. Verification, Validation, and Testing. In: J. Banks, ed. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. New York: John Wiley & Sons, pp. 335-393.
- Benonysson, A., Bøhm, B. and Ravn, H.F., 1995. Operational optimization in a district heating system. *Energy Conversion and Management*, May, 36(5), pp. 297-314.
- Braccoa, S., Denticib, G., and Sirib, S., 2013. Economic and environmental optimization model for the design and the operation of a combined heat and power distributed generation system in an urban area. *Energy*, Volume 55, pp. 1014-1024.
- Dahash, A., 2016. *A Comparative Study of Modeling Approaches for District Heating Systems*, Master thesis, Offenburg-University of Applied Sciences, Offenburg, Germany.
- Dearling, C. and Erdman, W., 2006. Minimize the Surface Area of a Cylinder . In: *Principles of Mathematics 9*. 1st ed. Canada: McGraw-Hill, p. 640.
- Elci, M., Oliva, A., Herkel, S., Klein, K. and Ripka, A., 2015. Grid-interactivity of a Solar Combined Heat and Power District Heating System. *Energy Procedia*, 5 June, Volume 70, pp. 560-567.
- Foschung für die Energieeffiziente Stadt, 2016. *Projekt: Modellhafte Stadtquartierssanierung Freiburg Weingarten-West*. [Online] Available at: <http://www.eneff-stadt.info/de/pilotprojekte/projekt/details/modellhafte-stadtquartierssanierung-freiburg-weingarten-west/>
- Jie, P., Neng, Z. and Deying L., 2015. Operation optimization of existing district heating systems. *Applied Thermal Engineering*, 6 January, Volume 78, pp. 278-288.
- Joelsson, A. and Gustavsson L., 2008. District heating and energy efficiency in detached houses of differing size. *Applied Energy*, May, pp. 126-134.
- Kelly, S. and Pollitt, M., 2009. *Making Combined Heat and Power District Heating (CHP-DH) networks in the United Kingdom economically viable: a comparative approach*, s.l.: University of Cambridge.
- Nicola Terry, N., Palmer, J. and Cooper, I., 2012. *State-of-the-Art Review: Insulation and Thermal Storage Materials*, Cambridge, UK: Eclipse Research Consultants.
- Olsthoorn, D., Haghighat, F. and Mirzaei, P.A., 2016. Integration of storage and renewable energy into district heating systems: A review of modelling and optimization. *Solar Energy*, 15 October, Volume 136, pp. 49-64.
- Reddy, T. A., Saman, N. F., Claridge, D. E., Haberl, J. S., Turner, W. E. and Chalifoux, A. T., 1997. Baseline Methodology for Facility-Level Monthly Energy Use-Part 1: Theoretical Aspects.
- Shipley, A., Hampson, A., Hedman, B., Garland, P., and Bautista, P., 2008. *Combined Heat and Power, Effective Energy Solutions for a Sustainable Future*, s.l.: Oak Ridge National Laboratory (ORNL).
- Smit, R., 2006. *Power Quality and Utilisation Guide*, s.l.: Copper Development Association.
- Wetter, M., 2016. *Modelica Library for Building Energy and Control Systems*. [Online] Available at: <https://simulationresearch.lbl.gov/modelica> [Accessed 14 August 2016].

Model Based Design of a Split Carrier Wheel Suspension for Light-weight Vehicles

Jakub Tobolář¹ Daniel Baumgartner¹ Yutaka Hirano² Tilman Bunte¹ Michael Fleps-Dezasse¹
Jonathan Brembeck¹

¹German Aerospace Center (DLR), Institute of System Dynamics and Control, Wessling,

{Daniel.Baumgartner, Jakub.Tobolar}@DLR.de

²Toyota Motor Corporation, Future Project Division, Shizuoka, Japan,

Yutaka_Hirano@mail.toyota.co.jp

Abstract

Applying light-weight construction methods to the design of future electric vehicles results in weight reduction of both the vehicle body and the chassis. However, the potential for percental reduction of the sprung mass is larger compared to that of the unsprung mass. Consequently, unfavorable consequences on the compromise, which always needs to be found between road contact and road holding, can arise. This requires additional arrangements in order to reach the performance of a state-of-the-art conventional vehicle. This paper presents a possible design solution. The wheel carrier is split into two parts, thus enabling to tune the frequency response correspondingly to reference vehicles. Besides the technical solution the Modelica modeling of the proposed suspension system as well as a vehicle dynamics and ride comfort assessment are presented.

Keywords: *split wheel carrier, vehicle suspension, unsprung mass, small electric vehicle, three mass system*

1 Introduction

Recently, to contribute to lower carbon dioxide emissions, the development of light-weight electric vehicles (LEV) has become more and more active in the automotive sector. These LEVs usually consume less energy in comparison to conventional vehicles. Though, because the sprung mass of those vehicles tends to be reduced relatively more than the unsprung mass of the suspension, the ratio of sprung mass to unsprung mass is directed toward lower values compared to those of conventional vehicles. This means that the resonance frequency of the sprung mass becomes closer to the resonance frequency of the unsprung mass which results in possible reduction of the ride comfort.

Furthermore, the resonance peaks associated with the sprung mass and the unsprung one are shifted to higher frequencies compared to those of conventional vehicles because of reduced masses of both parts. These phenomena arises when the resonance frequency of the unsprung mass becomes close to the drive shaft twisting

mode. Thus, possibly a resonance excitation of the unsprung mass and the drive train can occur induced by adverse propelling torques.

To prevent abovementioned problems, it is necessary to shift the resonance frequency of the unsprung mass to lower / higher values and also to reduce the magnitude peak of the vertical acceleration frequency response associated with the unsprung mass. However, as shortly discussed in the next section, it is theoretically proven that the resonance frequency and the magnitude peak of the unsprung mass cannot be influenced by adjusting spring and damper coefficients of the conventional suspension which constitutes a two mass system of the unsprung mass and the sprung one.

To solve this design conflict, the idea of a three mass suspension system is introduced. In Section 2 of this paper, theoretical analysis of the effect of a three mass suspension is analyzed at first. A technical solution of the mechanical structure of the three mass system is described in Section 3. Actual effects of the three mass system are finally investigated by simulations using a Modelica model (presented in Section 4) of the proposed suspension together with an advanced multi-body vehicle model in Section 5. Section 6 provides a conclusion of the investigations results.

2 Basic Idea of the Three Mass Suspension System

Figure 1 (left) shows a schematic quarter car of a conventional "two mass suspension system". Here, m_b denotes the mass of the sprung mass (representing a quarter of the car body) and m_w is the mass of the unsprung mass composed of the wheel and its carrier block. Figure 2 shows a set of Bode magnitude plots from the acceleration of road excitation to the sprung mass acceleration of this two mass system. The parameter varied between the set of plots is the suspension spring stiffness c_{bw} in the upper chart and damping d_{bw} in the lower one. It can be concluded that the resonance peak associated with the sprung mass mode (second peak in the gain plot) can be changed neither by varying the spring stiffness, nor by variation of the damp-

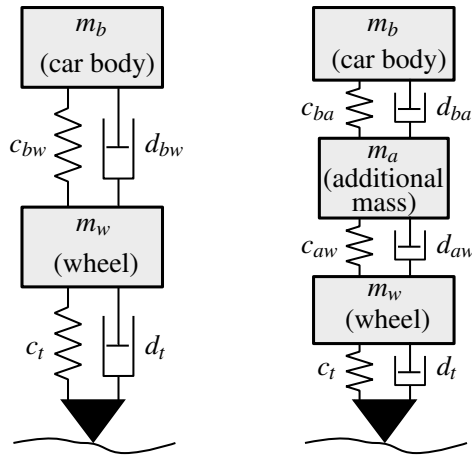


Figure 1. Schematic quarter car models of a conventional two mass suspension system (left) and of the three mass system (right).

ing of the suspension. This corresponds to the fundamental investigations of invariance properties of a two mass suspension system presented e.g. in (Hedrick and Butsuen, 1990) or (Savaresi et al., 2010), which emphasize that the sprung mass acceleration near to the wheel resonance frequency is invariant regarding body spring stiffness c_{bw} and body damping d_{bw} . To overcome this fundamental restriction, the serial three mass suspension system as shown in Figure 1 (right) is introduced. In this solution, the unsprung mass of the wheel carrier is divided into two parts being insulated by an additional spring and damping elements (c_{aw} and d_{aw} , respectively) from each other. A detailed theoretical comparison of three mass systems with classical two mass systems highlighting the advantages of three mass systems is given in (Ryba, 1974a) and (Ryba, 1974b).

Figure 3 shows a Bode magnitude plot comparison of the two mass system with a set of plots for the three mass system when changing the spring stiffness (c_{aw}) of the additional spring element. Here, the spring stiffness of the additional spring was set relative to the vertical tire stiffness c_t using a factor k as

$$c_{aw} = k \cdot c_t. \quad (1)$$

It is shown by Figure 3 that by selecting a proper value of the gain coefficient k such as $k = 1.0$, it is possible to shift the frequency of the mode related to the unsprung mass and also to reduce the magnitude of the resonance peak. After this encouraging preliminary result it was decided to design a mechanic realization of such a three mass suspension with split carrier which will be presented and discussed in the following section.

3 Technical Solution

The possible technical solution of the three mass system was developed assuming a small LEV with state-of-the-art double wishbone front and rear suspensions.

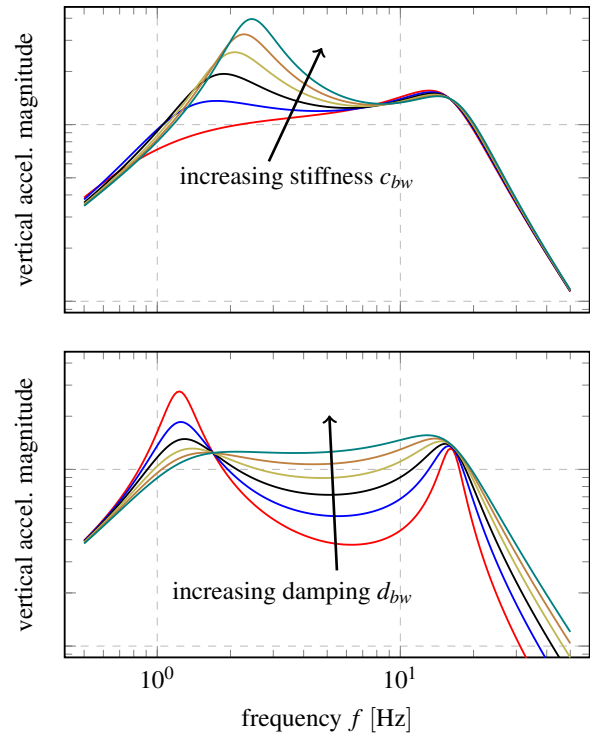


Figure 2. Bode magnitude plots from the acceleration of road excitation to the sprung mass acceleration of the two mass suspension system with varying spring stiffness (above) and varying damping (below).

The solution described below consists in splitting the wheel carrier into two parts – the wheel hub and the carrier itself – guided and suspended to each other. Thus, the lower part of the unsprung part (m_w) consists of a wheel, a tire, a brake disc and a carrier which supports the wheel hub bearing and the brake caliper. The upper part of the split unsprung mass (m_a) incorporates parts connecting suspension linkage mounts and the rest of the hub carrier. A mechanism to limit the relative motion between m_w and m_a has also to be considered. An additional spring element (c_{aw}) is assumed as a rubber bushing element or combination of a bushing element and a supplementary spring element.

Several technical solutions were examined to realize abovementioned arrangement. Finally, the solution with linear sliding mechanism, see Figure 4, was chosen. It consists of two sealed linear sliding bearings (light grey parts in Figure 4) connected firmly with the wheel hub (purple) and guided through the supporting wheel carrier structure (beige). This design resembles a mechanism of a conventional telescopic fork as utilized for motorcycle front suspension; see (Stoffregen, 2012). The advantages are a simple, sealed and long-time proven design and availability of standardized parts thus enabling relatively cheap production and assembly. Moreover, such a design offers high stiffness against external forces and torques. Consequently, the wheel attitude changes under common

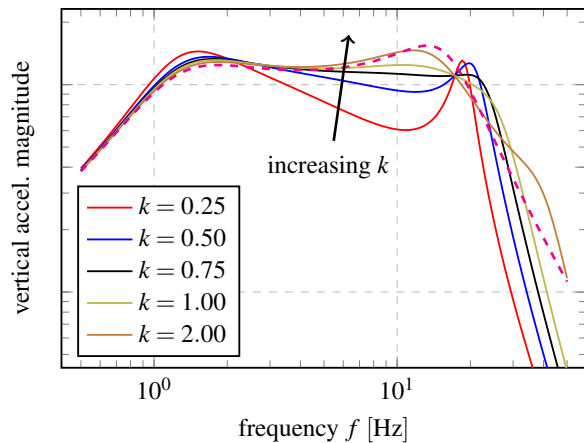


Figure 3. Bode magnitude plots of the three mass suspension system (using varying factor k according to equation (1)) compared to the conventional two mass suspension (marked with dashed magenta line).

operating loads can be minimized, as proven by vehicle dynamic tests, see Section 5.3. For the investigated LEV, this technical solution was applied for both the front and the rear suspension, the latter being of the driven axle.

The wheel carrier support structure is depicted in more detail in Figure 5. It consists of support tubes with integrated sliding bearings (part D in Figure 5, yellow) and a rubber sealing (part E, green/black/orange). The wheel hub (purple) together with the brake's caliper (not depicted in Figure 5) is attached to immersion tubes in order to provide vertical deflection only between the parts of the wheel carrier.

The vertical motion of the split wheel carrier is suspended and damped by a rubber bushing element (part A in Figure 5, dark grey) together with support coil springs (part C, red/orange) housed within one of the tubes (light grey). The load springs are supported by threaded head caps (part F, light red) for easy exchange and adjustment of the spring preload. A non-linear damping device (part B, pink) can be optionally installed in one of the immersion tubes in order to improve the damping behavior (if unsatisfactory) of the main bushing element.

Particular attention was given to the design of the suspension elements – the coil spring and the rubber bushing. To achieve consistent behavior of the suspension over a wide range of excitation frequencies – stimulated e.g. by road irregularities, the effect of the dynamic stiffening of the rubber has to be minimized, see e.g. (Mitschke and Wallentowitz, 2014). The parallel arrangement of the bushing and the springs enables shifting of the high portion of stiffness c_{aw} onto the coil spring, thus holding over 90 % of the total value. The remaining stiffness realized by the bushing is then relatively low, additionally resulting in low damping d_{aw} and reasonably compromised dynamic stiffening.

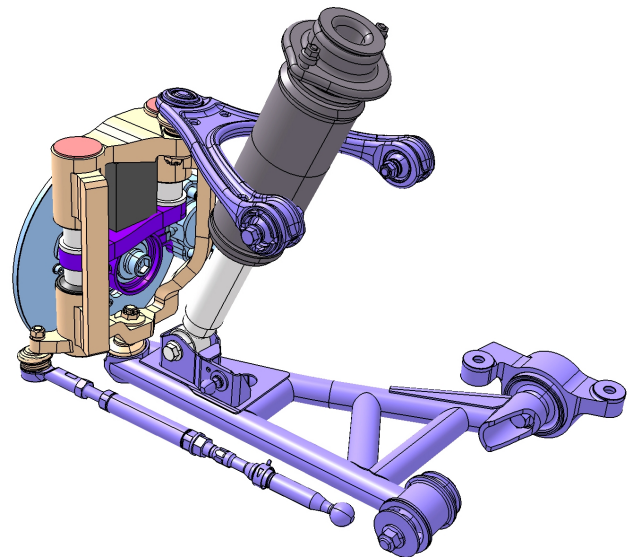


Figure 4. Technical solution of the split wheel carrier with sliding bearings in the context of a front right double wishbone suspension.

4 Modelica Model

To facilitate a simulative assessment the technical solution of the suspension presented in the previous section was modeled using Modelica. Using such a model, analyses on a multi-body quarter car model were performed, see Sections 5.1 and 5.2. Additionally, the suspensions were used within a total vehicle model for vehicle dynamics and driving comfort assessment, as documented in Sections 5.2 and 5.3.

In order to promote easy interoperability with the various existing automotive Modelica libraries, the created Modelica package containing all the models was consequently based upon the *VehicleInterfaces* base classes, see (Dempsey et al., 2006). The *VehicleInterfaces* library focuses on standardizing the assemblies interface definitions without presupposing a standard vehicle model architecture. Hence, the same assembly models can be reused in different model architectures.

The idea of template assembly models and parametrized models was introduced as also utilized in the *PowerTrain* library from DLR, see e.g. (Schweiger et al., 2005). Various parametrized models of realistic assemblies are thus inherited from template models which reflect different structure and level of model's detail. This facilitates redeclaration of the particular model within the overall vehicle models or virtual test rigs depending on the simulation purpose to be fulfilled.

The majority of virtual test rigs and maneuver scenarios as well as a couple of components and template models needed to model and evaluate the suspension concept had already been predefined within a DLR proprietary Modelica library for vehicle dynamics.

The multi-body models of a) the conventional two mass reference suspension (see model structure depicted in Fig-

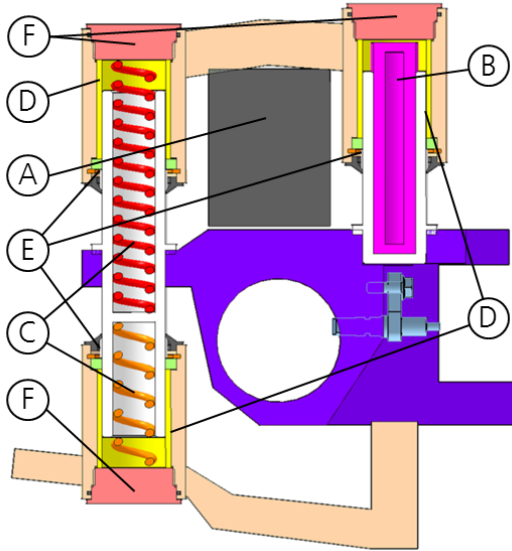


Figure 5. Cross-section view of the front right wheel carrier and components.

Table 1. Relevant masses of reference vehicle (2M) and of vehicle with proposed suspension solution (3M).

Parameter	Value [kg]	
	2M	3M
Total weight incl. driver of 75 kg	850	872
Front sprung mass m_b	191	189
Front unsprung mass m_w	26	23
Front add. unsprung mass m_a	N.A.	11
Rear sprung mass m_b	178	175
Rear unsprung mass m_w	29	23
Rear add. unsprung mass m_a	N.A.	15

ure 6) and b) the proposed split wheel carrier design (its structure is shown in Figure 7) incorporate all relevant mass parameters as well as the nonlinearity of the force elements. In both cases the models are parameterized according to the considered small LEV. The particular masses are listed in Table 1 for the conventional reference vehicle in comparison with the vehicle with split wheel carrier. A multi-body model of the latter is visualized in Figure 8.

For elastic wishbone mounts, a simplification was adopted in that the orthogonal deformations depend proportionally only on the corresponding action forces, thus

$$\begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} = \begin{bmatrix} f_x/c_x \\ f_y/c_y \\ f_z/c_z \end{bmatrix}, \quad (2)$$

and, correspondingly, for rotations. Such a mounting was additionally used to connect the toe control link of the rear suspension. For simplicity, the main bushing mounted between the wheel hub and the wheel carrier was modelled

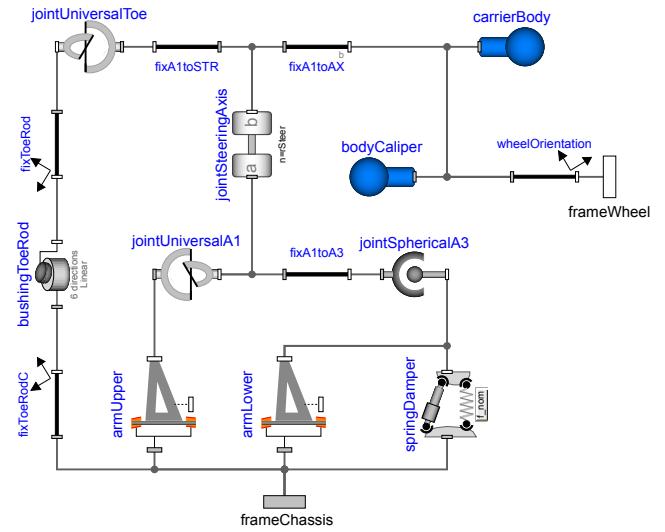


Figure 6. Modelica structure of the conventional rear suspension.

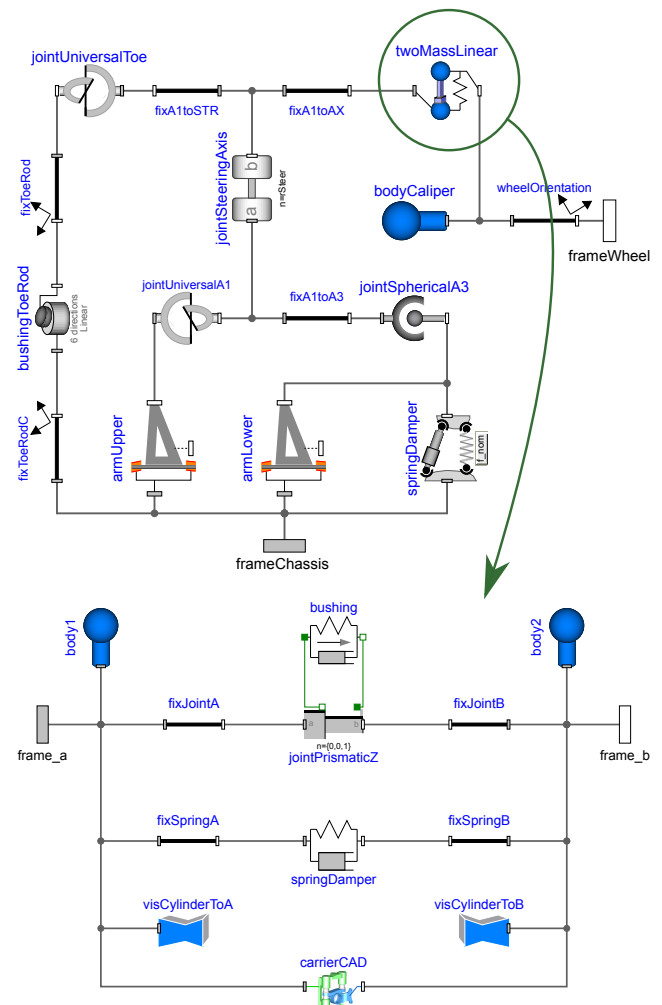


Figure 7. Modelica structure of the split wheel carrier rear suspension (above) and the split carrier submodel (below).

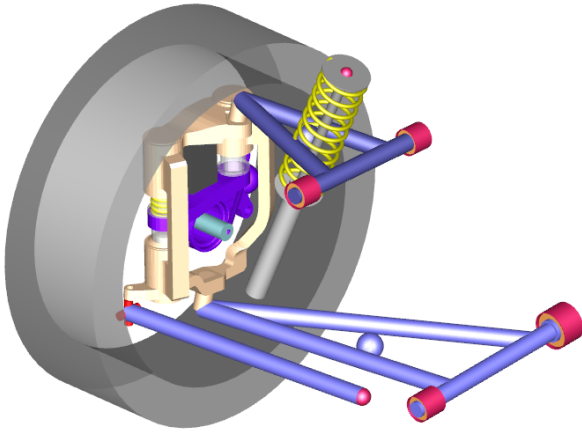


Figure 8. Visualization of the multi-body model of presented front suspension with split wheel carrier.

on the same principle. This is reasonable for low dynamic stiffening as explained in the previous section.

The vehicle body was modeled as a rigid structure with an extra mass of 75 kg at the location of a driver's hip joint. A linear spring-damper element was used to represent the vertical force/deflection of the tire. Additionally, the horizontal tire forces were modeled using the Pacejka Magic Formula (Pacejka, 2002) to assess the vehicle dynamics behavior.

In the following, the model of the vehicle of the two mass configuration and parametrization (2M in Table 1) is called *reference vehicle* or *simply reference*.

5 Simulation Results

For the proposed split wheel carrier suspension the ride comfort and tire/road contact were assessed simulating both the quarter car and the full vehicle model. Additionally, vehicle dynamics were evaluated utilizing the full vehicle model only.

5.1 Verification of the Technical Solution

For the first verification of the eligibility of the technical design, the frequency response analysis according to (Bunte, 2011) was performed on the nonlinear quarter car multi-body models with a) the reference suspension and b) the proposed one. The vertical excitation from 0.5 Hz to 30 Hz was considered, together with the frequency-dependent decrease of the amplitude in order to reproduce the amplitude progression of the road class D from (ISO 8608). Thus, the time excitation conditions were similar to those applied for linear system analysis depicted in Figures 2 and 3.

The comparison of time domain simulation results is done in Figure 9. The multi-body models prove the reduction of the second resonance peak by the three-mass system up to frequencies about 20 Hz. Above this frequency, the third resonance peak of the proposed suspension becomes significant which increases the response amplitude

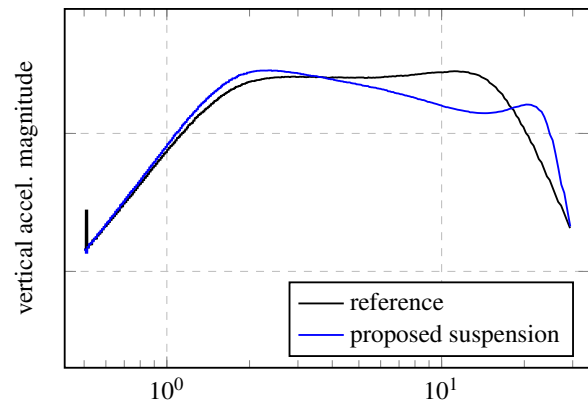


Figure 9. Frequency response of nonlinear quarter-car multi-body models.

compared to the reference vehicle. In summary, this proof of concept confirmed the preliminary assessment as discussed in Section 2.

5.2 Comfort Assessment

For ride comfort and tire/road contact assessment, a time domain simulation of the vehicle placed on a virtual four post rig was performed. The post excitation conforms to the road class D according to (ISO 8608) driven at constant velocity of 70 km/h. The road irregularities are generated by means of a colored noise signal matching a given power spectral density (PSD). For the generation of such a signal in Modelica, the *AdvancedNoise* library (Klöckner et al., 2015) was utilized.

For the full vehicle model, the vertical excitation signals for the left and right track were generated independent of each other, i.e. using uncorrelated signals. The excitation signals for the rear wheels were delayed by the ratio of wheel base and speed against the respective front wheels running ahead. The excitation signal together with its PSD are shown in Figure 10.

The generation of the excitation using a noise generator from the *AdvancedNoise* library has the advantage of a high quality stochastic signal. In contrast, this way of signal generation is computationally expensive and slows down simulation speed. Using DASSL as numerical solver on a quad-core personal computer the full vehicle simulation on average runs 50 times slower than real time. Therefore, we restricted the simulation time for the full vehicle simulation to 100 s which corresponds to a track length of 1944 m. For the less expensive quarter car simulations, the simulation time of 200 s was applied in order to reach a more significant assessment.

The evaluation with the quarter car model used the following criteria:

- RMS_{fz} : root mean square of vertical tire load,
- RMS_{az} : root mean square of body vertical acceleration,

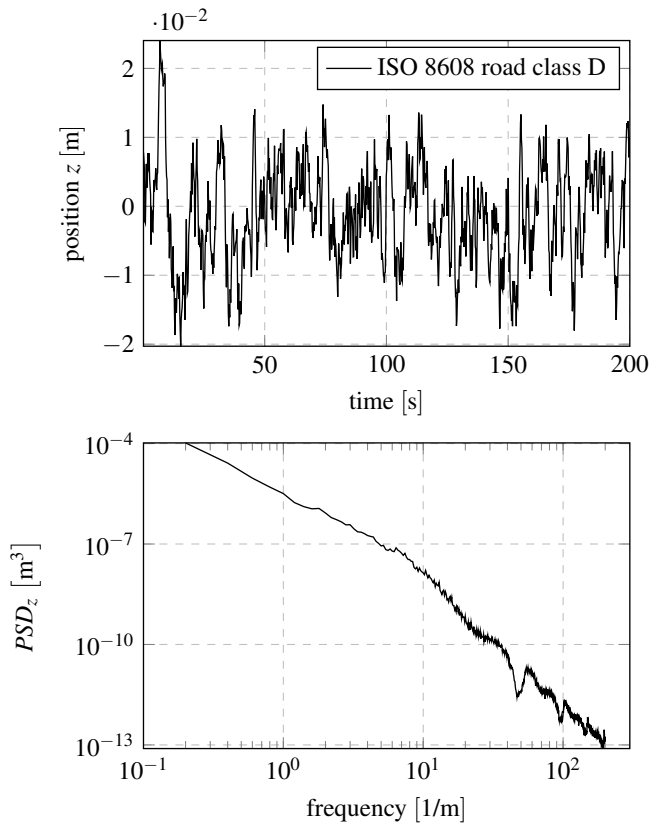


Figure 10. Exemplary road excitation signal used for vertical displacement of virtual post rig (above) and its power spectral density (PSD, below).

- K_{ges1} combined assessment criterion according to (Hennecke, 1995) for the time window of 10 s and
- K_{ges2} similar to K_{ges1} but for the total time of measurement.

For the full vehicle model, the following criteria were additionally evaluated:

- RMS_{pitch} : root mean square of body pitch angular acceleration,
- RMS_{roll} : root mean square of body roll angular acceleration.

The collectivity of criteria evaluated by simulations is shown in Table 2 and Table 3.

Supplementary to these concluding values, Figure 11 depicts the signals of the combined comfort assessment criteria K_{ges1} and K_{ges2} from simulations on the virtual quarter car test rig. These criteria show the improvement for the proposed suspension (plotted in red) against the reference vehicle (blue). For tire/road contact, the improvement of about 17 % is reached – as indicated by RMS_{fz} in Table 2. This trend is even more evident for the full vehicle simulation, see Table 3. Here, the comfort improvement is about 25 % and the road contact improvement is up to about 38 %. These results are achieved by

Table 2. Resulting comfort assessment criteria values for quarter car simulations with a simulation time of 200 s.

Criteria	Unit	Reference	Proposed suspension	Difference [%]
RMS_{fz}	N	112.50	93.30	-17.07
RMS_{az}	m/s ²	0.47	0.43	-7.89
K_{ges1}	–	9.13	8.50	-6.96
K_{ges2}	–	8.59	7.97	-7.24

Table 3. Resulting comfort assessment criteria values for full vehicle simulations with a simulation time of 100 s.

Criteria	Unit	Reference	Proposed suspension	Difference [%]
RMS_{fz}	N	702.9	433.2	-38.37
RMS_{az}	m/s ²	1.450	0.963	-33.59
RMS_{pitch}	rad/s ²	2.005	1.430	-28.68
RMS_{roll}	rad/s ²	4.707	3.462	-26.45
K_{ges1}	–	32.670	24.412	-25.28
K_{ges2}	–	34.260	26.135	-23.72

virtue of the proposed split wheel carrier system while using a preliminary parametrization.

5.3 Vehicle Dynamics Assessment

To achieve an evaluation of the proposed suspension concept also in terms of vehicle dynamics, two standard driving maneuvers were simulated: a) quasi steady state cornering on a circle with a radius of 40 m while slowly increasing the vehicle speed and b) steering angle sine sweep at a constant speed of 80 km/h. For a) the necessary steering wheel angle δ_H to keep the vehicle on the circle was recorded and the resulting self-steering gradient SSG was computed. During b) the magnitude $a_{y,gain}$ and phase angle $a_{y,phase}$ of the lateral acceleration frequency response were recorded and assessed.

The evaluation of the recorded criteria of both vehicles indicated just negligible difference for both maneuvers, i.e. the suggested technical solution of the proposed suspension has negligible influence on the lateral vehicle dynamics. This is particularly reached due to the sufficient stiffness of the mechanism when exposed to lateral and longitudinal forces and torques. Consequently, only marginal changes in the wheel attitude can be observed in most cases, as demonstrated in some extent in Figure 12 for a toe angle change under lateral force load, and in Figure 13 for a track change under variable vertical load. Both Figures result from the suspension elastokinematics analysis.

Concluding from the two simulated maneuvers, the vehicle equipped with the proposed suspension appears to have also a good driving behavior.

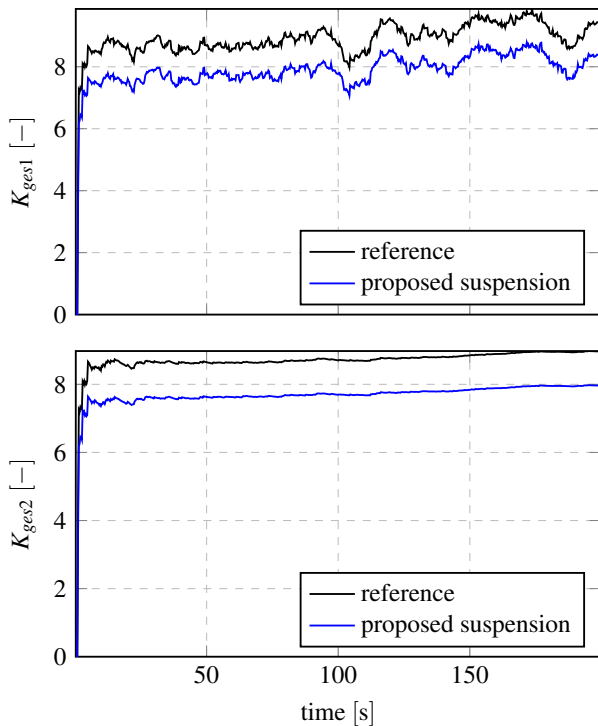


Figure 11. Comfort assessment criteria signals from quarter car simulations: K_{ges1} for time window of 10 s above and K_{ges2} for the total simulation time (200 s) below.

6 Conclusions

A suspension design solution was proposed to compensate for unfavorable effects on ride comfort and tire/road contact in the context of light-weight electric vehicles. The suggested mechanical design introduces the separation of the wheel hub from the wheel carrier allowing for a vertical relative movement between both parts by means of a prismatic joint. The realization utilizes two linear sliding bearings which house the auxiliary suspension elements – a design resembling a motorcycle front fork.

A comprehensive investigation on the influence of such suspension design on the ride comfort and tire/road contact was done. Simulation results show that the ride comfort can be improved significantly while there is negligible influence on the vehicle dynamics. The relative deflection in the prismatic joint introduced into the split wheel carrier is bounded to a few millimeters when the system is operated under common driving conditions.

Acknowledgements

The authors would like to thank Mr. Uwe Bleck for sharing his expertise on vehicle suspensions and vehicle dynamics.

References

T. Bunte. Recording of model frequency responses and describing functions in modelica. In *The 8th International Mod-*

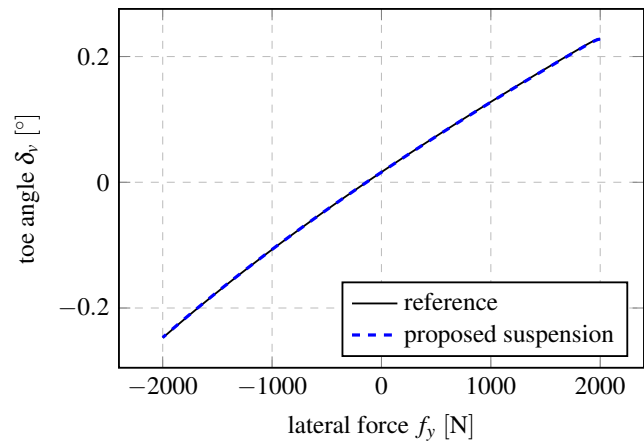


Figure 12. Change of the wheel toe angle of rear suspension under lateral load.

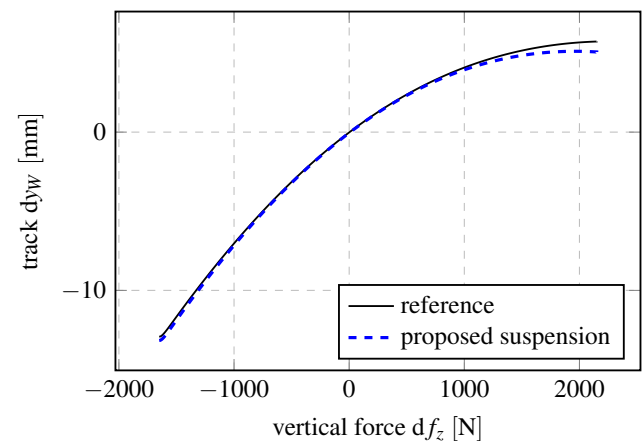


Figure 13. Change of the wheel track of rear suspension under vertical load change.

elica Conference, Dresden, Germany, 2011. URL <http://elib.dlr.de/68920/>.

M. Dempsey, M. Gäfvert, P. Harman, Ch. Kral, M. Otter, and Treffinger P. Coordinated automotive libraries for vehicle system modelling. In *The 5th International Modelica Conference*, Vienna, Austria, 2006.

J. K. Hedrick and T. Butsuen. Invariant properties of automotive suspensions. *Journal of Automobile Engineering*, pages 21–27, 1990.

D. Hennecke. *On the Assessment of the Riding Comfort of Passenger Cars under Transient Excitation*. PhD thesis, TU Braunschweig, Düsseldorf: VDI Verlag, 1995. In German.

ISO 8608. Mechanical vibration – road surface profiles – reporting of measured data, 1995.

A. Klöckner, A. Knobloch, and A. Heckmann. How to shape noise spectra for continuous system simulation. In *The 11th International Modelica Conference*, pages 411–418, Paris, France, 2015. Linköping University Electronic Press,

Linköpings Universitet. URL <http://elib.dlr.de/98408/>.

- M. Mitschke and H. Wallentowitz. *Dynamics of Motor Vehicles*. Springer Vieweg, 5 edition, 2014. ISBN 978-3-658-05068-9. In German.
- H. B. Pacejka. *Tyre and Vehicle Dynamics*. Elsevier Ltd, Oxford, 2002.
- D. Ryba. Improvements in dynamic characteristics of automobile suspension systems part 1: Two-mass systems. *Vehicle System Dynamics*, pages 17–46, 1974a.
- D. Ryba. Improvements in dynamic characteristics of automobile suspension systems part 2: Three-mass systems. *Vehicle System Dynamics*, pages 55–98, 1974b.
- S. M. Savaresi, C. Poussot-Vassal, C. Spelta, O. Sename, and L. Dugard. *Semi-active suspension control design for vehicles*. Butterworth-Heinemann/Elsevier, 2010.
- Ch. Schweiger, M. Dempsey, and M. Otter. The PowerTrain Library: New Concepts and New Fields of Application. In *The 4th International Modelica Conference*, Hamburg–Harburg, Germany, 2005.
- J. Stoffregen. *Motorcycle technology*. Springer Vieweg, 2012. ISBN 978-3-8348-1716-7. In German, DOI 10.1007/978-3-8348-2180-5.

Development of hierarchical commercial vehicle model for target cascading suspension design process

Kwang-chan Ko¹ Jong-chan Park¹ Dae-oh Kang² Jae-hun Jo³

Min-su Hyun³ Seung-jin Heo³

¹Hyundai Motor corporation, Korea, {kcko, impactpack}@hyundai.com

²Institute of Vehicle Engineering, Korea, bigfive@ivh.com

³School of Automotive Engineering, Kookmin University, Korea, {bluenice8,slay,sjheo}@kookmin.ac.kr

Abstract

This paper presents the development of framework and an industrial application of commercial vehicle suspension & steering system design based on the target cascading. This framework consists of 3 main modules, those are modeling, solving, and post-process module. Excel GUI is employed in order to give straightforward simulation way to the end users who are not familiar with vehicle dynamics simulation. End users are allowed to handle modeling parameters using Excel to build up models in the easy way. Key feature of solving module is that the simulation is conducted automatically with just selecting one of predefined scenario. The last module whose object is to calculate Ride and Handling performance index, is the post-process module.

A pilot study is applied to the practical issue to see the benefits of the framework, and design decision is made from the application results. This application study shows remarkable benefits not just in terms of Ride and Handling performance, but also in terms of solving cost. 15% of improved performance is produced regarding Ride and Handling, and 50% of development time is saved. It means that the framework allow to avoid time-consuming process to achieve required target in the vehicle development process.

Keywords: *Vehicle Dynamics, Target Cascading, Commercial Vehicle, Hierarchical Model, Suspension and Steering, Ride and Handling*

1. Introduction

The framework development is explained from section 2 to section 4. Section 2 contains the way of modeling and testing on the main sub-systems, and section 3 covers performance index calculator.

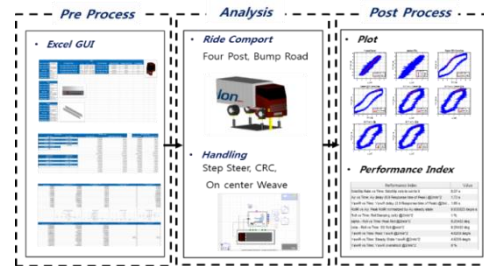


Figure 1. Overview of Frame Work

Section 4 is about the Excel interface development in order to give convenience to the end user. The optimization study is conducted to figure out the benefits of the developed framework in the section 5.

2. Library Establishment of Vehicle Dynamics

The main sub-systems of vehicle dynamics library consist of suspension, body, cab, and tire. For application to the target cascading process, each sub-system consists of geometrical and physical model.

2.1 Suspension and Steering system

2.1.1 Suspension Modeling

Full range of HMC commercial vehicle suspension types are modeled by using both Tubular Elastic Kinematic Suspension (TEKS) and Multi body Dynamics. TEKS use lookup table to specify suspension geometry, so TEKS model has to reflect the unique issue about commercial vehicle suspension. For instance, commercial vehicles have the suspension models of dependent type and independent type, the big difference between those suspension types is the roll motion. Generally in case of dependent suspension, left and the right movement is coupled in roll motion but the other is not.

The number of suspension types using in commercial vehicles are quite large, for efficient approach, object-oriented methodology is taken into the Multi body dynamic modeling. Figure 2.1 shows the modeling results.

Front			Rear		
Type	Steering + Susp.		Type	Susp.	
Rt_1	Independent type1		Rr_1	Dependent type1	
Rt_2	Independent type2		Rr_2	Dependent type2	
Rt_3	Dependent type1		Rr_3	Dependent type3	
Rt_4	Dependent type2		Rr_4	Dependent type4	
Rt_5	Dependent type3		Rr_5	Dependent type5	
Rt_6	Dependent type4				

Figure 2.1 Multi Body Dynamic model

Force element of physical model was modeled from the functional equation that has the design variables as its factors. The parts developed by the method above are leaf spring, coil spring, air spring, stabilizer bar, etc. Table 2.1 show some examples of force elements.

Table 2.1 Leaf Spring Model

Leaf Spring	
Design equation	$c_f = \frac{Enwh^3}{2Kl^3}$ $K = \frac{3n}{2n + n_-}$ $n_- = n - 1$
parameter	<p>E : modulus elasticity n : number of leaf w : Leaf Width h : Leaf thickness K : Defection Factor l : Leaf length n_- : Modified number</p>

We generate code of these functional equations in Modelica language like Figure 2.2.

```

model Leaf_spring1 "Linear 1D translational spring"
  extends Modelon.Mechanics.Translational.Templates.Compliant;
  parameter Modelica.SIunits.Position s0=0 "preload distance";
  parameter Modelica.SIunits.Force f0=0 "preload force";

  parameter Modelica.SIunits.Length w=0.06 "Leaf Width";
  parameter Modelica.SIunits.Length h=0.01 "Leaf thickness";
  parameter Modelica.SIunits.Length l=0.6 "Leaf length";
  parameter Real n=10 "Number of leaf";
  parameter Modelica.SIunits.ModulusOfElasticity E=21000000000
    "modulus of elasticity";

  Modelon.Units.SI.TranslationalStiffness c "Bending stiffness";

  Real k "Defection Factor";
  Real n_ "Modified Number";
  equation
    n_ = n - 1;
    k = (3*n) / (2*n + n_);
    c = (E*n*w*(h^3)) / (2*k*(l^3));
    -f + f0 = c*(s_rel - s0);
  B
end Leaf_spring1;

```

Figure 2.2. Leaf Spring Model (Modelica)

2.1.2 Steering Modeling

Once steering system is modeled. Rack&Pinion steering is modeled for the independent suspension and Pitman-Arm steering is modeled for the dependent suspension.

Table 2.2 Rack&Pinion, Pitman Arm Model

Type	Steering
1) Rack & Pinion (Independent)	
2) Rack & Pinion (Dependent)	

2.1.3 Validation

Suspension models are validated by K&C experiment. Figure 2.3 show the comparison results of the established suspension system models of independent type and dependent type with the test data. From the comparison, we reach the conclusion that the established models have reliability.

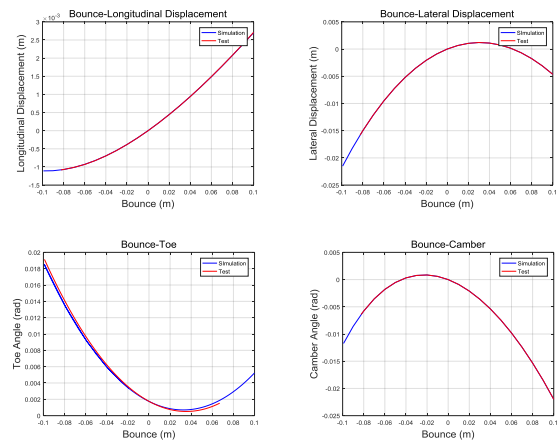


Figure 2.3. Parallel wheel travel

2.2 Cab, Frame & body Model

3 types of Cabin mounting are used by HMC commercial vehicles are built up in the library as shown in table 2.3

Table 2.3 Cab Mounting library

Type	Front	Rear	Animation
1) Large Air Spring			
2) Middle Air Spring			
3) Middle Coil Spring			

Lumped mass, C.G location, and moment of inertia are main input parameters in case of body model, but realistically bending and torsion can occur due to the long length of frame in the commercial vehicles, so bending stiffness and torsion stiffness to the lumped mass model are reflected.

2.3. Tire model

Pacejka 02 Tire are employed for tire library. In order to create reliable tire model, all the parameter that required for the Pacejka 02 are measured. Figure 2.4 is the description about one of the tire data.

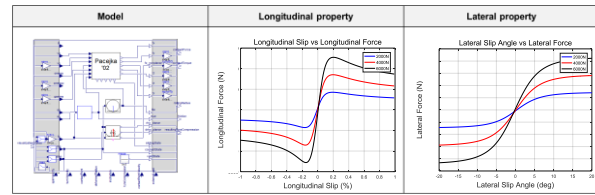


Figure 2.4. Pacejka 02 tire model

3. Post Processor

3.1 Performance Index of R&H

The higher priority way to set up quantified R&H performance Index is from analysing statistical relationship between subjective feeling evaluation and objective measurement data, but a lot more valid sample data are demanded for the statistical relationship analysis. Realistically it is not easy to collect enough valid sample data due to many reasons. Instead of those preliminary researches, 3 benchmarking vehicles are chosen and measured to set up R&H performance indexes in this stage.

(1) Test / simulation modes, measurement methods are established after 3 benchmarking vehicles are chosen with considering weight, wheelbase, and steering, suspension type. The specifications of 3 benchmarking vehicles are shown in Table 3.1.

Table 3.1 Specification of Benchmarking Vehicles

	Vehicle A	Vehicle B	Vehicle C
Wheel Base	3,670	3,665	3,935
Weight (FRT/RR)	3,020 (1,550/1,470)	2,250 (1,250/1,000)	3,230 (1,635/1,595)
Steering	Rack Pinion	Rack Pinion	Bell Crank
Front Suspension	MacPherson Strut	MacPherson Strut	Double Wish Bone
Rear Suspension	Rigid Axle	Rigid Axle	Rigid Axle

On-Centre Weave, Steady-State Cornering, Step Steer, Pulling Stability, Bumpy Ride are selected for test modes. Generally too many test / simulation modes cause a lot of solving cost in the optimization process, so test / simulation modes must be minimized.

(2) The 31 quantified indexes for R&H performance indexes are calculated using measurement data of benchmarking vehicles. Understeer gradient which decide cornering stability and Steering R2 value which decide cornering linearity are shown in Fig. 1 as examples

of graphic calculation.

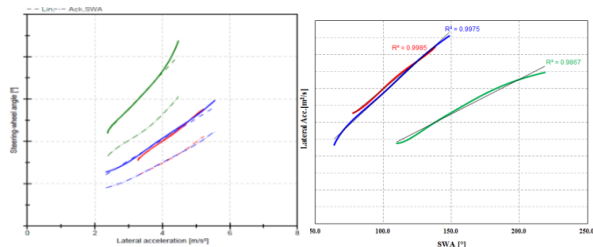


Figure 3.1. Steady-State Cornering Test

(3) The 31 quantified indexes are divided into 9 groups for mapping which represent subjective feeling. Those 9 groups consist of 3 controllability fields (Roll Control, Response Level, Cornering Controllability), 3 Stability fields (Understeer Balance, Response Velocity, Directional Stability), and 2 Steering Feel fields (Steering Sensitivity, Pulling), 1 Ride Comfort field (Bumpy Ride). Fig. 3.2 is example of the mapping results regarding Stability fields.

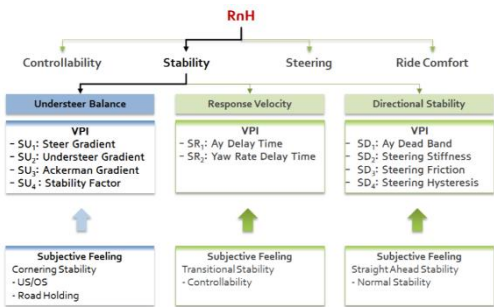


Figure 3.2. Stability Feeling Matching Map

(4) 31 indexes are taken into design of experiment (D.O.E) Screening to figure out the relationship between individual indexes of those 31. Finally 24 indexes are selected after D.O.E Screening except 7 indexes which have repeated performance meaning by other indexes. The screening results are shown in Fig 3.3.

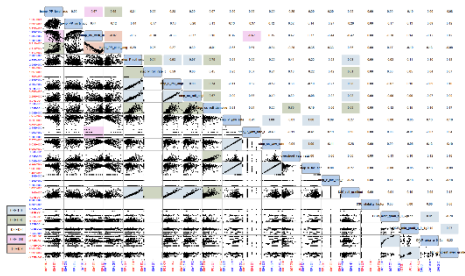


Figure 3.3. D.O.E Correlation of VPI

Table 3.2 Description of 24 Indexes

Steering Sensitivity			
Ay vs SWA: Average Gradient (between 1-2m/s²)	[m/s²] / [°]	On Centre Weave	Steering Sensitivity
Ay vs SWA: Min Gradient (between 1-2m/s²)	[m/s²] / [°]	On Centre Weave	Minimum Steering Sensitivity
Roll Control			
Roll vs Ay: Roll gradient @ linear range	[°] / [m/s²]	Steady State Cornering	Body Control: roll amplitude
Roll vs Time: Peak Roll @ 2m/s²	[°]	Step Steer	Body Control: roll velocity
Roll vs Time: Peak Roll @ 2m/s²	[°]	Step Steer	Amount of Maximum Roll
Roll vs Time: SS Roll @ 2m/s²	[°]	Step Steer	Amount of SS Roll
Roll vs Time: Roll Damping (Peak Roll / SS Roll) @ 2m/s²	[°]	Step Steer	Roll Damping
Understeer Balance			
SWA vs Ay: Steer gradient	[°] / [m/s²]	Steady State Cornering	Understeering Balance
SWA vs Ay: Understeer Gradient	[°] / [m/s²]	Steady State Cornering	Understeering Balance
SWA vs Ay: Stability factor	[°] / [m/s²]	Steady State Cornering	Understeering Balance
Response Level			
YawR vs Time: Peak YawR @ 2m/s²	[°]	Step Steer	Response Level Transient
YawR vs Time: Steady State YawR @ 2m/s²	[°]	Step Steer	Response Level Steady State
Cornering Control-ability			
SWA vs Ay: linear range bandwidth	[m/s²]	Steady State Cornering	Response Linearity
SWA vs Ay: R² Value between 1-3m/s²	[m/s²]	Steady State Cornering	Response Linearity
Directional Control-ability			
YawR vs Time: YawR overshoot @ 2m/s²	[°]	Step Steer	Directional damping
Ay vs SWA: S. doubled @ 10g	[°]	On Centre Weave	Steering hysteresis (measure of yaw lag)
Response Velocity			
Ay vs Time: Ay delay (90% Response time of Peak) @ 2m/s²	[s]	Step Steer	Response Delay @ off centre
YawR vs Time: YawR delay(90% Response time of Peak) @ 2m/s²	[s]	Step Steer	Response Delay @ off centre
Pulling Stability			
Brake Pulling vs Time: Peak Lateral Movement	[m]	0.3G Braking at 100 kph, SWA Lock	Brake Pulling
Steady State Pulling vs Time: Peak Lateral Movement	[m]	4s driving at 200kph, SWA Lock	Steady State Pulling
Ride Comfort			
Peak peak Longitudinal Acceleration	[m/s²]	Bumpy Ride	Impact Harshness
Peak peak Vertical Acceleration	[m/s²]	Bumpy Ride	Impact Harshness
Peak peak Vertical Displacement	[mm]	Bumpy Ride	Fitch&Bounce
Peak peak Pitch Angle	[°]	Bumpy Ride	Fitch&Bounce

3.2 Post-Processor

To automate the performance index developed at 3.1, post-processor was developed by using Matlab. The data used for the inputs in post-processor comes from Dymola as mat-file form. Figure 3.4 is GUI of the post-processor.

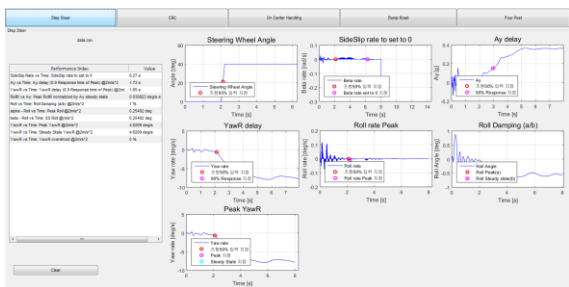


Figure 3.4. Performance index calculation program

4. Modeling tool based on Excel.

Pre-processing GUI is built based on Excel to give end users convenience. Like Figure 4.1, pre-processing is performed by inputting the model data first, and goes through the process that links the cells data inputted with the parameters on Modelica.

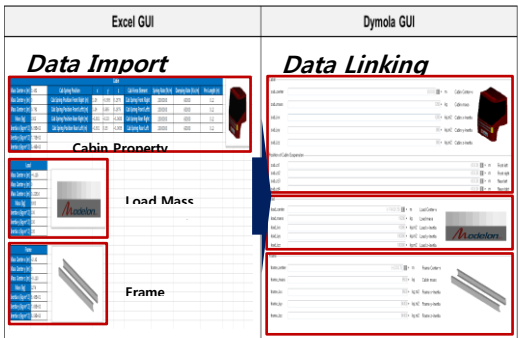


Figure 4.1. Parameter linking based on Excel

Linking on Excel and Modelica parameters was processed by using the external data library which is one out of Modelica share libraries. Excel GUI is divided The constituted GUI is shown on Figure 4.2.

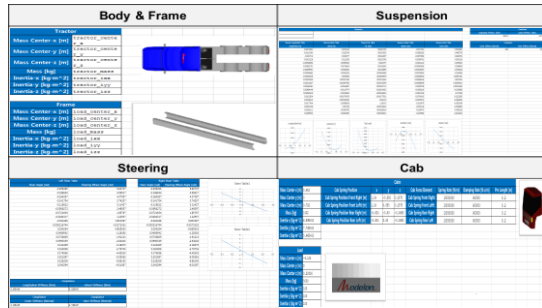


Figure 4.2. Excel GUI

5. Application

To review the validity of the simulation framework that was developed earlier, ride and handling performance simulation was conducted.

5.1 Ride performance test

5.1.1 Model explanation

Four-post test for the ride evaluation method are simulated, so the mulit body dynamics truck model waw combined with four-post test rig, and formed them as the test environment. The constituted truck chassis model was shown on Figure 5.1, and the four-post test environment was indicated on Figure 5.2.

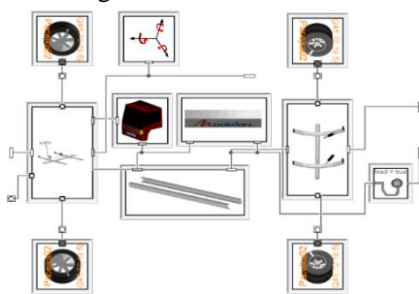


Figure 5.1. Truck Chassis Model

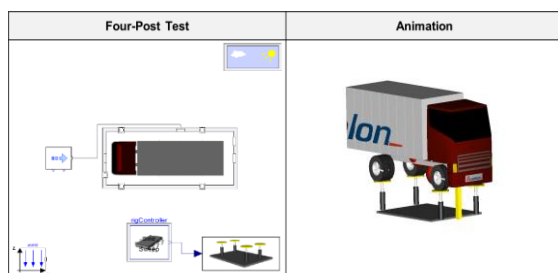


Figure 5.2. Four-Post Test Environment

5.1.2 Explanation on the evaluation method.

ISO C-Class road profile is used for the input of four-post test, and selected the vertical acceleration of body as performance index.

5.1.3 Comparison analysis of the results

As shown at Figure 5.3, the test results showed little error (RMS error: - %) in time domain.

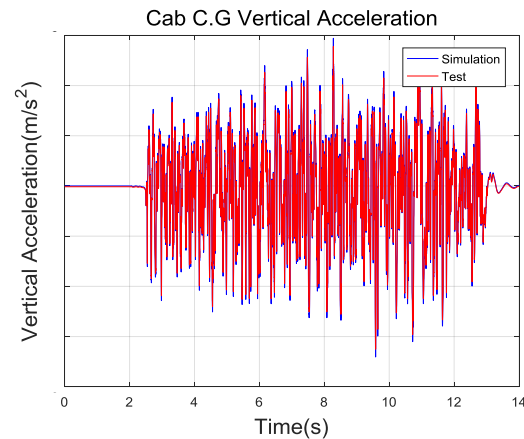


Figure 5.3. Four-Post Test Result Vertical Acceleration

Likewise, the gradient of PSD in frequency domain of Figure 5.4 was ilustraed with high accuracy as well.

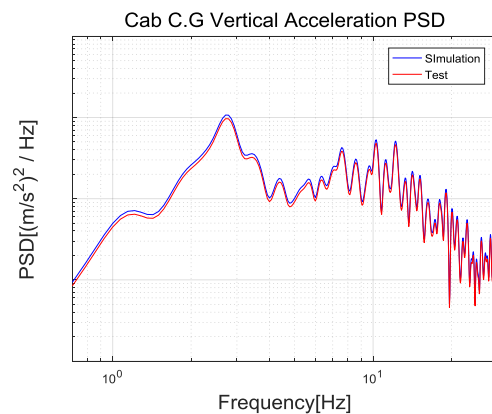


Figure 5.4. PSD

5.2 Handling performance test

5.2.1 Model explanation

CRC (Constant Radius Cornering) and step-steer maneuvers were taken into simulation to evaluate not just steady state condition but also transient condition response. The truck chassis model that used in ride test is linked with the steering

controller which controls the vehicle to drive in a steady curvature, and the velocity controller which controls to drive as the speed allowed. The step-steer test environment was established through combination of the steering actuator that controls to steer towards the allowed steering wheel angle and the velocity controller which controls to drive in steady speed shown as Figure 5.6.

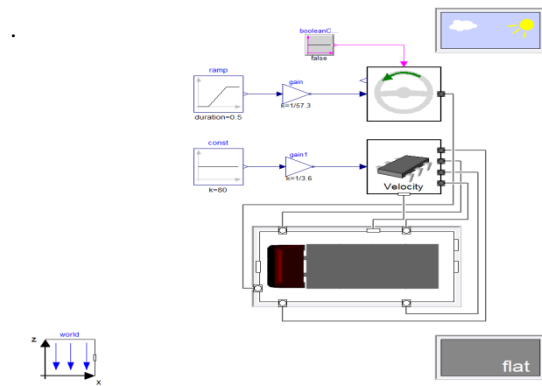


Figure 5.5. Step Steer Test Environment

5.2.2 Comparison analysis of the results

Figure 5.6 showed the gradient of steering wheel angle-lateral acceleration of CRC, and it proves that this model predicts the steady state condition response of the truck in high accuracy. The performance index was estimated with little error as well.

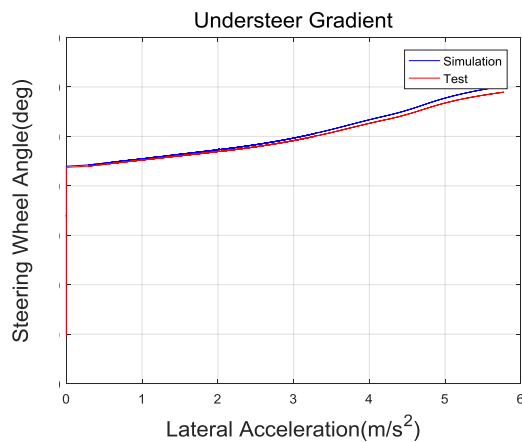


Figure 5.6. Understeer Gradient

Figure 5.7 showed the gradient of lateral acceleration of step-steer and yaw rate, and it proves that this model predicts the transient condition response in high accuracy. The performance index selected earlier was calculated with little error as well.

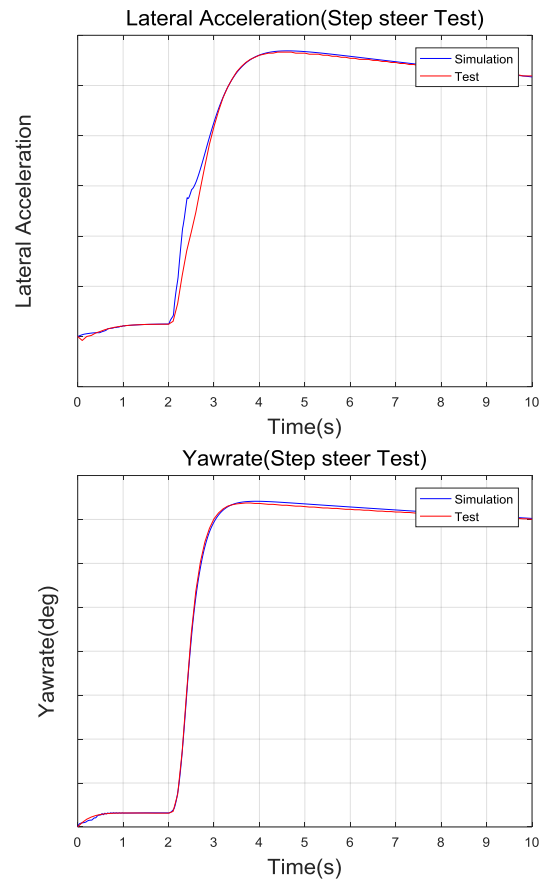


Figure 5.7. Lateral Acceleration, Yawrate

5.3. Application of Target Cascading Design

5.3.1 Formulation of design matters

For the application examples, RMS of body vertical acceleration, which is the ride performance index as the objective functions, is chosen, together with the parameters of air spring as the design variables. The matter was defined to find the design variable value to minimize the objective functions.

5.3.2 System level design

In system level, RMS of body vertical acceleration is extracted, and air spring property (F-D gradient) can be optimized through genetic algorithm in Optimization Library of Modelon. Genetic algorithm is a probability search algorithm, and it is favorable to the treatment for discrete variables, not influenced from the continuity and differentiability of functions, etc. which consist of the matters, and able to search globally. The formulation of design matters is the same as follows.

$$\text{Minimize } (x) = \sqrt{\frac{x_1 + x_2 + x_3 + \dots + x_n}{n}},$$

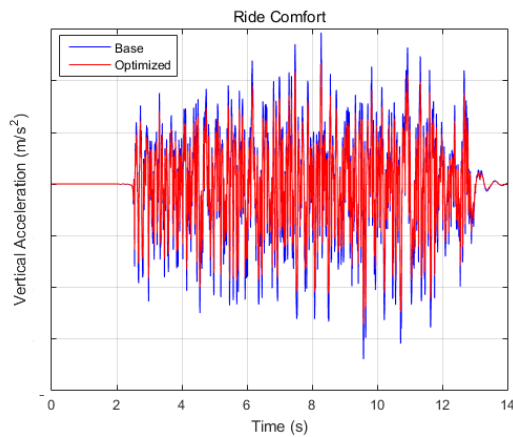
$x_i (i = 1:n)$ = Body vertical acceleration,

n = Number of data.

Subject to $0.8y_{base} < y_{base} < 1.2y_{base}$

y_{base} = Air spring characteristic data of base model

Figure 5.8 shows the change of performance index, and Figure 5.9 indicates the property of the optimized design variables.



RMS (m/s ²)	
Base	1.8410
Optimized	1.4912

Figure 5.8. Performance Index Variation

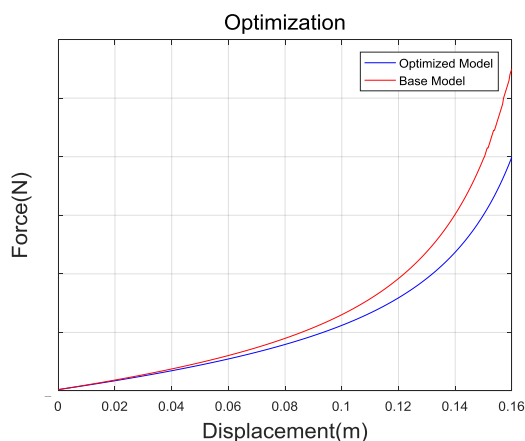


Figure 5.9. Optimization

5.3.3 Sub-system level design

In sub-system level, optimal value of parameters that can embody the optimized air spring derived

from system-level through genetic algorithm is selected. The formulation of the optimized design is the same as follows.

$$\text{Minimize } (x) = \sum_{i=1}^n (x_{s,i} - x_{ss,i})^2,$$

$x_{s,i}$ = System-level air spring data,

$x_{ss,i}$ = Subsystem-level air spring data,

n = Number of data

Subject to $0.8 * y_{base} < y_{base} < 1.2 * y_{base}$

y_{base} = Air spring parameters of base model

Figure 5.10 shows the comparison results between the property of air spring realized through the optimal parameters and the property of air spring derived from system-level, and we can see that the property value is embodied with little error.

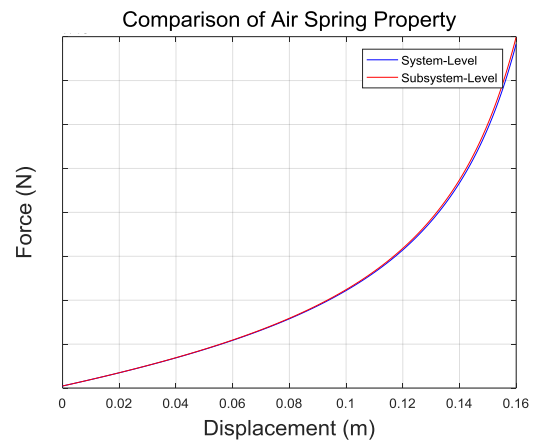


Figure 5.10. Comparison of Air Spring Property

Table 5.3 is one that compares the design variables of base model and the design variable values of the optimal model.

Table 5.1 Comparison of Air Spring Property

	Base	Optimal
Nominal Preload Force	20000 N	20000N
Polytropic Coefficient	1.38	1.317
Effective Area with Respect to Volume	0.077m ³	0.09343m ³
Effective Area with Respect to Load	0.07315m ³	0.0555m ³
Constant Pressure Spring Rate	1000000N/m	1000000N/m

6. Result

Through this study, we developed the interpretation of hierarchical structure and the design model through object-oriented modeling method. The merits of constituted hierarchical structure model are as follows. Firstly, both behavior model and physical model can be interpreted at one platform. Secondly, design objectives and design variables that are indispensable for target cascading can be shared without separate treatment of data. Thirdly, it is easy for users, if necessary, to modify the model since the model has been established through object-oriented modeling method. And, to enhance design efficiency, we raised efficiency by developing design process through linking with pre-processor (Excel), model (Dymola), post-processor (matlab). To test the effectiveness on this, we applied the established framework to the suspension system design matters that were considered to improve the performance of R&H. From the result of design, we verified that performance improved by 15%, and the time for design decreased more than 50% as well. These results proved that the developed framework is suitable for the suspension system design process of target cascading.

Reference

- Kang, Ph.D: Robust Design Optimization Process Development for Suspension System by using Target Cascading Method, *Kookmin University*, 2010.
- J. Rauh: Virtual Development of Ride and Handling Characteristics for Advanced Passenger Cars, J. Rauh, *Vehicle System Dynamics*, vol. 40, no. 1-3, pp. 135-155, 2003.

Model Based Analysis of Shimmy in a Racing Bicycle

Nicolò Tomiati¹ Gianantonio Magnani¹ Bruno Scaglioni¹ Gianni Ferretti¹

¹Politecnico Di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria DEIB

Via Ponzio 34/5, 20133 Milano, Italy,

{gianantonio.magnani,bruno.scaglioni,gianni.ferretti}@polimi.it,
nicola.tomiati@mail.polimi.it

Abstract

In this paper we are presenting a model of a racing bicycle, developed in Modelica language within the Dymola environment. The main purpose is to investigate the dynamic response of the bicycle and its modes of vibration, referring in particular to *shimmy*. This phenomenon occurs at high speeds and consists of sudden oscillations of the front assembly around the steering axis. Lateral accelerations on the horizontal tube of the frame can reach 5-10 g with a frequency that varies between 5-10 Hz. Even if it is quite uncommon, *shimmy* is a topic of great relevance, because it may be extremely dangerous for the rider. Thanks to this model, we can show that the main elements which contribute to the rise of the oscillations are the lateral compliance of the frame and the tyres' deformation.

Keywords: *bicycle, shimmy, flexible multibody systems*

1 Introduction

This paper will present a multibody model of a racing bicycle developed in Modelica, within the Dymola environment. The main purpose of this work is to investigate in depth the dynamic response of the bicycle and its modes, referring in particular to *shimmy*.

Any two-wheeled vehicle is subject, during its movement, to three modes of vibration: *capsize*, *weave* and *wobble*. The first two are always present; the third one occurs occasionally.

If the *capsize* mode is unstable, the bicycle follows a spiral path with increasing values of the roll angle that leads it to a lateral fall.

The *weave* mode consists, instead, in an oscillatory motion of the rear frame about the yaw axis together with oscillations about the roll axis. In this case, the frequency is of 1-2 Hz.

Finally, the *wobble* mode (which is often referred to as *shimmy*) is an oscillatory motion of the front assembly with respect to the steering axis. When it occurs, lateral accelerations on the horizontal tube of the frame can reach 5-10 g with a frequency that varies between 5-10 Hz (Magnani, Ceriani, and Papadopoulos 2013). This phenomenon is therefore very violent, unexpected and can lead to dramatic consequences, particularly if the rider does not know it and is not able to handle it. Fortunately, it does not occur so frequently and it is difficult that it can

lead to a fall, although this is the sensation perceived by the cyclist. Usually, this happens at high speed, such as the one that can be reached along a downhill road. The phenomenon is well known among cyclists and bicycle manufacturers. It is a topic of great relevance because it is not still clear what are the main causes that lead to these vibrations.

Thanks to experimental activities (Magnani, Ceriani, and Papadopoulos 2013) and by using numerical models (Plöchl et al. 2012; Klinger et al. 2014; Limebeer and Sharp 2006), the lateral compliance of the frame and the tyres' deformation have been found to be two essential contributors to the *wobble* mode. One of the goals of this article is to understand in detail what are the causes or factors that excite these vibrations, referring in particular to a racing bicycle.

The paper is organised as follows. Section 2 gives an overview of the overall bicycle model, describing all the components in detail. Section 3 explains how the elements are connected to each other and what assumptions have been made before running the simulations. In Section 4 simulation results are presented. Two different versions of the model will be analysed. At the end, in Section 5 the conclusions and some possible practical advice that may be helpful to the rider to damp out the *shimmy* oscillations are discussed.

2 Bicycle Model

The multibody model presented in this work is based (for some components) on the Modelica *MotorcycleDynamics* package, which is described in detail in (Donida, Ferretti, Savaresi, Schiavo, et al. 2006; Donida, Ferretti, Savaresi, and Tanelli 2008). This library, in turn, was developed by *VehicleDynamics*, which shares basically the same structure (Andreasson 2003).

The following step is to run simulations to study its dynamic behaviour. Our attention has been focused on a racing bicycle, which is described in more detail in (Klinger et al. 2014). Whenever possible, therefore, data reported in that article has been used in order to make the model as compliant as possible to the real behaviour.

The main components of the model are:

- the rear frame;

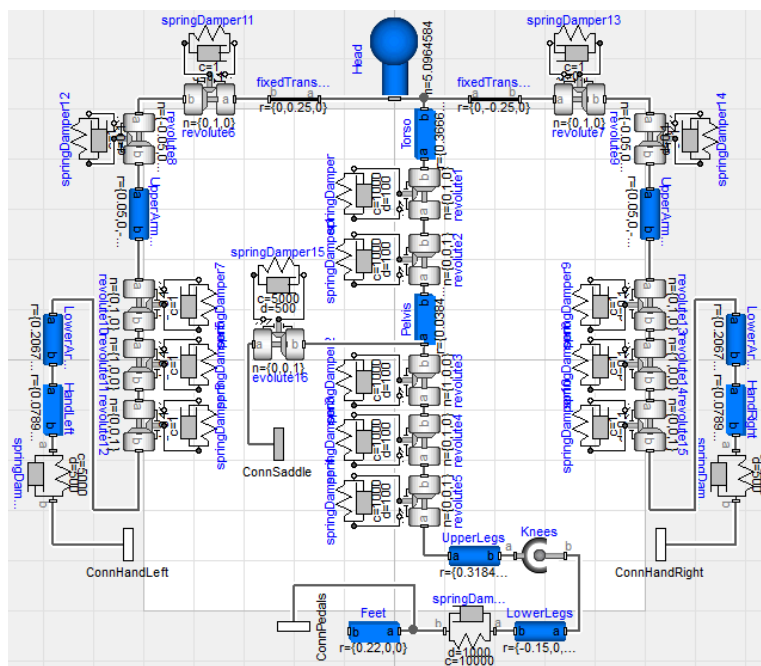


Figure 1. Rider block diagram in Dymola with the four interfaces to connect it to the other components. Four spring-damper elements have been introduced to model the compliance of the constraints between rider's hands-handlebar, feet-pedals and pelvis-saddle.

- the front assembly, which includes handlebar, stem and fork;
- the cyclist;
- the front and rear wheels;
- the road.

2.1 Rear Frame

The first component is modelled by a *BodyShape* element, *i.e.* a single rigid body characterised by centre frame, mass and inertia tensor. In order to associate to this body the true shape of the frame, we have used a CAD model. Secondly, we have added the saddle, which is connected to the rear frame with a *Revolute* joint. This type of connection allows the rotations around an axis passing through the saddle tube. In this way, it is possible to consider the compliance of the constraint between the saddle and the frame.

The rear frame model presents four interfaces that allow connecting this component with the rider (including the saddle and pedals), with the front assembly (through the steering axis) and with the rear wheel (at the hub).

2.2 Front Assembly

The front assembly has also been modelled as a rigid body with its inertia tensor and whose mass is concentrated in a single point. It consists of the fork, whose true shape has been defined in a CAD model, the stem and the handlebar. Four interfaces characterise this component; in fact, the front assembly can be connected with the rear frame, with

the front wheel at the hub, and with the cyclist at the two contact points on the handlebar.

2.3 Cyclist

The third component of the model is the rider. It has been modelled as a multibody system obtained by the connection of solid geometric elements having different shapes. In particular, *Cylindrical elements* have been used to model limbs (*i.e.* arms, forearms, thighs and legs) while *Rectangular parallelepipeds* for the torso, the pelvis, hands and feet. In regards to the head, a *Body* element has been chosen, which is characterised by mass and inertia tensor. It is visualised by a cylinder and by a sphere that has its centre at the centre of mass.

To model the human articulations two types of joints have been used, chosen depending on the possible relative movements between the parts connected. *Spherical* joints prevent all the translations but enable the rotations about three mutually orthogonal axes. On the other hand, *Revolute* joints prevent all the translations and the rotations about two axes. Therefore, they leave only one degree of freedom (a rotation about an axis). It is important to notice that a *Spherical* joint can be obtained by connecting to one another three *Revolute* joints, specifying for these objects three orthogonally axes of rotation (as has been done with the elbows).

To make the model more realistic, elements made up of a spring and a damper in parallel have been added: in this way it is also possible to take into account the contribution of stiffness and damping of human muscles. Figure 1 shows the rider block diagram in Dymola. Four interfaces

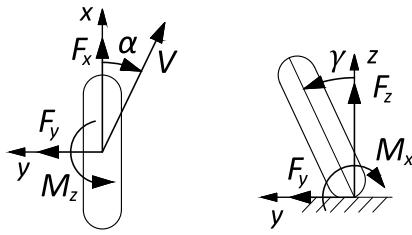


Figure 2. Tyre definitions: side-slip angle α is defined as the angle between the wheel centre plane and the direction of the forward velocity V . Camber angle γ is defined as the angle between the wheel centre plane and the vertical axis z of the road. F_x is the longitudinal force, F_y is the lateral force and F_z is the normal force. M_x is the overturning torque and M_z is the aligning torque. Positive values are shown. The left figure is a top view while the right one is a rear view.

have been included in the model. In so doing, the rider can be connected to the front assembly and to the rear frame. Two spring-damper elements have been added to cyclist's hands and the upper part of the front assembly to model his grip on the handlebar. The same has been done for the connection between rider's feet and bicycle pedals. Lastly, a spring-damper element has also been added between the cyclist's pelvis and the saddle in order to model the compliance of the sitting position (the rider, in fact, is not rigidly attached to the saddle).

In order to verify that the behaviour of the model was compatible with a rider's real movements, different simulations have been performed (for example, by simulating a turning manoeuvre or the execution of a curved trajectory).

2.4 Wheel Model

Wheels are modelled as rigid bodies with their mass concentrated in the hub. Afterwards, a *torus* model has been used to associate the real tyre shape to the wheel. The front and rear wheels have the same dimensions (*i.e.* the same radius), but different mass and inertial properties.

2.5 Tyres and Wheel-Road Interaction

As already mentioned in the Introduction, to highlight the *wobble* mode it is necessary to consider tyres' deformation.

The tyre allows the contact between the rigid part of the wheel (*i.e.* the hub) and the road surface. At the same time, it ensures adherence to the asphalt and generates distributed forces and torques within the contact region. In the following, it will be assumed that these forces and torques are instead concentrated and applied at the single contact point that represents the interaction between wheel and road surface. In order to compute these forces, four reference frames are needed, as explained in (Donida, Ferretti, Savaresi, Schiavo, et al. 2006). Figure 2 shows

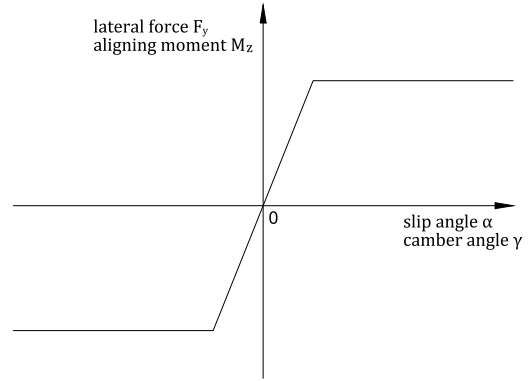


Figure 3. Qualitative trend of the lateral force F_y and aligning torque M_z as a function of the side-slip α and camber γ angles.

the sign convention adopted in this work.

As can be seen, α is the side-slip angle, which is defined as the angle between the forward velocity V and the wheel centre plane; γ is instead the camber angle, defined as the angle between the vertical axis z of the road and the wheel centre plane.

The following step is the determination of contact forces and torques. As stated in (Pacejka 2006), there are different relations between forces and angles. For our purposes, a linear relation has been chosen to describe tyres behaviour. Moreover, the model has taken into account the tyres' dynamic, *i.e.* the delay in the deformation due to the elasticity properties of the material. The tyre, in fact, does not respond immediately when it is rolled from the stand-still under a slip angle. It is necessary some time before the lateral force F_y approaches the stationary value. The same is true for the aligning torque M_z .

The longitudinal force F_x , which can represent both traction and braking forces, is defined as:

$$F_x = C_{F\kappa}\kappa, \quad (1)$$

where κ is the longitudinal wheel slip.

On the other hand, the lateral force F_y is the sum of two terms:

$$F_y = C_{F\alpha}\alpha' + C_{F\gamma}\gamma'. \quad (2)$$

The aligning torque M_z also depends on both the side-slip angle α' and the camber angle γ' , according to this equation:

$$M_z = -C_{M\alpha}\alpha' + C_{M\gamma}\gamma'. \quad (3)$$

The side-slip angle α' in (2) and (3) differs from α because of the delay in the tyre response after the deformation. The same is true for the camber angle γ' . These dynamics have been modelled by two first-order differential equations, *i.e.*:

$$\frac{\sigma_\alpha}{V_x}\dot{\alpha}' + \alpha' = \alpha, \quad (4)$$

$$\frac{\sigma_\gamma}{V_x}\dot{\gamma}' + \gamma' = \gamma, \quad (5)$$

where V_x is the longitudinal component of the forward velocity. The characterising parameter, called *relaxation length* σ , is similar to a time constant except that it has units of length rather than time. The relaxation length is a tyre characteristic that can be determined experimentally (Limebeer and Sharp 2006).

Finally, the overturning torque M_x has been also considered, defined as:

$$M_x = C_{Mx}\gamma. \quad (6)$$

To further improve the model, two saturation limits with respect to the lateral force F_y and the aligning torque M_z have been introduced. This means that, at high values of side-slip and camber angles, this force and torque are constant (see Figure 3). In this way, the trend of the curve is very similar to the one that can be obtained by applying the Pacejka's *magic formula* described in (Pacejka 1993). The linear approximation is valid only for small values of the two angles.

The stiffness coefficients inside equations (1)-(6) depend on the vertical force F_z transmitted on the ground at the contact point between tyre and road surface. Dymola computes its value at any given time (typically, in fact, the vertical force F_z is not constant during the movement) and this operation allows to compute all the stiffness coefficients. When contact forces and torques are known, a balance is carried out at the hub, *i.e.* the point where the wheel is connected to the other components of the bicycle.

2.6 Road

The road surface has been modelled through the `Environments` package of the `MotorcycleDynamics` library.

This package allows the user to select the road slope (level, uphill or downhill road) and its characteristics (dry asphalt, wet and so on). To run the simulations it has been chosen to work with a dry road, having a slope such as the bicycle forward speed increases linearly from 10 m/s to 20 m/s in 40 seconds (see Figure 4). From the results of the experimental activity described in (Magnani, Ceriani, and Papadopoulos 2013), it is shown that *shimmy* appears in this speed interval. The quote $z = f(x, y)$ of the road surface is defined by the equations:

$$z = \begin{cases} 0 & \text{if } x < 0 \\ -0.035k(x)x & \text{if } x \geq 0 \end{cases}, \quad (7)$$

where x is the position along the longitudinal direction, while:

$$k(x) = \frac{\arctan(10x + \frac{\pi}{2})}{\pi}. \quad (8)$$

Equation (8) is necessary to avoid discontinuities on the road surface, *i.e.* it guarantees an appropriate connection when the road slope changes.

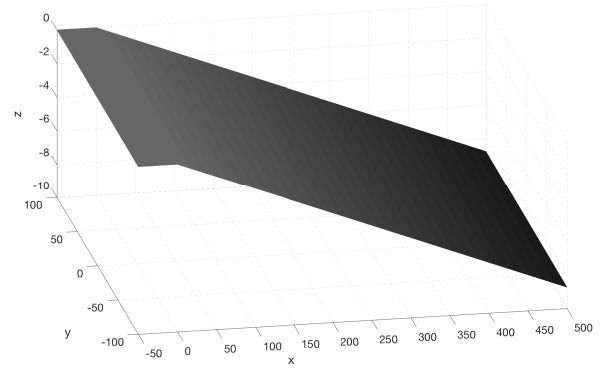


Figure 4. Road surface.

3 Model Assembly

Figure 5 shows the connections between the different models.

In more detail, the rider is connected to the rear frame and to the front assembly, including the saddle, the pedals and the two contact points on the handlebar. The front wheel is attached to the hub of the front assembly with a *Revolute* joint. This element simulates the behaviour of the ball-bearing. Similarly, the rear wheel is attached to the bicycle main frame.

Lastly, it is necessary to connect to one another the front assembly and the rear frame. Once again, a *Revolute* joint has been used: it introduces the rotation δ of the steering axis. As previously mentioned, there is another key element that is essential to trigger the *wobble* mode. This is the lateral compliance of the frame and it can be modelled by a second *Revolute* joint that allows the rotations of the front assembly around the β -axis (see Figure 6).

This axis is in the plane of symmetry of the vehicle and it is perpendicular to the steering axis, as suggested in (Klinger et al. 2014). The flexibility is lumped at the steering head. The user can set the values of stiffness k_β and damping c_β coefficients that represent the structural properties of the frame. Figure 7 shows the three-dimensional representation of the rider-bicycle model. As can be seen, the cyclist assumes the typical position for riding a racing bicycle, with his upper body in a bent-forward position and his hands firmly attached to the handlebar.

Some other simplifying assumptions are also needed. The gravity force acts on each component, and the aerodynamic drag force¹ has been neglected, assuming that the contribution related to this force is balanced by the component of the weight that appears when the bicycle is moving on a downhill road. Moreover, it has been assumed that the aerodynamic force does not change the vertical forces F_z acting on the wheels' contact points. Actually, the lift force reduces the vertical load on both front and

¹The aerodynamic force can be divided into two components: *drag* force, which is directed along the longitudinal axis, and *lift* force, which is directed along the vertical axis.

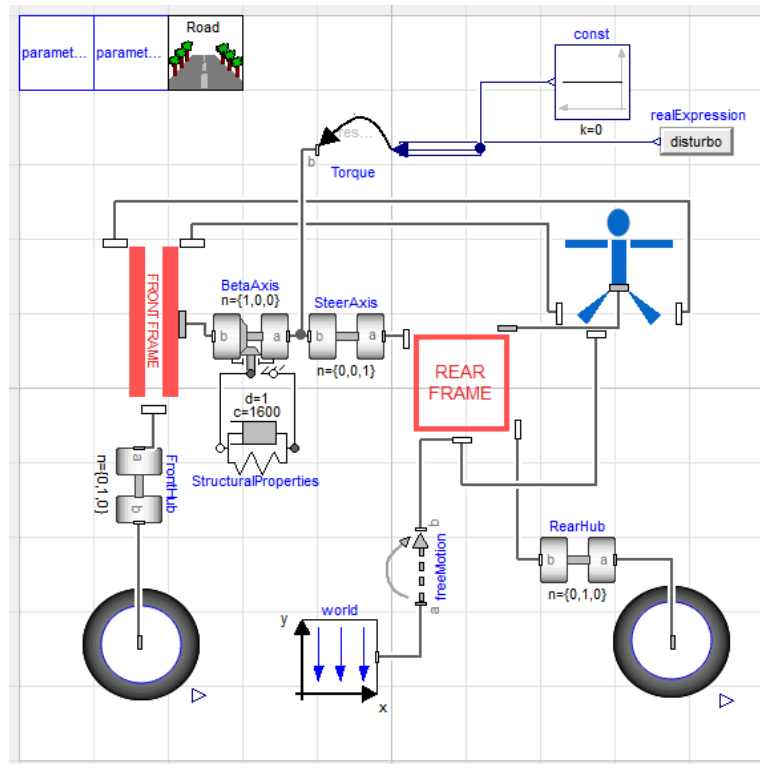


Figure 5. Model assembly that highlights the connections between components. A *Revolute* joint with a spring-damper element has been added to model the frame lateral compliance (β -axis).

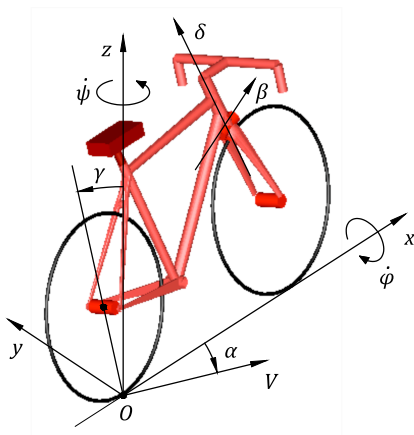


Figure 6. This figure shows the steering axis δ , the axis β that is necessary to model the frame lateral compliance, roll (ϕ) and yaw (ψ) angular velocities, the camber angle γ and the side-slip angle α . Positive values are shown.

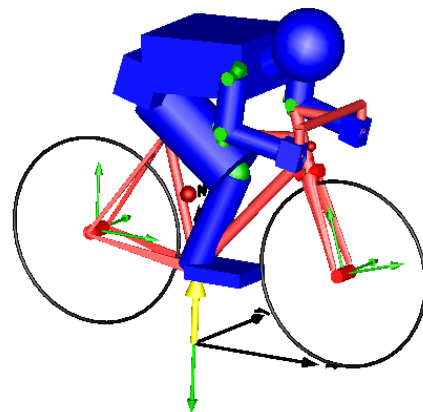


Figure 7. Three-dimensional representation of the racing bicycle model.

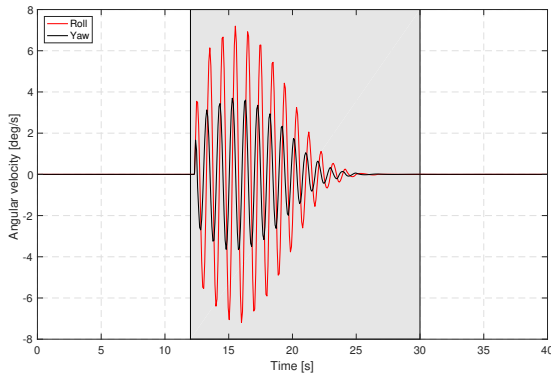


Figure 8. Roll and yaw angular velocities of the bicycle rear frame when $k_\beta \rightarrow \infty$ (*rigid frame*).

rear tyres, while the drag force increases the rear vertical load and decreases the front one.

4 Simulation Results

The aim of the simulations is to study the model dynamic response after the application of suitable perturbations, trying to point out the *wobble* mode. For this reason, an impulsive torque disturbance has been chosen. It is applied on the steering axis when the forward speed is equal to $v_s = 13$ m/s.

4.1 Rigid Frame Model

The first scenario considered is characterised by a *rigid* version of the bicycle model. It can be obtained by setting the frame stiffness coefficient $k_\beta \rightarrow \infty$. After the torque application, the steering axis is subject to oscillations that initially increase in amplitude and then decrease up to being completely damped. However, their frequency is approximately equal to 1 Hz, a value much smaller than 5-10 Hz that characterises the *wobble* mode. Although other simulations have been carried out by changing the type of the perturbation and some model parameters, we have not been able to trigger the *shimmy* using the bicycle model with a *rigid* frame. Figure 8 shows, instead, the rear frame roll and yaw angular velocities.

The oscillation trend is the same that characterises the steering axis response, *i.e.* with oscillations that initially increase and then disappear after a few seconds. As can be noticed, the two signals have a phase difference of 90° : when the roll angular velocity is zero, the yaw rate reaches its maximum (or minimum). This trend perfectly describes the *weave* mode. More specifically, supposing the rider to be sitting on the saddle, when a counter-clockwise torque is applied to the steering axis δ , the bicycle initially rotates counter-clockwise about the yaw axis z and then clockwise about the longitudinal axis x (see Figure 6)².

²This movement is consistent with the so-called countersteering: for example, to perform a right curve at high speed, what is being done is slightly push the handlebar as if you were to turn in the opposite direction (*i.e.* to the left). The bicycle responds by leaning correctly in

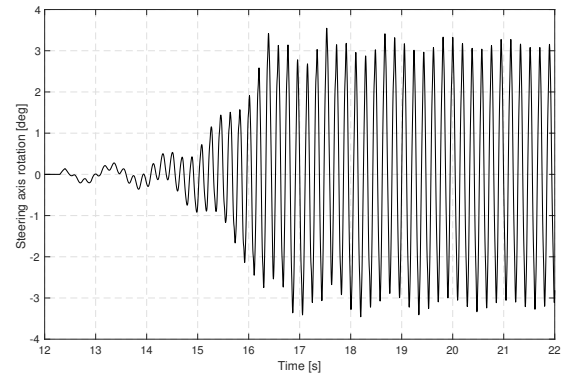


Figure 9. Zoom of the steering rotation response for the *lumped flexibility frame* model.

The oscillations related to the *weave* mode are damped because the *weave* eigenvalue computed on the linearized model passes through the imaginary axis, *i.e.* from the instability region of the complex plane (the right half-plane) to the stability area (the left half-plane). If this does not occur, the oscillations are different (not damped) and they lead to a fall of the bicycle.

4.2 Lumped Flexibility Frame Model

Simulations have been repeated considering the lateral compliance of the frame (hereinafter referred to as *lumped flexibility frame* model). A zoom of the steering axis response after the torque disturbance application is shown in Figure 9.

As can be seen, the model response to the disturbance consists of low-frequency oscillations with small amplitude (some tenths of a degree) together with high-frequency oscillations. Steering rotation reaches in a few seconds an amplitude of some degrees. Thanks to the saturation imposed to the lateral force F_y and to the aligning torque M_z , the oscillations do not diverge but their amplitude is limited in time. The initial behaviour of the steering rotation of the *lumped flexibility frame* model is very similar to the one that characterises the *rigid* version of the bicycle. This means that the degree of freedom which represents the lateral compliance β is, therefore, essential for the high-frequency contribution in the system response. Figure 10 shows the spectrograms related to roll and yaw angular velocities³.

As in the previous simulations, by applying the torque disturbance the *weave* mode is excited. Its frequency is now $f_{weave} = 0,98$ Hz. This mode is also stable: after a few seconds, in fact, the oscillations disappear because they are damped. When it happens, only the high-frequency oscillations remain in the system response. They represent the *wobble* mode. As can be seen from Figure 10, these oscillations are characterised by a fre-

the curve direction (Åström, Klein, and Lennartsson 2005).

³A spectrogram is a visual representation of the spectrum of frequencies in a signal as it varies with time.

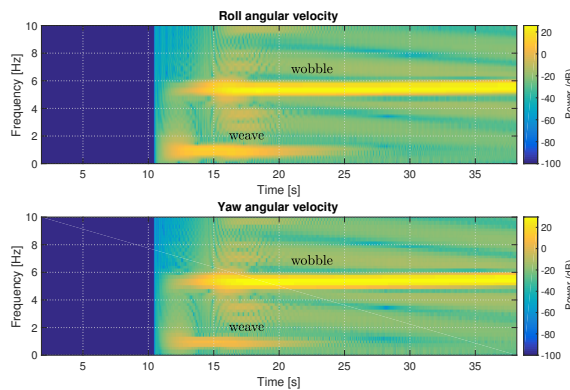


Figure 10. Spectrograms of roll and yaw angular velocities for the *flexible* bicycle. As can be noticed, the *wobble* frequency is independent with respect to the forward speed.

quency equal to $f_{wobble} = 5,43$ Hz. In the experimental activity described in (Magnani, Ceriani, and Papadopoulos 2013) it is reported that the frequency of *shimmy* for this particular racing bicycle is 7,5 Hz. This value is higher than the one obtained by the *lumped flexibility frame* model. By running other simulations, it was noted that the *wobble* frequency f_{wobble} changes varying the value of the parameter related to the frame stiffness, *i.e.* k_{β} . The same result can be achieved by changing the parameters of the spring-damper combination that models the rider's hand grip on the handlebar.

In (Magnani, Ceriani, and Papadopoulos 2013) it is said that the *wobble* frequency seems to be independent with respect to the bicycle's forward speed: this important result has been obtained also through the Dymola model (see again Figure 10).

5 Concluding Remarks

This work presented the development of a racing bicycle model in Modelica language. The model has been built trying to make it as compliant as possible to the real behaviour. For this reason, attention has been focused on the rider and on the wheel-road interaction.

By running simulations with the *rigid* model (without the frame lateral compliance), the only vibrational mode that has been excited is the *weave* mode. It has been necessary to modify the model by introducing an additional degree of freedom to highlight the *wobble* mode. This shows that it is necessary to consider both the frame lateral compliance and the tyres' deformation (also by taking into account their dynamic behaviour) to trigger the high-frequency oscillations characterising the *shimmy*.

The *wobble* mode appears when the forces and torques that arise at the contact point of the front wheel are larger than the value needed to guarantee the longitudinal alignment. In this case, the wheel begins to oscillate about the steering axis at a frequency that is too high to be counteracted by the cyclist. The use of a simple linear relation between forces and angles, as stated in equations (1)-(6),

is not sufficient. In fact, if the relation is linear, the oscillations are still present in the system response, but they are not limited in amplitude. As a consequence, both the rider and the bicycle fall in a few seconds. By adding instead a saturation at high angle values, the amplitude of the oscillations will remain limited in time.

Finally, some practical tips to be applied if the *shimmy* occurs are discussed. Overall, there is no way to stop a violent shimmy. These tips, however, are strongly recommended because they can contribute significantly to limit the amplitude of the oscillations. The first tip is the rider to assume an upright posture to increase the aerodynamic drag, thus promoting a deceleration of the bicycle. It is also suggested to tighten the horizontal tube of the rear frame with the legs, increasing in this way the structural stiffness. If necessary, gently use the rear brake. Usually, the oscillations are not divergent so it is difficult that they can lead to a fall, although this is the sensation perceived by the rider during the occurrence of the phenomenon.

References

- Andreasson, J. (2003). "Vehicle Dynamics Library". In: *Proceedings of the 3rd International Modelica Conference*.
- Åström, K. J., R. E. Klein, and A. Lennartsson (2005). "Bicycle dynamics and control". In: *IEEE Control Systems Magazine* 25.4, pp. 26–47.
- Donida, F., G. Ferretti, S. M. Savaresi, F. Schiavo, and M. Tanelli (2006). "Motorcycle Dynamics Library in Modelica". In: *Proceedings of 5th International Modelica Conference*. Vienna, Austria, pp. 157–166.
- Donida, F., G. Ferretti, S. M. Savaresi, and M. Tanelli (2008). "Object-oriented modelling and simulation of a motorcycle". In: *Mathematical and Computer Modelling of Dynamical Systems* 14.2, pp. 79–100.
- Klinger, F., J. Nusime, J. Edelmann, and M. Plöchl (2014). "Wobble of a racing bicycle with a rider hands on and hands off the handlebar". In: *Vehicle System Dynamics* 52.
- Limebeer, D. J. N. and R. S. Sharp (2006). "Bicycle, motorcycles, and models". In: *IEEE Control Systems Magazine*.
- Magnani, G., N. M. Ceriani, and J. Papadopoulos (2013). "On-road measurements of high speed bicycle shimmy, and comparison to structural resonance". In: *2013 IEEE International Conference on Mechatronics*, pp. 400–405.
- Pacejka, H. B. (1993). "The Magic Formula tire model". In: *Vehicle System Dynamics (supplement)* 21, pp. 1–18.
- (2006). *Tire and Vehicle Dynamics*. Ed. by Elsevier Ltd. second edition. Chap. chapter 4.
- Plöchl, M., J. Edelmann, B. Angrosch, and C. Ott (2012). "On the wobble mode of a bicycle". In: *Vehicle System Dynamics* 50.3, pp. 415–429.

Optimization-friendly thermodynamic properties of water and steam

Marcus Åberg¹ Johan Windahl² Håkan Runvik² Fredrik Magnusson¹

¹Department of Automatic Control, Lund University, Sweden, {marcus.berg@gmail.com, fredrik.magnusson@control.lth}

²Modelon AB, Ideon Science Park, Lund, Sweden, {johan.windahl@modelon.com, hakan.runvik@modelon.com}

Abstract

This paper describes the development of an optimization-friendly thermodynamic property model of water and steam that covers liquid, vapor, 2-phase as well as the super-critical region. All equations are at least twice continuously differentiable with respect to all model variables and can be used in dynamic optimization problems solved by efficient derivative-based algorithms. The accuracy has been verified against the industry standard IAPWS IF97 and performance and robustness have been tested by solving a trajectory optimization problem where the start-up time of a gas power plant has been minimized while satisfying constraints on temperature gradients, pressure and flows. Simulations of various plant models have also been performed to verify and benchmark the implementation. The results show that the new media can be used in both solving dynamic optimization and simulation problems yielding reliable results. The new media has been integrated into Modelon's Thermal Power library 1.13. This article is built upon the work in (Åberg, 2016).

Keywords: *Dynamic optimization, Thermodynamic properties, Power plant start-up, ThermalPower library, WaterIF97, Optimica, JModelica.org*

1 Introduction

During the last decade, optimization of large scale dynamical systems has become more common in both the industry as well as in academia (Magnusson, 2016). There are several interesting areas and applications where optimization can be used, e.g. to improve efficiency and economical aspects in energy applications. Examples where Modelica models have been used include start-up of power plants (Casella, Donida, & Åkesson, 2011), (Runvik, 2014), (Parini, 2015), production planning of district heating networks (Velut, et al., 2014) and power plant load scheduling (Kumar & Mathur, 2014). Modelica is well suited to describe the behavior of dynamical models and thereby also suitable to be used in the context of optimization.

Even if the usage is more common today, the use of dynamic optimization is still not widely spread among the engineering community as compared to simulation. There are several factors that have been limiting the deployment:

- Modelica does not support formulation of optimization problems. However, it can easily be formulated using the Modelica extension Optimica (Åkesson, 2008) or using custom annotations (Zimmer, Otter, Elmqvist, & Kurzbach, 2014)
- It is more challenging to create optimization models versus simulation models. Solving efficiently large-scale dynamical optimization problems requires the model equations to be at least twice continuously differentiable. In the general case when solving non-convex dynamic optimization problems good initial guess values, appropriate model dynamics and as well as good numerical properties are required to find the optimal solution (Nocedal & Wright, 2006)
- Modelica libraries such as the Modelica standard library have been designed for simulation and not optimization. The lack of libraries for optimization is usually a stopper as creating robust models is a large effort and requires an understanding of numerical aspects.

This work targets the last issue and is intended as a first step to bridge the gap between simulation and optimization of thermo-fluid systems. We do so by implementing an optimization-friendly water and steam property model that fulfills a generic media interface.

Modelica is object oriented and supports design of interfaces and classes. This allows a library designer to create models of various fidelity and assumptions. Users can then change between classes that fulfill the constraining interface. Examples include switching to a media of lower fidelity that is less computationally demanding

for e.g. real-time applications or to a model suitable for optimization.

The choice of focus on water and steam properties is due to its large usage in power and heat applications. Traditional electricity-generation sources such as coal, nuclear and natural combined gas plants are based on a steam-cycle. Other applications are hydro power plants and heating and cooling distribution networks.

2 Background

The availability of a Modelica implementation of the industry standard of water and steam properties IF97 (Wagner, et al., 2000) helped to spread the usage of the Modelica technology to the energy and power sector (Windahl, et al., 2014). But the high accuracy implementation Modelica.Media.WaterIF97 is targeting the usage of simulation and not optimization. The main issues with using Modelica.Media.WaterIF97 for optimization are:

- limited support of first order and no support of second order partial derivatives of thermodynamic properties
- discontinuous first order partial derivatives at the phase borders between liquid and steam
- discontinuous first order partial derivatives at the region boundaries. IF97 is divided into 5 regions that have their own implementation (Wagner, et al., 2000)

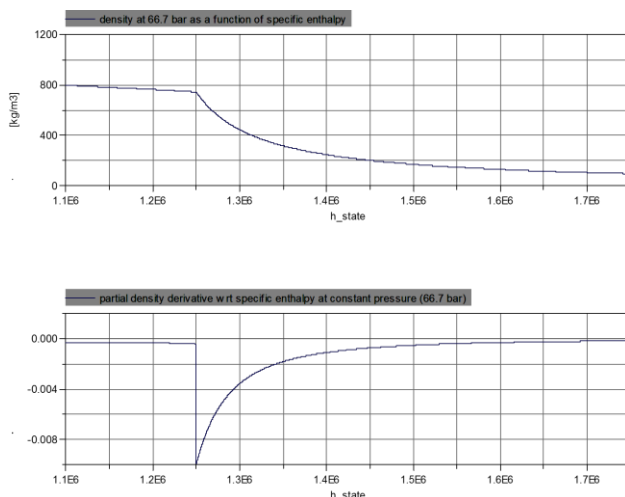


Figure 1 Density (upper) and its partial derivative with respect to specific enthalpy at constant pressure as a function of specific enthalpy. At $h=1250$ kJ/kg is the bubble saturation line for water which introduces a discontinuity in the partial derivative.

The lack of support of derivatives is an implementation issue. Modelon.Media.WaterIF97, a similar implementation, has support for first order derivatives. But the discontinuity at the phase regions, as illustrated in Figure 1, is a fundamental limitation. The formation or depletion of a phase is a strong non-linear process and needs to be approximated to be twice continuously differentiable. The models in this work are implemented to be compatible with JModelica.org's dynamic optimization framework (Magnusson & Åkesson, 2015). This framework uses CasADi (Andersson, 2013), to efficiently compute sparse first and second order derivatives using algorithmic differentiation (Griewank & Walther, 2008).

2.1 Previous work

To the authors knowledge there is no published work related to dynamic optimization of energy and power systems that focus on a generic media implementation. (Velut, et al., 2014) and (Runvik, 2014) mention the use of “smooth media model functions” but don't go into any detail. (Casella, Donida, & Åkesson, 2011) use simplifications such as incompressible fluids with constant heat capacity for non-saturated liquid and steam. (Parini, 2015) approximates the subcooled liquid as incompressible fluid and describe the superheated vapor using a cubic equation of state but does not describe any accuracy or region of validity. (Windahl, et al., 2014) investigate requirements for a new media interface, mentioning the benefit of an interface that supports analytic calculations of the Hessian but don't go any further. (Schulze, 2014) focuses on numerically efficient implementation but does so from a simulation perspective. This is also the focus of the guideline on the fast calculation of steam and water properties with the spline-based table look-up method (International Association for the Properties of Water and Steam, 2015)). The latter uses quadratic splines that are continuously differentiable once and therefore not suitable for dynamic optimization.

This article is built upon the work in (Åberg, 2016). To this publication the media implementation has been updated with some minor modifications that have made the model more numerically efficient compared to the implementation used in that thesis. Therefore the results in this article have been updated too.

3 Implementation

The approach chosen was to approximate the thermodynamical functions with polynomials over different operating regions in the p - h , p - s , p - T and d - T plane. These approximations are then connected via smooth step functions from one region to another. In that way, the functions are twice continuously differentiable over the whole working regime.

Polynomial approximation over different regions has the main advantage that it is smooth over the defined region. The main challenge here is to find a way to accurately and smoothly connect the different regions. In this implementation step functions are used to make a smooth transition between the regions. These can be defined so that the functions are twice continuously differentiable and the smoothness requirement hence is fulfilled.

The functions that were implemented can be divided into 1D and 2D-functions. 1D-functions describe the saturated behavior in the two-phase region. Saturation temperature, bubble and dew enthalpy are quantities that can be calculated directly from the pressure. The 2D-functions take two independent state properties (Thorade & Saadat, 2013) and calculate thermodynamic properties and a few partial derivatives.

The methods of least squares are used to fit a univariate or bivariate polynomial to the specified data set. The maximum order of the polynomials was set to $k = 9$ on following form.

$$p(x_1, x_2) = \sum_{i=0}^k \sum_{j=0}^k b_{ij} x_1^i x_2^j$$

$$p(x) = \sum_{i=0}^k b_i x^i$$

If weights are used in the least-squares regression, certain data points can be given a greater importance in the fitting process. This is used to give points closer to the phase border a greater weight in the fit. Making the residual smaller close to the border allows for a smoother transition between the different phases.

3.1 Regions

The regions are referred to as liquid, vapor and two-phase region. The liquid and vapor regions are divided into sub- and super-critical areas. The region of a certain point is decided by its p and h values. Figure 2 shows the phase diagram in the p - h plane with all of the regions.

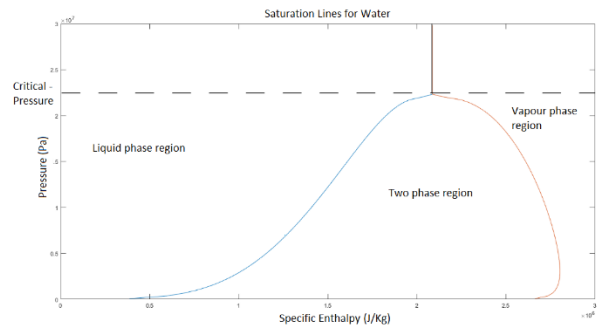


Figure 2 Phase diagram of water, saturation lines are drawn with approximated functions. Regions are divided into super- and sub-critical for both liquid and vapour.

Furthermore, it was noticed later in the process that accurate media calls were needed for very low pressures. Thus, a super low pressure region was added to the functions.

3.2 The smooth step function

The method used for making a smooth transition between regions is via a smooth step function S . The idea is to multiply the polynomial defining the function over a certain region with a function so that the function assumes the polynomial fit's value within the region and goes to zero outside this specific region. The desired properties of S are

$$S(x) = \begin{cases} 0, & x \leq 0 \\ x, & 0 \leq x \leq 1 \\ 1, & 1 \leq x \end{cases}$$

The right- and left borders of the step has been chosen to 1 and 0 for easy implementation and the scaling can be done when calling the function by scaling the input parameter.

Since the overall goal with this implementation is to make the media implementation twice continuously differentiable the step function must also be so.

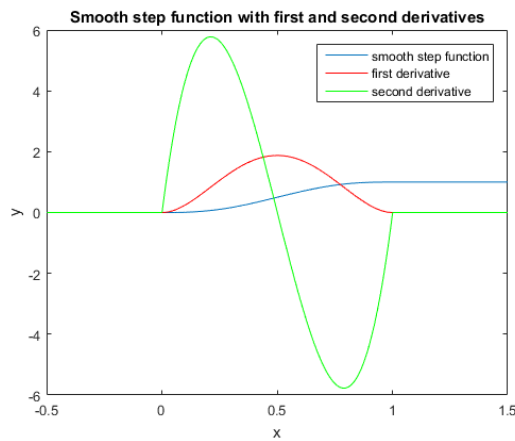


Figure 3 Smooth step function with its first and second derivative.

For this purpose, a generic 5th order polynomial can be used. If the boundary conditions on the derivatives and the function are applied the following solution is found.

$$S(x) = \begin{cases} 0, & x \leq 0 \\ 6x^5 - 15x^4 + 10x^3, & 0 \leq x \leq 1 \\ 1, & 1 \leq x \end{cases}$$

3.3 The approximating polynomials

The data needed for making the polynomial fits was extracted from Modelica.Media.WaterIF97.

The grid in the p-h plane that was used for data extraction was 100x100 points and linear along the h-axis with range [1.0e5, 4.0e6] (J/kg). A logarithmic scale was used for the p-axis with approximately the range [7.6e4, 3.0e7] (Pa).

For 2D-functions that use d, T and s as inputs, the response data from IF97 for constructing the functions that calculates these properties from p and h were used instead. This was done since it is hard to construct a grid that does not contain points outside of the domain of definition for these properties.

It is of extra importance that the polynomial fits have high accuracy close to the borders to other regions, since they are to be connected to another polynomial function there. Big differences in the values of the different surfaces close to the border will lead to a "leap" in the function value at the border. Even though the step function smoothes this leap out and makes sure the function is continuous it is of extra importance that this difference is made as small as possible since the model is to be used in optimization algorithms which can get stuck at inconsistencies like this. The approach used for handling this problem is to give data points at

the borders between different regions a higher weight to make the linear regression generate polynomials which are accurate at the border. However this might cause "overshoots" in the rest of the region if the border points are weighted too much. This method was therefore used only when this phenomenon did not cause relatively large residuals inside the considered region.

Furthermore, weighting is used to make the least-square algorithm minimize the relative errors instead of absolute. Since some of the approximated functions range largely in value, data points which have small response reference values (close to zero) will get very large relative errors if weighting is not performed.

3.4 Accuracy of implemented media functions

Since 18 functions have been implemented, only a few important examples will be accounted for in this section.

3.4.1 Temperature

The temperature function is shown in Figure 4 and has a maximum relative error of around 0.8% as seen in Figure 5. The red line in the figure represents the phase border. The relative error is calculated as the percentage difference between the implemented approximation and IAPWS IF97.

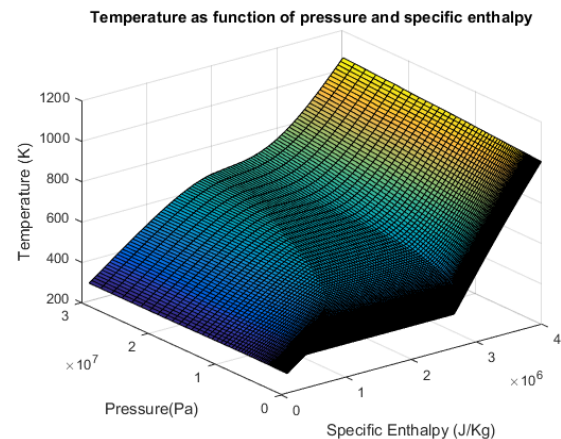


Figure 4 The approximated temperature function.

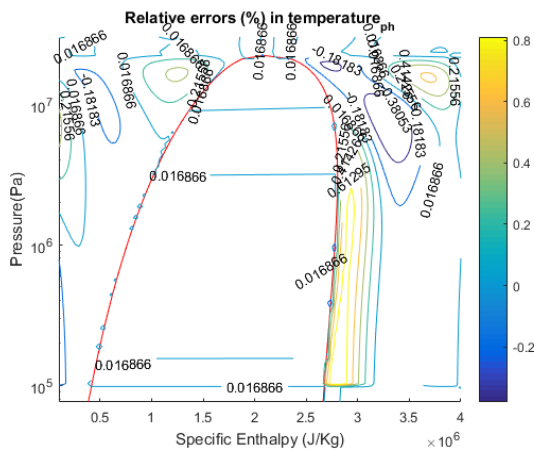


Figure 5 Contour plot of relative errors of the approximated temperature function.

There are a couple of interesting things to note from the relative error plot. At low pressures and high specific enthalpies there is a distinct drop in the relative error. This is because a new region was added for sub-1 bar pressures in the vapor region to get higher accuracy at components such as condensers which operate at very low pressures. Another thing to note is that the highest relative error is located after the phase border at the vapor side. The coefficients here have been weighted in a way to be consistent with the saturated properties at the phase border. This weighting might cause this bulge as the least squares-algorithm prioritizes minimizing the error at the border instead of inside the region but with the benefit of better consistency at the phase border.

3.4.2 Density

The density function and the corresponding relative errors can be viewed in Figure 6 and 7, respectively. As can be seen, there is a "spike" in the relative error around the critical point, which is due the connection of the three regions, and the value there is an interpolation between the function approximations in all three regions.

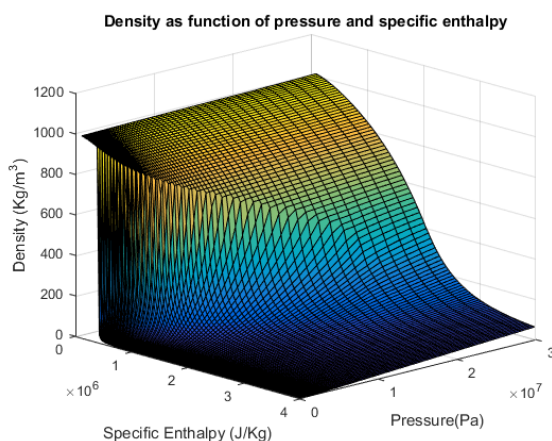


Figure 6 The approximated density function.

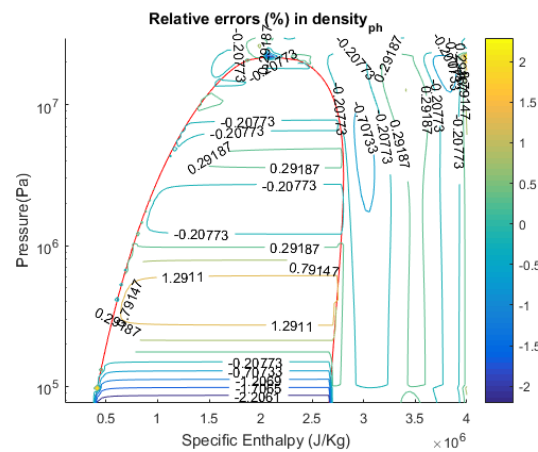


Figure 7 Contour plot of relative errors of the approximated density function

The same behavior at low pressures in the vapor region can be seen in the relative error plot as in the temperature function due to the same reasons, that is, an added region at low pressures. The spike in relative errors is due to the fact that 3 regions meet at the critical point. If the density function is compared with the temperature function, which has a similar point, it can be seen that the temperature function is rather flat at the critical point where the density is rather steep.

4 Optimization benchmarking case: Start-up of a Heat Recovery Steam Generator (HRSG)

For testing the implementation in optimization applications, a model describing a start-up phase of a heat recovery steam generator (HRSG) has been chosen.

4.1 Description of the HRSG model

The model used in this thesis has been built upon a model developed for a tutorial, for further information and material from this tutorial please see (Larsson, 2015). A similar model and optimization problem is investigated thoroughly in a master thesis previously written in cooperation with Modelon (Runvik, 2014).

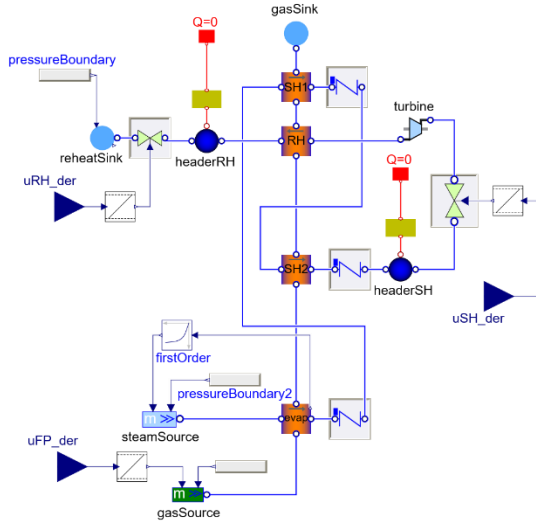


Figure 8 Model diagram view of the system used in the optimization.

The main working components of the model are a series of heat exchangers transferring heat from a flue gas source to the water medium until the steam reaches a desired working state. The flue gas is led into an evaporator where the water is evaporated into steam. A feedback loop connected to the evaporator keeps the water level in the evaporator on a constant level. From here, the steam goes through two superheaters where the steam pressure and temperature increase to reach the desired working levels. After superheater 2 the steam is collected in a superheater header, where in reality the pipes are collected and the steam can be redirected into a turbine step. There is also a wall model connected to the header. One of the main issues with the start-up phase of the power plant is the exposure of thermal stresses in the components, and thus this has to be modelled. The wall models allows for the modelling of these stresses. There is a valve located after the header which can be used to control the temperature and pressure inside the header. When the steam has reached high enough temperature and pressure it can instead of going through the control valve, be redirected into a turbine step. To maximize energy output of the plant, the steam is thereafter led into a reheater step. After going through the reheater header it could once again be led through another turbine step. Again, a control valve is added to be able to control the pressure and the temperature inside the header.

The controllable inputs of the model are the firing power of the gas source and the opening of the valves located after superheater 2 and the reheater header. These inputs can be used to control the pressure and temperature inside the headers and consequently can be used to limit the thermal stresses inside the header walls. As can be seen in Figure 8 integrator steps are added to the control inputs. This was done to be able to

put constraints on the rate of change of the optimizing input signals.

4.2 Optimization problem formulation

The aim of the start-up is to take the plant from the initial operating point to another operating point as fast as possible without violating the problem constraints. The preferred properties of this process to reach this point are:

- Control the system from the initial operating point to a point where the steam in the plant has high enough quality to be redirected to turbines.
- The thermal stresses inside the header walls should be limited to extend the lifespan of the components.
- The controllable inputs have rate of change constraints which must be obeyed.

The optimal control problem defined over the time interval $[0, t_f]$ is stated as

$$\begin{aligned} \min \int_0^{t_f} & w_{TSH2} (T_{SH2} - T_{SH2ref})^2 \\ & + w_{pSH2} (p_{SH2} - p_{SH2ref})^2 \\ & + w_{pRH} (p_{RH} - p_{RHref})^2 \\ & + w_{SH2v} \dot{u}_{SH2v}^2 + w_{RHv} \dot{u}_{RHv}^2 \\ & + w_b \dot{u}_b^2 dt \end{aligned}$$

subject to

model equations

$$\begin{aligned} dT_{SH2} & < dT_{SH2}^{max} \\ dT_{RH} & < dT_{RH}^{max} \\ |\dot{u}_{SH2v}| & < \dot{u}_{SH2v}^{max} \\ |\dot{u}_{RHv}| & < \dot{u}_{RHv}^{max} \\ |\dot{u}_b| & < \dot{u}_b^{max} \end{aligned}$$

The first three terms of the objective function correspond to the penalties on temperature and pressure deviations from the desired values inside the heat exchangers (same as in the headers). T_{SH2} and p_{SH2} are the temperature and pressure inside superheater 2 (T_{SH2ref} and p_{SH2ref} the desired values), p_{RH} the pressure inside the reheater. w are the corresponding weights. The last three terms represent the derivatives of the control inputs, \dot{u}_{RHv} is the reheater valve signal, \dot{u}_{SH2v} the superheater 2 signal and \dot{u}_b the boiler control signal.

The model equations are the equations that describe the dynamics of the system. dT_{SH2} is the thermal gradient in the superheater 2 header wall and dT_{RH} is its counterpart in the reheater header wall. The last three

constraints put upper and lower limits on the derivatives of the three control signals.

4.3 Optimization

A direct collocation method (Magnusson & Åkesson, 2015) is used for solving the optimization problem. The time horizon is divided into 12 elements, using 4 collocation points in each element. The element grid points are located so that they are closer together in the first part of the time horizon, to better capture the transient behavior at the beginning of the start-up. 3/4 of the elements are in the first 3/8 of the time horizon and 1/4 in the last 5/8.

Optimization statistics are summarized in Table 1. The optimization model has 8 continuous time states and 85 algebraic variables. This model is translated into a non-linear program with 5184 variables.

Table 1 Optimization statistics

DAE model	
Number of states	8
Number of algebraic variables	85
NLP model	
Total number of variables	5184
Solution statistics	
CPU-time in IPOPT (s)	1.45
CPU-time in NLP function evaluations (s)	1.56
Solution time (s)	3.11

4.4 Verification through simulation

To verify the result the optimized signals were extracted and used in a simulation experiment using Water-IF97 media functions. The trajectories for these simulations are displayed in Figures 9 and 10 alongside the trajectory from the optimization.

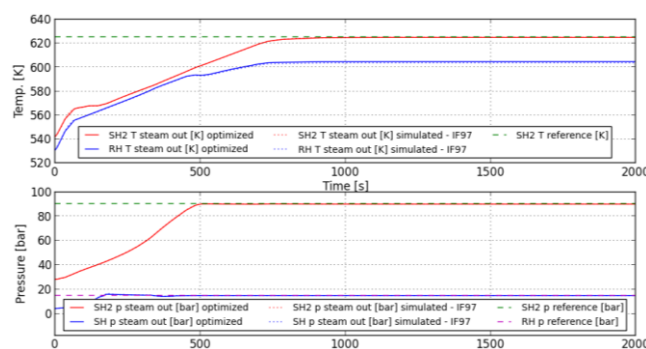


Figure 9 Temperature and pressure signals from optimization (solid) and simulation (dotted). In simulation, the optimal input signals are used as input, and the medium is modeled with Water IF97 thermodynamic property functions.

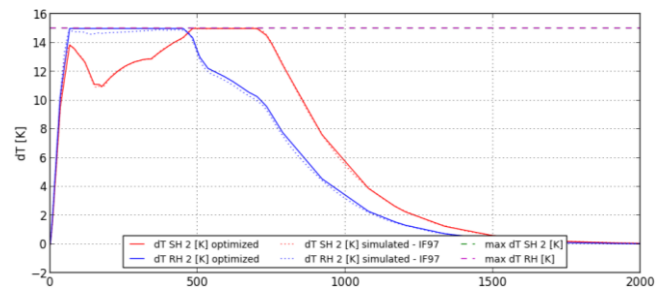


Figure 10 Metal wall temperature gradient signals from optimization (solid) and simulation (dotted). The dashed line represents the maximal allowed wall stress.

The simulation results match the optimized trajectory well, which indicates two things. Firstly, it indicates that the time discretization of the optimization model is sufficient to capture the dynamics of the model. Secondly, it indicates that the implemented media functions give very similar results to the IF97 functions.

5 Dynamic simulation

To verify that the media model can also handle industrial relevant dynamic simulation use cases, it was tested with large dynamic simulation examples in the Thermal Power Library. These tests expose the media implementation over various properties and under different operating conditions. As throughout this article, the Water-IF97 media implementation will be used as the reference medium.

Three use cases were set up:

1. Coal fired 400 MW electrical super-critical steam cycle that operates at a maximum pressure of 300 bar and 580 °C. The model consists of 5683 equations and 193 continuous time states.
2. Heat recovery steam generator (HRSG) that operates at a pressure around 84 bar and in a temperature interval of 175-500 °C. The model consists of 1616 equations and 39 continuous time states. In comparison with the optimization test case, this model includes more dynamics and describes the considered system more thoroughly.
3. Nuclear steam generator system that operates at a maximum pressure of 70bar and 285 °C. The model consists of 6708 equations and 147 continuous time states.

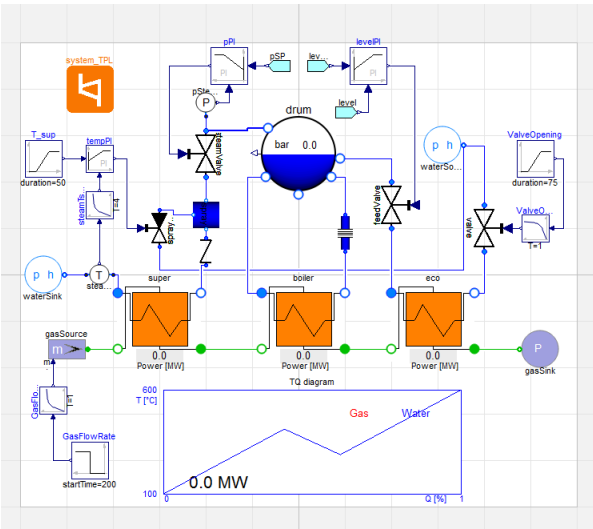


Figure 11 Model diagram view over the HRSG-model (test case 2)

The test cases were simulated on a standard laptop (Dell Latitude E7470, Intel i7-6600U) using the Modelica simulation tool Dymola 2017 with the solver Dassl and a tolerance of 1e-5.

5.1 Result

The result is summarized in the tables below. Using the new media implementation, a speed-up of up to 40% can be achieved in an industrial relevant large-scale power plant simulation. The difference in result of selected important variables is below 0.6% in use case 1 and 2 and 2.9% in use case 3, however it may be larger for certain intermediate pressure variables. The larger deviation in use case 3 is mainly due to a deviation in the isentropic efficiency calculation at the last turbine stage. This may be improved by dividing the specific entropy polynomial into regions at lower pressure in a similar way as was done with the density function. If trajectories from the simulations are compared, the results seem to match well as can be seen in Figure 11, showing the total power transferred from the exhaust gas to the steam through all three heat exchanger stages in use case 2.

The CPU-time is a combination of the computational effort that is required to do one integrator step and the number of steps. Even if a media implementation is faster the CPU-time of a simulation may increase due to an increase of the number of integrator steps. This may happen if the implementation contains variations due to the use of e.g. higher order polynomials or transitions between computational regions. The F-evaluations describe the number of function evaluations of all system equations. They are used in the integration process to evaluate derivatives and calculate numerical system Jacobians. Dassl use the Jacobian in its internal solver process (Petzold, 1982).

Table 2 Simulation statistics use case 1 (super-critical power plant simulated 25000s).

	Optimization media	WaterIF97 (reference)
Simulation statistics		
CPU-time (s):	24.5	34.6
Solver steps	825	862
F-evaluations	7310	8857
Jacobian-evaluations	125	158
Steady-state results		
Generated power	405.8 MW	408.2 MW
Condenser temperature	297.64K	297.60

Table 3 Simulation statistics use case 2 (HRSG simulated 200s).

	Optimization media	WaterIF97 (reference)
Simulation statistics		
CPU-time (s):	7.98	8.38
Solver steps	320	296
F-evaluations	3819	3412
Jacobian-evaluations	105	94
Steady-state results		
Total heat transfer	286.8 MW	287.1 MW
Steam outlet flow	10.69 kg/s	10.75 kg/s
Gas exhaust temperature	554.3K	554K

Table 4 Simulation statistics use case 3 (nuclear plant simulated 25000s).

	Optimization media	WaterIF97 (reference)
Simulation statistics		
CPU-time (s):	12.9	18.3
Solver steps	1112	1044
F-evaluations	14150	16048
Jacobian-evaluations	448	392
Steady-state results		
Generated power	432 MW	420 MW
Condenser temperature	33.67 °C	33.37 °C

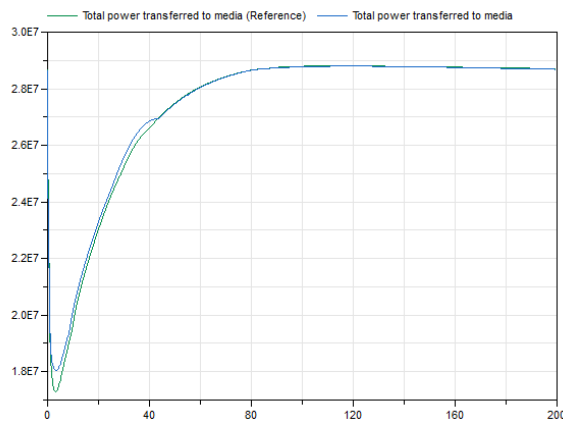


Figure 12 Total power transfer from exhaust gas to steam in use case 3.

6 Conclusions

Efficient optimization-friendly properties of water and steam covering sub-critical and super-critical regions have been implemented in Modelon's Modelica Thermal Power library 1.13. The new medium can bridge the gap between simulation and optimization and was tested against industrial relevant thermo-fluid systems. It was shown in the optimization benchmarking case that the implemented media functions could be used to provide results that coincide well with IF97 simulation results using the resulting optimal control inputs. This shows that the implementation suggested in this article can yield reliable results.

The simulation benchmarking test cases aimed at comparing the accuracy and performance of the new implementation with the existing Water-IF97 media implementation. The results from these simulations show that there are some slight deviations in the results between the implementations. However, the dynamics of the system are captured accurately and the relative errors are small. The largest deviations are observed at rapid transients. That there are deviations is expected as the implementation approximates the Water-IF97 standard. The question is whether these differences are small enough to yield acceptable results and in the tested simulation models this seems to be the case for a majority of the use cases. Comparing the simulation statistics of the large plant use cases shows that the new implementation is up to 40% faster.

6.1 Future work and possible improvements

It is desirable that the media is accurate at the phase borders. The length of the smoothing interval impacts the derivatives of the functions in the implementation. A smaller delta makes the transition between the polynomials go faster and hence making the function

"less smooth" even though the implementation in theoretical sense still is twice continuously differentiable. However, making this parameter too big will instead decrease the accuracy in a larger region around the region borders.

When modelling thermodynamic properties, there are many natural laws to consider, which might not totally be satisfied by the approximations made as there is no check on whether such relations are fulfilled. An example of this is that by nature the density must increase with increased pressure if the temperature is kept constant. Iterative solvers that use gradients based on the function approximations might be affected if there are inconsistencies in such relations.

Furthermore, the choice of functional form in the least squares approximations might be investigated. There might be better forms of functions to represent the functions. In (Aute & Radermacher, 2014) the use of Chebyshev Rational polynomials is proposed for fast evaluation of thermodynamic properties. The use of different functional forms might be a way of making the implementation faster and more accurate.

For easy implementation of similar models describing the thermodynamic properties of other media than water, it is desirable to standardize the implementation. Ideally the whole work-flow would be automated so that the only thing that would have to be provided to create a new media model is the tables containing the thermodynamic property data. This has however been hard to achieve, as the many different functions that have been approximated have different shapes and appearances making it hard to construct an automated form for all these functions. It has been necessary to make specialized forms and adaptations for many of the functions to achieve good accuracy.

7 Acknowledgements

Fredrik Magnusson acknowledges support from the LCCC Linnaeus Center and eLLIIT Excellence Center at Lund University.

8 References

- Andersson, J. (2013). *A general-purpose software framework for dynamic optimization*. Ph. D. thesis. Faculty of Engineering, KU Leuven, Leuven, Belgium.
- Aute, V., & Radermacher, R. (2014). Standardized polynomials for fast evaluation of refrigerant thermophysical properties. *International Refrigeration and Air Conditioning Conference at Purdue*. Purdue, Indiana.

- Casella, F., Donida, F., & Åkesson, J. (2011). Object-Oriented Modeling and Optimal Control: A Case Study in Power Plant Start-Up. *18th IFAC World Congress*.
- Griewank, A., & Walther, A. (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, second edition*. ISBN: 978-0-89871-659-7.
- International Association for the Properties of Water and Steam. (2015). *Guideline on the Fast Calculation of Steam and Water Properties with the Spline-Based Table Look-Up Method (SBTL)*.
- Kretzschmar, H.-J., & Wagner, W. (2008). *International Steam Tables. [electronic resource] : Properties of Water and Steam Based on the Industrial Formulation IAPWS-IF97*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg.
- Kumar, S., & Mathur, T. (2014). Dynamic Load Scheduling of Optimization of Power Plants. *Advanced Control of INdustrial Processes (AdCONIP)*. Hiroshima University, Hiroshima, Japan.
- Larsson, P.-O. (2015, October). Report from the modelon tutorial at the 2015 modelica conference. Retrieved from <http://www.modelon.com/blog/articles/report-from-the-modelon-tutorial-at-the-2015-modelica-conference/>
- Magnusson, F. (2016). *Numerical and symbolic methods for dynamic optimization*. PhD thesis, Lund University, Department of Automatic Control, Lund, Sweden.
- Magnusson, F., & Åkesson, J. (2015). Dynamic Optimization in JModelica.org. *Processes*, 3(2), 471-496.
- Nocedal, J., & Wright, S. (2006). *Numerical Optimization*. New York, NY: Springer New York.
- Parini, P. (2015). *Object Oriented Modeling and Dynamic optimization of energy systems with application to combined-cycle power plant start-up*. Msc thesis, Politecnico di Milano, Milano.
- Petzold, L. R. (1982). *A Description of DASSL: A Differential Algebraic System Solver*. Presented at IMACS World Congress, Montreal, Canada, August 8-13, 1982.
- Runvik, H. (2014). *Modelling and start-up optimization of a coal-fired power plant*. Master's thesis, Lund University, Department of Automatic Control, Lund.
- Schulze, C. (2014). *A Contribution to Numerically Efficient Modeling of Thermodynamic Systems*. PhD thesis, Technische Universität Braunschweig, Fakultät für Maschinenbau.
- Thorade, M., & Saadat, A. (2013). *Partial derivatives of thermodynamic state properties for dynamic*. *Environmental Earth Sciences*, 70, 8, 3497-3503.
- Wagner, W., Cooper, J., Dittmann, A., Kijima, J., Kretzschmar, H.-J., Kruse, A., . . . Trübenbach, J. (2000). The IAPWS Industrial Formulation 1997 for the Thermodynamic properties of Water and Steam. *J. Eng. Gas Turbines Power* 122, 150-182.
- Velut, S., Larsson, P.-O., Runvik, H., Funqvist, J., Bohlin, M., Nilsson, A., & Modarrez Razavi, S. (2014). Production Planning for Distributed District Heating Networks. *11th International Modelica 2015 Conference*. Versailles, France.
- Windahl, J., Prölss, K., Bosmans, M., Tummescheit, H., van Es, E., & Sewgobind, A. (2014). MultiComponentMultiPhase - A framework for thermodynamics in Modelica. *Proceedings of the 11th International Modelica Conference*. Versailles, France.
- Zimmer, D., Otter, M., Elmqvist, H., & Kurzbach, G. (2014). Custom Annotations: Handling Meta-Information in Modelica. *Proceedings of the 10th International Modelica 2014 Conference*. Lund, Sweden.
- Åberg, M. (2016). *Optimisation-friendly modelling of thermodynamic properties of media*. Master's thesis, Lund University, Department of Automatic Control, Lund. Retrieved from <http://lup.lub.lu.se/student-papers/record/8888181>
- Åkesson, J. (2008). Optimica - An Extension of Modelica Supporting Dynamic Optimization. *Proceedings of the 8th International Modelica 2008 Conference*. Bielefeld, Germany.

Modeling of a Thermosiphon to Recharge a Phase Change Material Based Thermal Battery for a Portable Air Conditioning Device

Rohit Dhumane Jiazhen Ling Vikrant Aute Reinhard Radermacher

Center for Environmental Energy Engineering, University of Maryland, College Park, 4164 Glenn L. Martin Hall
Bldg., MD 20742, USA

{dhumane, jiazhen, vikrant, raderm}@umd.edu

Abstract

Closed loop two phase thermosiphons have a broad range of applications due to their simplicity, reliability, low cost and the ability to dissipate high heat fluxes from minimal temperature differences. The present study focuses on one thermosiphon operation which solidifies a phase change material (PCM) based thermal battery for a portable air conditioner called Roving Comforter (RoCo). RoCo uses vapor compression cycle (VCC) to deliver cooling and stores the heat released from the condenser into a compact phase change material (PCM) based thermal battery. Before its next cooling operation, the PCM needs to be re-solidified. This is achieved by the thermosiphon, which operates within the same refrigerant circuitry with the help of a pair of valves. The molten PCM which acts as heat source affects the dynamics of the thermosiphon which in turn affects the solidification process. Thus the dynamics of both the PCM and thermosiphon are coupled. For accurate transient modeling of this process, the PCM model considers the solidification over a temperature range, variable effects of conduction and natural convection during the phase change and variable amounts of heat release at different temperatures within the temperature range of phase change. The paper discusses component modeling for this transient operation of thermosiphon and its validation with experimental data.

Keywords: Thermosiphon, Thermosyphon, Phase Change Materials

1 Introduction

A thermosiphon is an energy transfer device capable of transferring heat from a heat source to a heat sink over a relatively long distance, without the use of active control instrumentation and any mechanically moving parts such as pumps (Dobson and Ruppersberg, 2007). Thermosiphons are used in diverse applications like cooling of electronic components, light water reactors, solar water heating systems, geothermal systems, and thermoelectric refrigeration systems due to simple designs, simple operating principles and high heat transport capabilities (Franco and Filippeschi, 2011). Lack of moving component for pumping refrigerant also leads to higher reliability of the system. Thermosiphons may operate with

single phase fluid or two phase fluid, may consist of a co-current or counter-current flow (Haider et al., 2002) and have open or closed loops (Benne and Homan, 2009). The counter-current thermosiphons are also referred to as heat pipes. Industrial applications typically involve the co-current thermosiphons and the term thermosiphon used henceforth in this paper, will refer to these co-current thermosiphons.

A closed-loop two-phase thermosiphon consists of a closed circuit of refrigerant tube filled with a working fluid (referred to as refrigerant in this paper) and oriented in a vertical plane. The refrigerant evaporates in the lower portion of the loop (called evaporator) due to a heat input. The resultant vapor then travels upwards through a vertical tube called as the riser to reach the condenser, where it rejects its latent heat. The condenser is located vertically above the evaporator and the condensed refrigerant from its outlet trickles down into the evaporator by gravity through the downcomer tube. The cycle repeats until the heat source is exhausted.

The current study is motivated by a need to understand the dynamics of a thermosiphon used to recharge the thermal battery of a portable air conditioning device called Roving Comforter (RoCo) (Du et al., 2016). The dynamic model is expected to aid the improvement of design and development of controls. A brief description of RoCo is given in the next section.

2 System Details

Traditional HVAC (Heating, Ventilation and Air Conditioning) systems consume significant amounts of energy to maintain a uniform temperature in the buildings within a narrow range, neither of which is necessary for delivering comfort (Hoyt et al., 2015). Personal conditioning systems like RoCo provide an opportunity to save the building energy by relaxing the building thermostat settings without compromising occupant thermal comfort.

RoCo uses vapor compression cycle (VCC) to deliver cooling for building occupants and stores the waste heat from the condenser in a compact PCM based thermal battery. The schematic of the two modes of operation of RoCo is shown in Figure 1. When the PCM is molten, the VCC operation is terminated. Due to the poor thermal conductivity of PCM, the molten PCM cannot solid-

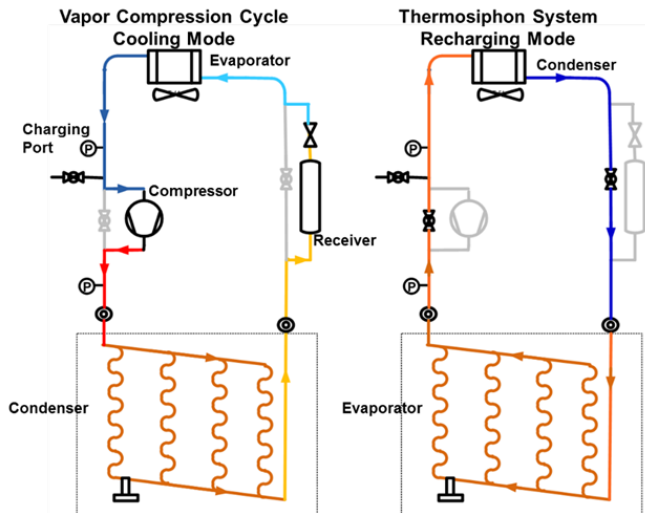


Figure 1. Schematic of two modes of operation of RoCo with the thermal battery marked in grey box

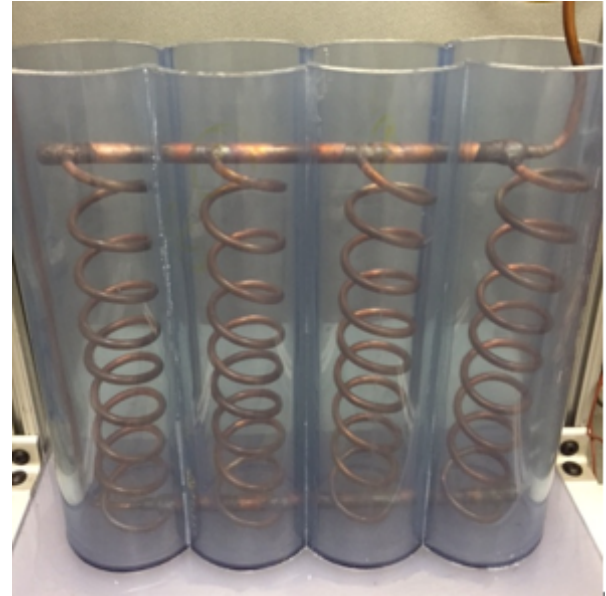


Figure 2. The thermal battery of RoCo in the experimental setup (Du et al., 2016)

ify by itself within a reasonable time duration by rejecting heat to ambient air. Consequently, to enable a faster recharge of the thermal battery (i.e. PCM solidification), a thermosiphon is used. The thermosiphon operation is ideal in this situation because of its high rate of heat dissipation even from relatively small temperature differences between the heat source and the heat sink. The refrigerant circuitry is designed to enable a single direction flow of refrigerant. By operating a pair of valves, the refrigerant circuit switches from VCC circuit to thermosiphon circuit.

The PCM selected for the current application is paraffin-based, with the midpoint of its solidification temperature range at 35°C. The temperature choice is based on a trade-off between two opposing factors. The temperature should be high enough so that the PCM does not solidify at typical room temperatures (< 26°C). At the same time, the temperature should also be low enough so that the condenser temperature for VCC operation is not very high. Higher condenser temperature leads to poor coefficient of performance (COP). Thus, a very narrow range of temperature range is applicable for the solidification temperature of PCM in RoCo. Paraffin based PCM is chosen because as a class paraffin is safe, reliable, predictable, less expensive and non-corrosive. It melts and freezes repeatedly without phase segregation and consequent degradation of its latent heat of fusion (Sharma et al., 2009). It crystallizes with little or no supercooling (Sharma et al., 2009). The only major disadvantage of paraffin based PCM is its low thermal conductivity. To address this issue, the thermal battery consists of helical coils of refrigerant tubing enclosed within PCM volume (See Figure 2). This arrangement enables higher surface area of heat transfer and better reach within the PCM volume.

3 Model Development

The system model for the thermosiphon consists of several components which are shown in Figure 3. The evaporator (See Figure 2) consists of four symmetric refrigerant circuits and to save computational effort, only one of them is modeled. The `splitter` and `mixer` components are used to scale the dynamic behavior of a single refrigerant circuit to the complete evaporator. The `splitter` divides refrigerant mass flow rates equally into four, while the `mixer` combines them. The refrigerant then flows into the riser, condenser, refrigerant tube and downcomer before flowing back to the evaporator. The refrigerant tube is a non-adiabatic flow passage for the refrigerant. The PCM blocks are connected to a tube control volume, which is a simple model of circular wall for pipes. The tube control volume component is also used to model the pcm container. Finally, the heat losses by natural convection and radiation from the pcm container to the ambient are incorporated. Detailed description of the component models is provided in this section.

3.1 Phase Change Material

Recall that the PCM Heat Exchanger (PCM-HX) consists of helical refrigerant tubes surrounded by PCM. The PCM solidification is a complex phenomenon due to the fact that the solid-liquid boundary moves depending on the rate of heat transfer and hence its position with time forms part of the solution (Zalba et al., 2003). The rate of heat transfer varies progressively during the phase change due to the varying effects of conduction and natural convection which depends on the state of PCM. Thus, the dynamics of PCM and thermosiphon are coupled. The helical nature of the refrigerant tube further increases the complexity by making the problem three-dimensional.

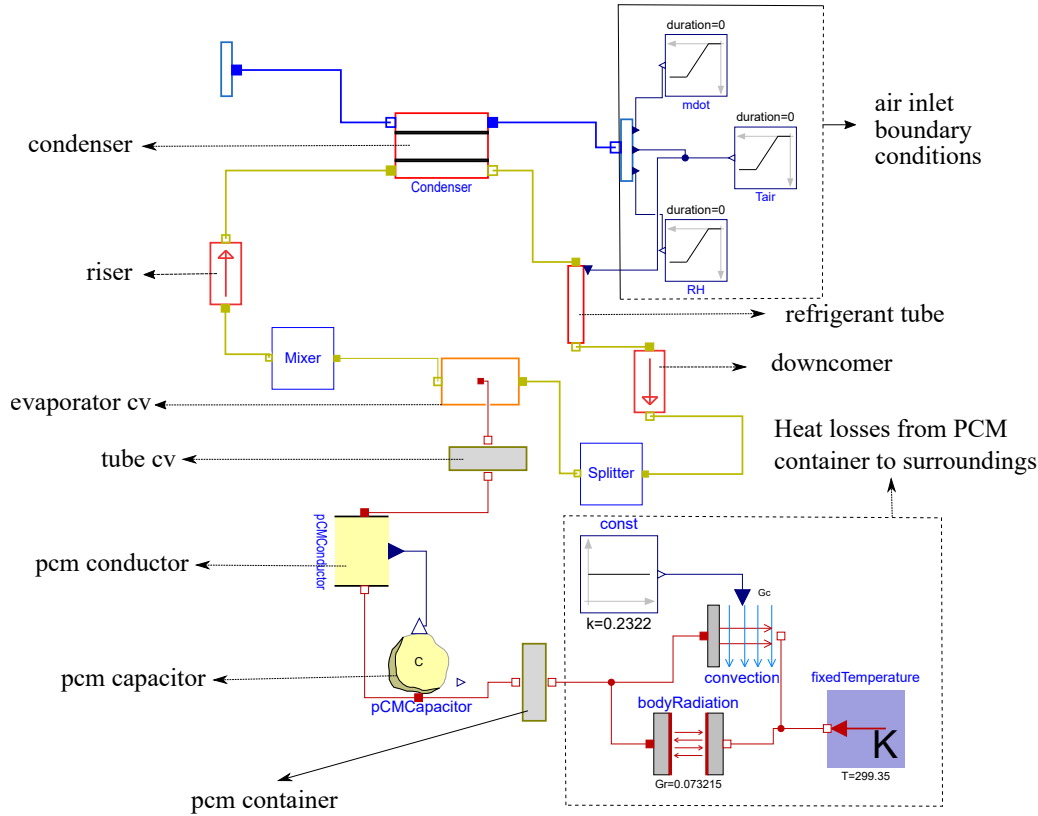


Figure 3. Schematic of System Model for Thermosiphon.

The model used in the current work is a trade-off for accuracy, complexity and usability. The PCM block is taken as a lumped control volume to eliminate the modeling of momentum equation for the molten PCM flow from natural convection. Two components are used to model PCM: `PCMConductor` to model the rate of heat transfer from the PCM and `PCMCapacitor` to model the PCM heat storage.

3.1.1 PCM Capacitor

The PCM-HX is the heat source for the thermosiphon and consequently dictates its dynamics. Very accurate description of its solidification is required. The energy equation applied to PCM control volume gives rise to:

$$m_{pcm} \frac{dh}{dt} = \dot{Q} \quad (1)$$

where, m_{pcm} [kg] is the mass of PCM, h [Jkg⁻¹] is the specific enthalpy, t [s] is the time and \dot{Q} [W] is the rate of heat transfer.

The enthalpy method by Voller (1990) is used to model the energy equation. This method requires an input of enthalpy-temperature function of PCM solidification which is created using data from DSC readings of the PCM. This ensures accurate temperature prediction of the PCM state during solidification. The benefit of the enthalpy method is that it allows calculations on a fixed grid with implicit treatment of the phase change boundary.

`Modelica.Blocks.Sources.CombiTable1D` block is used for input of enthalpy-temperature profile.

The enthalpy-temperature profile is calculated as shown in equation (2)

$$h(T) = \begin{cases} \int_{T_A}^{T_A} c_s dT, & \text{solid} \\ \int_{T_B}^{T_A} c(T) dT, & \text{two phase} \\ h_{fg} + \int_T^{T_B} c_L dT, & \text{liquid} \end{cases} \quad (2)$$

h_{fg} [Jkg⁻¹] is the latent heat of melting, the PCM melts from temperature T_A [K] to T_B [K], c_s [Jkg⁻¹ K⁻¹], $c(T)$ [Jkg⁻¹ K⁻¹] and c_L [Jkg⁻¹ K⁻¹] are specific heat capacities of PCM in the respective phases.

The melt fraction (λ) of PCM is calculated from its enthalpy by the following equation:

$$\lambda = \max(0, \min(1, \frac{h}{h_l})) \quad (3)$$

where h_l [Jkg⁻¹] is the enthalpy at the point where the PCM just turns liquid. The equation is simplified because of the fact that the enthalpy scale is defined as zero for the point where the PCM starts to melt. The melt fraction is made available for the PCM capacitor block through the `RealOutput` interface.

3.1.2 PCM Conductor

The PCM Conductor block captures the variable effects of conduction and natural convection during the solidification of PCM. It calculates heat transfer coefficient as a function of melt fraction.

PCM Conductor block connects the refrigerant control volume of the condenser to the PCM Capacitor block. It extends `Modelica.Thermal.HeatTransfer.Interfaces.Element1D` block and provides for the heat flow, which is calculated using `CombiTable1D` fitted function for heat transfer coefficient as a function of melt fraction. The `RealInput` interface is used to obtain melt fraction input from PCM Capacitor.

Table 1 contains the anchor points given to the `CombiTable` block used as input for the normalized heat transfer coefficient as a function of melt fraction. The constant value used to multiply the normalized function to obtain heat transfer coefficient (HTC) is $116 \text{ W m}^2 \text{ K}^{-1}$. These numbers are obtained by matching the condenser pressure from simulation to the experiment since there are no correlations to capture the behavior in literature. Pal and Joshi (2001) discuss the heat transfer variation in the four regimes captured by Table 1. The initial heat transfer occurs in a conduction dominated regime. Then there is a reduction in heat transfer coefficient with the appearance of small melt layer because the velocity of the liquid PCM due to buoyancy force is small. The melting then progresses to a convection dominated regime where the velocity of liquid PCM increases causing a higher rate of heat transfer. Finally, the magnitude of velocity decreases as the temperature in the molten PCM becomes more uniform with time due to natural convection stirring, leading to decreased buoyancy force for convection.

Table 1. Input table for PCM Conductor block.

Melt Fraction	Normalized HTC
0	1
0.2	0.9
0.4	1
0.7	0.9
1	0.8

3.1.3 PCM-HX Refrigerant Control Volume

The PCM is modeled using a lumped control volume (CV) and accordingly a lumped control volume on the refrigerant side is required. These two CVs are connected using `Modelica.Thermal.HeatTransfer.Interfaces.HeatPort` interface.

The liquid refrigerant from the downcomer reaches the bottom header of the PCM-HX (See Figure 2). Then it absorbs heat from the PCM, vaporizes and rises up into the riser. The flow of refrigerant into and out from the CV, is modeled using

`Modelica.Fluid.Interfaces.FluidPort` interface.

To define the state of refrigerant inside the lumped control volume two properties are required. The average density, ρ_{avg} [kg m^{-3}] for the two phase refrigerant can be obtained as shown below:

$$V_{tot} = V_v + V_l \quad (4)$$

$$m = \rho_v V_v + \rho_l V_l \quad (5)$$

$$\rho_{avg} = \frac{m}{V_{tot}} \quad (6)$$

V [m^3] refers to the volume of the refrigerant, m [kg] its mass. The subscripts v and l refer to vapor and liquid phases while tot stands for total.

The pressure of the refrigerant is the average of the pressure at its inlet and outlet fluid ports. A medium record for the refrigerant is created and these thermodynamic properties are set to determine its state.

```
medium.d = rho_avg;
medium.p = 0.5*(port_a.p+port_b.p);
```

The net pressure drop between the inlet and outlet ports is assumed to equal to the gravitational head offered by the refrigerant column.

The evaporator CV consists of liquid refrigerant with vapor escaping from the top after absorbing heat from the surrounding PCM. If the flow were to reverse, liquid refrigerant will leave out from the inlet port. Thus the stream variable of enthalpy in the fluid connectors are equated to the enthalpies of saturated liquid and vapor.

```
port_a.h_outflow = h_f;
port_b.h_outflow = h_g;
```

The heat flow term of the `HeatPort` is calculated by multiplying heat transfer coefficient by the product of surface area and temperature difference between `HeatPort` temperature and medium temperature.

```
heatPort.Q_flow = htc * A * (heatPort.T -
medium.T);
```

The two phase heat transfer coefficient for the refrigerant inside the helical coils is calculated first by using Schmidt (1967) correlation to obtain single phase liquid only heat transfer coefficient which is then used in Shah Chart correlation (Shah, 1982).

Finally, the mass and energy balance equations are written down and state transformations applied to update the values of pressure and enthalpy of the refrigerant CV. This approach is pretty standard in two phase refrigerant system and is discussed in Tummescheit et al. (2000). The equations for energy, however, involve stream connector variations as described in Franke et al. (2009).

3.2 Condenser

The condenser in the system is a standard air to refrigerant heat exchanger. It is modeled using the heat exchanger developed by Qiao et al. (2015). The model neglects gravitational pressure drops. Thus, it can be visualized as if

it is placed in a horizontal plane as opposed to vertical in the real case. However, the height of the condenser is small compared to the riser and gravitational effects can hence be ignored for model simplicity. The refrigerant side heat transfer coefficients for two phase flow are calculated using Shah (2016) correlation. The airside heat transfer coefficient is calculated using (Wang et al., 2000) correlation.

3.3 Riser

Recall that the riser is that portion of the refrigerant circuit where the refrigerant vapor rises from the evaporator into the condenser. The timescale over which its dynamics evolves is much faster than the heat exchangers. As a result, it is modeled as what is described as Flow Model in literature (Tummescheit et al., 2000). Only the momentum equation is used in the model and the mass and energy storage in the control volume are ignored. The momentum equation for riser contains balances for the pressure force, frictional force and gravitational force as shown in Equation 7 in which L_t [m] is the length of the riser, $\frac{dm}{dt}$ [kg s⁻²] is the rate of change of refrigerant mass flow rate, A [m²] is the cross-section area for refrigerant flow in the tube, p_{in} [Pa] and p_{out} [Pa] are inlet and outlet pressures, f is friction factor, S [m] is the perimeter of the flow section of the tube, ρ [kg m⁻³] is refrigerant density and g [ms⁻²] is the acceleration due to gravity.

$$L_t \frac{dm}{dt} = A(p_{in} - p_{out}) - \frac{1}{2} \frac{m^2}{\rho A^2} f S L_t + A \rho g L_t \quad (7)$$

The friction factor equation incorporates laminar and turbulent flow regimes by merging Hagen-Poiseuille and Blasius equations. Both these equations are taken from Bergman and Incropera (2011).

3.4 Downcomer

The model for downcomer is similar to that of riser except for the momentum equation in which the direction of gravitational effects are reversed. The momentum equation for downcomer is shown in Equation 8.

$$L_t \frac{dm}{dt} = A(p_{in} - p_{out}) - \frac{1}{2} \frac{m^2}{\rho A^2} f S L_t - A \rho g L_t \quad (8)$$

3.5 Heat Losses

The PCM loses heat by natural convection and radiation with the surroundings. The heat loss by these modes are about 15-20% of the heat removed by the thermosiphon. For accurate prediction of solidification time, it is necessary to include these heat losses.

The PCM is contained in a PVC container. A simple Tube model of circular wall with one-dimensional heat conduction and capacitance lumped at arithmetic mean temperature is used. The equations for this model can be found in `Modelica.Fluids.Examples.`

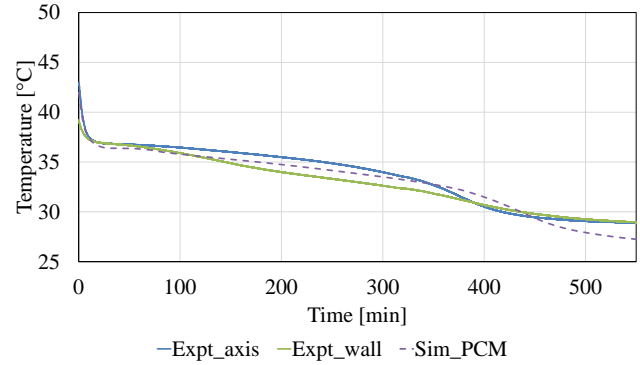


Figure 4. Comparison of PCM temperature prediction with experimental data

`HeatExchanger.BaseClasses.WallConstProps.` The Tube model has two `HeatPort` interfaces, one of which is connected to the PCM Capacitor block. The second `HeatPort` is connected to a `Modelica.Thermal.HeatTransfer.Sources.FixedTemperature` block which has surrounding temperature, via `Modelica.Thermal.HeatTransfer.Components.Convection` and `Modelica.Thermal.HeatTransfer.Components.BodyRadiation` blocks to model the heat losses.

The heat transfer coefficient by convection from the container walls is obtained using Churchill and Chu (1975) correlation for natural convection for vertical plates. Radiation is calculated by taking a value of emissivity $\epsilon = 0.9$ for the material. The container is assumed to be a convex body in a large enclosure. The heat transfer coefficient from the top surface of the container is calculated using Lloyd and Moran (1974) correlation. However, this value is negligible in comparison to the net heat loss and omitted from the simulation.

4 Results and Discussion

Figure 3 shows the thermosiphon model with all the components described in the previous section. The boundary conditions and initial state points are provided using the experimental results from Du et al. (2016).

Figure 4 shows a comparison of PCM temperatures from the experiment with the model. There are two points from the experiment with subscripts axis and wall. The subscript axis refers to temperature probe near the PCM container wall while the subscript axis refers to temperature probe at the axis of the helical coil. The dotted line in the Figure 4 shows the lumped PCM temperature from the model. As can be observed, the overall prediction is good until roughly 430 minutes. The results deviate significantly from this point. This deviation can be attributed to the assumption of the adiabatic riser. In the experiment, there is heat loss from the refrigerant vapor as it passes through the riser leading to its condensation. The heat absorption from PCM drops significantly when the PCM is

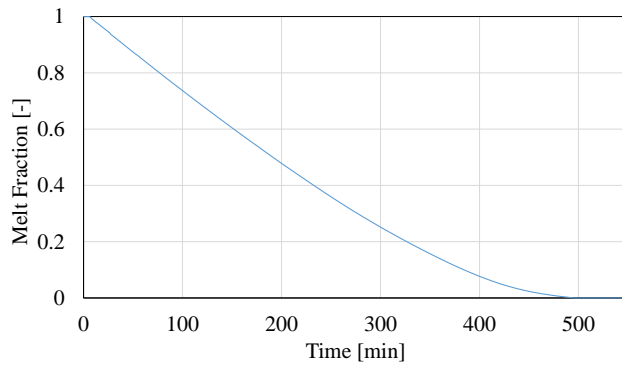


Figure 5. Percentage of PCM Molten with time

solidified. The model predicts a small rate of the mass flow rate at this point, but in reality there is no mass flow rate. The refrigerant vapor rises up but gets condensed and falls back down. This phenomenon is not captured by the model.

The recharge time calculated by the model is 489 minutes when the PCM fully solidifies (See Figure 5). However, 94% percent of PCM is solidified at the 400 minute mark. For a good overall cycle COP for RoCo, the thermosiphon can be operated for only 400 minutes and VCC operation started at this point.

Figure 6 shows the temperatures on the airside of the condenser. The prediction of air outlet temperature is slightly lower in the initial 20 minute interval. This can be attributed to the receiver present in the circuit which is filled with hot liquid refrigerant. For the setup, the receiver is sized in such a way that the downcomer is completely filled with liquid refrigerant. This results in larger thermal mass of refrigerant to be cooled.

5 Conclusions

A fully transient model for two phase closed loop thermosiphon is developed from first principles and used to study the dynamics of a thermosiphon used to recharge the thermal battery of a portable air conditioner. Equations to model various components of the thermosiphon

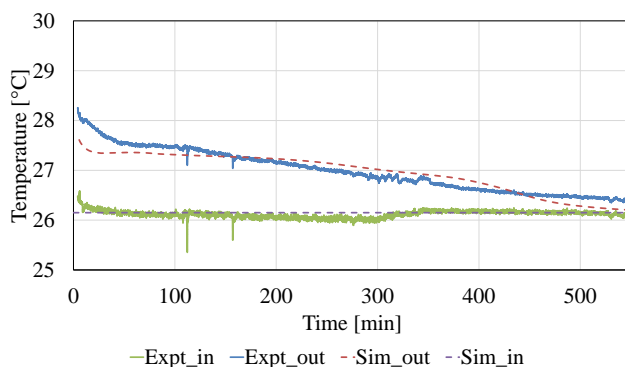


Figure 6. Comparison of air inlet and outlet temperatures at the condenser

are discussed. The heat source of the thermosiphon is finite and the coupled dynamics is successfully predicted by the model. It is observed that solidifying 94% of PCM is better than full solidification for better system COP. The model is expected to be an invaluable tool in designing the future versions of the portable air conditioning device with different requirements.

6 Acknowledgment

This research was supported by the Advanced Research Projects Agency - Energy (ARPA-E) with Award Number DE-AR0000530. We thank the members of Center for Environmental Energy Engineering (CEEE) and team members of the Roving Comforter Project for their support.

References

- K S Benne and K O Homan. Transient Behavior of Thermosiphon-Coupled Sensible Storage with Constant Temperature Heat Addition. *Numerical Heat Transfer, Part A: Applications*, 55(2):101–123, 2009. doi:10.1080/10407780802552062.
- Theodore L Bergman and Frank P Incropera. *Introduction to heat transfer*. John Wiley and Sons, Chichester, New York, 6 edition, 2011. ISBN 978-0470-50196-2.
- Stuart W Churchill and Humbert HS Chu. Correlating equations for laminar and turbulent free convection from a vertical plate. *International journal of heat and mass transfer*, 18(11):1323–1329, 1975. doi:10.1016/0017-9310(75)90243-4.
- R T Dobson and J C Ruppertsberg. Flow and heat transfer in a closed loop thermosiphon. Part I—Theoretical simulation. *J. Energy South. Afr*, 18:32–40, 2007.
- Yilin Du, Jan Muehlbauer, Jiazhen Ling, Vikrant Aute, Yunho Hwang, and Reinhard Radermacher. Rechargeable Personal Air Conditioning Device. In *ASME 2016 10th International Conference on Energy Sustainability collocated with the ASME 2016 Power Conference and the ASME 2016 14th International Conference on Fuel Cell Science, Engineering and Technology*. American Society of Mechanical Engineers, 2016. doi:10.1115/ES2016-59253.
- Alessandro Franco and Sauro Filippeschi. Closed Loop Two-Phase Thermosiphon of Small Dimensions: a Review of the Experimental Results. *Microgravity Science and Technology*, 24(3):165–179, 2011. doi:10.1007/s12217-011-9281-6.
- Rüdiger Franke, Francesco Casella, Martin Otter, Michael Sielemann, Hilding Elmqvist, Sven Erik Mattson, and Hans Olsson. Stream Connectors – An Extension of Modelica for Device-Oriented Modeling of Convective Transport Phenomena. 43:108–121, 2009. doi:10.3384/ecp09430078.
- S I Haider, Yogendra K Joshi, and Wataru Nakayama. A natural circulation model of the closed loop, two-phase thermosiphon for electronics cooling. *Journal of heat transfer*, 124(5):881–890, 2002. doi:10.1115/1.1482404.

- Tyler Hoyt, Edward Arens, and Hui Zhang. Extending air temperature setpoints: Simulated energy savings and design considerations for new and retrofit buildings. *Building and Environment*, 88:89–96, 2015. doi:10.1016/j.buildenv.2014.09.010.
- JR Lloyd and WR Moran. Natural convection adjacent to horizontal surface of various planforms. *Journal of Heat Transfer*, 96(4):443–447, 1974. doi:10.1115/1.3450224.
- Debabrata Pal and Yogendra K Joshi. Melting in a side heated tall enclosure by a uniformly dissipating heat source. *International Journal of Heat and Mass Transfer*, 44(2):375–387, 2001. ISSN 0017-9310. doi:10.1016/S0017-9310(00)00116-2.
- Hongtao Qiao, Vikrant Aute, and Reinhard Radermacher. Transient modeling of a flash tank vapor injection heat pump system—part I: model development. *International journal of refrigeration*, 49:169–182, 2015. doi:10.1016/j.ijrefrig.2014.06.019.
- Eckehard F Schmidt. Wärmeübergang und Druckverlust in rohrschlangen. *Chemie Ingenieur Technik*, 39(13):781–789, 1967. doi:10.1002/cite.330391302.
- M M Shah. Chart correlation for saturated boiling heat transfer: equations and further study. *ASHRAE Trans.:(United States)*, 88(CONF-820112-), 1982.
- Mirza M Shah. Comprehensive correlations for heat transfer during condensation in conventional and mini/micro channels in all orientations. *International journal of refrigeration*, 67:22–41, 2016. doi:10.1016/j.ijrefrig.2016.03.014.
- Atul Sharma, V V Tyagi, C R Chen, and D Buddhi. Review on thermal energy storage with phase change materials and applications. *Renewable and Sustainable Energy Reviews*, 13(2):318–345, 2009. doi:10.1016/j.rser.2007.10.005.
- Hubertus Tummescheit, Jonas Eborn, and Falko Wagner. Development of a Modelica base library for modeling of thermohydraulic systems. In *Modelica Workshop 2000 Proceedings*, pages 41–51, 2000.
- V R Voller. Fast implicit finite-difference method for the analysis of phase change problems. *Numerical Heat Transfer*, 17(2):155–169, 1990. ISSN 1040-7790. doi:10.1080/10407799008961737.
- Chi-Chuan Wang, Kuan-Yu Chi, and Chun-Jung Chang. Heat transfer and friction characteristics of plain fin-and-tube heat exchangers, part II: Correlation. *International Journal of heat and mass transfer*, 43(15):2693–2700, 2000. doi:10.1016/s0017-9310(99)00333-6.
- Belen Zalba, Jose Ma Marin, Luisa F Cabeza, and Harald Mehling. Review on thermal energy storage with phase change: materials, heat transfer analysis and applications. *Applied thermal engineering*, 23(3):251–283, 2003. doi:10.1016/S1359-4311(02)00192-8.

Extended Modelica Model for Heat Transfer of Two-Phase Flows in Pipes Considering Various Flow Patterns

Timm Hoppe¹ Friedrich Gottelt¹ Stefan Wischhusen¹

¹XRG Simulation GmbH, Harburger Schlossstr. 6-12, 21079 Hamburg Germany,
{hoppe,gottelt,wischhusen}@xrg-simulation.de

Abstract

Boiling in vertical and horizontal pipes is a complex process defining transient and static performance of various technical applications. This work presents an extended heat transfer model which takes the complete boiling process into account. Models from the literature for the different boiling regimes are evaluated with respect to accuracy and suitability for system simulation application. A set of sub-models for each of the existing boiling phenomena is implemented and applied to the global boiling model. Special attention is paid to smooth transition between the sub-models and to numerical efficient solutions with respect to the consideration of the boiling crisis. The simulation results show good accordance with literature data.

Keywords: boiling model, heat transfer, two phase flow, pipe flow, evaporation, critical heat flux, boiling crisis, subcooled boiling, saturated boiling, flow pattern, Fluid-Dissipation, ClaRa

1 Introduction

The heat transfer from an evaporator wall to a flowing two-phase fluid is called flow boiling. During flow boiling three different regimes can be identified. The first one is the subcooled boiling regime. The bulk of the fluid is still subcooled but bubbles can already form in the wall layer, are cooled by the surrounding liquid and thus enhance the heat transfer. It is followed by the saturated boiling regime in which the wall is predominantly covered by the liquid phase. Bubbles are formed there, leave the wall and exchange heat and mass with the surrounding liquid at saturation temperature. The critical heat flux marks the beginning of the post critical heat flux regime, where the gas phase becomes the dominating phase in the wall layer. In all of these regimes the slope of the pipe plays an important role. For example, a vertical pipe has a distinct point at which the regime changes from saturated boiling to the post critical heat flux regime. In a horizontal pipe there is a gradually transition, as the top of the pipe may be already covered by steam, while the bottom of the pipe is still cooled by liquid.

During flow boiling of fluid flows high heat fluxes can be transferred at low temperature differences. Flow boiling occurs in many industrial applications, such as thermal power plants, air conditioning or heat exchangers in the

process industry. Exact knowledge about the two-phase heat transfer and pressure loss is important to determine behaviour of these applications. However, it is quite common to reduce the boiling process to the saturated boiling regime in system simulation, although several boiling regimes can be identified which require specific models. This assumption neglects important effects like the critical heat flux at which the heat transfer coefficient drops by magnitudes.

An industrial example where it is important to include this effect are different kinds of thermal power plants like conventional coal fired steam generators, solar steam generators, natural circulation heat recovery steam generators or nuclear steam generators.

In coal fired once-through steam generators the critical boiling state occurs during normal operation at the end of the evaporator. In normal operation the mass flow rate is high enough to sustain a sufficient cooling of the pipes, see (Brinkmeier, 2015). However, in abnormal working conditions, e.g. maldistribution of mass flow between parallel pipes or failure of feed water supply systems the mass flow rate may be significantly lower than during nominal operation. A detailed calculation of the heat transfer is needed to determine the wall temperatures in this abnormal situations.

2 State of the Art of Two-phase Heat Transfer Modelling

2.1 Literature Review

The actual steam quality of the flow is described by the local vapour mass fraction defined by the ratio of vapour mass flow \dot{m}_{vap} to total mass flow \dot{m} :

$$x_{act} = \frac{\dot{m}_{vap}}{\dot{m}} \quad (1)$$

If thermodynamic equilibrium is assumed the steam quality can as well be defined by the local specific enthalpy h , the specific enthalpy at bubble point h' and the specific evaporation enthalpy Δh_V :

$$x_{eq} \equiv x = \frac{h - h'}{\Delta h_V} \quad (2)$$

The thermodynamic equilibrium steam quality is used in the further course of the paper and will be referred to as

x. In the subcooled area it can also be negative. The overall boiling process with the different boiling regimes is displayed schematically for a horizontal pipe in Figure 1. Two different heating situations are shown, one with a high heat flux resulting in a nucleate boiling dominated flow and one with a low heat flux resulting in a convective boiling dominated flow. The circumferential averaged heat transfer coefficient is plotted against the actual steam quality and the steam quality assuming thermodynamic equilibrium.

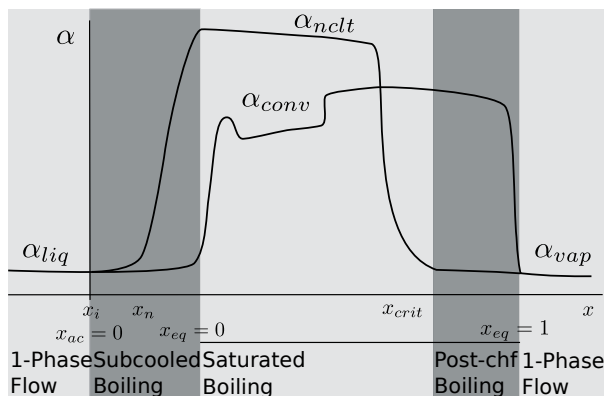


Figure 1. Schematic circumferential averaged heat transfer coefficient for an evaporating flow in a horizontal pipe with a high heat flux (nucleate boiling dominated) and a low heat flux (convective boiling dominated), according to (Steiner, 2002), flow regime descriptions refer to nucleate boiling dominated flow

Subcooled Boiling

Bubble formation starts in the wall layer of the current already at bulk enthalpies below the bubble enthalpy, i.e. $x < 0$. These bubbles deteriorate as they move to the subcooled core of the flow, thus enhancing the heat transfer coefficient. This is known as subcooled boiling, see Figure 1 for a schematic behaviour of the heat transfer coefficient in the subcooled boiling regime of a nucleate boiling dominated flow.

In general the literature on subcooled boiling is sparse compared to other boiling regimes. A comparison of several subcooled boiling models was done by Spindler (K. Spindler, 1990), the model proposed in the VDI heat atlas (Schröder, 2002) shows the least error in prediction of the measurements. At the end of the subcooled boiling regime at a steam quality $x = 0$ the VDI model is by construction equivalent to the saturated boiling heat transfer model for a vertical pipe from the VDI heat atlas. Therefore, a smooth transition to the following boiling regime is ensured.

Saturated Boiling

In the saturated boiling regime two different boiling modes can be observed. This is convective boiling on the one hand and nucleate boiling on the other hand. Convective boiling describes the convective process between the wall and the liquid phase, whereas nucleate boiling describes the heat transfer induced by formation, growth and

departure of the bubbles. In horizontal pipes a stratification of the fluid can occur, depending on the present flow pattern. As the upper pipe wall is partly dry, i.e. only covered by the gaseous phase, the circumferential heat transfer coefficient is lower compared to flow patterns which cover the wall completely with the liquid phase.

All relevant saturated boiling models are a function of the convective boiling heat transfer coefficient, basically calculated with convective one phase heat transfer correlations and the nucleate boiling heat transfer coefficient, basically calculated with pool boiling heat transfer correlations. A simple addition of the coefficients is done by the Chen model (Chen, 1966). He introduced a boiling suppression factor and a two phase multiplier to fit pool boiling heat transfer correlation and the one phase convective heat transfer correlation to his flow boiling data. Shah (Shah, 1982) proposed a model which is not superimposing the convective and the nucleate heat transfer coefficients but chooses the larger of the two. Gungor and Winterton (Winterton, 1986) developed a new form of the Chen model, basing on a larger database. They later also proposed a simpler version of the model (Gungor and Winterton, 1987). The VDI heat atlas (Steiner, 2002) proposes an asymptotic model which incorporates natural limitations of the flow boiling coefficients, instead of suppression and two phase factors as used by the other models. It is also referred to as the Steiner-Taborek model as it bases on a publication of the two authors (Steiner and Taborek, 1992). The model is recommended by ASHRAE (Owen, 2005) and Thome (Thome, 2006a).

Critical Heat Flux

In the further course of the evaporation process the critical boiling state x_{crit} , also known as critical heat flux, is reached and the gas phase becomes the dominating phase at the wall. In a vertical pipe it is a distinct point at which the heat transfer coefficient drops suddenly by magnitudes. In a horizontal pipe the critical boiling state at the top of the pipe can already be reached while the bottom of the pipe is still covered with liquid. The circumferential averaged heat transfer coefficient in such a situation can be seen in Figure 1.

Several approaches for prediction of the critical heat flux can be found in literature. This are on the one hand the *local hypothesis* correlations which take the local conditions at the point of boiling crisis into account. Then there are the *global hypothesis* correlations which consider the conditions at the pipe inlet. A third method are the *look-up tables*.

The VDI heat atlas (Auracher et al., 2002) gives the Groeneveld tables (Groeneveld, 2007) as reference for look-up tables and the Katto-Ohno model (Katto Y., 1984) as reference for the global hypothesis correlations. The Katto-Ohno correlation is also proposed by Thome (Thome, 2006a). For medium to high pressures and mass flow rates the VDI heat atlas proposes the local hypothesis correlations of Doroshchuk and Kon'kov. They are con-

sidered as superior to the other two methods. Furthermore, they differentiate between dry out and departure from nucleate boiling.

Post Critical Heat Flux Boiling

At least two boiling regimes can be recognized after the critical boiling state. On the one hand there is the dry out of the surface. The wall is not necessarily dry, as the remaining liquid is entrained as droplets in the vapour flow, also referred to as mist flow. Droplets may then hit the wall and wet it temporarily. The second regime is called film boiling or departure from nucleate boiling. It is characterized by the formation of a gas film, the corresponding flow pattern is called inverted annular flow. The heat transfer coefficient is significantly lower than in mist flow. For each regime different heat transfer models are required. If the wall temperature is very high compared to the gas temperature radiation will also play an important role in the heat transfer.

For dry out there are two different kinds of correlations. The simpler correlations assume thermodynamic equilibrium between gas and liquid phase. The model of Groeneveld (Groeneveld, 1973) is proposed by the VDI heat atlas (Katsaounis, 2002) and by Thome (Thome, 2006a).

The more complicated models account for thermodynamic non-equilibrium effects by an apparent superheated gas temperature. The Köhler model (Köhler, 1983) proposed by the VDI heat atlas is to mention here. A model which also includes radiation was published by Ganic and Rohsenow (Ganic and Rohsenow, 1977).

Summary

The VDI heat atlas proposes correlations for all different boiling mechanisms, thus the complete boiling process is covered by the VDI proposal. In some extent the models refer to each other. For example one model merges by construction into the model for the following boiling regime. This smooth transition between the different models is important in system simulation. It should be continuous, continuous differentiable, with low gradients and without hysteresis. The proposed models are also considered among the most reliable ones by other authors. Therefore, the models of the VDI heat atlas are implemented. Where it is possible the references between the models are used to smooth or simplify the transitions.

2.2 Library Review

It is not known to the authors that an existing Modelica library incorporates a heat transfer model for the complete boiling process. The *AirConditioning* library and the *TIL* library, which are commercially used for simulation of automotive refrigeration cycles, cover only the saturated boiling regime. The *AirConditioning* includes implementations of the simple Gungor-Winterton and the Chen correlation. The *TIL* library includes the Chen correlation and the convective boiling correlation of the Steiner-Taborek model. In the freely available *ThermoCycle* library the Shah and the simple Gungor-Winterton correlation are im-

plemented.

In power plant modelling the critical heat flux plays an important role. In the freely available *ClaRa* library only the saturated boiling regime is covered. Also other libraries, the freely available *ThermoSysPro*, the freely available *ThermoPower* and the commercial *ThermalPower*, provide no correlations for determination of the boiling crisis.

3 Implementation

The different boiling regime correlations are implemented in the *FluidDissipation* which is an open source library and freely available, see (XRG Simulation). For implementation the functional approach described in (Vahlenkamp and Wischhusen, 2009) is used. The main aspects of the approach are:

- Independence of thermo-hydraulic framework
- Use of function calls
- Inputs are delivered by records

In contrast to the models already implemented in the *FluidDissipation* each boiling regime is implemented in a separate model. Thus, the overall heat transfer models are built outside of the *FluidDissipation*. The advantage of this procedure is that models can be developed which are tailored to the corresponding problem. In the following the implementation of the separate models is shortly described focussing on main declarative equations and differences of the implementation to the VDI heat atlas models. For detailed description of the models see the *FluidDissipation* documentation.

3.1 Subcooled Boiling Heat Transfer

From the VDI heat atlas (Schröder, 2002) the models for determination of the position, in terms of a steam quality, of initial bubble formation x_i and of net vapour forming x_n are implemented. Both values are by definition of the subcooled boiling regime always negative. For $x_i < x < x_n$ the proposed model for calculation of the heat transfer coefficient from VDI heat atlas is used. For $x_n < x < 0$ the VDI heat atlas proposes a superposition of the nucleate boiling heat transfer coefficient and the subcooled convective heat transfer coefficient. However, this model incorporates the wall temperature in the heat transfer coefficient calculation, which would introduce additional iterations for the solver. Furthermore, the transition to the saturated boiling heat transfer coefficient is only smooth if the influence of the flow pattern on the saturated boiling heat transfer coefficient is neglected. To simplify this model, thus making it numerical more stable, a simple smoothing from the subcooled convective heat transfer coefficient $\alpha_{sc}(x = x_n)$ to the saturated heat transfer coefficient $\alpha_{sat}(x = 0)$ is done. The Stepsmoother from the *FluidDissipation* is used which makes use of the smooth transition

of the following equation

$$y_{reg} = (1 + \tanh(\tan(x))) / 2 \quad (3)$$

The output y_{reg} is between 0 and 1 for $x = [-\pi/2, \pi/2]$.

3.2 Saturated Boiling Heat Transfer

The Steiner-Taborek model from the VDI heat atlas (Steiner, 2002) is implemented. The following equation results in an asymptotic combination of the convective boiling α_{conv} and the nucleate boiling heat transfer coefficient α_{nclt}

$$\alpha_{sat} = \sqrt[3]{\alpha_{conv}^3 + \alpha_{nclt}^3} \quad (4)$$

Convective Boiling

The VDI heat atlas distinguishes between horizontal and vertical pipes, in consequence two different correlations are provided. Both depend on the one phase heat transfer coefficients for liquid and steam α_{liq} and α_{vap} , the density ratio ρ'/ρ'' and the steam quality x . For horizontal pipes the VDI correlation includes a correction for stratified flow patterns. The information of the present flow pattern and the angle of the unwetted circumference of the pipe is needed. The correction weights the one phase liquid and steam heat transfer coefficients according to the unwetted angle. In the implementation of these functions a smoothing with the stepsmoother from the *FluidDissipation* is applied for the transition between different flow patterns.

Nucleate Boiling

The used correlation from the VDI heat atlas is a function of heat flux \dot{q} , pressure p , pipe diameter d and surface roughness W . For horizontal pipes the correlation shows an additional dependency on the mass flow rate \dot{m} and on the steam quality x . Furthermore, a correction includes the dependency of the present flow pattern. The function expects the present flow pattern of the flow as an input and aligns a correction factor to each flow pattern. These correction factors depend on the thickness and conductivity of the wall. Also in this implementation the transition between flow patterns is smoothed.

3.3 Boiling Crisis

The position of the critical boiling state in terms of a critical steam quality x_{crit} is determined by the correlations which are explained in this subchapter. The critical steam quality marks the end of the saturated boiling regime and the begin of the post critical heat flux regime and therefore determines in an overall boiling model at which point the heat transfer coefficient calculation has to change from the saturated boiling correlation to the post critical heat flux correlation.

The local thesis correlations of the VDI heat atlas (Auracher et al., 2002) of Konkov and Doroshchuk for water are implemented. They depend on the local mass flow

rate, the local pressure, diameter of the pipe and the local heat flow rate.

The Konkov correlation predicts the critical steam quality for dry out, the Doroshchuk correlation the critical steam quality for departure from nucleate boiling. A logic chooses the smaller of the two. The used correlation determines whether dry out or departure from nucleate boiling is present. Thus, it can be decided which post critical heat flux correlation has to be used.

For horizontal pipes gravitational effects have to be considered. With the modified Froude number Fr these effects can be described

$$Fr = \frac{x_{crit}\dot{m}}{\sqrt{\rho''}\sqrt{9.81d(\rho' - \rho'')\cos(\theta)}} \quad (5)$$

The angle θ is the inclination of the pipe. For Froude numbers < 10 the stratification of the flow induces an occurrence of the boiling crisis at the upper side of the pipe $x_{crit,up}$ at much lower steam qualities than at the bottom side of the pipe $x_{crit,low}$. This difference Δx_{crit} can be determined with the modified Froude number

$$\Delta x_{crit} = x_{crit,low} - x_{crit,up} = \frac{16}{(2 + Fr)^2} \quad (6)$$

With the difference Δx_{crit} the transition zone in horizontal pipes from the saturated boiling regime to the post critical heat flux regime can be determined.

3.4 Post Dry-Out Heat Transfer

Two different kinds of models exist for the post dry out heat transfer. One type assumes thermodynamic equilibrium between the liquid and the vapour phase, i.e. the temperatures are equal. The other type calculates a superheated gas temperature, which determines the temperature difference for the heat transfer.

Thermodynamic Equilibrium

The proposed model from the VDI heat atlas (Katsaounis, 2002) is implemented. According to the heat atlas the scope of the model is only for large mass fluxes $\dot{m} > 2000 \text{ kg}/(\text{m}^2\text{s})$ and high pressures with $\rho'/\rho'' \leq 6$. However, the model bases on data of a much wider range, according to (Thome, 2006b).

Thermodynamic Non-Equilibrium

The proposed model for thermodynamic non-equilibrium from the VDI heat atlas (Katsaounis, 2002) is implemented. The assumption of the vapour liquid equilibrium is not valid in this boiling regime. The heat transferred from the wall to the fluid is not used completely for vaporising the liquid but also for superheating the gas phase. Thus, the temperature difference for heat flow rate calculation is calculated with the temperature of the superheated gas and the wall temperature within the scope of the model, i.e.:

$$\dot{Q} = \alpha_{chf} A (T_w - T_g) \quad (7)$$

For calculation of the gas temperature the model provides an empirical correlation. It is no result of energy balancing.

The steam quality x_{lim} gives the end of validity of the model in terms of steam quality and is an output of the model. For $x > x_{lim}$ the original source of the model (Köhler, 1983) suggests to keep the wall temperatures constant. As the object-oriented modelling approach does not allow a direct manipulation of the wall temperature calculation, this behaviour is approximated implicitly by keeping the superheated gas temperature constant. This temperature is used for calculation of the heat flux until the equilibrium temperature of the fluid is greater.

3.5 Flow Pattern Map

The calculation bases on the flow pattern map model described in the VDI heat atlas (Steiner, 2002). To determine the flow pattern the angle of the unwetted circumference of the pipe φ is needed. The VDI heat atlas model uses an iteration process to calculate the unwetted angle. To reduce iterations and make the model suitable for system simulations some modifications have been made. The unwetted angle is calculated directly with the approximation suggested by Biberg (Biberg, 1999).

The output of the function is the unwetted angle φ and a real variable *flowPattern*. To each number a flow pattern is aligned to, see Figure 2. The variable *flowPattern* and the unwetted angle are used by the saturated boiling heat transfer model, see section 3.2. They identify with the value of the variable the present flow pattern and correct the heat transfer coefficient, accordingly. The transition between two flow patterns is smoothed using the *FluidDissipation Stepsmoothen* see, equation 3.

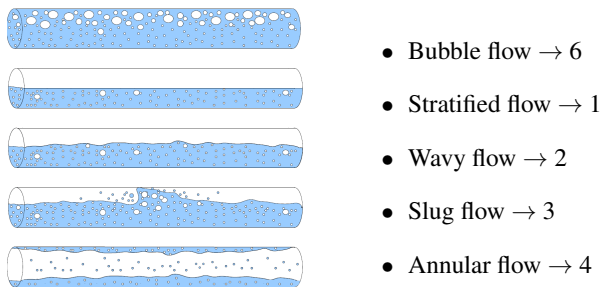


Figure 2. Flow patterns for two phase flow in horizontal straight pipes.

4 Application

In this section an exemplary application of the functions is described. The overall boiling process including flow pattern effects is implemented in the model. The combination of the subfunctions is done in a replaceable model applying the *FluidDissipation* functions. This makes the introduction of additional states possible. These states are necessary due to numerical reasons. The thermo-hydraulic

framework of the *ClaRa* library, (The ClaRa development team) and (Brunnemann et al., 2012), is used. The application is done for the three conservation equation pipe model. The main features of the pipe model are:

- homogeneous single phase, i.e. thermodynamic equilibrium between the phases and same velocities for gas and vapour phase
- one dimensional flow direction
- dynamic energy and mass balances, static momentum balance
- balance equation spatially discretised in flow direction

4.1 Transition of subfunctions

Main task of the application model is to ensure a smooth transition between the different heat transfer modes. The smoothing function, defined by equation 3, is used to ensure a continuous and numerical stable transition. In Figure 3 an example for the transition from saturated boiling to post critical heat flux boiling is given. In the example the transition zone is defined by

$$tz = \Delta x_{crit} \quad (8)$$

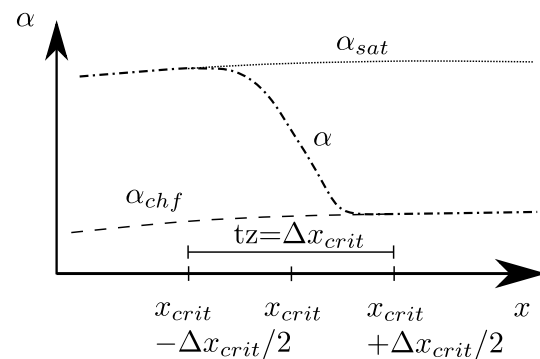


Figure 3. Schematic picture for transition from saturated to post critical heat flux boiling

In the following the used submodels and the switching between the submodels is described in detail. Figure 4 gives an overview of the used models and the selection sequence which is run by the model for determining the heat transfer mode. The corresponding transition zone is displayed in the figure as well. The chosen values of the transition zone are a trade-off between accuracy and numerical performance.

1. **Superheated one-phase flow:** The one-phase heat transfer model (Dittus-Bölder) from the *FluidDissipation* is used. The transition zone depends on the choice of the post critical heat flux heat transfer model. For the equilibrium model the transition zone is defined from $x = 0.9$ to $x = 1$.

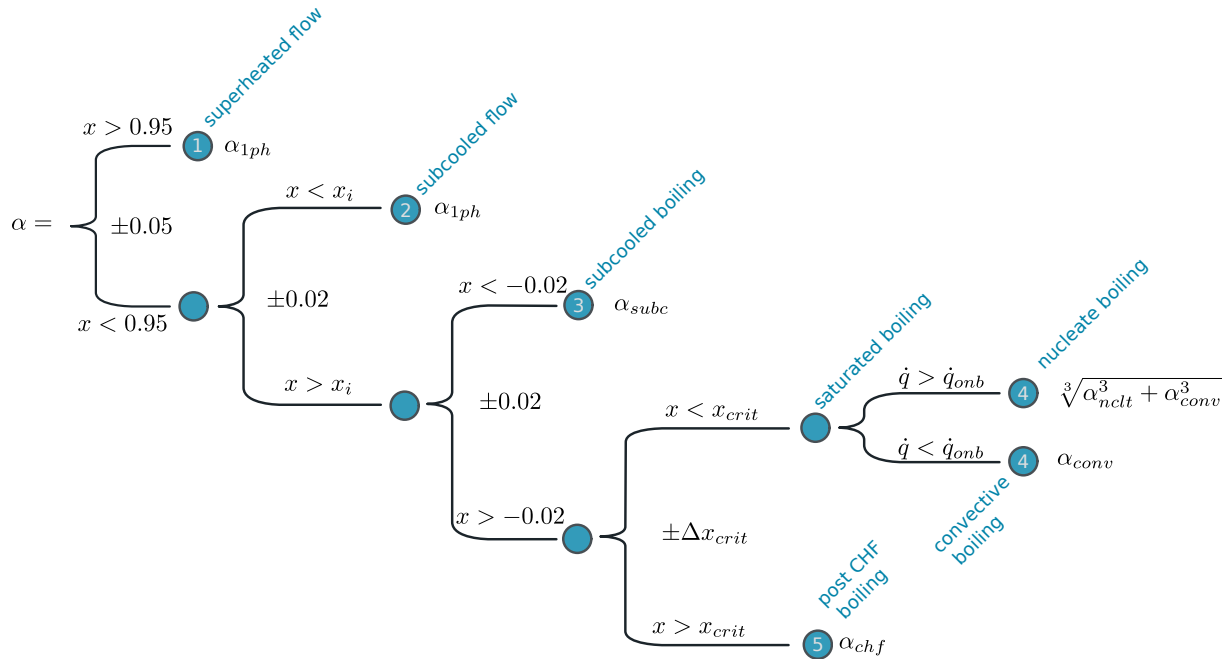


Figure 4. Transition between different Boiling Regimes

2. **Subcooled one-phase flow:** The one-phase heat transfer model (Dittus-Bölder) from the *FluidDissipation* is used. The point of initial bubble formation x_i is calculated by the subcooled boiling model from section 3.1. The transition zone is defined from $x_i - 0.02$ to $x_i + 0.02$.
3. **Subcooled boiling:** The model for subcooled boiling described in the previous section is used. At $x = 0$ the model is merged by construction of the equations into the saturated boiling model described in the previous section.
4. **Saturated boiling:** The model for saturated boiling described in the previous section is used. If the minimum heat flux for onset of nucleate boiling is exceeded the saturated boiling heat transfer coefficient is calculated according to equation 4 otherwise only convective boiling is present. The critical steam quality x_{crit} is calculated by the boiling crisis model. The transition area in terms of a steam quality difference is provided for a horizontally oriented tube by the boiling crisis model. For vertically oriented tubes a fixed value of $\Delta x_{crit} = 0.05$ is set. Thus, the transition area is defined by $tz = \pm \Delta x_{crit}$.
5. **Post-CHF boiling:** The models described in the previous section are used. It can be decided whether the thermodynamic equilibrium or the non-equilibrium model is used.

4.2 Calculation of critical heat flux

The position of the critical heat flux is calculated using the functions of Kon'kov and Doroshchuk (Auracher et al.,

2002) which assume a local hypothesis for the critical heat flux. However, the position is calculated for the total pipe and not for each cell. Thus, situations are avoided in which the model calculates several boiling crises at a time in different cells. Otherwise inconsistent results could be obtained with the previously described selection sequence of the model. As inputs to the critical heat flux model the pressure and mass flow rate of the last cell are used. The heat flow rate is averaged over the total pipe. Thus, a strong numerical coupling between the wall temperatures, heat flows and the heat transfer coefficients of all cells is introduced. To uncouple these variables and help the simulator to break up the resulting non-linear systems of equations a stabilizer state for the heat flow rate \dot{Q}_- is introduced with the following equation:

$$\frac{d\dot{Q}_-}{dt} = \frac{\dot{Q} - \dot{Q}_-}{\tau} \quad (9)$$

During stationary conditions both heat fluxes are equal, thus no new information is included, the stationary results do not change. During transient conditions the additional state lags behind the real heat flux \dot{Q} with the time constant $\tau = 0.1s$.

In consonance with the calculation of the critical heat flux position also the point of initial bubble formation x_i and the point of net vapour generation x_n are calculated for the total pipe and not for each cell. The fluid properties of the first cell are handed over to the two models. The average stabilizer state heat flow rate of equation 9 is used as a heat flow rate in the models.

5 Verification and Discussion of first Results

5.1 Verification of overall heat transfer model

For verification of the total model which is described in section 4 two simulations of evaporation of water in a horizontal pipe are conducted. A screenshot of the used model built with the *ClaRa* library is shown in Figure 5. The re-

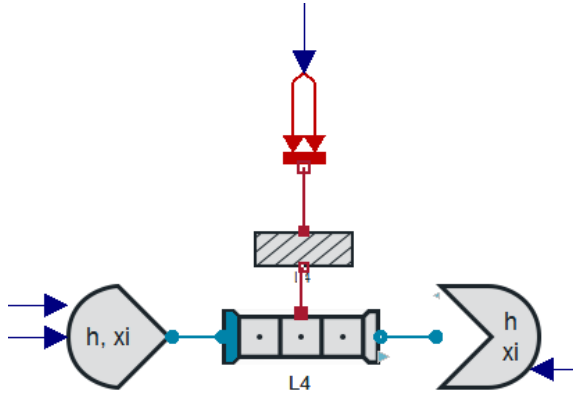


Figure 5. Screenshot of test model

sults should show the principle behaviour of the heat transfer coefficient as can be seen in Figure 1. The boundary conditions and geometric parameters are shown in Table 1, column "Sim. 1". The first values in column one refer to the extreme case of a convective boiling dominated flow, the second values to the extreme case of a nucleate boiling dominated flow. Furthermore, an independency of the flow pattern is assumed, which occurs in pipe walls with a good conductivity (Steiner, 2002). A circumferential uniform heating is assumed. The pipe is discretised with 60 control volumes. The pipe length is chosen such that the fluid is evaporated completely in both cases.

The simulation results are shown in Figure 6. The cir-

Table 1. Boundary conditions and geometric parameters of verification simulation

Parameter	Sim. 1	Sim. 2	Unit
Mass flow	500/400	500	kg/(m ² s)
Heat flow	20/750	600	kW/(m ²)
Outlet pressure	50	50	bar
Inlet spec. enthalpy	550	800	kJ/kg
Hyd. diameter	0.032	0.032	m
Pipe length	600/15	15	m
Wall conductivity	high	low	-

cumferential averaged heat transfer coefficient is plotted against the steam quality. The convective boiling dominated case shows a behaviour which is to be expected from Figure 1. The heat transfer coefficient rises with a steam quality of zero, thus no subcooled boiling is present. It rises further with rising steam quality until it peaks at a

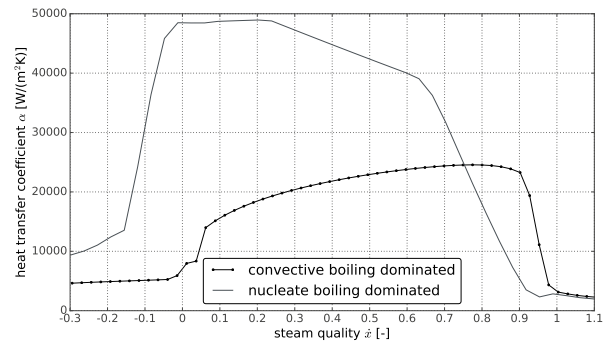


Figure 6. Simulation results of evaporation of water in a horizontal pipe

steam quality of $x = 0.8$. From that point on the heat transfer coefficient drops until the one phase heat transfer at a steam quality of 1 is reached, thus no post critical heat flux boiling is present. The nucleate boiling dominated case shows a distinct subcooled boiling regime. A plateau of the heat transfer coefficient follows at which a limit of the effect of mass flow is reached, according to the nucleate boiling model from the VDI (Steiner, 2002). After that plateau the nucleate boiling heat transfer coefficient drops as one could expect from the schematic Figure 1. At a steam quality of $x = 0.65$ the critical heat flux at the upper side of the pipe is reached. At $x = 0.9$ the critical heat flux is reached also at the bottom side, a post critical heat flux regime follows.

In summary it can be said that the results for both extreme conditions are in very good consonance with the schematic figure. The model predicts the occurrence of the different heat transfer regimes correctly. Also the transition between the different subfunctions is handled in a plausible way.

5.2 Comparison with a simple saturated boiling heat transfer model

A simulation with the boundary conditions from Table 1, column "Sim. 2", is conducted with the overall heat transfer model described in section 4 and the Gungor-Winterton 1986 (Winterton, 1986) heat transfer model which is widely used in system simulation. The boundary conditions are chosen such that they lie in between the extreme conditions of a nucleate and a convective dominated flow. Furthermore, a pipe with a low thermal conductivity of the wall is used.

The results are shown in Figure 7. As an interpretation of the results a schematic pipe with the present flow pattern is drawn below the plot. The flow pattern is calculated from the functions described in section 3.5. The overall heat transfer model predicts a distinct subcooled boiling regime. This boiling regime is neglected in the Gungor-Winterton model and the one phase heat transfer coefficient is used instead. Thus, the heat transfer coefficient of the overall model is over a wide range multiple times larger than the one of the simple Gungor-Winterton

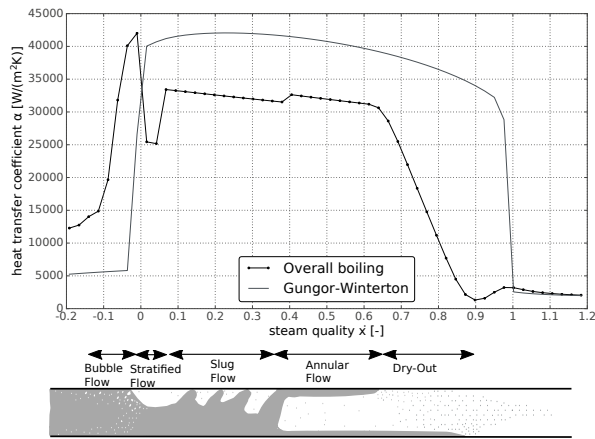


Figure 7. Simulation results of evaporation of water in a horizontal pipe

model.

At steam qualities slightly below zero the heat transfer coefficient of the overall model peaks before it drops again at steam qualities larger than zero. This is due to the fact that the subcooled boiling model does not depend on the flow pattern. Whereas with begin of saturated boiling the heat transfer coefficient shows a high dependency on the flow pattern. At low steam qualities a stratified flow is present. As the upper surface of the pipe is not wetted due to stratification, the circumferential averaged heat transfer coefficient is significantly lower than the Gungor-Winterton model. It predicts a up to 60% larger heat transfer coefficient, as it has no correction due to stratification effects of the flow.

In the further course of the evaporation process the overall boiling model predicts two more different flow patterns. Both induce only a partial wetting of the upper pipe wall, thus the heat transfer coefficient is reduced compared to the simple Gungor-Winterton model.

In the overall heat transfer model the saturated boiling regime ends at a steam quality of $x = 0.65$. The dry out begins at the upper side of the pipe. At the end of the dry out process at a steam quality of $x = 0.9$ the heat transfer coefficient is approximately 10 times smaller than predicted by the simple Gungor-Winterton model.

The results show that the neglect of effects can lead to over- or underestimation of the heat transfer coefficient by magnitudes. For applications, in which crucial variables depend strongly on the heat transfer coefficient, simple models fail to produce reliable results. For example, this may be the case for wall temperatures in the post dry out regime. In that case it is important to include more complex heat transfer models which cover the overall boiling process.

5.3 Validation with measurements

As a validation case the Becker experiments (Abel-Larsen et al., 1985) are presented. They are a series of steady state critical heat flux experiments, all of them with a dry-out

heat crisis. The test section is a vertical pipe, electrically heated with a length of 7 m and a diameter of 0.0149 m.

In Figure 8 the results of a simulation with the overall model using the thermodynamic equilibrium model for the post critical heat flux heat transfer are shown. The

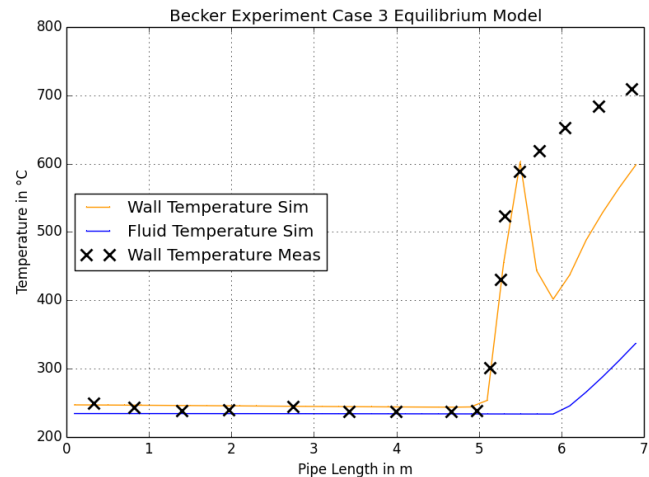


Figure 8. Becker Experiment Case 3 Equilibrium Model

wall temperatures before the heat crisis and the location of the heat crisis are predicted correctly. The predicted wall temperatures after the heat crisis until a length of approximately 5.5 m match also the measurement. Beyond that point the post critical heat flux heat transfer model is not valid, the overall boiling model changes to the one-phase heat transfer correlation. This change leads to an underestimation of the wall temperature, as the gas phase is superheated in the post critical heat flux area. This underestimation has two main reasons. On the one hand the definite temperature is underestimated as the temperature difference for calculation of the heat flux is formed with the equilibrium temperature. On the other hand, the heat transfer coefficient is calculated with fluid properties at the equilibrium temperature, the fluid properties at the superheated gas temperature should be used instead.

In Figure 9 the test case is simulated with the overall model using the thermodynamic non-equilibrium model. The effect of the superheating of the wall near gas phase is included in the non-equilibrium model. Thus, it matches the measurement much better than the simple equilibrium model. At a length of 6 m the model keeps the wall temperatures constant. This assumption fits also better to the measurement than the use of the one phase heat transfer coefficient as it is done by the simple thermodynamic equilibrium model.

The thermodynamic non-equilibrium model is able to predict the wall temperatures more accurate than the simple equilibrium model. Especially, the transition to the purely convective one phase heat transfer regime fits much better to the experiment data. However, this advantage is dearly bought by a loss of numerical stability and efficiency. This is due to the introduction of a fictive gas

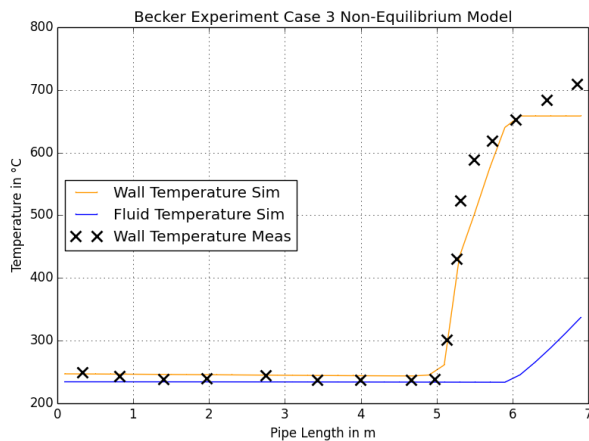


Figure 9. Becker Experiment Case 3 Non-Equilibrium Model

temperature and the indirect manipulation of the wall temperatures. These means are introduced to overcome the drawback of an equilibrium temperature for gas and liquid phase which is used for calculation of the heat balance in the three equation pipe model. For a more accurate modelling of the effects in the post critical heat flux regime a separate energy balancing of the liquid and the gas phase is necessary. This would require a thermo-hydraulic framework which includes volume models for separate balancing of the phases, the so called five or six conservation equation models, see (Hänninen and Ylijoki, 1992).

6 Summary

In this paper an implementation of an extended heat transfer model for two phase flow in pipes is presented.

The various correlations of the different boiling regimes are implemented in separate subfunctions. The functional based approach of the *FluidDissipation* (XRG Simulation) is used. It enables users to create models which are tailored to their problem.

An exemplary application within the thermo-hydraulic framework of the *ClaRa* library is given. This model is used to verify the implementation. A comparison with a widely used saturated boiling model is conducted. The predicted heat transfer coefficients of the two models differ by magnitudes. These results show the importance of complex heat transfer models in system simulation. These models are relevant in applications, in which crucial variables depend strongly on the heat transfer coefficient, e.g. a safety analysis concerning the pipe wall temperatures.

For validation of the extended heat transfer experiment data from the literature are used. The simulations match the measurement data well. Especially, the position of the critical boiling state is predicted correctly. For more accurate results concerning the post critical heat flux regime a separate balancing of the phases is necessary, which is a feature of the six conservation equations models.

References

- H. Abel-Larsen, A. Olsen, J. Miettinen, T. Siikonen, J. Rasmussen, A. Sjöberg, and K. Becker. Heat transfer correlations in nuclear reactor safety calculations. Technical report, Nordic liaison committee for atomic energy, 1985.
- H. Auracher, G. Drescher, D. Hein, O. Herbst, A. Katsaounis, V. Kefer, and W. Köhler. *VDI Heat Atlas*, chapter Hbc - Kritische Siedezustände. 9th edition, 2002.
- D. Biberg. An explicit approximation for the wetted angle in two-phase stratified pipe flow. *The Canadian Journal of Chemical Engineering*, 1999.
- N.O.W.W. Brinkmeier. *Flexibilisierung von Kraftwerken*. PhD thesis, Technische Universität Braunschweig, 2015.
- J. Brunnemann, F. Gottelt, K. Wellner, A. Renz, A. Thüring, V. Roeder, C. Hasenbein, C. Schulze, G. Schmitz, and J. Eiden. Status of ClaRaCCS: Modelling and Simulation of Coal-Fired Power Plants with CO₂ Capture. *Proceedings of the 9th International Modelica Conference, Munich, Germany*, pages 609 – 618, 2012.
- J. C. Chen. Correlation for boiling heat transfer to saturated fluids in convective flow. *Industrial and Engineering Chemistry Process Design and Development*, 5:322–329, 1966.
- E.N. Ganic and W.M. Rohsenow. Dispersed flow heat transfer. *International Journal of Heat and Mass Transfer*, 20:855–866, 1977.
- D.C. Groeneveld. Post-dryout heat transfer at reactor working conditions. In *Proceedings of the National Topical Meeting on Water Reactor Safety*. Atomic Energy of Canada Limited, 1973.
- D.C. Groeneveld. The 2006 CHF look-up table. *Nuclear Engineering and Design*, 237:1909–1922, 2007.
- K.E. Gungor and R.H.S. Winterton. Simplified general correlation for saturated flow boiling and comparisons of correlations with data. *Chemical Engineering Research and Design*, 1987.
- M. Hänninen and J. Ylijoki. The one-dimensional separate two-phase flow model of apros. Technical report, Technical Research Centre of Finland, 1992.
- E. Hahne K. Spindler, N. Shen. Vergleich von Korrelationen zum Wärmeübergang beim unterkühlten Sieden. *Wärme- und Stoffübertragung*, 1990.
- A. Katsaounis. *VDI Heat Atlas*, chapter Hbd -Wärmeübergang nach der Siedekrise. 9th edition, 2002.
- Ohno H. Katto Y. An improved version of the generalized correlation of critical heat flux for the forced convective boiling in uniformly heated vertical tubes. *Int. Journal for Heat Mass Transfer*, 27, 1984.
- W. Köhler. *Einfluß des Benetzungszustandes der Heizfläche auf Wärmeübergang und Druckverlust in einem Verdampferrohr*. PhD thesis, 1983.

- M.S. Owen, editor. *ASHRAE Handbook - Fundamentals*, chapter 5 - Two Phase Flow. ASHRAE, 2005.
- J.J. Schröder. *VDI Heat Atlas*, chapter Hba - Strömungssieden unterkühlter Flüssigkeiten. 9th edition, 2002.
- M. M. Shah. Chart Correlation for Saturated Boiling Heat Transfer: Equations and Further Study. *ASHRAE Transaction* 1982, 88, 1982.
- D. Steiner. *VDI Heat Atlas*, chapter Hbb - Strömungssieden gesättigter Flüssigkeiten. 9th edition, 2002.
- D. Steiner and J. Taborek. Flow Boiling Heat Transfer in Vertical Tubes Correlated by an Asymptotic Model. *Heat Transfer Engineering*, 13(2):43–69, 1992.
- The ClaRa development team. ClaRa - Simulation of Clausius-Rankine cycles. URL www.claralib.com. fetched Dec, 15th 2016.
- J. R. Thome, editor. *Engineering Data Book III*, chapter 10 - Boiling Heat Transfer inside Plain Tubes. Wolverine Tube Inc., 2006a.
- J. R. Thome, editor. *Engineering Data Book III*, chapter 18 - Post Dry-Out Heat Transfer. Wolverine Tube Inc., 2006b.
- T. Vahlenkamp and S. Wischhusen. FluidDissipation for Applications - A Library for Modelling of Heat Transfer and Pressure Loss in Energy Systems. In *Proceedings 7th Modelica Conference, Como, Italy*, September 2009.
- K.E. Gungor R.H.S Winterton. A general correlation for flow boiling in tubes and annuli. *Int. J. Heat Mass Transfer*, 29(3): 351–358, 1986.
- XRG Simulation. URL <http://www.xrg-simulation.de/de/produkte/xrg-library/xrg-fluiddissipation-library>. fetched Dec., 15th 2016.

Improved Model of Photovoltaic Systems

Dmitry Altshuller¹ Peter Hüsson¹ Christopher Alain Jones¹ Leonard Janczyk²

¹Dassault Systems, USA, {dmitry.altshuller, peter.huesson, christopher.jones}@3ds.com

²Dassault Systemes, Germany, leonard.janczyk@3ds.com

Abstract

The paper describes a model of a typical photovoltaic (PV) system. Unlike models previously discussed in literature, heat transfer phenomena are accounted for simultaneously with the electrical dynamics. Furthermore, the model is simulated for a time scale of one full year.

Keywords: Photovoltaic system, Modelica, Thermal effect, Modeling thermal effect, Solar power, Dymola

1 Introduction

The importance of obtaining energy from renewable resources cannot be overestimated. However, harnessing these resources often presents considerable technical difficulties. Furthermore, the effectiveness of using resources such as wind or solar power depends on the weather conditions. It is, therefore, critically important to develop mathematical models that can reliably predict power output before any significant investment is made.

The main difficulty in modeling photovoltaic (PV) systems lies in the complexity of accounting for all the factors that may influence the performance of a PV cell. Most of the existing models of PV systems tend to focus only on some of these factors while simplifying the influence of others. For example, modeling the influence of solar irradiance is emphasized in (Tian et al, 2012) as well as in (Khatib and Elmenreich, 2016) while the dependence of temperature is simplified. By contrast, a detailed thermal model is developed in (Jones and Underwood, 2001) but the influence of electric power output on the temperature of the system, which, in turn, affects this power output, is considerably simplified.

The most commonly used tool for modeling PV systems is Simulink. PSpice is used in (Castaner and Silvestre, 2002) and MATLAB is used in (Khatib and Elmenreich, 2016). In this paper we propose to use Modelica and the tool Dymola to account for the mutual influence of the power output and temperature variation. To this end, we start with the circuit model proposed in (Pandiarajan and Muthu, 2011) which is then combined with the thermal model from (Jones and Underwood, 2001). The simulation is run using the weather module from the HVAC Library developed by XRG Simulation.

2 Mathematical Background

2.1 Equivalent Circuit

The concept of using an equivalent circuit to model a PV cell goes back to the book (Angrist, 1982). The model was subsequently improved in the book (Masters, 2004) and is to this day used. The circuit consists of a signal-dependent current source connected anti-parallel to a diode, parallel to a resistor, and in series with another resistor. The generic schematic of the equivalent circuit is shown in Figure 1.

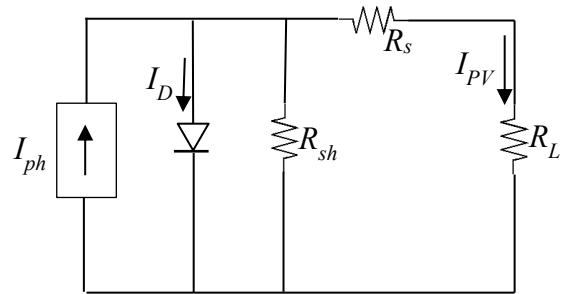


Figure 1. Schematic of a PV equivalent circuit.

The resistors R_{sh} and R_s are the intrinsic shunt and series resistances of the cell. Typically, the value of R_{sh} is several magnitudes higher than R_s . Therefore, they can be neglected to simplify some of the equations. These parameters also affect the characteristic of the diode.

The current I_{ph} depends on the solar radiation and the temperature of the system. In addition, the diode current depends on several other voltages and currents in the circuit. In the next subsections we will give the details of these models.

2.2 Modeling of the Current Source

The current source is the element where the solar irradiance energy is converted into the electrical energy. The equation for the current I_{ph} has the form (Seyedmahmoudian et al, 2013):

$$I_{ph} = [I_{sc} + K_i(T - T_{ref})] \frac{G}{G_{ref}} \quad (1)$$

In this equation I_{sc} is the short-circuit current, K_i is the temperature coefficient, T is the temperature of the

PV cell, $T_{ref} = 298$ K is the reference temperature, G is the solar irradiance, and $G_{ref} = 1000$ W/m² is its reference value.

The value of the solar irradiance is obtained from the weather model which is taken from the HVAC Library developed by XRG Simulation. The temperature is obtained by modeling various heat transfer phenomena described in Subsection 2.4.

2.3 Modeling of the Diode

The diode current is described by the following equation (Pandiarajan and Muthu, 2011):

$$I_D = I_0 \left[\exp \frac{q(V_{PV} + I_{PV}R_s)}{N_S A k T} - 1 \right] \quad (2)$$

The parameters in this equation are as follows:

$q = 1.6 \times 10^{-19}$ C is the electron charge;

N_S is the number of PV cells connected in series;

A is the ideality factor;

$k = 1.3805 \times 10^{-23}$ J/K is the Boltzmann constant.

Also, involved in the equation are voltage through and current at the load. These are obtained while the simulation is running.

The saturation current I_0 is calculated from (Angrist, 1982):

$$I_0 = I_{RS} \left(\frac{T}{T_{ref}} \right)^3 \exp \frac{q E_{g0} (T - T_{ref})}{B k T T_{ref}} \quad (3)$$

In addition to the already defined variables and parameters, B is the ideality factor and E_{g0} is the band gap. The equation for the reverse saturation current I_{RS} is (Pandiarajan and Muthu, 2011):

$$I_{RS} = \frac{I_{SC}}{\exp \frac{q V_{OC}}{N_S k A T} - 1} \quad (4)$$

The new parameters in this equation are the short-circuit current I_{SC} and the open-circuit voltage V_{OC} .

The reader is referred to the paper (Pandiarajan and Muthu, 2011) and references therein for a more detailed discussion of these equations and parameters.

2.4 Modeling of the Heat Transfer

There are four heat transfer mechanisms that must be modelled: heating by short-wave and long-wave radiation as well as cooling by free and forced convection. In addition, there is heat loss equal to the power output of the PV system. Let us describe each of these following (Jones and Underwood, 2001).

The short-wave radiation is directly proportional to the solar irradiance and is equal to $\alpha A G$, where α is the absorptivity, A is the surface area, and G is solar irradiance.

The long-wave radiation comes from two sources: sky and ground. It obeys the Stefan-Boltzmann law but the terms must be multiplied by respective emissivity of the sky, the ground, and the PV module. Furthermore, for the purposes of radiation modeling, the temperature of the sky is increased by 20 K for clear sky. This number is reduced down to zero for the overcast sky, proportionately to cloud cover. In addition, adjustments need to be made for the tilt angle of the panel.

The convection flow rate is directly proportional to the difference between the temperature of the panel and the ambient temperature. The coefficient for the free convection is proportional to the cubic root of this temperature difference (Holman and Bhattacharyya, 2011, p.335) but the coefficient for the forced convection is treated as a parameter since there are presently no known reliable correlations for it.

The equation has the form (Jones and Underwood, 2001):

$$C \frac{dT}{dt} = \alpha A G + \sigma A \left(\frac{1 + \cos \beta}{2} \varepsilon_{sky} T_{sky}^4 + \frac{1 - \cos \beta}{2} \varepsilon_{ground} T_{ground}^4 - \varepsilon_{module} T_{module}^4 \right) - (h_{c,forced} + 1.31 \sqrt[3]{T - T_{amb}}) A (T - T_{amb}) - P_{out} \quad (5)$$

In this equation, in addition to the already defined variables and parameters, C is the heat capacity of the module, σ is the Stefan-Boltzmann constant, β is the tilt angle of the panel, ε refers to the emissivity, and $h_{c,forced}$ is the forced convection coefficient. The indices for the temperatures are self-explanatory.

3 Model Implementation

3.1 Electrical System

3.1.1 Overall System Model

In order to model the electrical part of a PV system, the following Modelica classes were created: Current Source, Diode, PV array, and PV cell.

The PV cell class connects all of these elements and also includes the heat transfer model described below in Subsection 3.2. It includes a two-pin electrical port which is to be connected to the load and a temperature port to receive the ground temperature. Additional inputs are solar irradiance, the numerical value of the ambient temperature, and the cloud cover fraction. Parameters are propagated from its included classes. The model is shown in Figure 2

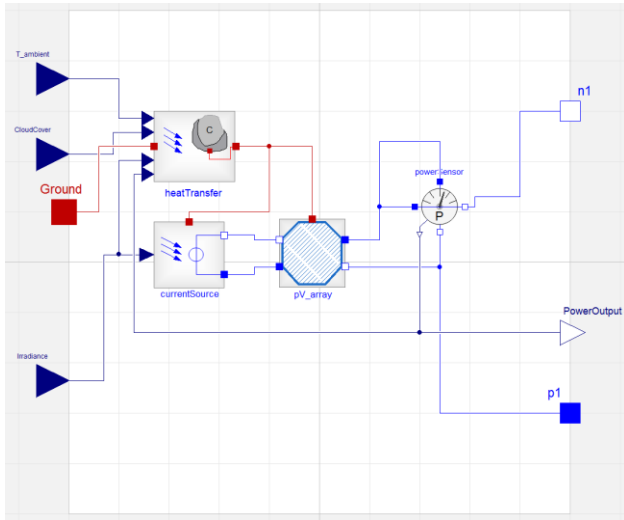


Figure 2. PV cell model.

The Current Source class has a two-pins electrical port connected to the signal-driven current source. The signal input for the source is calculated using the equation (1) using the solar irradiance as input. The class also has a thermal port that provides the temperature of the cell. Parameters for this class are the same as in the equation (1) and are propagated to the PV cell class.

PV array is a two-port electrical system. One port is connected to the Current Source and the other to the load. It also has a thermal port. The model includes the shunt and the series resistances and the diode model modified as described in the next Subsection. Parameters are propagated from its included classes and are further propagated to the PV cell class. The model is shown in Figure 3.

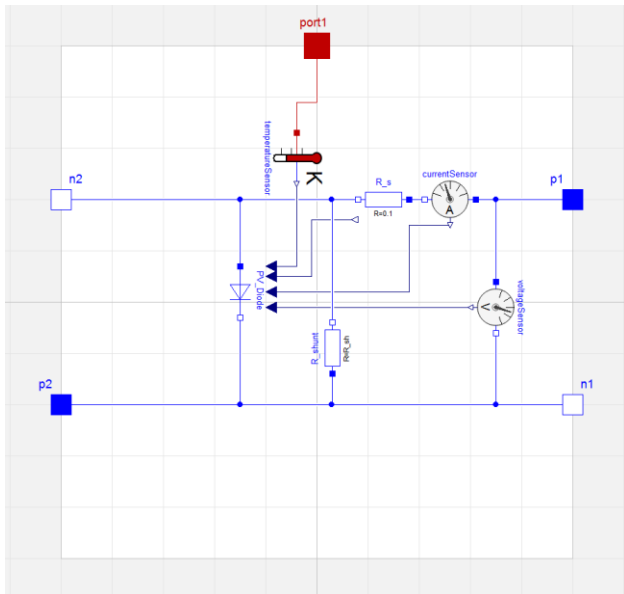


Figure 3. PV array model.

3.1.2 Diode Model

The PV_Diode model implements equations described in Subsection 2.3. It has been modified from the MSL Diode2 model.

The model itself has a basic electrical OnePort which are connected in the equivalent circuit model. Additionally, the model has four real inputs. These inputs are the temperature of the PV module, the serial resistance of the equivalent circuit model as well as the current and voltage signal of the load.

In comparison to the Diode2 model of the MSL we do not need to smooth the function, because we do not have to consider any switching behavior.

Parameters for the diode model and their default values are shown in Figure 4.

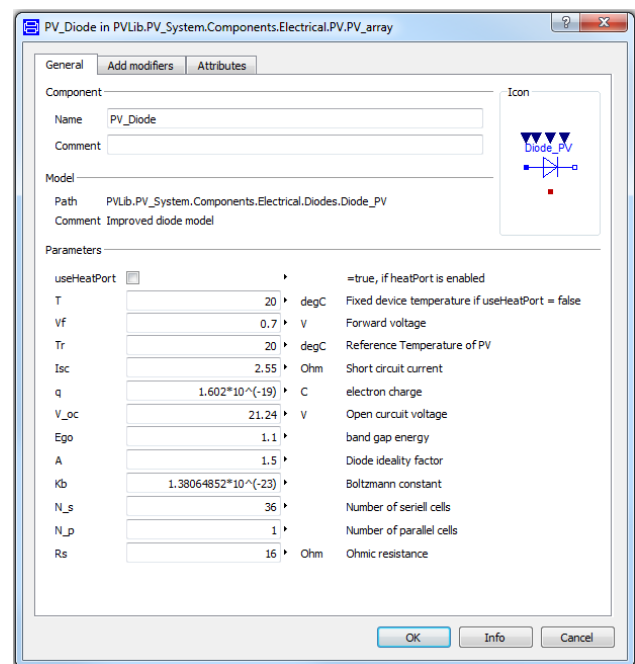


Figure 4. Parameters for the diode model.

The model equations are implemented as follows:

equation

```
Tk = temperature_PV;
Rs = resistance_Rs;
Ipv = current_Ipv;
Vpv = voltage_Vpv;
```

```
Irs = Isc / (exp((q*V_oc) / (N_s*Kb*A*Tk)) - 1);
```

```
I_01 = Irs * ((Tk/Tr)^3) * exp(((q*Ego) / (A*Kb)) * ((1/Tr) - (1/Tk)));
```

```
I_0 = I_01 * (exp((q*(Vpv + Ipv*Rs)) / (N_s*A*Kb*Tk)) - 1);
```

```
i = I_0;
LossPower = i*v;
```


3.1.3 Verification

After modeling the electrical part of the PV-array, we want to verify our model. Most of the literature use the same output characteristics of PV modules to characterize the system behavior. We use these characteristics to verify the correct behavior of our model.

The tests are made at a constant temperature of 293.15 K. Two variables are varied to get the typical curve. First the system load will be varied. Second, the irradiation will be changed.

Figure 5 shows the test setup of the verification experiment.

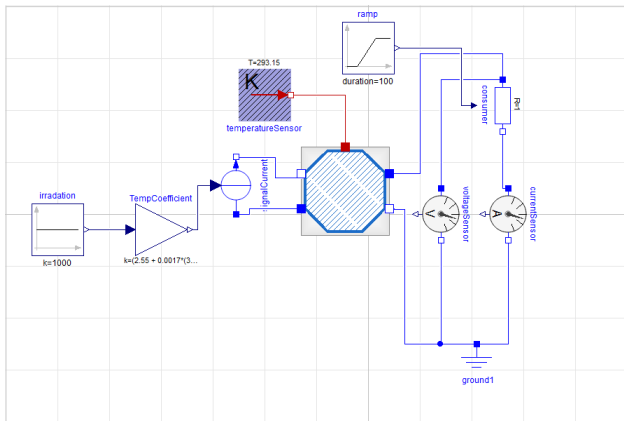


Figure 5. PV array test setup.

For the electrical parameters we use the references of a Solkar 36W PV module. The electrical characteristics can be seen in Table 1 (Pandiarajan and Muthu, 2011).

Table 1. Electrical characteristic data of Solkar 36W

<i>Electric parameters</i>	
Open circuit voltage (V_{OC})	21.24 V
Short circuit current (I_{SC})	2.55 A
Number of cells in series (N_s)	36
Number of cells in parallel (N_p)	1

For the first simulation run we take the irradiation of 200 W/sqm and change the electrical load outside of the PV cell. This load represents the consumer. We repeat this experiment with the irradiation of 600 W/sqm and 1000 W/sqm.

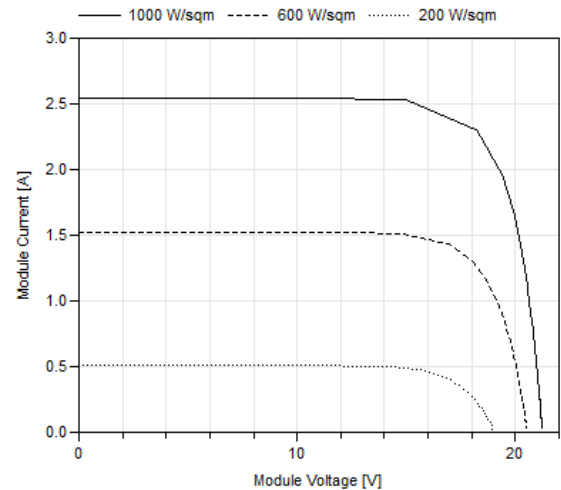


Figure 6. I-V characteristics with varying irradiation.

Figure 6 shows the result of the simulation runs. You can see the characteristic I-V output diagram of a PV cell. The results match the results from the paper (Pandiarajan and Muthu, 2011). Our electrical model is therefore considered verified.

3.2 Thermal System

The thermal system implements the equations described in Subsection 2.4. It has two thermal ports: one for the ground temperature and one for the temperature of the cell. The latter is connected to both Current Source and the PV array classes. Additional inputs are ambient temperature, solar irradiance, the cloud cover ratio, and the electrical power output computed in the PV cell model. Parameters for the heat transfer model are also described in Subsection 2.4 and they are propagated to the PV cell class. The model is shown in Figure 7.

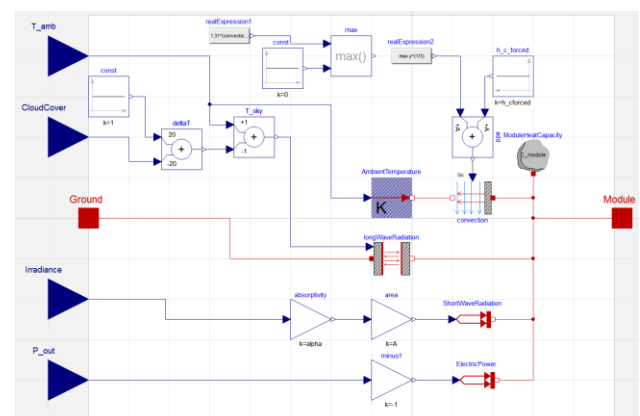


Figure 7. Heat transfer model.

Convection is modelled with the MSL Thermal Convection component. The forced convection coefficient is implemented as a parameter and the free convection coefficient is computed using the cubic root law with the provision that it can never be less than zero.

An additional class has been created for computing the long wave radiation by modifying the MSL thermal Body Radiation component. The two thermal ports from this component are used to connect the ground and the PV cell. A real input port has been added for the temperature of the sky. The parameters for this class are emissivities, the area of the panel and the tilt angle and they are propagated to the heat transfer model. The governing equation has been modified as follows in order to comply with the equation (5):

equation

```
Q_flow = A*Modelica.Constants.sigma*((
(1 + cos(beta_surface))/2)*
epsilon_sky*T_sky^4+((1 - cos(
beta_surface))/2)*epsilon_ground*port_
a.T^4 - epsilon_module*port_b.T^4);
```

The variables in this equation correspond to those in the equation (5). For example, β_{surface} means β , the tilt angle of the panel.

Unfortunately, data in (Jones and Underwood, 2001) does not provide enough information to verify our model in simulation experiments.

4 Experiments and Results

Simulation experiments have been run using the tool Dymola and the weather models from the HVAC Library. The setup is shown in Figure 8.

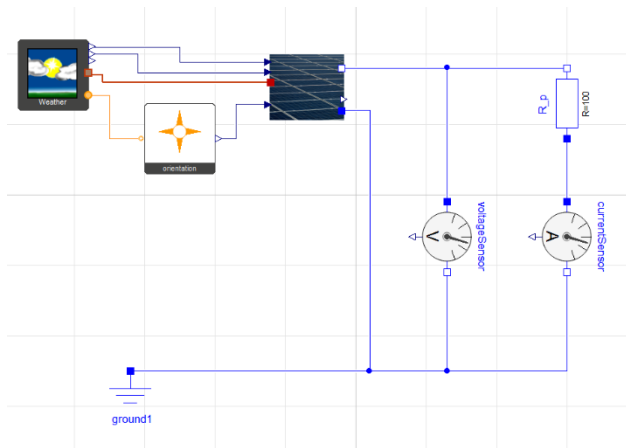


Figure 8. Experimental setup.

The weather and the orientation models are taken directly from the above-referenced library. The weather model uses the file specifying the weather condition for the entire year. It provides values of the ambient temperature, the cloud cover, the ground temperature and sun position in the sky. The latter is used by the orientation model to calculate the solar irradiance.

We have used the following values of the parameters for the PV cell as shown in Figure 9.

To get a better understanding of the system it makes sense to simulate the PV cells over the course of a year. To see the effect of the weather on the energy output of our solar system, it is interesting to compare different climate conditions. To keep the simulation results simple and comparable we only varied the cloud coverage between 0% and 50% for this investigation. The remaining weather conditions were kept the same for both simulation runs.

Current source			
T_ref	298	Reference temperature	
K_J	0.0017	Absolute Temperature coefficient	
G_ref	1000	Reference irradiance	
I_SC	2.55	Short-circuit current	
Heat Transfer			
A	1.51	Area of the module	
epsilon_sky	0.95	Sky emissivity. Set 0.95 for clear sky; set to 1.0 for overcast	
epsilon_ground	0.95	Ground emissivity	
epsilon_module	0.9	Module emissivity	
C_module	2918	Heat capacity of the module (= cp*m)	
alpha	0.7	Absorptivity of the surface	
h_cforced	2	Forced convection coefficient	
beta_surface	0	Tilt angle. Horizontal: 0; vertical 90 degrees	
T_init	20	Initial Temperature of the Module	
Electric behaviour			
V_oc	21.24	V	Open circuit voltage
N_s	36		Number of seriell cells
N_p	1		Number of parallel cells
Tr	20	degC	Reference Temperature of PV
R_sh	1000	Ohm	Shunt resistance

Figure 9. Parameters for the simulation.

In Figure 10 the generated energy over the course of a year is shown. One can see, that the cloud coverage has an effect on the energy output of the PV system. Although the simple variation of this weather input is just effecting the long wave radiation, described in Subsection 2.4. To get a better understanding of the electric outputs and the possible use-cases of the complete system, we would have to make simulations with varying set of weather data.

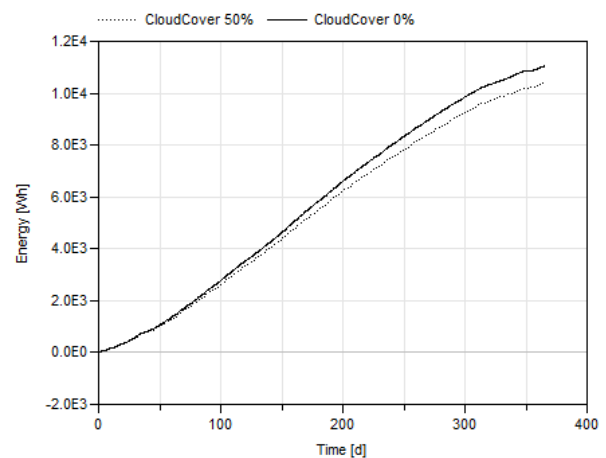


Figure 10. Electric energy over one year.

5 Conclusions and Outlook

The results of the simulation experiments demonstrate the fundamental soundness of our modeling approach, which is to combine the electrical and the thermal models.

Future research may proceed along the following lines. First, it may be beneficial to consider other weather conditions and/or climates. Second, it may be of interest to implement control systems and algorithms so that various parameters can be changed in response to changing weather conditions in order to obtain maximum power output. Finally, it will be interesting to investigate the feasibility of using PV systems in electric vehicles and/or battery charging stations.

References

- S. W. Angrist. *Direct Energy Conversion*, 4th ed. Boston: Allyn and Bacon, Inc., 1982
- L. Castaner and S. Silvestre. *Modeling Photovoltaic Systems Using PSice*. Chichester: Wiley and Sons, Inc., 2002.
- J. P. Holman, S. Bhattacharyya. *Heat Transfer. In SI Units.*, 10th ed. McGraw Hill, 2011.
- A. D. Jones and C. P. Underwood. A Thermal Model for Photovoltaic Systems. *Solar Energy*, Vol. 70, No 4, pp. 349–359, 2001.
- T. Khatib and W. Elmenreich. *Modeling of Photovoltaic Systems Using MATLAB*, John Wiley & Sons, 2016.
- G. M. Masters. *Renewable and Efficient Electric Power Systems*. Hoboken; Wiley and Sons, Inc., 2004.
- N. Pandiarajan and R. Muthu. Mathematical Modeling of Photovoltaic Module with Simulink. *International Conference on Electrical Energy Systems (ICEES)*, 3-5 Jan 2011, pp. 314-319.
- M. Seyedmahmoudian, S. Mekhilef, R. Rahmani, R. Yusof and E. T. Renami. Analytical Modeling of Partially Shaded Photovoltaic Systems. *Energies*, 2013, 6, pp. 128-144. doi: 10.3390/en6010128.
- H. Tian, F. Mancilla-David, K. Ellis, E. Muljadi and P. Jenkins. A Cell-to-Module-Array Detailed Model for Photovoltaic Panels. *Solar Energy*, 86, pp. 2695-2706, 2012.

Modelling of a Hydro Power Station in an Island Operation

Arndís Magnúsdóttir¹ Dietmar Winkler²

¹Verkís hf, Iceland, arm@verkis.is

²University College of Southeast Norway, dietmar.winkler@usn.no

Abstract

There is a strong focus on new renewable energy sources, such as, solar power, wind energy and biomass, in the context of reducing carbon emissions. Because of its maturity, hydropower is often overlooked. However, there is an era of hydro oriented research in improving many aspects of this well established technology.

Representing a physical system of a hydropower plant by mathematical models can serve as a powerful tool for analysing and predicting the system performance during disturbances. Furthermore it can create opportunities in investigating more advanced control method.

A simulation model of a reference hydropower station located in northwest of Iceland was implemented using the modelling language Modelica[®]. The main simulation scenarios of interest were: 20 % load rejection, worst-case scenario of full shut-down and pressure rise in the pressure shaft due to the water hammer effect. This paper will show that the different simulation scenarios were successfully carried out based on the given the data available of the Fossárvirkjun power plant. The load rejection simulation gave expected results and was verified against a reference results from manufacturer.

Keywords: Hydropower in Iceland, modelling, simulation, island operation, Modelica, Dymola, Electric Power Library, Hydro Power Library, water hammer effect

1 Introduction

The process of using the energy of moving water to create electricity is a long-standing, well-proven and reliable technology. Unlike other renewable energy sources, hydropower is not a recent development but has been around for several hundredths of years. As of today the availability of hydropower has been associated with kick-starting economic growth (International Hydropower Association 2016).

There is a strong focus on renewable energy sources in the context of the desired global reduction in carbon emissions. Technologies such as solar power, wind energy and biomass are in focus while hydropower is often overlooked. Hydropower has many advantage when it comes to the effect of climate change as it is renewable, efficient and reliable source of energy that does not directly emit greenhouse gasses. Because of its maturity, hydropower is often associated with conservative and perhaps stagnant technology development. However, there is an area

of hydro-oriented research in improving many aspects of this well established technology, taking full advantage of progress in science and engineering (Munoz-Hernandez, Mansoor, and Jones 2013).

Around 70 % of Iceland's electricity is produced from hydroelectric power and is the world's largest electricity producer per capita. In cooperation with Icelandic oldest and leading consulting engineers in energy production, Verkís hf, a complete dynamic hydropower model was implemented based on a reference power station, Fossárvirkjun, located in the northwest region of Iceland. The objective of developing such model is to study the dynamic characteristics of the plant, such as load rejection and to explore worst-case scenario of a full shut-down of the plant. Furthermore, the effect of water hammer, following pressure rise in the pressure shaft will be of outermost interest since Fossárvirkjun's water-way has no surge tank installed. Water inertia is the main aspect that influences the water hammer waves in the pressure shaft.

To build such model and to simulate these different scenarios the object-oriented modelling language, Modelica[®], is used to model the complex, physical power plant. The commercial modelling and simulation environment Dymola (Dassault Systèmes 2016), a product of Dassault Systèmes, was used. In addition, two separate libraries, the Hydro Power Library(HPL) and the Electric Power Library(EPL) (Modelon AB 2016) will be coupled together in order to represent the complete hydro power system.

1.1 Fossárvirkjun

In the year 1937, a hydropower station was built to serve Ísafjörður, located in the northwest region of Iceland in Skutulsfjörður, in the Westfjords. At that time, it was the only electric power source for the Ísafjörður area. Since then there has been no refurbishment until now. The Westfjord Power Company has refurbished the existing power station with a new turbine/generator and electrical equipment. A new pressure shaft and a new powerhouse were constructed about 800m from the existing one and the new power station is named Fossárvirkjun. The existing 600kW Pelton machine was replaced by a new 1200kW Pelton turbine. The new refurbished power plant serves Súðavík in an island operation (*Refurbishment of the Fossár hydro Power Plant* 2015). Figure 1 shows the new power house of Fossárvirkjun.

The reference system used for the modelling part is the



Figure 1. Power house of Fossárvirkjun (*Refurbishment of the Fossár hydro Power Plant 2015*)

new refurbished Fossárvirkjun that started operation in autumn of 2016.

The reservoir is Fossavatn, a fresh water which is mostly fed by direct runoffs and springs. The intake is at 343 m.a.s.l. and the rated discharge is at $0.45 \text{ m}^3/\text{s}$. The pressure shaft is around 1900 metres long consisting of a *DN500 GRP* pipe with no surge facility. The turbine is a two-nozzle horizontal Pelton turbine. Since Fossárvirkjun will be running in island operation two simulation scenarios are of interest.

As has been mentioned, there is no surge tank to absorb a sudden rise of pressure in the pressure shaft. Therefore, the pressure at the bottom of the pressure shaft, has to be closely monitored. Table 1 summarises the general information data of the system.

Table 1. General data table of Fossárvirkjun

<i>Properties</i>	<i>Values</i>	<i>unit</i>
<i>Pressure shaft</i>		
Length	1 900	[m]
Inner Diameter	0.50	[m]
Nominal pressure in pressure shaft	32	[bar]
Maximum over pressure	15	[%]
<i>Pelton Turbine</i>		
Number of Nozzles	2	
Rated Discharge	0.45	[m^3/s]
Rated Net Head	308	[m]
Turbine Efficiency	91	[%]
<i>Synchronous Generator</i>		
Power	1404	[kVA]
Max mechanical power	1325	[kW]
Nominal Voltage	400	[V]
Nominal Current	2026.5	[A]

A rough sketch of the real water-way of Fossárvirkjun is depicted in Figure 2. The intake is at 343 m.a.s.l. and the connection to the turbine at 38 m.a.s.l. The length of the water-way roughly 1900 m, keeping in mind that the

actual length of the pipe segments is longer.

The turbine runner is fixed on the generator's shaft. The generator is a standard 400V AC synchronous machine with a brush-less excitation system. The governor is a PID controller.

2 Modelling

The Modelica simulation environment used in this project was Dymola which is commercial tool for modelling and simulation of complex systems. It is a product of Dassault Systèmes. Dymola allows the user to create a graphical representation of a physical system and has different solvers to choose from. Modelica is multidomain modelling language which means that different libraries produced by sometimes several developers can be coupled together if needed. Taking the advantage of this multidomain modelling, two types of libraries were used to build the dynamic model of Fossárvirkjun; Hydro Power Library and Electric Power Library.

The complete power system of Fossárvirkjun can be seen in Figure 3. The model entails different source components that are connected together.

The reason why the EPL has to be coupled with the HPL is that even though the HPL contains an electrical system, it does not give information about active or reactive power, that is, it is only calculating active power quantities.

2.1 The Water-Way

The water-way was modelled using components from the HPL that calculate the media state vectors ($f(p, T)$) and media flow of the water.

An important assumption made in the modelling is that the states are uniformly distributed. It is assumed in the upcoming modelling that the water head is constant, that is, assuming that the water source is an infinite. Figure 4 shows the water-way sub-component.

Mass, energy and momentum balance equations are discretised with the finite volume method using an upwind discretisation scheme. State variables are pressure, temperature and mass-flow for each pipe segment. Each pipe segment is split up by a combination of closed volume models and mass flow models. For each pipe segment the two models contain the following

Closed Volume Models

- Conservation laws: Energy Balance and Mass Balance
- State variables: Pressure (p) and temperature (T)
- Inflow and outflow: Flow of mass and enthalpy

Mass Flow Models

- Conservation Laws: Momentum Balance
- State variables: Mass flow \dot{m}
- Outflow: \dot{m}_{out}

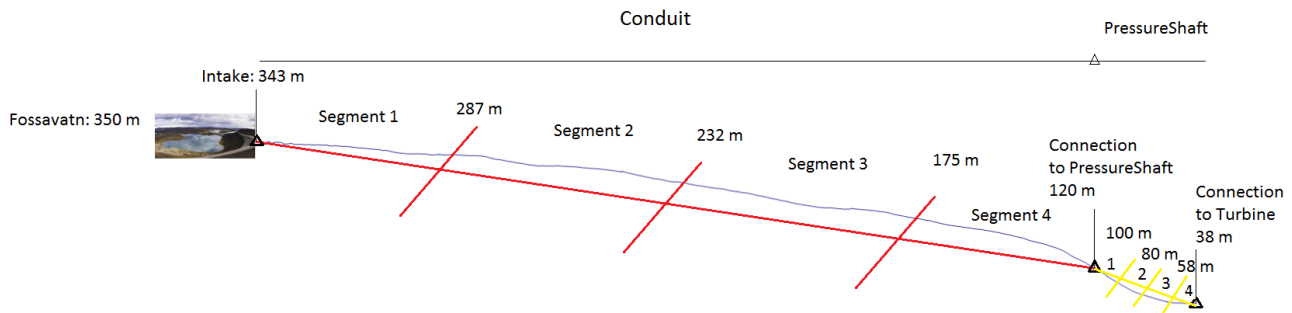


Figure 2. From the real water-way of Fossárvirkjun to modelled water-way in Modelica, split by segments.

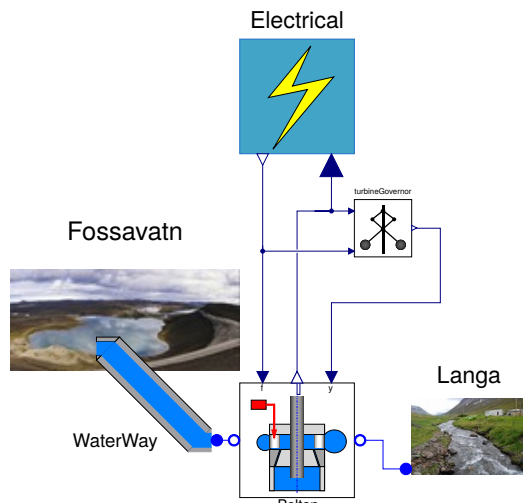


Figure 3. Complete model of Fossárvirkjun in Dymola

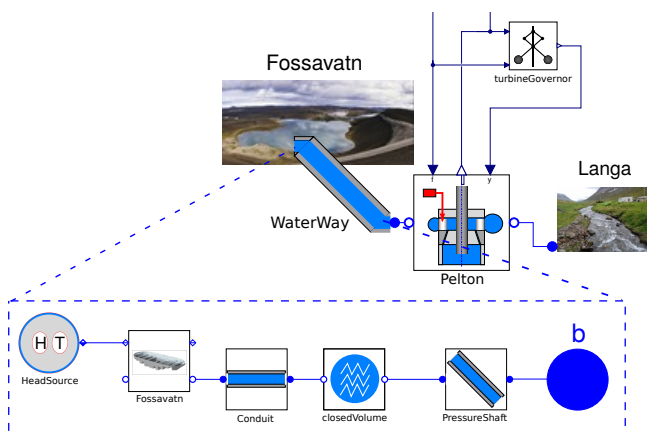


Figure 4. Submodel: Water-way

2.1.1 Finite Volume Method

For one phase flow models, the partial differential equations of mass, energy and momentum are discretised and solved with the finite volume method where they are integrated and approximated by ordinary differential equa-

tions. The Finite Volume Method is considered to be particularly good at maintaining the conserved quantities (Elmqvist, Tummescheit, and Otter 2003).

The conduit in Fossársvirkjun is a uniform pipe, but modelled with two separate pipes, the conduit and the pressure shaft. This was done in order to be able to analyse the pressure shaft in more details because of the special interest in the pressure rise.

The water-way sub-component consists of a head source, reservoir (Fossavatn), conduit, closed volume and pressure shaft:

Head Source Infinite source of volume with prescribed details about water height and temperature.

Reservoir/Fossavatn Detailed reservoir built with n segments. Using massflow models which calculates using momentum balance for fluid segments that is between two open channel segments/reservoir.

Conduit/Pressure shaft Model of discretised pipe with massflow models at inlet and outlet. Using the upwind scheme of finite volume method to discretise the balance equations; Mass, Momentum and Energy. Pressure, temperature and mass-flow are the state variables. This pipe is made up of n segments.

Closed Volume Used to connect the conduit and pressure shaft together. As the name implies, it is a closed volume with state variables as pressure and temperature.

In relation to the model of the water-way in Figure 4 where different sub-components come together to create the water-way,

The earlier Figure 2 shows also how the different sub-components were used in order to build the model of the head-race water-way. The conduit model (red line in the figure) is divided into four segments. It begins at the intake and ends at the junction with the pressure shaft. The pressure shaft then starts descending at this junction and continues all the way to the turbine inlet. The real water-way of Fossárvirkun is the blue line in the background.

As Figure 2 shows, there is some loss in detail in the water-way while modelling. From one junction to another, the conduit pipe is modelled as a straight line. In theory, you could have numerous of segments throughout the conduit and subsequently minimising the loss of detail but with the cost of the simulation being computational demanding.

As mentioned before, the conduit is composed of two main elements; closed volume and mass flow component. To calculate the dynamics all three conservation equations; Energy, mass and momentum; are used. The HPL calculates the mass and energy balance in the closed volume and the momentum balance in the mass flow component. One of the benefits of using Modelica language is the transparency, that is, behind the sub-components/models are the corresponding equations that describe the dynamics of the model. For example, the reservoir model that represents *Fossavatn* uses the momentum balance to calculate the mass flow models.

2.1.2 Pelton Turbine

The HPL offers two types of turbine models; the Kaplan turbine with guide vanes and runner blades and a basic turbine with guide vane servo which can be used for both Francis and Pelton turbines. The latter turbine model was the preferred choice for Fossárvirkjun.

The turbine model is controlled via a `gateActuator` input signal from the controller changing the discharge of the turbine. For Pelton turbines this corresponds to the nozzle opening which dictates the flow through the turbine based on a look-up table, *i.e.*, *TurbineTable*. This turbine look-up table contains information about:

- Nozzle Opening [pu]
- Volume Flow Rate [m^3/s]
- Turbine Efficiency [pu]

Based on the nozzle vane opening, the volume flow rate and turbine efficiency can be calculated. Therefore, the behaviour of how the turbine responds to the control signal depends on the *TurbineTable*.

The corresponding plot can be seen in Figure 5. The red line represent the turbine efficiency [pu] and the blue line the volume flow rate [m^3/s] corresponding to the gate actuator signal [pu] on the x-axis.

The Pelton turbine contains two nozzle jets. The first nozzle operates alone under relatively low flow rate ($0.124 - 0.224 \text{ m}^3/\text{s}$) until the second nozzle steps in to aid with the increased flow at $0.225 \text{ m}^3/\text{s}$. This is clearly visible in Figure 5 where there the blue line becomes suddenly steeper. At this time, the efficiency also increases as the red line displays.

Equation (1) describes the power from the Pelton turbine.

$$\begin{aligned} P_{\text{turbine}} &= \eta_{\text{hydro}} \cdot \Delta P_{\text{available}} \cdot Q_{\text{max}} \\ &= \eta_{\text{hydro}} \cdot H_{\text{available}} \cdot g \cdot \rho \cdot Q_{\text{max}} \end{aligned} \quad (1)$$

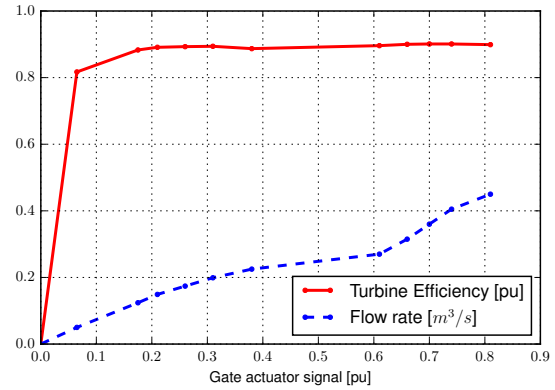


Figure 5. Plot from the *TurbineTable*

For Fossárvirkjun the maximum power arriving at the turbine shaft is calculated using the maximum efficiency of 91 % (from the *TurbineTable*):

$$\begin{aligned} P_{\text{turbine}_{\text{max}}} &= 0.91 \cdot 304 \text{ m} \cdot 9.81 \frac{\text{m}}{\text{s}^2} \cdot 1000 \frac{\text{kg}}{\text{m}^3} \cdot 0.45 \frac{\text{m}^3}{\text{s}} \quad (2) \\ &\approx 1.221 \text{ MW} \end{aligned}$$

2.1.3 Langá

The Langá component consists simply of a pipe model and a fixed source of temperature and pressure, as can be seen in Figure 6.

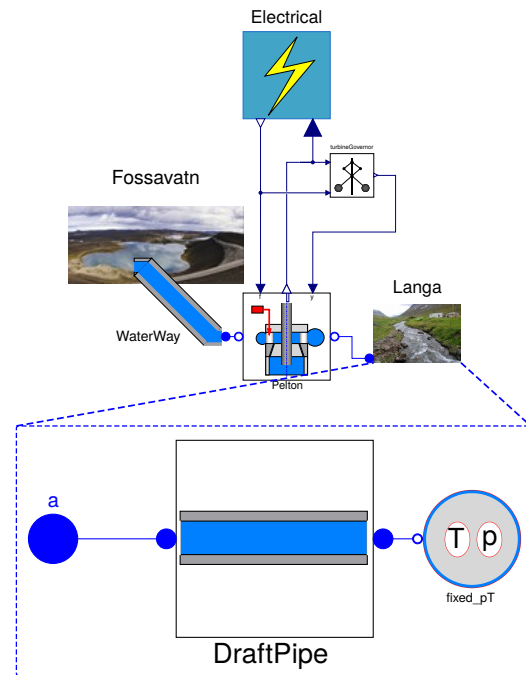


Figure 6. Details of the Langá model

Since the Pelton turbine does not require a draft tube, the pipe that is connected to the output of the turbine is

only put in there to have the connectors compatible with the fixed source. The fixed source is simply a constant pressure, which is set near to atmospheric pressure.

2.1.4 Governor

The governor component is situated above the Pelton turbine as can be seen in Figure 3. The governor is an analogue PID controller where it takes in both the power from the generator and the frequency. The PID controller works under two conditions; No-load and under load. These conditions are set with a Boolean condition; `true` when *no-load*, `false` when under *load*. This Boolean condition allows to run with two sets of parameters, one for speed control and one for power control. The calculations for the error signal into the PID controller is shown here below in Equation (3).

$$e = (f_0 - f) + (P_{in} - P_{ref}) \quad (3)$$

Since the power system will be run in speed control the governor will have an open MCB breaker, that is the Boolean condition is set to `true`. The signal will be the speed of the rotor connected to the generator. The governor will therefore control the output by keeping the signal at a speed of 1 pu, *i.e.*, 50 Hz.

2.2 Electrical grid

For the modelling of the electrical grid the Electric Power Library was used. It is a library for electric power systems. The library offers a choice of different phase systems:

- DC system
- AC one-phase system
- AC three-phase abc (non-transformed)
- AC three-phase dq0 (dq0-transformed)
- AC three-phase dq (dq-transformed) — for a balanced system

The electrical grid was modelled for a balanced system, that is, represented by the AC three-phase dq0 system but omitting the zero-component creating the AC three-phase dq-transformation. Figure 7 shows the details of the electrical grid component.

The power generated from the Pelton turbine goes as an input to the single mass rotor in per unit which is then connected to the generator through a flange. The synchronous generator generates power with positive direct-quadrature representation. The voltage and reactive power is controlled by the first order control exciter which is connected to the field voltage. In between the load/consumer, is the transformer.

The transformer is a step-up type, from 0.4 kV to 11 kV. The 5 km transmission line then carries the alternating current to the consumer. The consumer is a small fishing village, Súðavík, located on the west coast of Iceland.

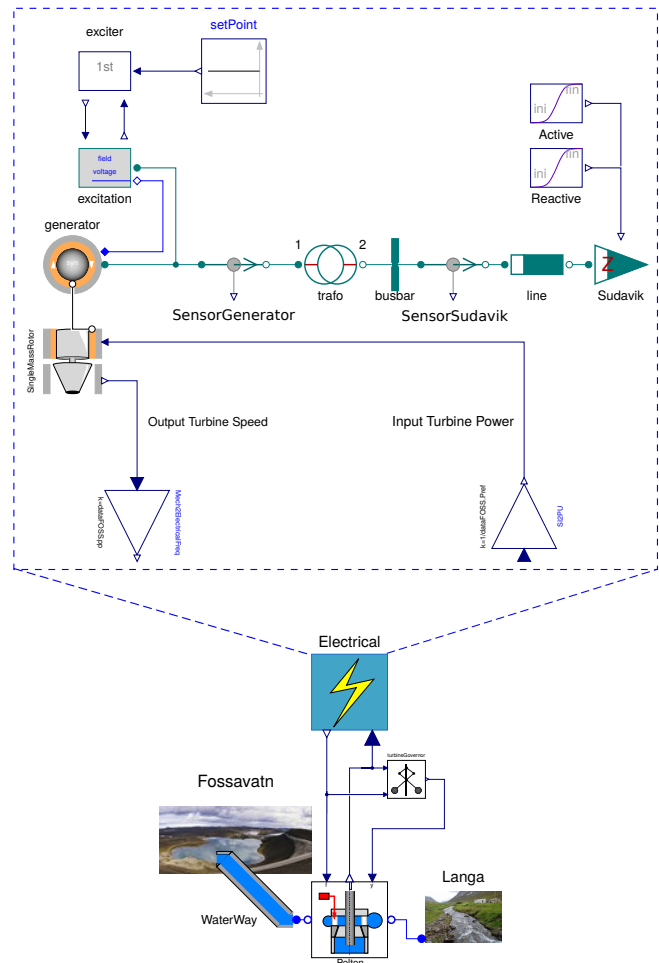


Figure 7. Electrical Component in EPL

20 km from Ísafjörður. Half of the power consumed is from households and the other half is consumed by a fish factory.

The EPL is highly complex, where all the components involved are fully mathematically represented. Since EPL is very detailed, the amount of input parameters required by the user is plentiful. This can be beneficial for accuracy reasons but does invite parameterisation error. There are a great number of input parameters that have to be known and correspond to a real scenario power system. Compared to the HPL, EPL is very sensitive to parameter inconsistencies.

The main components involved are:

2.2.1 Single Mass Rotor

Represents one single stiff rotating mass, defined with inertia constant H [s]. The single mass rotor is used as a connector between the generator and the Pelton turbine. A power signal from the HPL turbine model is used to calculate the rotational speed based on the load that the connected generator represents.

2.2.2 Synchronous Generator

This component is a three-phase-balanced-dq, AC synchronous machine with electric excitation. The user can choose from a Y or Delta topology.

2.2.3 Exciter

The exciter controls the excitation DC voltage with first order control which is directly determined by the per unit voltage control signal. The exciter controls both the reactive power and the voltage in the field.

2.2.4 Transformer

Ideal three-phase-balanced-dq step-up transformer. The magnetic coupling is ideal with no stray-impedance and zero magnetisation current. The user then chooses between Y and Delta topology at primary and secondary side. On the primary side there is the 0.4 kV from the generator and on the secondary side the resulting 11 kV from the transformer.

2.2.5 Súďavík Load

Inductive three-phase-balanced-dq load. Consumes active and reactive power of nominal voltage. Power is derived from the apparent power multiplied with the power factor input.

3 Simulation

The act of simulation is the experiment done on the model. The simulation results depend highly on how well the model represents the real system. One should always note that the simulation is only valid under the limitation and conditions given and can never represent the system completely, but is mainly an approximation for understanding the system. The simulation is only valid for the given input data (Tiller 2016). There were two types of simulation scenarios of interest.

- 20 % load rejection
- The water hammer effect

Since the power system is in an island operation it is important to monitor the behaviour of any disturbances in the system. The load rejection simulation was constructed by a 20 % sudden load rejection. This scenario is trying to imitate the incidence when there is a power shut-down, *e.g.*, a shut-down of a large factory. The water hammer effect is particularly of interest for two reasons: There have been incidents where the pressure on the bottom of the pressure shaft raised above the pressure threshold of the pipe's material, resulting in an outburst. Second reason is the lack of surge tank in the power system. The objective of the surge tank is to absorb the pressure and therefore take care of the sudden pressure rise in the pressure shaft, like has been stated. Omitting the surge tank leads to an increase in the travel distance of the impact waves in the conduit which causes increase in inertia of the water mass (Kiselev 1974).

3.1 Load Rejection

The load rejection simulation was constructed in a way that the induction load modelled was changed from its original steady active power load of 1.239 MW to a sudden drop of 20 % resulting in an active power of 0.996 MW. Figure 8 illustrates the model basis for the simulation consisting of the water-way, governor and electrical part.

The results from the simulation can be seen in Figure 9 where the plot illustrates the expected changes in active power, reactive power and the flow into the turbine. The aim here was to keep the rotor speed (frequency) consistent at 1 per unit (50 Hz). The upper plot shows the rotor speed [pu] as the red line and the flow m^3/s in to the turbine as the blue line. The control action taken is to decrease the nozzle opening to compensate for the power loss caused by the load rejection. Similarly, the active and reactive power [W] decreased accordingly.

Similarly, it is interesting to see if the voltage stays constant since the aim of the exciter (voltage regulator) is to keep the voltage steady. On the upper plot in Figure 10 the results from the 20 % Load Change illustrate the effect it has on the voltage both on the low voltage side and the high voltage side, that is, before and after the transformer. On the lower plot in the same Figure 10 the pressure at inlet of the turbine rises from 27.47 bar to 29.19 bar, thus the pressure increase is 1.72 bar. This increase in pressure is a result of the output of the controller, closing the valve to reduce the flow.

To summarise, Table 2 reflects the numerical results from the 20 % load rejection.

Table 2. 20 % load rejection

	Original	Change	Difference [%]
Active P. [MW]	1.239	0.996	−19.61
Reactive P. [Mvar]	0.138	0.111	−19.56
Pressure [bar]	27.47	29.19	5.89
Flow [m^3/s]	0.454	0.341	−24.89

Since the objective of the controller is to keep the rotor speed constant, three different load rejections were implemented to see the reaction of the rotor. Figure 11 shows the results after the following load rejections; 20 %, 40 %, 60 % and 80 %. The desired outcome is to keep the speed at 1 pu (50 Hz) after each load-rejection.

As can be seen in Figure 11 it follows that higher the load rejection the more amplitude the oscillations have at the instance when the load changes.

3.2 The Water Hammer Effect

The following simulations were done in order to investigate pressure rise in the pressure shaft and the effect it has on the governing stability due to the oscillations in the pressure shaft. A rapid change in the flow can lead to major oscillations in the water-way, also called the water hammer effect. Figure 12 shows the model constructed

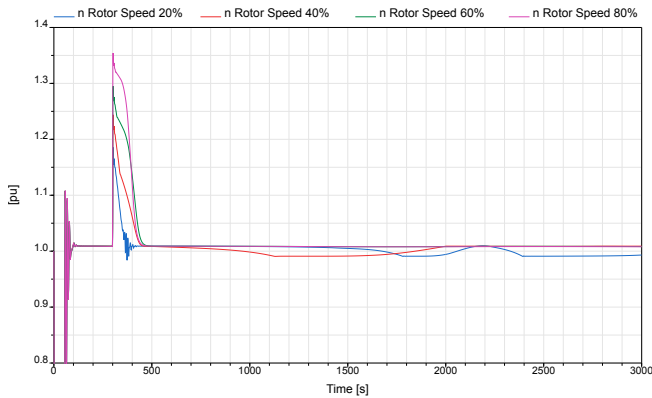


Figure 11. Rotor speed after various load rejections

The reason for the creating a stand-alone water-way is simply to allow more direct flow changes and investigations without a controller modifying the control signals because of some safe-guard and control delay restrictions that might be present/activated.

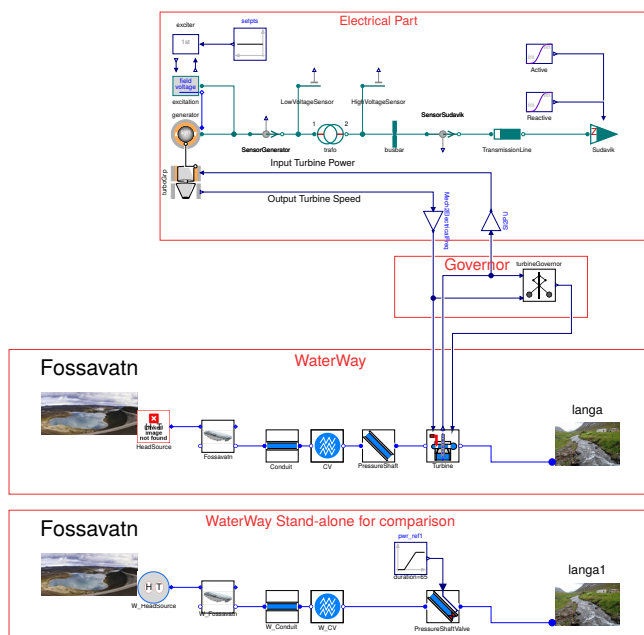


Figure 12. Overview of the model used for the water hammer effect scenario

It follows that in order to compare these models, the control signal from the governor in the upper model has to be the same as the valve/nozzle closing time. The control signal to the valve in the stand-alone model is a simple ramp function. The resulting plot can be seen in Figure 13. Since the control system is involved in the complete model, it is not possible to simply close the nozzle in the Pelton turbine. To achieve a fully closed turbine the load has to be shut-down first. Therefore, the load is set to zero at time 250 seconds, from its original load. The time it takes to fully close the turbine until there is no flow

through, is 65 seconds.

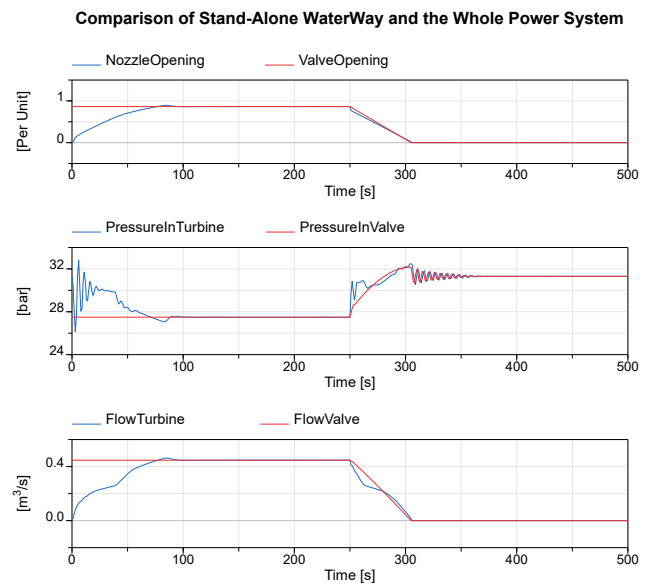


Figure 13. Water hammer plot comparing both models

The top plot shows the nozzle closing signal from the governor and the equivalent ramp signal to close the valve. As can be seen in Figure 13 they are almost identical. The most important is that their closing time is the same, which it is.

The comparison between the pressure drop in the turbine and valve can be seen on the middle plot. As expected, for the whole power system there is fluctuation in the pressure at the beginning since the governor is reacting to the full load. However, for the stand-alone water-way the valve starts fully opened. Eventually after 100 seconds the pressure in the turbine settles to the same pressure as the valve. At the 250 seconds the turbine and valve close. Apart from the pressure oscillation in the whole system, the models respond in a similar dynamic behaviour. Similarly, on the bottom plot the flow out from the turbine and the valve behave in a similar manner.

Since the comparison between the stand-alone water-way and the whole system gave identical results the stand-alone water-way can undergo further analysis. It was important to confirm that for the same opening degree, pressure and flow the results are identical before and after closing. For worst-case scenario in terms of the water hammer effect is if the load in Súðavík completely shuts-down. This can be seen in the resulting plot on Figure 13. There the time it takes to close the turbine is 65 seconds.

Having now an identical water-way with a simple pressure shaft with valve, an analysis of a faster closing of the valve can take place to test the minimum closing time to see the maximum allowable pressure in the pressure shaft.

Stated in the technical data from the manufacturer the allowable pressure rise in the pressure shaft is 15 %. We

investigated how quickly the valve can close. This was done by gradually decreasing the closing time starting from at 56 seconds as shown in Figure 13 and then inspecting the pressure rise for smaller closing times. Table 3 displays the peak/maximum pressure rises for a series of faster closing times.

Table 3. Closing time in water-way analysis

Closing time [s]	Pressure	
	Max [bar]	Rise [%]
56	32.26	0.8
40	32.58	1.8
15	34.76	8.6
12	35.59	11.2
10	36.71	14.7

The corresponding plots can be seen in Figure 14. The upper plot shows the closing signal to the valve and the bottom plot shows the pressure oscillations at the inlet of the valve. The most aggressive pressure rise is between closing time (10-15 seconds), resulting in heavy oscillating dynamic of the water wave.

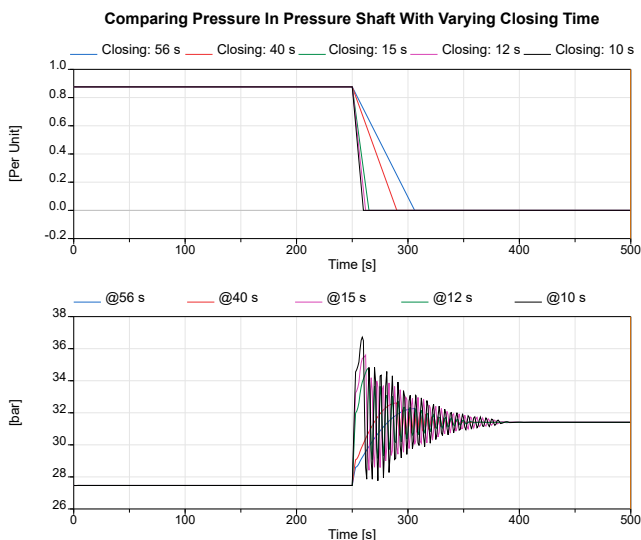


Figure 14. Closing time analysis on stand-alone water-way

Figure 15 shows a schematic of the water-way where the blue line represents the actual pipe alignment and the red/yellow lines represent the pipe as modelled in Modelica split up by segments. Each pipe is divided into four segments of equal length. One could increase the resolution by using more segments but in this case the default of four was sufficient. Both the elevation of the pipe segments and corresponding pressure is marked on the schematic. The pressure build-up due to the closing of the valve from the intake at 343 m and down to the turbine inlet can be seen in Figure 16.

4 Conclusion

4.1 Load Rejection

The load rejection was carried out while monitoring the flow into the turbine, speed of the rotor, pressure, voltage and power. The variables of interest gave a promising outcome indicating in a dynamic model that should represent Fossárvirkjun power plant adequately. Since having information regarding 20% load change from the manufacturer, similar load change scenario was implemented in order to validate the results.

As for the change in active and reactive power due to the load change, both decreased immediately around 19.6% in power. They are controlled by separate controllers, active power by the PID governor and the reactive by the voltage regulator, therefore a good indicator that both controllers are taking similar action. When looking into whether the results are as expected is to

Also the in (1) calculated theoretically available Pelton turbine power of 1.221 MW compares well with the simulated active power of 1.239 MW.

The same can be said for the voltage in Figure 10. The objective of the voltage regulator is to keep the voltage constant during load rejections. The voltage on both, the low voltage side and the high voltage side, remains constant throughout the disturbance which results in a good performance from the voltage regulator.

4.2 The Water Hammer effect

The analysis of the water hammer effect was implemented in Section 3.2 where the stand-alone water-way was compared to the whole power system. The results in Figure 13 were promising as both models yielded to similar behaviour. Since both water-ways are identical, apart from the valve in the pressure shaft on the stand-alone unit, it was expected that the pressure would be the same. The pressure and the flow in the turbine are of course more oscillating since being represented by the whole power system and thus controlled by the governor while the stand-alone model shows a more ideal behaviour.

After having the above results confirm that the stand-alone unit had identical result to the whole power system. More aggressive worst-case scenario shut-down of the valve took place. Closing time analysis was therefore implemented while observing the pressure in the pressure shaft of the stand-alone unit. Figure 14 showed the pressure increases with different closing times. To no surprise, the pressure increased as expected from the original closing time of the valve of 56 seconds down to 10 seconds.

The worst-case scenario shut-down of the valve indicated that a closing time of 10 seconds creates a maximum pressure increase to 36.71 bar. This is something that is dangerously near the maximum allowed pressure of 32 bar + 15%, see Table 1. Therefore, the results indicate that the valve/turbine should not be closed/shutdown in under 12 seconds.

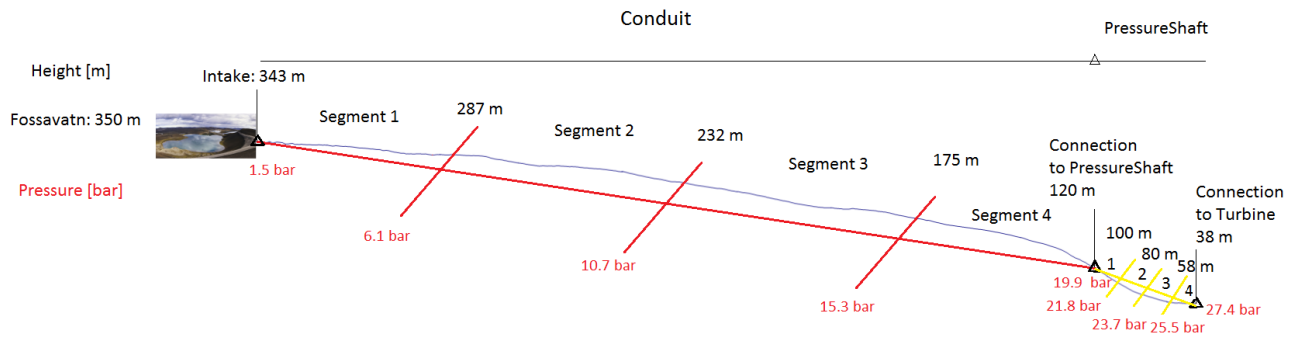


Figure 15. Pipe segments Fossavatn to turbine/valve inlet

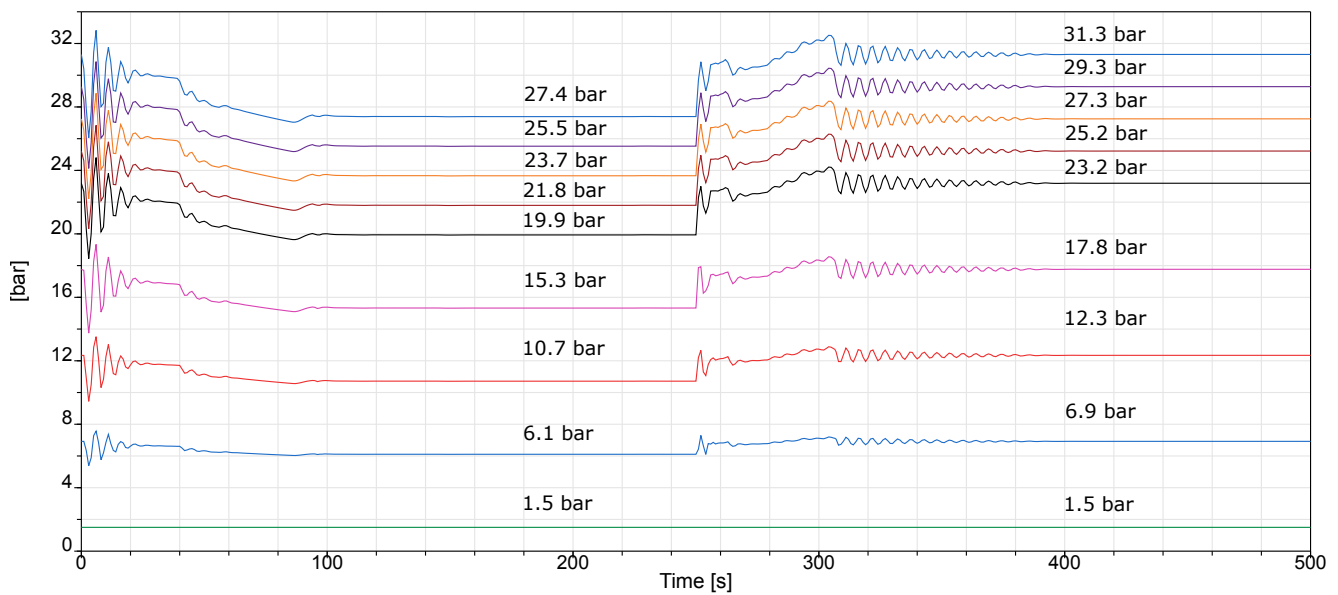


Figure 16. Pressure build-up in pipe segments from segment 1 (bottom) through to turbine connection (top)

References

- Dassault Systèmes (2016). *Dymola*. Modelon. URL: <http://www.dymola.com> (visited on 05/28/2016).
- Elmqvist, Hilding, Hubertus Tummescheit, and Martin Otter (2003). "Object-oriented modeling of thermofluid systems". In: pp. 269–286. URL: <http://elib.dlr.de/11988/> (visited on 05/30/2016).
- International Hydropower Association (2016). *A brief history of hydropower*. International Hydropower Association. URL: <http://www.hydropower.org/a-brief-history-of-hydropower> (visited on 05/28/2016).
- Kiselev, G. S. (1974). "Effect of water inertia in penstocks on regulating characteristics of hydraulic units". In: *Hydrotechnical Construction* 8.4, pp. 337–341. ISSN: 1570-1468. DOI: 10.1007/BF02406941. URL: <http://dx.doi.org/10.1007/BF02406941>.
- Modelon AB (2016). *Modelon Libraries*. Modelon. URL: <http://www.modelon.com/products/modelica-libraries/> (visited on 05/28/2016).
- Munoz-Hernandez, German Ardul, Sa'ad Petrous Mansoor, and D. I Jones (2013). *Modelling and controlling hydropower plants*. London; New York: Springer. ISBN: 978-1-4471-2291-3. URL: <http://public.eblib.com/choice/publicfullrecord.aspx?p=973672> (visited on 05/25/2016).
- Refurbishment of the Fossár hydro Power Plant* (2015). Verkís. URL: <http://www.verkis.com/about-us/news/refurbishment-of-the-fossarvirkjun-power-plant> (visited on 05/28/2016).
- Tiller, Michael M. (2016). *Modelica By Example*. Ed. by Michael M. Tiller. URL: <http://book.xogeny.com/> (visited on 01/20/2017).

Periodic Steady State Identification for use in Modelica based AC electrical system simulation

Martin Raphael Kuhn

German Aerospace Center (DLR e.V.), department of system dynamics and control, Germany
Martin.Kuhn@dlr.de

Abstract

Analysis of dynamic systems is often carried out at steady state condition. For cyclic systems like rotating machinery, it is not possible to detect this condition by simply monitoring the change rate of their variables, due to their periodicity. This paper focuses on methods for stationary periodic steady state identification of AC electrical systems. An overview of relevant methods is given and mappings of periodic variables to equivalent stationary variables are discussed. Two new periodic steady state monitors based on Short Time Fourier Transformation are proposed. The study was motivated by the need to identify the steady state condition of an aircraft electrical network for power quality checks. An implementation with Modelica tools is demonstrated.

Keywords: *periodic systems, steady state identification, wavelet, FFT*

1 Introduction

Testing of power quality criteria of electrical components and networks according to industrial standards, as (MIL-STD-704F,2004), often demands testing in settled condition. When the data is generated from a simulation of the physical system, at best, the system might be initialized in steady-state condition already. For non-linear switching and periodic systems this condition might not be found easily or only approximately from alternative representations, as in (Kuhn et al.,2012). In this case, the time-domain simulation of the system may converge to the exact periodic steady-state condition from a start condition, if the system is internally stable and well damped. The correct estimation of the convergence time becomes crucial if the evaluation of the quality criterion is part of a closed-loop optimization of the system itself. Then the time for simulation to reach steady-state condition, may affect the total time for the optimization process significantly. While the convergence rate may be known analytically for simple systems, generally this is not the case for arbitrary systems. This chapter shows practical methods for testing on the periodic steady-state condition of AC electrical circuits to reduce unnecessary simulation time. Input signals can be simulation results or measurements. It is assumed that the differential algebraic equation system or the loosely coupled subsys-

tem of interest is completely observable via the chosen output.

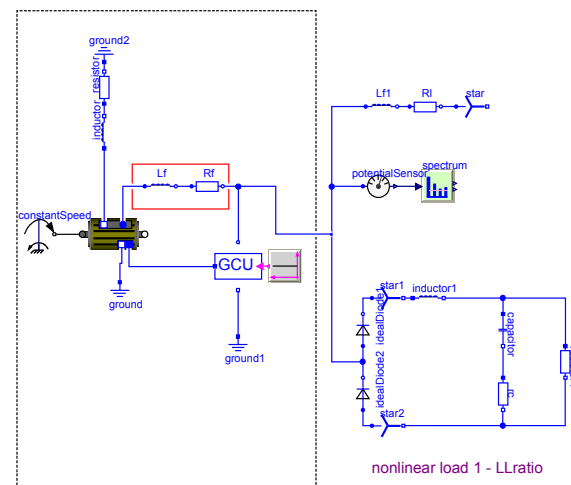


Figure 1: Simulation model for power quality of a small aircraft electrical network

To demonstrate the requirement, we will use the following example of a small aircraft electrical network in Figure 1. It was used as part in a loop of an industrial design process of a generator, whose design parameters are generated by a foregoing routine (Kuhn et al., 2012). The generator feeds a mixed AC resistive and DC 6-pulse switching load. The model simulates through an initial transient phase, till it reaches periodic steady-state. The design is then tested for conformance with industrial standards on power quality in the AC distribution line, which is found between the generator on the left and loads on the right. Power quality is tested via a Fast Fourier Transformation (FFT) Routine block.

The fast identification of periodic steady-state is of wider interest in simulation technique; for example for the non-linear transfer analysis in Saber (Saber,2016) or a similar Modelica-based tool (Bunte,2011). Both record the input/output behavior of a system, where the input is a frequency sweep signal. Its rate of change is limited in order to arrive -hopefully- in steady-state at the output. The sweep rate may need manual tuning for the specific condition, which may be circumvented by an automatic steady-state detection.

This paper focuses purely on the detection of the periodic steady-state of systems with output $x(t)$, which

can be represented by a superposition of band-restricted time-varying harmonic phasors $X_k(t)$ with base angular velocity ω_{base} .

$$x(t) \approx \sum_{k \in K} X_k(t) \cdot e^{j k \omega_{base} t}, K \in \mathbb{Z}, X \in \mathbb{C}, x \in \mathbb{R} \quad (1)$$

The time variant complex variable $X_k(t)$ is called dynamic phasor

$$X_k(t) = \frac{1}{T} \int_{t-T}^t x(\zeta) e^{-j k \omega_b \zeta} d\zeta \quad (2)$$

The mathematics can be found for example in (Demiray, 2008). An example of such a system with time-varying content around distinct frequencies is displayed in Figure 2.

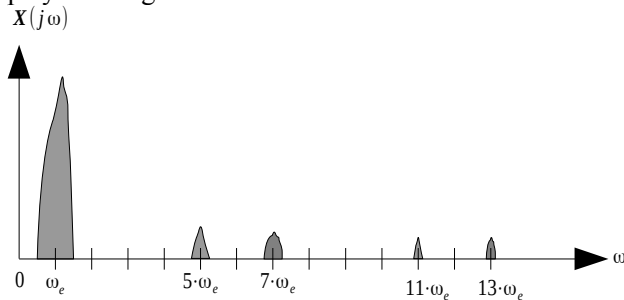


Figure 2: Fourier spectrum of nonstationary signal, with spectral content around $\sin(\omega_e)$, $\sin(\omega_e \cdot (6 \pm 1))$, $\sin(\omega_e \cdot (12 \pm 1))$

This paper is structured as follows: In the following section the difference between steady-state and periodic steady-state is highlighted. An overview on applicable methods for Steady State Identification is given in section 3. Section 4 discusses the transformation from periodic to non-periodic domain by pre-operators. The main theory of three selected methods for steady-state detection is presented and tested in section 5. This is followed by a conclusion. A theoretical investigation on parametrization of Discrete Fourier Transformation (DFT) for the purpose of Total Harmonic Distortion (THD)-based steady-state detection is given in the Appendix.

2 Steady state versus periodic Steady State Identification

In general, a time-variant system $F(x, \dot{x}, u) = 0$, excited by input u or autonomous, may show stable-stationary, unstable-stationary, stable-periodic, unstable-periodic or chaotic behavior of the state variables and possibly of the outputs. For non-linear systems, the system may bifurcate into several possible periodic steady-state conditions (Schupp, 2003). For linear differential algebraic systems, a steady-state detection mechanism may search for the condition

$$x(t) - x(t - \Delta t) = 0 \text{ or } \dot{x} = 0 \quad (3)$$

In practical applications only the detection of a minimum convergence rate $\dot{x} < \alpha_1$ may be feasible, since a longer duration of $\dot{x} = 0$ may not appear because of asymptotic convergence and/or additive noise. In the case of periodic systems, the steady-state definition has to be adapted. It is called a periodic steady-state condition, where consecutive cycles do not deviate, which means they have an auto-correlation of 1. This can be expressed by

$$x(\tau) - x(\tau - T) = 0 \quad \forall \tau \in [t - T, t] \quad (4)$$

, where the periodicity time constant T replaces the infinitesimal Δt in equation 3.

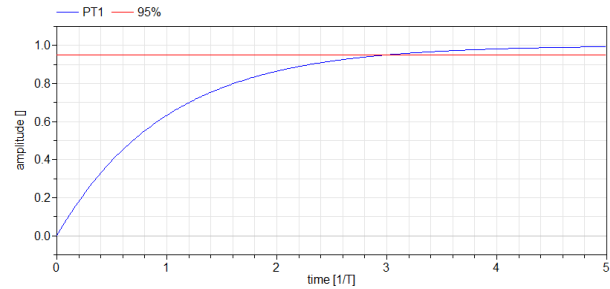


Figure 3: Transient of PT_1 system

To display the difference between steady-state and periodic steady-state, Figure 3 shows the output of a very basic first-order lowpass (PT_1) system, excited by a unit step at $t=0$. The system is asymptotically internal stable and converges to 1. An amplitude of 0.95 may be seen as quasi steady-state condition, appearing after 3 times characteristic time T .

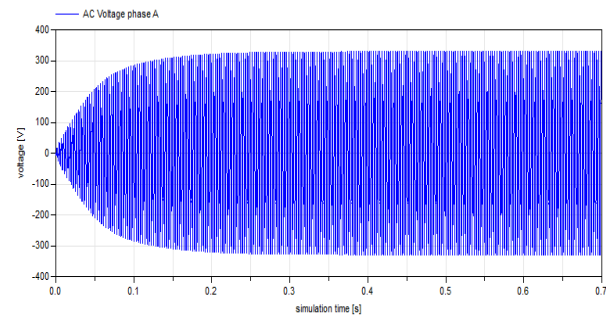


Figure 4: Transient of AC voltage of small aircraft electrical network example (original data)

In contrast to this, the transient of phase A of a three-phase AC voltage of the aircraft electrical network example is plotted in Figure 4. While it is oscillating, at the same time, it shows a first-order like transient behavior of the envelope.

3 Overview of methods

The process of signal-based steady-state detection has remarkable analogies with the theory of fault detection. The signal-based fault detection observes the behavior of a system on the change from its nominal (dynamic) behavior. Steady-state detection basically observes the behavior of a system on its change from past behavior.

They differ, as fault detectors generally are designed offline with specific fault data and models; absolute values on nominal or faulty conditions are known. Steady-state observers do not necessarily rely on detailed knowledge of the system. Isermann (Isermann,2006) classifies methods for single signal fault-detection into methods with “limit or trend checking”, and methods with “signal models”. “Limit and trend checking” methods are applicable for measurable absolute values or measures from statistical observers. Detection by “signal models” include correlation methods, spectrum analysis and wavelet analysis. Isermann (Isermann,2006) defines the basic steps of a scheme for fault detection with signal models, as preparation and transformation into “signal model“, extraction of relevant measures by “feature generation” and detection of faults, or by comparison to the nominal behavior in “change detection”.

Similar to it, steady-state detection can be separated into the steps:

- “Signal model” preparation, for periodic systems with removal of oscillation by an operator: The prepared signal can be any property in time domain, frequency domain or stochastic property.
- Application of test on steady-state: The test itself is based on the signal model.
- Decision making: the steady-state decision has to be made. It is very specific to the system, where noise and additional dynamics superpose the potential periodic system and the threshold has to be set based on prior knowledge.

(additive high-frequency noise is not correlated by definition, and should be filtered out from the original signal before, by low pass filtering)

For the steady-state detection, the following methods attracted attention in research in recent years:

The F-like test- developed first by Cao and Rhinehart (Cao and Rhinehart,1995) - belongs to the class of index-based change-detection methods¹. It relies on statistical methods to identify steady-state in noisy processes. It was tested and expanded on afterwards by Rhinehart for a multi-variable case (Brown and Rhinehart,2000). Applications included different processes, especially in chemical engineering. Other works by Kelly and Hedengren (Kelly and Hedengren,2013) concentrated on slow varying drifts in non-stationary processes with application to a windowed signal.

Wavelet transformation can be used to analyze characteristics of a specific system and match its specific out-

put patterns. Based on this, Jiang (Jiang et al.,2000) developed a method for identification of steps, peaks, noises, abnormal sudden changes and similar for chemical processes and reciprocally steady-state. The technique is not adapted to on-line steady-state detection. However, in an independent work, Korbel (Korbel et al.,2014) developed a steady-state identification for on-line reconciliation, based on wavelet transform and filtering for real-time data.

THD is a quality criterion, which is a measure of the distortion of a base oscillation through its harmonics (multiples). In case where industrial standards demand testing for a specific maximum THD, the criterion needs to be evaluated at periodic steady-state condition. When THD is evaluated repeatedly, observation of convergence of ΔTHD can be used as a direct indicator of the steady-state condition. This definition is industrially sufficient for the purpose of testing of THD. It was proposed in (Kuhn et al.,2015).

A further method for detecting steady-state is to use auto-regressive exogenous models with exogenous inputs (ARX). This method allows the SSI by system identification, where an auto-regressive model is tuned from the results of simulation or measurements. It is not based on detailed knowledge of the system equations. The identifiability of the system is checked where singularities in the model matrices appear in case of steady-state. Based on this singularity, an index is proposed (Rincón et al.,2015).

From these methods, the “F-like test”, wavelet-based test, THD-based test, and an adaptation of the THD-based test in frequency domain will be discussed in detail in the next sections. The first, due to its popularity and simplicity. The second, as a promising approach and to test the new Modelica Wavelet library. The THD criterion and the adapted frequency-based criterion is chosen, since it relies on the objective criterion directly. An overview is shown in Figure 5.

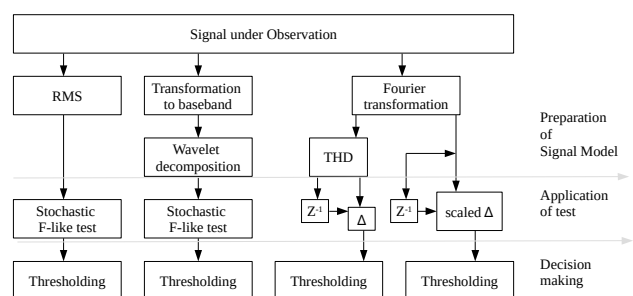


Figure 5: Overview on applied methods

All methods are tested for detection of steady-state on the small electric on-board network example from Figure 1.

¹ A “F-Test” is a detector of the change in variance

4 Mapping of periodic to non-periodic variables

For AC circuits, the method for steady-state testing has to be capable of detecting periodic steady-state. Either by itself, or the signal model preparation has to transform it to an oscillation-free measure. The problem can be overcome by mapping of the periodic signal to a non-cyclic equivalent and identification with standard methods. For a system of type (1), knowledge of a dominating, excited oscillation can be exploited, to identify the steady-state condition of the AC voltage signal. The signals main content is a modulation of a baseband signal x_{bb} and forced oscillation as

$$x(t) = \Re \{ x_{bb}(t) e^{j\omega_{base} t} \} \quad (5)$$

plus harmonic content at $k \cdot \omega_{base}$, plus uncorrelated noise. The minimum periodic cycle is the forced oscillation's time constant $T_{base} = 2\pi / \omega_{base}$, $2 \cdot T_{base}$ in case of additive odd harmonics, or arbitrary in case of non-harmonic content.

Equation (4) is not useful to implement, since the condition is only fulfilled for perfect congruence. Instead, it can be simplified by using a norm $\check{x}(t)$. The steady-state condition can be identified directly via $\check{x}(t) < \epsilon$ or via some more advanced methods on $\check{x}(t)$, listed next.

The test on steady-state can be seen as testing of the auto-regression of the signal, separated in intervals of length T . And it is similar to regression testing of two signals by the use of norms (e.g. (Pollok and Bender, 2014)). The **maximum error norm** of consecutive periods generates a periodic sampled one-dimensional output:

$$\check{x}_{me}[t] = \max \left(\frac{\|x(\tau) - x(\tau - T)\|}{\|x(\tau)\|} \right) \forall \tau \in [t - T .. t] \quad (6)$$

The norm is quite efficient, due to its simplicity. Since it is a norm on signal amplitude rather than energy, it will penalize sharp discontinuities and noise.

Similar to this and even more easy to implement, by a rough knowledge of the period, only peak values within consecutive periods can be selected. The signal corresponds to sample-and-hold of the peak values with sample period T . In aeronautical standards, this is often called the **“envelope”**:

$$\check{x}_e[t] = \frac{x(t - \hat{T} .. t) - x(t - 2\hat{T} .. t - T)}{x(t - \hat{T} .. t)} \quad (7)$$

Only one sample is gained within one interval at maximum or in case of application to the absolute value, an additional sample at minimum. Peak values may be prone to noise as some electronics, as rectifiers, add high portions of distortion to the high amplitude part of a voltage wave.

The **temporal** (time limited) **auto-correlation** treats not only minimum and maximum values, but all data of a period. It normalizes the signal to

$$\begin{aligned} x_{auto}^\vee[t] &= \frac{\int_{t-T}^t x(\tau - T) x^*(\tau) d\tau}{\left(\int_{t-2T}^{t-T} |x(\tau)|^2 d\tau \right)^{1/2} \left(\int_{t-T}^t |x(\tau)|^2 d\tau \right)^{1/2}} \\ &= \frac{\int_{t-T}^t x(\tau - T) x^*(\tau) d\tau}{\left(\int_{t-2T}^t |x(\tau)|^2 d\tau \right)^{1/2}} \end{aligned} \quad (8)$$

This norm is tolerant to noise and time shifts but highly prone to incorrect estimation on length of period T . The temporal auto-correlation measure is similar to the temporal auto-co-variance γ_{yy} of stochastic signals. It is common to think complex or unmodeled processes as stochastic processes (Oppenheim, 1999), which opens the field of stochastic data analysis for the problem. Other coherency metrics on spectrum, energy and time or phase-shift are listed in (Marple and Marino, 2004).

Alternatively the steady-state condition can be seen as the steady-state condition of the baseband signal. When the condition of a cycle is known exactly, it can be identified by one of the following methods:

AC coupled RMS (Root Mean Square): This method is best known for power supply networks at a fixed frequency of 50 or 60 Hz. It can be calculated as by MIL 704f, where RMS is the “value for one half-cycle measured between consecutive zero crossings of the fundamental frequency component”. Information on harmonic contents is lost by the integration.

$$X_{RMS} = \sqrt{\frac{1}{T} \int_0^T x(t)^2 dt} \quad (9)$$

When the phase angle θ is known, mathematical transformations to **phase-fixed reference system** can be applied (e.g. dq0/Park system or Fortescue transformation): For simulation, the phase angle is known. For real electrical systems, for single synchronous generator fed networks, it can be obtained by measuring a machines angular position. Without position measurement, the phase can be derived from the AC voltage by Phase Locked Loops (PLL). A PLL is a control circuit which generates an output signal in proportion to the phase difference of a reference signal to a measured signal. It can be used to adapt the frequency and phase of an observer to the measured signal (Krause et al., 2002).

Alternatively, the base band and harmonics can also be identified by **frequency selective filtering**: Signals can be analyzed in the spectral domain, where the base fre-

quency is usually associated with the spectral content of maximum amplitude. The frequency spectrum can be computed as the correlation of the signal with theoretically infinite sinusoidal waves at certain frequencies (Fourier transformation) or the correlation to finite wave packages at prevailing base frequencies (wavelet transformation (Mallat,2008)). For wavelet transform, one has to distinguish between direct application on sinusoidal signals and application on the pre-processed oscillation-free signal. For finite signals, the Fourier transformation is called Short Time Fourier Transformation, which can be implemented efficiently using Fast Fourier Transformation (FFT) (Cooley and Tukey,1965).

5 Implementation and validation of tests

In the following section, the selected theories of Steady State Identification are summarised and the steady-state monitors are tested through experiments.

5.1 F-like test

The F-like test, by Cao and Rhinehart (Cao and Rhinehart,1995) is based on statistical measures. The algorithm tests a signal on showing settled distribution at an associated level of significance. Possible distributions are uniform and Gaussian distribution. Measures are variance between data, moving average value and variance in the data itself. This method relies on sampled data.

The following steps can be implemented at low computational effort: First, the sampling vector is filtered by a filter factor of λ_1 .

$$X_f[i] = \lambda_1 X[i] + (1 - \lambda_1) \cdot X_f[i-1] \quad (10)$$

Where $X[i]$ are sampled data, $X_f[i]$ are filtered values and λ_1 is a filter factor. In the second step, a measure of the variance v_f^2 is computed with a moving average filter factor of λ_2 :

$$v_f[i]^2 = \lambda_2 (X[i] - X_f[i-1])^2 + (1 - \lambda_2) v_f[i-1]^2 \quad (11)$$

The unbiased estimate of the variance based on the filtered squared deviation from previous filtered values var_1 is given by:

$$var_1[i] = (2 - \lambda_1) \frac{v_f[i]^2}{2} \quad (12)$$

A measure on the second filtered variance estimate δ_f^2 is calculated based on the filtered square differences of successive data:

$$\delta_f[i]^2 = \lambda_3 (X[i] - X[i-1])^2 + (1 - \lambda_3) \delta_f[i-1]^2 \quad (13)$$

The formula includes a moving average filter with factor λ_3 . This second variance var_2 is given by:

$$var_2[i] = \frac{\delta_f[i]^2}{2} \quad (14)$$

Finally, the Steady State Identification index R is obtained as the ratio of the two variances:

$$R = \frac{(2 - \lambda_1) v_f[i]^2}{\delta_f[i]^2} \quad (15)$$

While R is a continuous measure, decision making needs tuning of a threshold R_t to distinguish between steady-state $R < R_t$ and non steady-state $R > R_t$. Filter values have to be tuned to match the time constants of the system under observation. Some more theoretical considerations on correct and incorrect identification of steady-state are given in (Cao and Rhinehart,1995), with respect to different types of error signals.

In a first trial, the F-like test was applied directly on the sinusoidal phase voltage. No useful results could be gained (not plotted), which can be explained by the strong correlation of the sinusoidal shaped signal. Therefore, isolation of the signal of interest had to be conducted first. For this example, simulation results did not show significant difference between several methods of RMS detection. Those are transformation by phase angle, integration over one period with start and end conditions identified by zero crossing detection, and peak-value detection. Figure 7 (top plot) shows the source signal of the test. The AC voltage is mostly settled after 0.1 seconds simulation time with an additional step of 10% at 0.3 seconds.

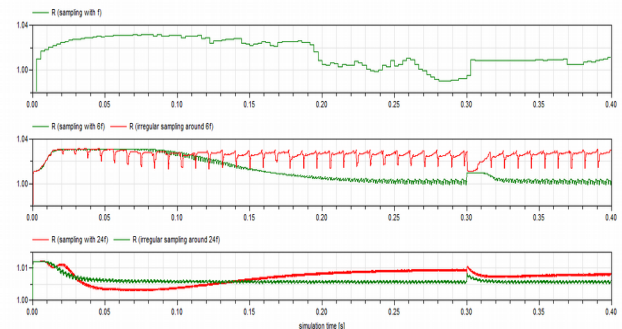


Figure 6: “R” index of F-like test for several sampling rates; input signal see Figure 7, top diagram

The influence of different types of sampling, and therefore different dominating noise on identification index R , can be seen in Figure 6. The plots present the results of the F-like test, applied on the same source data but with regular sampling intervals 6f and 24f, and irregular sampling around 6f and around 25f. The lambda factors were tuned manually.

It can be clearly seen that the quality of the results diverge on a significant scale. The results based on the 1f

sampling show some slow transient behavior, which is hard to interpret. In contrast, 6f and 24f sampling clearly identify changes. The R statistics show low pass behavior at steps of the input signal. A decision value on steady-state can be set but needs to deal with the chattering around the boundary value R_t .

5.2 Wavelet test

In wavelet analysis, the one-dimensional time variant input signal is decomposed into time variant subspaces with bandpass characteristics. By iterative wavelet multi-level decomposition, the original signal $f(t)$ is projected into a sequence of nested subspaces; each subspace is characteristic for a spectral content, similar to the indices of the Discrete Fourier Spectrum:

$$f(t) = \sum_{i \in I_j} c_{j,i} \varphi_{j,i} + \sum_{j=1}^J \sum_{k \in k_1} d_{j,k} \psi_{j,k} \quad (16)$$

The first sum represents low frequency content, while the right part represents higher frequency content. The wavelet spectrum originates from iterative bisection of the high-frequency signal up to scale J . $\psi_j(t)$ are scaled mother wavelets which define orthogonal spaces. Filtering of a signal corresponds to variation and limiting of its wavelet coefficients $c_{j,i}$ and $d_{j,k}$. Adaptive methods for filtering of Gaussian noise exist in many wavelet toolboxes. The filtered signal in the time domain can be restored by inverse transformation of the conditioned data. Formulas for discrete wavelet transformation are similar.²

Similar to the F-like test, this method needs separation of the fundamental of the amplitude-modulated wave first. While this can theoretically be done by an additional wavelet transformation, there is no benefit compared to the RMS method presented before. Next, the signal can be de-noised if necessary. Jiang (Jiang et al., 2003a) proposes to separate the baseband signal into the desired process trend $T(t)$ and process noise $N(t)$, by wavelet multi-level decomposition, filtering and reconstruction. Any other type of continuous or discrete filters may be used equivalently. Although the signal will suffer from a frequency dependent group delay by the filter, for steady-state detection, this can be seen as negligible compared to the typical time scales.

The wavelet-based detection itself uses the fact that a wavelet transform $Wf(t)$ of a signal $f(t)$ is proportional to the time derivative of the signal smoothed by the scaling function φ (see wavelet theory for details):

$$Wf(t) = 2 \frac{d}{dt} (f * \varphi)(t) \quad (17)$$

Furthermore, by the wavelet transform of the wavelet transform $WWf(t)$ one gets an analogon to the second-order derivative. Analogue to assumption of a

steady-state condition as a local extremum where first and second time derivative being zero, single and double wavelet transform can be applied. At a (local) minimum, the conditions

$$Wf(t) < \alpha_{w1}, \quad d(Wf(t))/dt < \alpha_{w2}. \quad (18)$$

must hold true. Similarly, for steady-state detection in the time domain, specific scaling of the α would be necessary. Where an ideal temporal derivative function is unspecific of the frequency and a Fast Fourier Transformation based spectral decomposition lacks information on the temporal variation, a wavelet can be adapted to the “characteristic scale”. This means, the frequency of the wavelet is chosen close to the characteristic response time τ of a system which acts as a kind of a bandpass filter. This can be realized by the sampling frequency directly, or iteratively by fragmentation into a wavelet spectrum with narrower bands of equation (16) which is called multi-resolution representation or alternatively Jiang (Jiang et al., 2000) calls it multi-scale process data analysis.

The steady-state index $\beta(t)$ is calculated from equations (19-21), where $\theta(t)$ is a factor of combined contributions from the first and second order wavelet and $\gamma(t)$ is an amplitude-limiting signal operator on the second order wavelet transform.

$$\theta(t) = |Wf(t)| + \gamma(WWf(t)) \quad (19)$$

$$\gamma(WWf(t)) = \begin{cases} 0 & , |WWf| \leq T \\ (|WWf| - T_w) / 2 \cdot T_w & , |WWf| \in [T_w, T_u] \\ 1 & , |WWf| \geq 3 \cdot T_w \end{cases} \quad (20)$$

β itself calculates as a threshold comparator from the contributions factor $\theta(t)$, with smoothed transient from 0 to 1.

$$\beta(t) = \begin{cases} 0 & , \theta(t) \geq T_u \\ \frac{1}{2} \left[\cos \left(\frac{\theta(t) - T_s}{T_u - T_s} \cdot \pi \right) + 1 \right] & , T_s < \theta(t) < T_u \\ 1 & , \theta(t) \leq T_s \end{cases} \quad (21)$$

Where T_s = standard deviation of Wf , $T_u = 3 \cdot T_s$, T_w = median (WWf). In β , “zero” indicates unstable status and “one” steady-state condition. For details, see (Jiang et al., 2003b) and for advanced end-of-steady-state-detection see (Korbel et al., 2014).

(Jiang et al., 2003b) demonstrates steady-state detection but does not focus on online implementation. It may look straightforward to perform the analysis continuously on a window of past samples. Practical implementations for this thesis showed the correct choice of the limits T_s , T_u and T_w often fails when considering only one window. The median especially moves quite arbitrarily. Therefore, limits are calculated non-causally by using the full data set. This proves the con-

² For background on wavelet analysis, one may see Debnath (Debnath and Shah, 2002), section “Wavelet bases and Multiresolution Analysis”.

siderations of Korbel (Korbel et al., 2014) who proposes to choose the limits from past measurements.

To implement the wavelet test, the Modelica wavelet library (Gao et al., 2014) was used. The library is similar to MATLAB's wavelet toolbox. Since the Modelica wavelet library does not support online computation yet, this study is an offline demonstration only. The library can be developed further for online computation, if issues regarding the initialization of buffers, data storage and allocation of vector sizes of intermediate variables are solved. Furthermore, the plotting relies on Dymola-specific Modelica scripting.

The test makes use of the interpolation routine, definition of a wavelet function and the discrete wavelet transform:

```
Wavelet.General.interpL()
Wavelet.Families.wavFunc(Wavelet.Records.waveletDefinition());
Wavelet.Transform.dwt();
```

Results of the test are displayed in Figure 7: From the original signal, the RMS value is calculated via Park transformation using generator angular information. The RMS value is processed by first and second order wavelet transformation. They show a clear relationship to the temporal derivatives. β is calculated via formulas 19-21. The first steady-state condition is detected at around 0.05 seconds. This assumption is based on the limits T_s , T_u and T_w and may be changed by different settings.

In summary, the wavelet-based method identifies the steady-state condition of the base harmonic well for the example. The signal can not be processed directly but has to be transformed to a non-periodic representation (RMS). The time scale for the wavelet transform and the limits need to be adapted to the model, based on known prior results. The computational efficiency has to be questioned critically for the wavelet transform. It may be improved in a future, real-time capable implementation of the Modelica library, by use of fast wavelet algorithms.

5.3 Discrete Fourier transformation based THD criterion

In (Kuhn et al., 2015) a Total Harmonic Distortion based steady-state detector was proposed. Its "signal model" relies on the Fourier spectrum. According to (Isermann, 2006), Fourier spectra are well suited for identification of periodic, stochastic, and non-stationary properties, and therefore for periodic Steady State Identification.

In a first step, a vector of sampled data of the input signal is decomposed into a discrete amplitude-frequency spectrum by a short time Discrete Fourier Transform algorithm. THD is calculated from the spectrum by ³

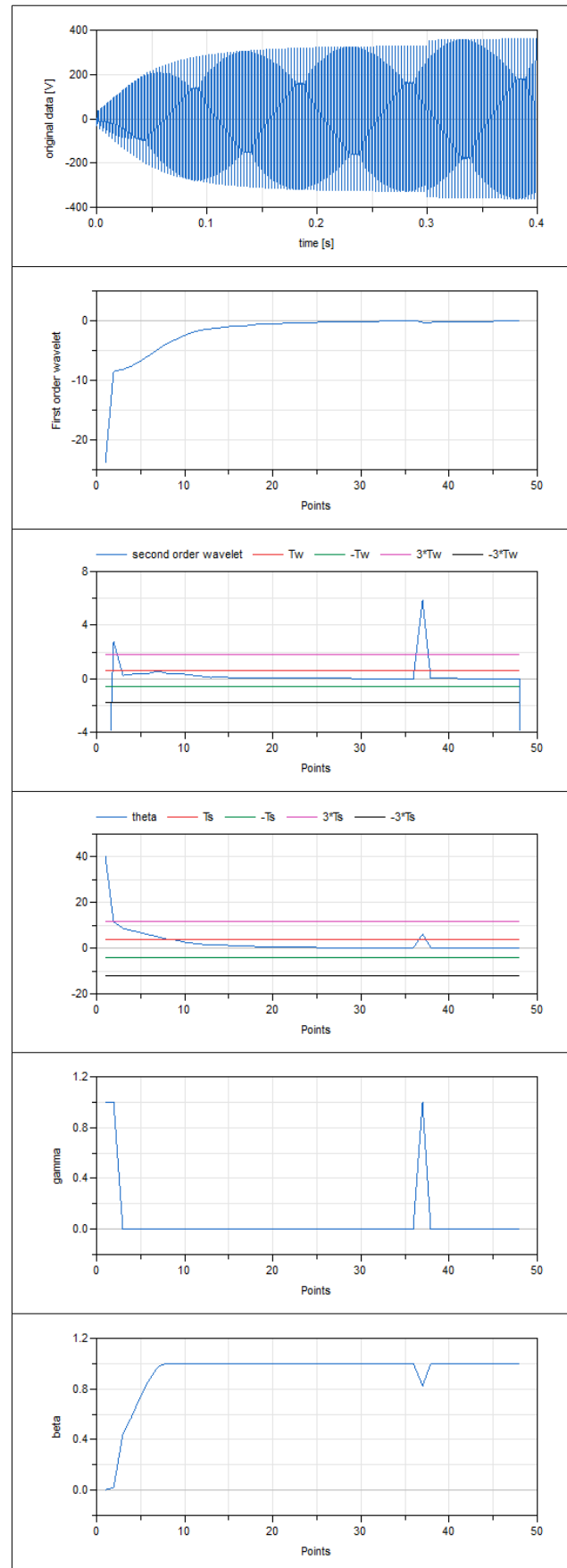


Figure 7: Wavelet based test, results

³ e.g., IEEE Standard 519-2014 (IEEE, 2014)

$$THD = \sqrt{\frac{\sum_{h=2}^M A[h \cdot f_{base}]^2}{A[f_{base}]^2}} \quad (22)$$

It is a one-dimensional norm on the $M-1$ amplitudes of the harmonics, which is normalized by the amplitude of the base frequency. The phase information and DC component is not considered.

Finally, the steady-state test is designed as a normalized “trend checking”, based on ΔTHD :

$$y_{THD} = \frac{|THD(t_x) - THD(t_x - N \cdot p)|}{\max(THD, \epsilon)} \quad (23)$$

?

$$y_{THD} < \delta \Rightarrow \text{steady state}$$

The criterion relies on consecutive evaluations of the spectra at times t_x and $t_x + \Delta T$. Each evaluation is based on data sets of length $N[s]$. The time delay between the two THD windows is defined in proportion p of the data set length, where an overlap of 50% is proposed for the data sets. Theoretical considerations are derived in section 7. ϵ prevents division by zero and influence of noise at small values of THD . While THD could be evaluated at every sampling interval, for efficiency reasons, the Modelica algorithm is only evaluated every $N \cdot p$. When the frequency resolution is set to $1/r \cdot f_{base}$, the total data set length is $(1+p) \cdot r \cdot T_{base}$ and evaluated no later than $p \cdot r \cdot T_{base}$ after an event. Example: $r=4$ $p=1/2 \rightarrow$ criterion evaluation not later than in $2 \cdot T_{base}$, based on data set length = $6 \cdot T_{base}$.

The main features of the implementation by the Modelica block “WithinAbsoluteFFTDomain_THD” were already discussed in (Kuhn et al., 2015). It is a big advantage of this method, that the AC signal can be taken directly as an input. There is no need for pre-processing as RMS or transformation to base band. While the expected base frequency should be given roughly, the Modelica-based algorithm can tune itself to the dominating peak in the nearby-spectrum. Also, the block features the option to use the criterion as an indicator for termination of simulation; the THD is delivered as a final result at this steady-state condition. No extra FFT computation is necessary for this, as the computa-

tion of THD and THD-based steady-state criterion rely on the same FFT data.

The THD-based criterion was tested with the small grid example. Here, the criterion could NOT identify steady-state condition. This shortcoming can be better understood from the plot of the THD in relation to V_{rms} in Figure 8, rather than the criterion itself.

As can be seen in the upper plot, the THD is not correlated with the main trend, even at steps. This is a special property of the small grid example. There exist higher harmonics because of the rectifier, but they are in fixed proportion to the base harmonic with fast and well damped filter dynamics on the DC side. Therefore, normalization of THD by the base amplitude prevents a change of the criterion in this case. Furthermore it can be proven easily, it gives the same THD if a Δ on one harmonic amplitude compensates for the amplitude on other.

$$THD(t_1) = \sqrt{\frac{A_1^2 + A_2^2 + \dots}{A_b^2}} \quad (24)$$

$$= THD(t_2) = \sqrt{\frac{(A_1 + \Delta_1)^2 + (A_2 - \Delta_2)^2 + \dots}{A_b^2}}$$

With proper choice of Δ_1 and Δ_2 . Strictly speaking, for the THD identification according to industrial standards, no “real” steady-state condition would be necessary here, as the THD does not change. But since it is not a proper indicator, it is not generally recommended. But it can be adapted to overcome the obstacles as shown next.

Adapted discrete Fourier Transformation-based criterion

In order to overcome the problems of the THD-based steady-state monitor, the new “THD-similar” criterion is proposed:

$$y_{THD\text{similar}} = \max \left(\frac{|A[h \cdot f_{base}[t_x]]|^2 - |A[h \cdot f_{base}[t_x + \Delta T]]|^2}{|A[f_{base}(t_x)]|^2 + \epsilon \cdot |A_{nom}|^2} \right) \quad (25)$$

?

$$y_{THD\text{similar}} < \delta \Rightarrow \text{steady state}$$

It is also based on the DFT spectrum and is inspired by the THD criterion, maximum error norm and variation in base amplitude. In contrast to THD, also the first (=base) harmonic is considered. An educated guess of a factor ϵ of the nominal base amplitude A_{nom} prevents division by zero and smooths the result. The decision threshold δ has to be set based on knowledge from past results.

The criterion and parameterization of FFT is discussed in detail in section 7. It is shown, that this criterion is well suited for identification of steady-state of dynamic

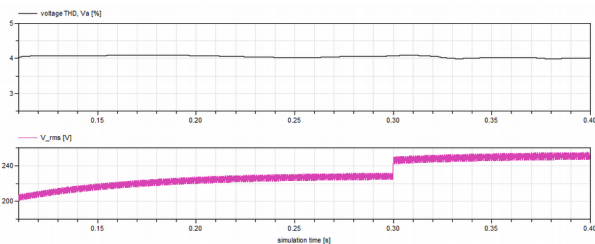


Figure 8: Investigation on spectrum: THD vs. signal (FFT window 0.017s)

systems (1), where unmodeled dynamics are treated as uniform noise.

The implementation of the function is based on Within-AbsoluteFFTdomain_THD, with the same parameters unless stated otherwise. It shares the benefits with the THD approach, with little simulation overhead through the efficient FFT algorithm. As soon as steady-state is detected, the test on conformance with the standards on THD can be performed. The quality test is based on the same FFT data without need for an additional FFT calculation. Results analogue to Figure 6 are shown in Figure 9.

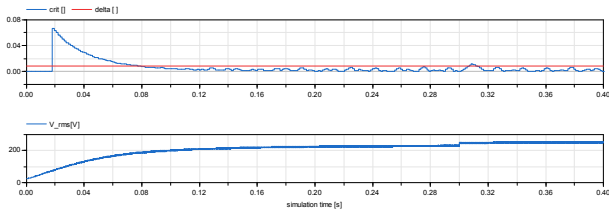


Figure 9: Investigation on new steady-state criterion vs. signal (FFT window 0.017s)

It can be seen that the steady-state condition is found reliably, with proper detection of the initial transient period. The change in amplitude at 0.3 seconds is detected shortly after the event.

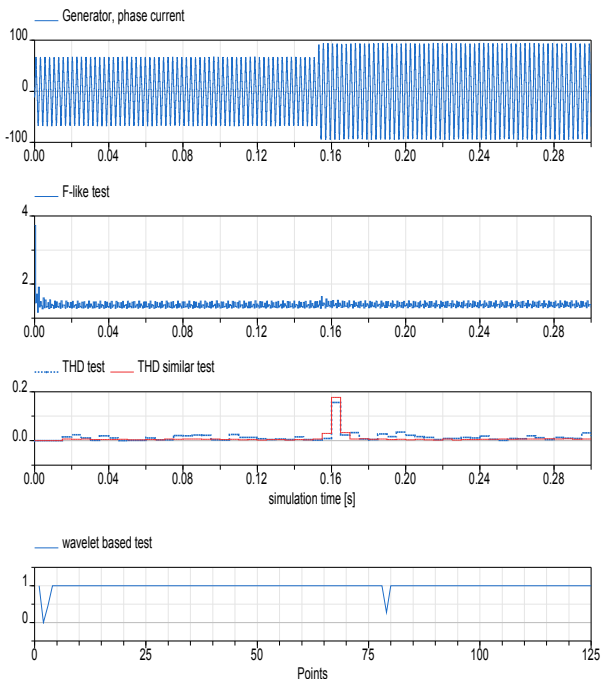


Figure 10: Identification of steady-state, based on real test data

Lastly, all methods are tested with an example based on hardware tests. The top plot in Figure 10 shows the measurement data of a generator connected to an electrical-driven Wing Ice Protection System (WIPS). The load is increased at 0.15seconds. It can be seen that the F-like criterion detects the event, but the output is noisy although care was taken for proper parameteriza-

tion. In contrast to this, the beta parameter of wavelet-based test and THD and THD-similar criterion detect the event reliably, with high signal-to-noise ratio.

6 Conclusion

In this paper, procedures for Steady State Identification were tested with an AC electrical circuit, with dominant main amplitude and harmonic distortion, and a second example. Both methods from literature demand a mapping of the periodic to non-periodic signals. The F-like test showed good performance and short delays. However, it was difficult to parameterize, and detection was weak. The wavelet-based test was very successful, but computational overhead and delay is high. Alternatively, an experiment based on a variation of THD was tested. The monitor can treat the periodic signal directly, at medium computational overhead. The delay is high but it can be seen as not critical, since evaluation of THD in steady state is requested. This criterion was not able to detect a transient period, where the signal had a fixed ratio of the base amplitude and harmonics. The THD-similar criterion was designed to also consider the base. Tests were very promising, at medium efficiency and medium delay. Due to its generality and efficiency, this method is proposed as the best choice for the application. The results are summarized in Table 1. Any generalization of the methods demands an investigation with more examples.

Test	Quality of SSI for the examples	Pre-operator needed for AC	Delay	Computation Efficiency
F-like	bad	yes	Short	high
Wavelet based	Very good	yes	high	low
THD criterion	Only partial	no	Medium-high	Medium, low if THD is needed
THD-similar criterion	good	no	Medium-high	Medium, low if THD is needed

Table 1: Evaluation matrix of proposed methods

Acknowledgements

Some preliminary studies were performed together with Mr. Mohamed Jmari, who did an internship at DLR as part of his studies at ENSMM, Besançon, France.

Valuable input was given by K. Chong.

7 Appendix: parametrization of FFT and derived measures for detection of steady-state condition

The following section builds on the results in (Kuhn et al., 2015) and goes into deeper discussion on the parameterization of the Discrete Fourier Transformation needed for THD evaluation, and their influence on steady-state detection. Use of Discrete and Fast Fourier Transformation itself is not discussed here but (Kuhn et

al.,2015) gave a practical approach to the generation of Fourier spectra in Modelica.⁴ Computation of FFT might sound numerically demanding, but efficient routines are available as public domain software, or as proprietary software down to chip-optimized routines from Intel and AMD. Cyclic FFT can evaluate the FFT at each sampling step, where results from earlier computations can be reused rather than freshly computed. For practical reasons, one may not evaluate the FFT and THD at every sample, since the convergence of the signal may happen within some AC periods, but not within some sampling intervals.

The FFT algorithm for use by THD calculation is well parameterized by

- The expected base frequency f_{base} and number of harmonics demanded $n_{harmonics}$; the maximum frequency in the spectrum $f_{max,FFT}$ needs to be well above the highest treated harmonic: $f_{max,FFT} > n_{harmonics} \cdot f_{base}$, where sample frequency $f_s = 1/T_s = 2 \cdot f_{max,FFT}$
- The type of window function, (e.g. rectangular, Hamming or Butterworth)
- The window length $N = n_s \cdot T_s = 1/f_{resolution}$, with the spectral resolution $f_{resolution}$ and the number of sample points n_s

(proper anti-aliasing by a low-pass filter is assumed).

“Windows” transfer the theoretical unlimited data set to finite length by selection of N samples, where the signal is multiplied by the window function before DFT. Such a window function starts near or at zero, then increases smoothly to a maximum at the center of the time series and decreases again (see Figure 11 for a Hamming window). The theory of DFT implicitly postulates that the input is periodic, where any waveform must repeat itself after the window of sampled signals. This means, for signals with sinusoidal content, the Fourier spectra of temporal consecutive windows coincide: if the windows are of length $l_p \cdot T_p$, and time shifted by $r_p \cdot T_p$; with arbitrary integer numbers l_p and r_p , and wavelength T_p for each sinusoidal content p .

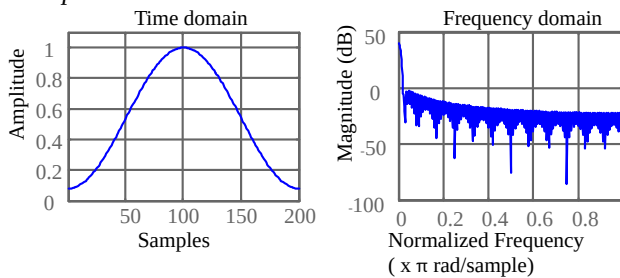


Figure 11: Hamming window

⁴ In the meantime the underlying FFT algorithm found its way into the Modelica standard library 3.2.2 as tool independent implementation “Modelica.Math.FastFourierTransform.realFFT()”

In the following, the properties of the spectral analysis are discussed with the purpose of steady-state identification. For better understanding, Figure 12 shows two spectra of the voltage transient of the small aircraft electrical network example: The amplitudes spectrum on the initial transient phase (red) differs from the spectrum of the settled phase (blue) in amplitude and distinctiveness of the peak (a sinusoidal oscillation of infinite length would result in a distinct Dirac impulse). The example shows that the spectra clearly differ and can be used for distinction of steady-state and non steady-state.

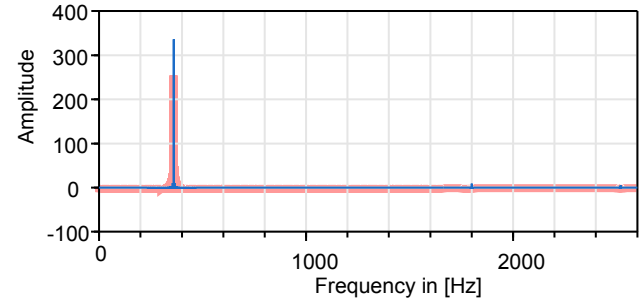


Figure 12: Voltage spectrum of small aircraft electrical network example, with dominant signal $\sin(2 \cdot \pi \cdot 360 \text{ Hz})$; blue: spectrum of period $[0..0.2s]$, red: spectrum of period $[0.2..0.4s]$,

The spectrum can be affected by:

- a) Smearing of peaks, from non-periodicity (energy conservation by Parseval’s theorem) or mismatch of period by window length,
- b) Spectral leakage, from convolution of the spectrum X by the window’s spectrum W
- c) Band restricted variation and smearing of peaks, from unmodeled dynamics

Case a) might be used as an indicator for the variation of the wavelengths, where non-integer l_p distort the spectrum. This is not recommended. The exact finding of the wavelength or phase information is highly prone to errors. Instead, the discontinuity can be removed by application of a non-rectangular window (Henning etc.)

Case b) can be seen a requirement on the shape and length of the window function. For better understanding, the effect of windowing is demonstrated in Figure 13. It shows the windowing the input signal X (grey peaks) by “rectangular” window (blue) and a “flattop” window (green). The width of the window in frequency domain is indirectly proportional to its length in time domain. The window type itself is characterized by the peak flatness (3dB bandwidth) and peak level of the sidelobes (see overview of window types in (Heinzel et al.,2002)).

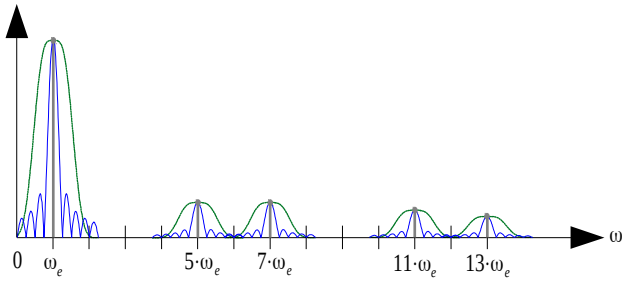


Figure 13: Influence of windowing and sampling

$\mathbf{X}(j\omega)$ (grey): Dirac peaks in continuous Fourier domain, e.g. from sine and cosine

$\mathbf{Y}(j\omega)$ (blue): convolution of “rectangular” window with $\mathbf{X}(j\omega)$

$\mathbf{Y}(j\omega)$ (green): convolution of “flattop” window with $\mathbf{X}(j\omega)$

The convolution of the window with the signal in frequency domain is:

$$Y(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\theta}) W(e^{j(\omega-\theta)}) d\theta \quad (26)$$

The following requirements result, to distinguish tight steady-state and wider non steady-state spectra X of a signal of type (1):

1) The DFT has to resolve the individual base-band signals of the spectrum, without overlapping caused by the window; (e.g. in Figure 13, the adjacent blue wave packs shall not merge). The window type and length $n_s \cdot T_s$ have to be chosen with focus on their broadening and height of sidelobes.

2) The steady-state and non-steady-state condition in the basebands of X , need to have distinguishable amplitudes in discrete Y as well: The discretization of (26) at a given sampling rate f_s results in

$$Y[k] = Y(e^{j\omega}) \Big|_{\omega_k = (2\pi/N)k} = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\theta}) W(e^{j(\omega_k - \theta)}) d\theta \Big|_{\omega_k = (2\pi/N)k}, \quad k=0..N-1 \quad (27)$$

2) is similar to 1), but includes a further demand: $Y[k]$ may not be undifferentiated for the shapes of X with the same local area c :

$$\int_{(2\pi/N)(k-1/2)}^{(2\pi/N)(k+1/2)} X(e^{j\omega}) d\omega = c \quad (28)$$

. This can be the case with flat top windows which makes them not favourable for the purpose of identification of band restricted disturbances: They exhibit broad peaks, with 3dBwidths starting from 2.9 bins. This gives them an approximate characteristics of $W(e^{j\omega})=1$ in the interval around a discretized angular velocity $\omega_k = (2\pi/N)k \pm 1/2$ (called “bin”). This certainly has benefits for the correct identification of

amplitudes, in case of a frequency mismatch of signal and discretized frequency; but overlapping and visibility of narrow banded effects had to be prevented by high spectral resolution and therefore be paid by large window lengths.

Case c) is by wide the most interesting effect. Steady-state identification can be based on prior knowledge, with measures from a single spectrum. Measures are the amplitudes of the main peaks, or their (3dB) widths, or their amplitude to width ratio, or the ratios of the main peaks. Or it can also be based on the temporal change of these measures. For this work, we assume there is little information on the spectrum given. Furthermore, there is no need to tune the algorithm for a special spectral shape, since any distinct change is seen as non-periodic condition. Instead, a measure is proposed based on the variation of the noise from unmodeled dynamics.

In the 1970s a method called “Welch’s method of averaging modified periodograms” was developed to improve the accuracy of periodograms. Periodograms are estimates of the spectral density of a signal. In this context, “modified” means the window is not of type “rectangular”. According to (Oppenheim and Schaffer, 1998), the estimate r , of a sequence of K periodograms is given by

$$I_r(\omega) = \frac{1}{NU} |Y_r(e^{j\omega})|^2 \quad (29)$$

, where estimates are based on non-overlapping data segments of length N , which are taken from a total data set length Q by a window. The correction factor U normalizes the amplitudes of the windows (if not already included in Y_r):

$$U = \frac{1}{L} \sum_{n=0}^{N-1} (w[n])^2 \quad (30)$$

Averaging of the K estimates results in the averaged periodogram

$$\bar{I}(\omega) = \frac{1}{K} \sum_{r=0}^{K-1} I_r(\omega) \quad (31)$$

, respectively

$$\bar{I}[k] = \frac{1}{K} \sum_{r=0}^{K-1} \frac{1}{NU} |Y_r[k]|^2 \quad (32)$$

of a discrete spectrum. In case the properties of the signal remain stationary, and noise is additional and uniformly distributed, the variance of $\bar{I}(\omega)$ is reduced by a factor of $1/K$ (Heinzel et al., 2002). Welch (Welch, 1967) proved that other types of windows may be used with similar reduction in variance (modified periodogram). Also he found, that HALF-overlapped windows (see Figure 14) reduce the variation in the spectral components approximately by an additional factor of 2, if this increases the number of windows on the data. More than 50% overlap usually gives no additional benefit, since the cross-correlation of the windows grows. Detailed considerations on the optimal

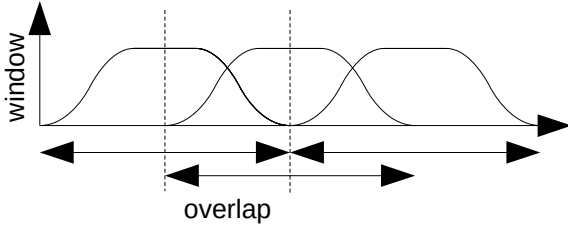


Figure 14: Segmented signal, with three windows and 50% overlap

usage of the information in relation to window overlap, are summarized in (Heinzel et al., 2002). The author lists 33 types of windows with amplitude flatness and power flatness in relation to overlap correlation. Results clearly show, that an overlap of 50% is a good choice for all windows except the “flat top” windows. By Welch’s method it is possible to get better, unbiased estimates of the spectrum, and therefore better inputs for THD calculation. Additionally, with the data of the periodogram (32), the standard deviation of the estimate can be computed with little extra effort. It is possible to construct an F-like test upon these measures, where transition to steady state can be associated with decreasing noise, and therefore decreasing standard deviation. This is not recommended since the large data vector would effect a substantial delay in the steady state detection. Instead, an indicator named ‘randomness’ (Heinzel et al., 2002). is more applicable. It is the ratio of the standard deviation to the averaged estimate of the signal, that dominates the frequency bin under consideration. “Randomness” is near unity for stochastic signals such as noise, and small for coherent signals such as sinusoidal wave:

$$'randomness' = \frac{\sigma(\bar{I}[k])}{E(\bar{I}[k])} \quad (33)$$

This “randomness” criterion is proposed as base of the THD-similar steady-state detector. Since reduction of delay is of highest interest, the set of input data must be kept short. This directly results in a number of 2 windows, with an overlap of 50% (more windows might be used to filter noise). The choice of only two windows transforms the $\sigma(\bar{I}[k])$ operator into a $\Delta(I[k])$ operator. $\Delta(I[k])$ is evaluated per “bin” $[k]$. Since any variation can be seen as “non steady-state”, it is sufficient to map the data vector to a single value by a maximum norm. (Euclidean norm might work as well, with smoother output). The criterion can be made less prone to noise if the variances $[k]$ are normalized by the expectation value of the main amplitude, rather than the expectation value $[k]$. This results in

$$y_{randomness} = \max \left(\frac{|A[k[t_x]]|^2 - |A[k[t_x + \Delta T]]|^2}{|A[k_{base}(t_x)]|^2} \right) \forall k \quad (34)$$

The SSI delay needs to be kept to a minimum, where delay is proportional to the window length, which in turn is proportional to the resolution of the DFT spectrum. The minimum delay is attained, when each band restricted variation is included by one bin each, but the window is not “flat top” whilst the resolution is high enough to prevent overlap with the adjacent harmonic by the window. Since we assume that all non-steady-state-caused distortion is centered around the base-frequency and the harmonics, the set of all k in criterion (34) can be limited to all bins which represent a harmonic of f_{base} . With the usual notation of expressing the number of the bins by their equivalent frequency, the k s in (34) are replaced by $k = h \cdot f_{base}$, with $h = [1..M]$. Inserting an additional ϵ in the denominator to prevent division by zero and influence of noise directly results in

$$y_{THD-similar} = \max \left(\frac{|A[h \cdot f_{base}[t_x]]|^2 - |A[h \cdot f_{base}[t_x + \Delta T]]|^2}{|A[f_{base}(t_x)]|^2 + \epsilon \cdot |A_{nom}|^2} \right) \quad (35)$$

$\forall h \in [1..M]$

The windows of type Bartlett, Hamming or Hanning are especially recommended due to their small 3dB peak width of 1.2736, 1.3008 and 1.4382 bins. For these windows the resolution of the spectrum should be at least $1/3 \cdot f_{base}$ to prevent overlap.

References

- Bünthe, T.. Recording of Model Frequency Responses and Describing Functions in Modelica, *Proceedings of the 8th International Modelica Conference*, 2011.
- Brown, P. R. and Rhinehart, R. R.. Demonstration of a method for automated steady-state identification in multivariable systems. *Hydrocarbon processing* 79:79-83, 2000.
- Cao, S. and Rhinehart, R. R.. An efficient method for on-line identification of steady state, 5:363 - 374, 1995.
- Cooley, J. W. and Tukey, J. W.. An algorithm for the machine calculation of complex Fourier series. *Math. Comput.* 19:297-301, 1965.
- Debnath, L. and Shah, F. A.. *Wavelet transforms and their applications*. : Springer, 2002.
- Demiray, T.. *Simulation of Power System Dynamics using Dynamic Phasor Models*. Ph.D. thesis, ETH Zurich, 2008.
- Gao, J., Ji, Y., Bals, J. and Kennel, R.. Wavelet library for Modelica, *Proceedings of the 10th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, 2014.
- Heinzel, G., Rüdiger, A., and Schilling, R.. *Spectrum and spectral density estimation by the Discrete Fourier transform (DFT), including a comprehensive list of window functions and some new at-top windows...*, online. http://www.rssd.esa.int/SP/LISAPATHFINDER/docs/Data_Analysis/GH_FFT.pdf, 2002.
- IEEE. Recommended Practice and Requirements for Harmonic Control in Electric Power Systems. *IEEE Std 519-2014 (Revision of IEEE Std 519-1992)* :1-29, 2014.
- Isermann, R.. *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer (Ed.). : Springer Science & Business Media, 2006.
- Jiang, T., Chen, B. and He, X.. Industrial application of Wavelet Transform to the on-line prediction of side draw qualities of crude unit. *Computers & Chemical Engineering* 24:507-512, 2000.
- Jiang, T., Chen, B., He, X. and Stuart, P.. Application of steady-state detection method based on wavelet transform. *Computers & Chemical Engineering* 27:569 - 578, 2003b.
- Kelly, J. D. and Hedengren, J. D.. A steady-state detection (SSD) algorithm to detect non-stationary drifts in processes. *Journal of Process Control* 23:326-331, 2013.
- Korbel, M., Bellec, S., Jiang, T. and Stuart, P.. Steady state identification for on-line data reconciliation based on wavelet transform and filtering. *Computers & Chemical Engineering* 63:206-218, 2014.
- Krause, P. C., Wasynczuk, O. and Sudhoff, S. D.. *Analysis of Electric Machinery and Drive Systems*. : WileyBlackwell, 2002.
- Kuhn, M. R., Otter, M. and Giese, T.. Model Based Specifications in Aircraft Systems Design, *11th international Modelica Conference*, 2015.
- Kuhn, M. R., Rekik, M. and Bals, J.. Modelling and Use of an Aircraft Electrical Network Simulation for Harmonics Consideration in Generator Design, *SAE Technical Paper*, 2012.
- Mallat, S.. *A wavelet tour of signal processing: the sparse way*. : Academic press, 2008.
- Marple, S. L. and Marino, C.. Coherence in signal processing: a fundamental redefinition, *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on* 1:1035-1038 Vol.1, 2004.
- US Department of Defense. *Aircraft electric power characteristic*. <http://www.wbdg.org/ccb/FEDMIL/std704f.pdf>, 2004.
- Oppenheim, A. V.. *Discrete-time signal processing*. : Pearson Education India, 1999.
- Oppenheim, A. V. and Schaffer, R. W.. *Zeitdiskrete Signalverarbeitung*. : Oldenbourg Wissenschaftsverlag, 1998.
- Pollok, A. and Bender, D.. Using Multi-objective Optimization to Balance System-level Model Complexity, *Proceedings of the 6th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools* :69-78, 2014.
- Saber. *Integrated Environment for Physical Modeling and Simulation*. Synopsys, I., www.synopsys.com/prototyping/saber, 2016.
- Schupp, G.. *Numerische Verzweigungsanalyse mit Anwendungen auf Rad-Schiene-Systeme*. Ph.D. thesis, Universität Stuttgart, 2003.
- Welch, P.. The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics* 15:70-73, 1967.

Discrete-time models for control applications with FMI

Rüdiger Franke¹ Sven Erik Mattsson² Martin Otter³ Karl Wernersson² Hans Olsson²
Lennart Ochel⁴ Torsten Blochwitz⁵

¹ABB, ruediger.franke@de.abb.com, ³DLR, martin.otter@dlr.de,

²Dassault Systèmes, {svenerik.mattsson, karl.wernersson, hans.olsson}@3ds.com,

⁴Uni Linköping, lennart.ochel@liu.se, ⁵ESI ITI, torsten.blochwitz@esi-group.com

Abstract

The paper proposes an extension of FMI 2.0 for the rigorous treatment of discrete-time models. This includes the introduction of discrete-time states, the declaration of clocks in the model description and an extension of the calling interface for the external activation of clocks by an importing environment.

The synchronous discrete-time extension enables for the first time the synchronization of FMUs with the environment and with other FMUs. It specializes the existing generic event mechanism of FMI 2.0 and maps to synchronous features of Modelica.

Discrete-time FMUs are needed for the generation of controller code from functional models. This paper outlines different use cases, including a simple PI controller, feed forward control with a nonlinear inverse model and nonlinear model predictive control.

The FMI change proposal FCP-001 and the Modelica change proposal MCP-0024 describe the proposed extensions in more detail. Test implementations exist in the simulation tools Dymola and OpenModelica and in the importing optimization solver HQP. The use cases given in this paper served for further refinement of the change proposals and the test implementations.

Keywords: Modelica, Synchronous modeling, Inline Integration, Model-based Control, Nonlinear Inverse Model, Feed Forward Control, NMPC.

1 Introduction

Control systems are composed of interconnected control blocks that must synchronize with each other and with real time. This requires precise time event handling and discrete states.

Modelica 3.3 extends the scope from a language primarily intended for physical systems modeling to modeling of complete systems. In particular, new synchronous language primitives were introduced for increased correctness of control systems implementation (Elmqvist et al, 2012).

Version 2.0 of the FMI standard omitted precise time event handling. The design was considered complicated at the time of the release of FMI 2.0 since several aspects have to be considered (Blochwitz et al, 2012):

- The synchronous features of Modelica 3.3 should be supported.
- FMI should also be useable by tools that do not support synchronous time event handling.
- The time event handling is to be defined in a way that allows backward compatible extensions.

This paper discusses the progress made recently. The work resulted in a new version of the FMI change proposal FCP-001 (Otter et al, 2016) and in the Modelica change proposal MCP-0024 (Franke, 2016). This paper summarizes the change proposals, provides use cases and investigates examples using test implementations in the simulation tools Dymola and OpenModelica and in the optimization solver HQP (Franke and Arnold, 1997).

2 Synchronous Modelica

Modelica has always supported continuous-time variables and discrete-time variables defined as piecewise continuous and piecewise constant functions of time, respectively. Both may change discontinuously at time instants, so called events. Events are treated at runtime.

The synchronous features of Modelica 3.3 introduce a new Clock type. Clock variables $c(t^k)$ are special discrete variables that are active (are ticking) at particular time instants, see Figure 1.

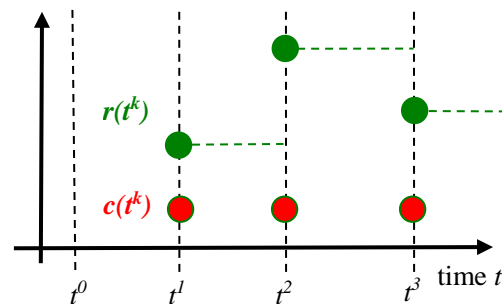


Figure 1: Clock variable c and clocked variable r

A clocked discrete-time variable $r(t^k)$ is associated with exactly one clock. This enables the partitioning of a model into sub-models for each clock at translation time. A clock defined for one variable of a partition automatically propagates to all other variables of this partition. This enables generic discrete-time models with inferred sample times.

A clocked discrete-time variable only has a value when the clock ticks. Continuous-time variables may be converted to clocked variables with the sample operator. A clocked variable may be converted to a continuous-time variable with the hold operator.

A clocked partition is mathematically defined as:

$$\begin{aligned} x^k &= f^k(x^{k-1}, u^k, t^k), & k &= 0, 1, 2, \dots, k_{end} - 1, \\ x^{-1} &= x_{start} \end{aligned} \quad (1)$$

$$y^k = g^k(x^{k-1}, u^k, t^k), \quad k = 0, 1, 2, \dots, k_{end} \quad (2)$$

Here x^k are discrete-time states, u^k are inputs, y^k are outputs, k is the k -th tick of the associated clock and k_{end} is the final tick. The discrete-time states are defined with difference equations as function f^k of the previous values x^{k-1} and the inputs u^k .

2.1 Clocked continuous-time models

A clocked partition may contain differential equations. This allows the embedding of regular continuous-time models from given Modelica libraries. The Modelica translator brings the equations of a clocked partition to the form of an ODE or semi-explicit index-1 DAE:

$$\begin{aligned} \frac{dx(t)}{dt} &= f[x(t), u(t)] \\ 0 &= h[x(t), u(t)] \end{aligned} \quad (3)$$

The translator then applies a specified solver method to convert continuous-time differential equations to discrete-time difference equations. This mixed symbolic/numeric approach is also known as inline integration (Elmqvist et al, 1995).

Basic solver methods are implicit Euler, explicit Euler and semi-implicit Euler. Application of implicit Euler results in:

$$\begin{aligned} \frac{x^k - x^{k-1}}{t^k - t^{k-1}} &= \text{if } k = 0 \text{ then } 0 \text{ else } f(x^k, u^k) \\ 0 &= h(x^k, u^k) \end{aligned} \quad (4)$$

Explicit Euler avoids the implicit equation system for the states x^k in (4) for non-stiff models. It results in:

$$\begin{aligned} \frac{x^k - x^{k-1}}{t^k - t^{k-1}} &= \text{if } k = 0 \text{ then } 0 \text{ else } f(x^{k-1}, u^{k-1}) \\ 0 &= h(x^{k-1}, u^{k-1}) \end{aligned} \quad (5)$$

The use of u^{k-1} in (5) leads to the introduction of additional discrete-time states for the delay of inputs by one sample period, even though this is typically not wanted. Semi-implicit Euler avoids the delay of inputs and implicit dependencies of states for non-stiff models. It results in:

$$\begin{aligned} \frac{x^k - x^{k-1}}{t^k - t^{k-1}} &= \text{if } k = 0 \text{ then } 0 \text{ else } f(x^{k-1}, u^k) \\ 0 &= h(x^{k-1}, u^k) \end{aligned} \quad (6)$$

Many more solver methods exist with specific advantages and drawbacks. The choice of the best solver method depends on the model at hand. This is why it is advantageous that inline integration embeds the most appropriate solver method into an exported model.

Modelica 3.3 defines the operators `previous(x)` to access x^{k-1} and `interval()` to determine $t^k - t^{k-1}$.

The Modelica change proposal MCP-0024 introduces the operator `firstTick()` to determine if $k = 0$ (Franke, 2016).

3 FMI extension

FMI 2.0 defines a generic event mechanism that also covers synchronous models. The drawbacks of this generic mechanism are that discrete states are hidden in the FMU and that the environment does not know any details about the events. This makes it impossible to synchronize events with the environment of an FMU. Thus, it is not possible to re-import an exported FMU with synchronous discrete-time features and achieve a deterministic behavior. Neither it is possible to exploit a discrete-time FMU for advanced applications such as parameter estimation or model predictive control, because the discrete states are hidden.

It is proposed to extend FMI by the following:

1. Declare clocks in `modelDescription.xml`
2. Declare discrete-time states in `modelDescription.xml`
3. Let the environment activate clocks in order to enable synchronization with the environment and with other FMUs.

This extension is optional. A model can always hide event details according to FMI 2.0.

3.1 Extension of `modelDescription.xml`

The “TypeDefinitions” section is extended with a “Clocks” subsection that contains one or more “Clock” entries.

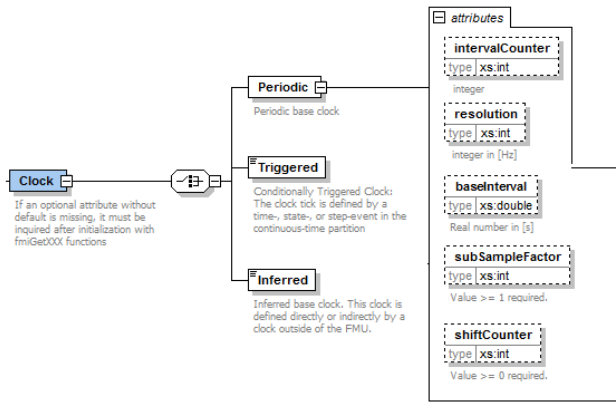


Figure 2: Kinds of Clock

Each Clock may be one of (see Figure 2):

- **Periodic:** the clock ticks periodically with an a priori known interval specified in the model description XML file. A priori known values make the sampling a structural model property for increased correctness at runtime.
- **Triggered:** the clock is activated by a Boolean condition in the model, e.g. for an interval that depends on model variables.
- **Inferred:** the clock is activated from outside the model, e.g. for a generic discrete-time model with arbitrary sample interval. Synchronous models do not require a parameter for the sample time; the clock propagates with clocked variables. Synchronous Modelica models use the interval operator instead of a parameter.

The attributes of Periodic define the clock interval and offset time. The basic clock interval is either specified with a double valued `baseInterval` or with integer valued `intervalCounter` and `resolution`. Both definitions relate to each other with

$$\text{baseInterval} = \text{intervalCounter} / \text{resolution}$$

Periodic clocks may be further refined with the attributes `subSampleFactor` and `shiftCounter`. This results in the actual

$$\text{interval} = \text{baseInterval} / \text{subSampleFactor}$$

that is delayed by

$$\text{offsetTime} = \text{interval} * \text{shiftCounter}$$

The attributes of “ScalarVariable” are extended with two new attributes:

- **previous** marks a discrete-time state, similar to the derivative attribute of continuous-time states. The value is an index to the variable providing the previous value of the discrete-time state.
- **clockIndex** associating a variable uniquely with a clock in the “Clocks” section.

Finally, the “ModelStructure” section is extended with a subsection “DiscreteStates”. It provides an ordered list of all exposed discrete states with their indices in the “ScalarVariable” list. Each entry of “DiscreteStates” may declare the dependencies from known inputs, continuous-time states and other discrete-time states. The dependencies are defined under the assumption that the respective clock ticks.

3.2 Extension of the C calling API

The C calling API is extended with four new functions that can be called during the event mode of an FMU.

A clock is activated by the environment for the current time instant by the function `fmi2SetClock`, and the status of a clock can be queried with the function `fmi2GetClock`:

```
fmi2Status fmi2SetClock (
    fmi2Component c,
    const fmi2Integer clockIndex[],
    size_t nClockIndex,
    const fmi2Boolean tick[],
    const fmi2Boolean* subactive);
```

Set the clock activation status by providing the indices of the corresponding clocks with respect to the xml element “<TypeDefinitions><Clocks>” and values. A clock is activated at the current time instant if `tick[i] = fmi2True`, otherwise the clock is deactivated. The environment may set `subactive[i] = fmi2True` to only evaluate the output equations (2) and replace the state equations (1) with

$$x^k = x^{k-1} \quad (7)$$

This is similar to the treatment of clocked continuous states at initial time, see (4), (5) and (6). The argument `subactive[i]` defaults to `fmi2False` if a NULL pointer is passed.

```
fmi2Status fmi2GetClock (
    fmi2Component c,
    const fmi2Integer clockIndex[],
    size_t nClockIndex,
    fmi2Boolean tick[]);
```

Query whether a set of clocks is active by providing the indices of the corresponding clocks with respect to the xml element “<TypeDefinitions><Clocks>”.

A clock interval is set by the environment for the current time instant by the function `fmi2SetInterval`, and it can be queried with the function `fmi2GetInterval`:


```
fmi2Status fmi2SetInterval(
    fmi2Component c,
    const fmi2Integer clockIndex[],
    size_t nClockIndex,
    const fmi2Real interval[]);
```

Set the interval value between the previous and the present tick of the clock.

```
fmi2Status fmi2GetInterval(
    fmi2Component c,
    const fmi2Integer clockIndex[],
    size_t nClockIndex,
    fmi2Real interval[]);
```

Query the interval value for the provided clocks (periodic or non-periodic). If the clocks are non-periodic, the interval has to be queried at every clock tick, to define the follow-up clock tick.

3.3 Extension of importing environment

The importing environment parses the model description XML file and activates periodic and inferred clocks during simulation. It activates periodic clocks at sample intervals specified in the model description XML file. It activates inferred clocks as needed by the environment (e.g. with an externally specified sample interval or if the clock of a connected FMU ticks). The FMU itself activates Triggered clocks.

This extension does not change the overall calling sequence of C functions for model exchange. The environment calls the new API functions additionally during event mode as follows:

0. **Enter event mode:**
FMI 2.0 enters the event mode either after initialization (call to function `fmi2ExitInitializationMode`) or during simulation (call to function `fmi2EnterEventMode`).
1. **Activate clocks and set inferred intervals:**
An FMU activates triggered clocks itself. The environment may query the clock activation status with the function `fmi2GetClock`. The environment sets the activation status of periodic and inferred clocks by calling `fmi2SetClock`. Moreover, the environment calls `fmi2SetInterval` for inferred clocks. It may query the clock interval, e.g. for triggered clocks, with the function `fmi2GetInterval`.
2. **Evaluate clocked equations:**
The evaluation is triggered by `fmi2GetXXX` for clocked variables during event mode or by

`fmi2NewDiscreteStates`. The FMU copies x^k to x^{k-1} and evaluates the discrete-time equations, updating x^k , if the corresponding clock is active. The FMU resets the clock activation after one evaluation. This means that the environment must activate the clock again if it wants to re-evaluate clocked equations, for instance to treat an algebraic loop (see below)

3. **Leave event mode:**

The functions `fmi2NewDiscreteStates` and `fmi2Reset` leave event mode and deactivate all clocks.

The environment might need to evaluate clocked discrete-time equations multiple times at one time instant, for instance to iteratively solve an algebraic loop among multiple connected FMUs or to calculate partial derivatives for optimization. The environment can either call `fmi2GetXXX` within event mode, triggering the evaluation of clocked equations if the respective clocks are active. The FMU will update discrete-time states and deactivate the clocks. The environment may reset discrete-time states by calling `fmi2SetXXX`, re-activate clocks and call `fmi2GetXXX` again for multiple evaluations. This also applies to all kinds of clocks, including also triggered clocks. Alternatively, the environment may enter event mode multiple times and reset discrete-time states for multiple evaluations.

The environment might be interested in the dependencies of model outputs from inputs and given discrete-time states, independently of the state equations. This can be achieved by passing `subactive=fmi2True` to `fmi2SetClock`.

3.4 Relation to Simulink S-functions

The basic concept of the proposed FMI extension is well known from other simulation technologies. The widely used simulation tool Simulink, for example, supports an arbitrary number of discrete sample times in an S-function, in addition to continuous-time equations. Lacking an XML file, the sample times are defined in S-function methods (C functions). The most important methods are listed here and related to the proposed FMI extension.

mdlInitializeSizes(SimStruct *S)

This method declares the number of sample times with

```
ssSetNumSampleTimes(S, n);
```

It corresponds to the number of Clock entries in the model description XML file.

mdlInitializeSampleTimes(SimStruct *S)

This method initializes each sample time $i = 0, \dots, n-1$ with an interval and an offset time by calling

```
ssSetSampleTime(S, i, interval);
ssSetOffsetTime(S, i, offsetTime);
```

The argument `interval` may take the special values `CONTINUOUS_SAMPLE_TIME` for a continuous-time model and `INHERITED_SAMPLE_TIME`, corresponding to an inferred sample time of this proposal.

Moreover, the argument `interval` may take the special value `VARIABLE_SAMPLE_TIME` and the argument `offsetTime` may take the special value `FIXED_IN_MINOR_STEP_OFFSET`, relating discrete-time sub-models to numerical integration steps of continuous-time sub-models. Such sampling can be implemented with triggered clocks of this proposal, if the FMU activates clocks itself during transitions between continuous-time mode and event mode.

Simulink will activate any sample time from outside S-functions in the case of sample hits and call the function

`mdlUpdate(SimStruct *S, int_T tid)`

A model must query the activation status and evaluate the respective discrete-time equations.

```
if (ssIsSampleHit(S, i, tid)) {
    // update discrete states that belong
    // to sample time i
}
```

Discrete states are accessed with

```
real_T *x = ssGetRealDiscStates(S);
```

This FMI proposal uses variable references to access discrete states. It introduces optional previous values for discrete-time states. Previous values allow the definition of dependencies on x^{k-1} in the model structure, see (1), (2). The environment only sets the actual value x^k . An FMU with previous values copies x^k to x^{k-1} prior to the evaluation of clocked equations.

4 Use Cases

This section lists use cases for control applications. A chemical process model serves as an example.

4.1 Exemplary chemical process model

We consider a continuous stirred-tank reactor (CSTR) with cooling jacket published by (Engell, Klatt, 1993). This highly nonlinear model exhibits interesting properties, like nonminimum phase behavior and change of steady-state gain at the main operating point. (Chen et al, 1995) propose this example as a benchmark problem for nonlinear control system design.

The following reaction describes the chemical process:



The reactor primarily transforms cyclopentadiene (substance A) to the product cyclopentenol (substance B). An unwanted subsequent reaction transforms B to cyclopentanediol (substance C). Another unwanted parallel reaction transforms A to the by-product dicyclopentadiene (substance D). The mathematical model contains the component balances for A and B:

$$\begin{aligned} \frac{dc_A}{dt} &= \frac{\dot{V}_F}{V_R} (c_{A,F} - c_A) - k_1(T)c_A - k_3(T)c_A^2 \\ \frac{dc_B}{dt} &= -\frac{\dot{V}_F}{V_R} c_B + k_1(T)c_A - k_2(T)c_B \end{aligned} \quad (9)$$

with the reaction coefficients

$$k_i(T) = k_{i,0} e^{\frac{E_i}{T}}, \quad i = 1, 2, 3 \quad (10)$$

as well as the energy balances for the reactor and the cooling jacket:

$$\begin{aligned} \frac{dT}{dt} &= \frac{\dot{V}_F}{V_R} (T_F - T) + \frac{k_w A_R}{\rho C_p V_R} (T_K - T) \\ &\quad - \frac{1}{\rho C_p} [k_1(T)c_A H_1 + k_2(T)c_B H_2 + k_3(T)c_A^2 H_3] \\ \frac{dT_K}{dt} &= \frac{1}{m_K C_{p,K}} [\dot{Q}_K + k_w A_R (T - T_K)] \end{aligned} \quad (11)$$

Table 1 lists the model parameters.

Table 1: Parameters of CSTR model

Na me	Value	Description
$k_{1,0}$	1.287 h^{-1}	Collision factor one
$k_{2,0}$	1.287 h^{-1}	Collision factor two
$k_{3,0}$	$9.043 (\text{molA h})^{-1}$	Collision factor three
E_1	-9758.3 K	Activation energy one
E_2	-9758.3 K	Activation energy two
E_3	-8560 K	Activation energy three
H_1	4.2 kJ/molA	Reaction enthalpy one
H_2	-11.0 kJ/molB	Reaction enthalpy two
H_3	-41.85 kJ/molC	Reaction enthalpy three
ρ	0.9342 kg/l	Density reactant
C_p	3.01 kJ/(kg K)	Heat capacity reactant
k_w	$1.12 \text{ kW/(m}^2 \text{ K)}$	Heat transfer jacket
A_R	0.215 m^2	Surface reactor
V_R	0.01 m^3	Volume reactor
m_K	5.0 kg	Mass cooling jacket
$C_{p,K}$	2.0 kJ/(kg K)	Heat capacity coolant

Table 2: Desired steady operating point

Name	Value	Description
$c_{A,F}$	5.10 molA/l	Feed concentration
T_F	104.9 °C	Feed temperature
\dot{V}_F/V_R	14.19 h ⁻¹	Feed flow rate
Q_K	-1113.5 kJ/h	Heat removal
c_A	2.14 mol/l	Concentration A
c_B	1.09 mol/l	Concentration B
T	114.2 °C	Reactor temperature
T_K	112.9 °C	Coolant temperature

Table 2 gives the desired operating point for optimal yield. The following subsections use this CSTR model to outline different use cases.

4.2 Functional Engineering

Modelica system models combine physical plant models with control models. This enables the study the functional behavior of a system with simulation. Having a functional model available, the actual controller code shall be generated automatically from the control models.

Figure 3 shows a system model with a CSTR and a PI control for the coolant temperature. The PI controller uses a clock and sample blocks from the Modelica_Synchronous library (Otter et al, 2012). The clock also defines the solver method ImplicitEuler to convert the controller model to discrete time.

The control task is to hold the coolant temperature at the desired operating point, in order to keep the desired concentration of product B.

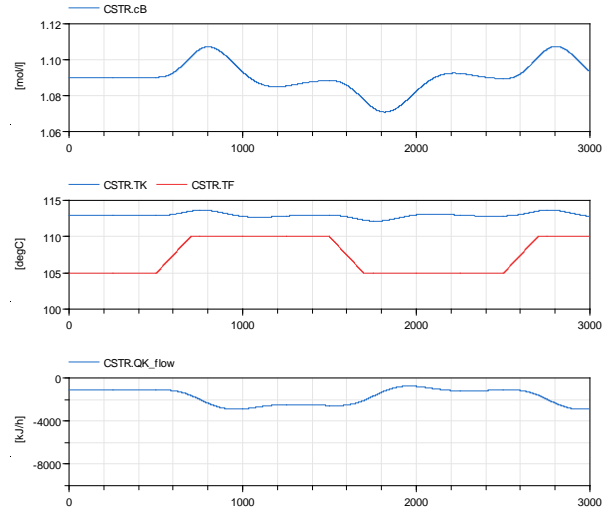
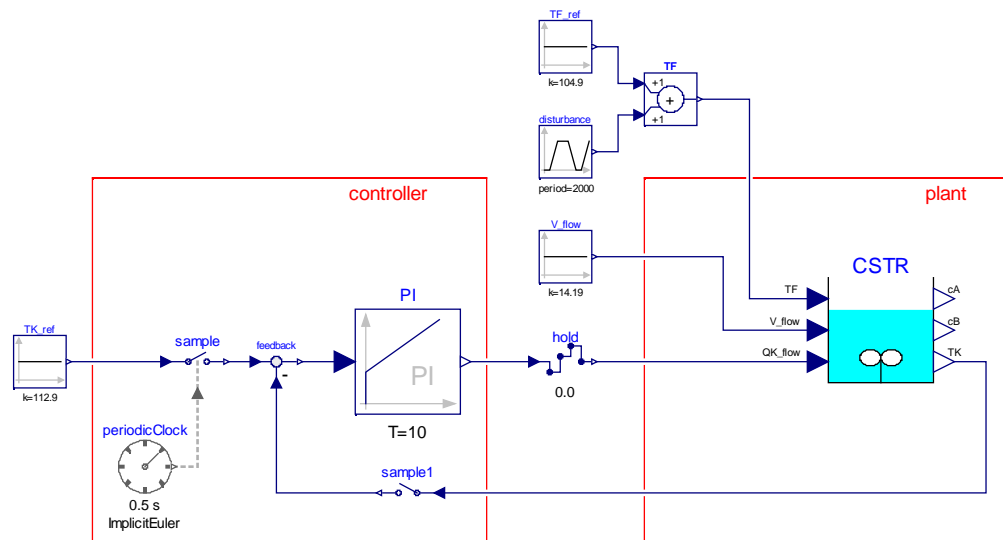
**Figure 3: Simulation results for the functional model with plant and controller over 3000s**

Figure 6 shows simulation results. The feed temperature CSTR.TF is increased periodically by 5 K. This results in higher reactor temperature and increased concentration CSTR.cB. The PI controller increases heat removal to bring the reactor back to the desired operating point.

Overall the disturbance leads to large deviations of the concentration of the product CSTR.cB from the desired operating point of 1.09 mol/l. This is because the controller sees the disturbance only indirectly if the coolant temperature increases. Moreover the reference value of the coolant temperature is not adjusted to the disturbance.

**Figure 4: Functional model of a plant with controller**

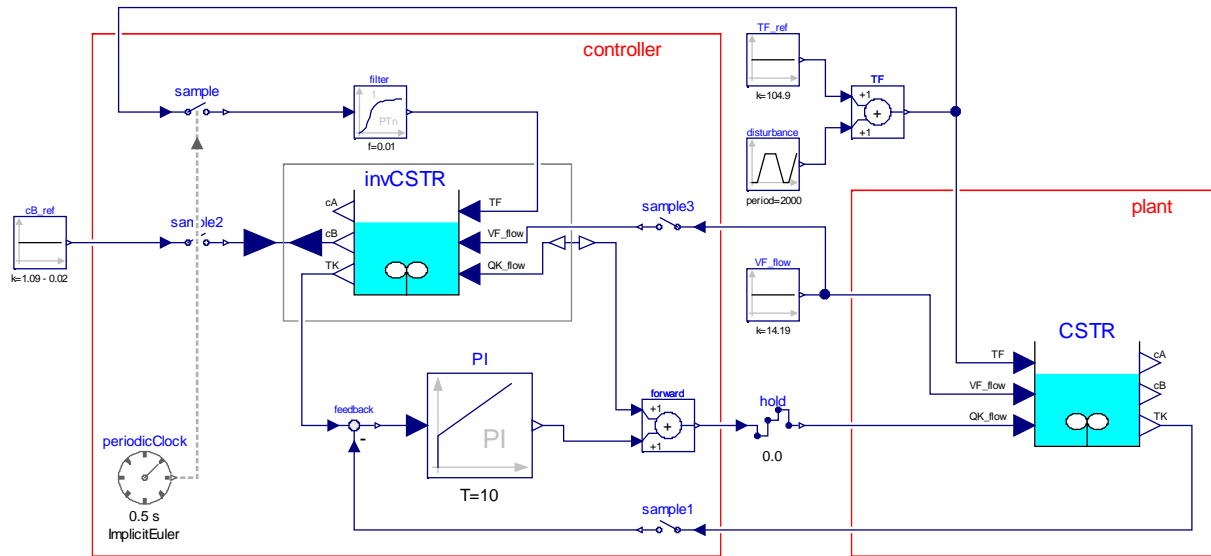


Figure 5: Functional model with advanced controller containing a nonlinear inverse plant model

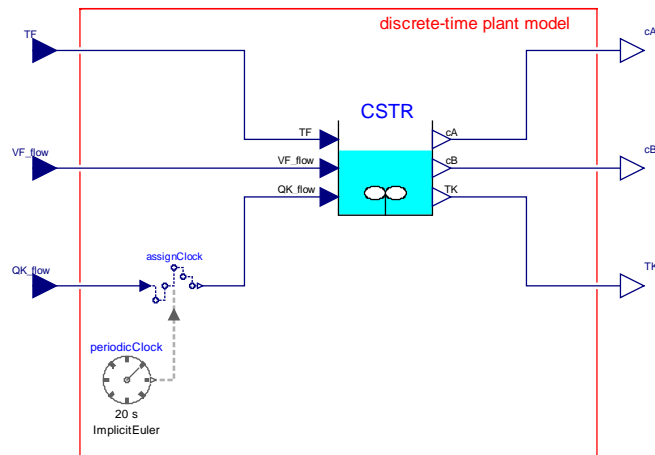


Figure 6: Discrete-time plant model for nonlinear model predictive control

4.3 Nonlinear inverse models for control

Feed forward is a well-known strategy to increase dynamic control performance. Modelica can invert a physical plant model analytically to get an inverse model for the feed forward path of a controller (Looye et al, 2005).

Figure 4 shows an advanced controller with nonlinear inverse model. This increases controller performance for disturbance rejection by converting feed temperature to an appropriate set point for heat removal and reference point for the coolant temperature TK_{ref} . Moreover, this enhances the controller with an external set point for the concentration of the product B.

Figure 7 shows simulation results. During the first 1000 s the controller adjusts the heat removal for the modified reference cB_{ref} of 1.07 mol/l. Afterwards the disturbance in the feed temperature is rejected considerably better with feed forward control.

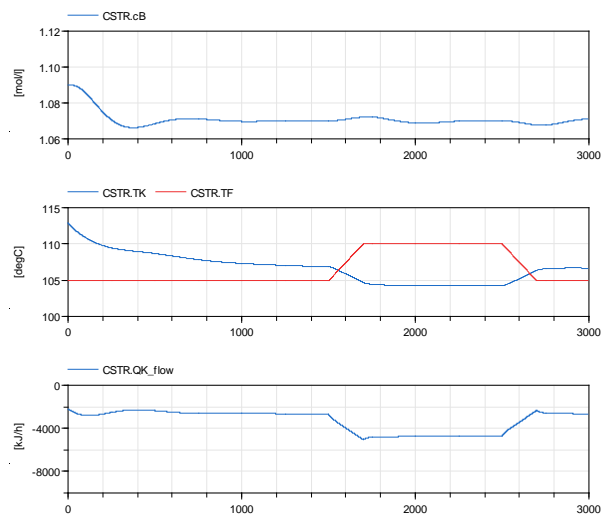


Figure 7: Simulation results for feed forward control with inverse plant model over 3000s

4.4 Discrete-time plant models for nonlinear model predictive control

Nonlinear model predictive control (NMPC) treats an optimal control problem for a given plant model at runtime. The model is used as is, without analytical inversion. This simplifies the treatment of multi-variable constrained problems at the cost of increased computing requirements for numerical optimization at runtime. A model predictive controller takes the following steps during each cycle (Franke et al, 2015):

1. Convert continuous-time physical model to discrete-time model for control.
2. Calculate model sensitivities.
3. Formulate a large-scale nonlinear optimization program spanning multiple time steps.
4. Solve the large-scale nonlinear optimization program.

The synchronous features of Modelica and the discrete-time extension of FMI enable to shift steps 1 and 2 from the runtime to model translation time. Figure 5 shows the CSTR model with clock and solver method assigned.

The resulting exported FMU has the discrete-time states $x = (c_A; c_B; T; T_K)$, the inputs $u = (\dot{Q}_K; T_F; \frac{\dot{V}_F}{V_R})$ and the outputs $y = (c_A; c_B; T_K)$. The control task is formulated as discrete-time optimal control problem over the time horizon of 3000s with $k_{end} = 150$ intervals of length 20s. The optimization objective is to minimize quadratic deviations of the concentration of substance B from the desired operating point. A second objective term applies a small penalty to control moves:

$$J = \sum_{k=0}^{k_{end}} (c_B^k - 1.07)^2 + \sum_{k=0}^{k_{end}-1} \left(\frac{\dot{Q}_K^{k+1} - \dot{Q}_K^k}{10^7} \right)^2$$

$$\rightarrow \min_{\dot{Q}_K^k}$$
(12)

The manipulated extraction of heat is constrained by

$$-9000 \text{ kJ/h} < \dot{Q}_K^k < 0 \text{ kJ/h}, \quad k = 0, \dots, k_{end} - 1$$
(13)

The discrete-time state equations in the FMU define further constraints:

$$x^{k+1} = f^k(x^k, u^k), \quad k = 0, \dots, k_{end} - 1$$

$$x^0 = (2.14; 1.09; 114.2; 112.9)$$
(14)

The solver HQP collects all states and the control inputs of all time intervals into one large vector of optimization variables

$$v = (x^0, u^0, x^1, u^1, \dots, x^{k_{end}-1}, u^{k_{end}-1}, x^{k_{end}}).$$
(15)

This results in the large-scale mathematical program

$$\begin{aligned} J(v) &\rightarrow \min_v & J: \mathbb{R}^n &\rightarrow \mathbb{R}^1 \\ h(v) &= 0 & h: \mathbb{R}^n &\rightarrow \mathbb{R}^{m_e} \\ g(v) &\geq 0 & g: \mathbb{R}^n &\rightarrow \mathbb{R}^m \end{aligned}$$
(16)

with $n = \dim(v)$, $m_e = (k_{end} + 1)\dim(x)$ and $m = 2k_{end}$. HQP applies Sequential Quadratic Programming (SQP) with a sparse Interior Point QP solver to the numerical solution of the mathematical program.

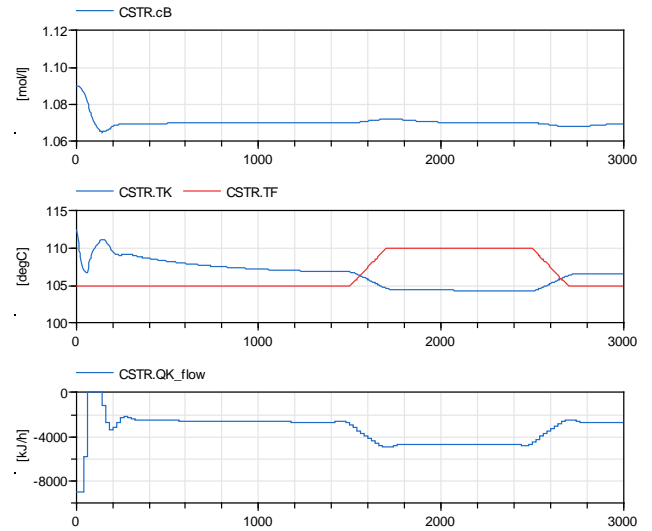


Figure 8: Results of the optimal control problem over a time horizon of 3000s

Figure 8 shows simulation results of the CSTR model for the optimized control trajectory 0 kJ/h. The optimal solution exploits the full range between -9000 kJ/h and 0 kJ/h to arrive at the new reference value $c_{B,ref} = 1.07 \text{ mol/l}$ significantly faster. It rejects the disturbance for the feed temperature T_F similar to the controller with nonlinear inverse model.

5 Conclusions

Modelica 3.3 introduced synchronous features that enable the rigorous treatment of discrete-time models. The Modelica_Synchronous library demonstrates the relevance of these features for control (Otter et al, 2012). The simulation tools Dymola and OpenModelica support Modelica_Synchronous so far.

This paper proposes an extension of FMI 2.0 to make rigorous discrete-time models available for control applications. The extension is backwards compatible. It specializes generic events towards clocks for discrete-time models. Tools that do not support synchronous time event handling can export the same model using generic events as known from FMI 2.0. An importing tool should parse the extensions of the XML file, in particular the Clocks section, activate periodic clocks at the specified intervals and activate inferred clocks on environment needs. Alternatively, an importing tool might reject the FMU if it finds inferred or periodic clocks in the Clocks section. Triggered clocks are activated by the FMU itself and need no support by the importing environment.

The basic concept of activation of sample times by a tool is well known from other simulation technologies, such as Simulink S-functions. The proposed FMI extension exploits the XML model description to associate clocks with variables. This enables deterministic clock propagation among multiple connected FMUs. The optional specification of integer valued clock intervals further enhances clock inference for system level design.

FMI export with synchronous features was implemented in the tools Dymola and OpenModelica. Import was implemented in the optimization solver HQP. The paper motivates the FMI extension with use cases for a highly nonlinear chemical process model. The use cases include functional engineering, nonlinear inverse models for control and nonlinear model predictive control.

The synchronous features of Modelica also include the automatic conversion of continuous-time models to discrete-time models with inline integration. This mixed symbolic/numeric approach simplifies model-based control applications considerably, because it releases an importing environment from the treatment of continuous-time differential equations and sensitivity equations. Run-time efficiency increases.

Discrete-time FMUs with inline integration are a work in progress. Development versions of OpenModelica, Dymola and HQP were used for the optimal control problem in section 4.4. Dymola 2017 was used for the nonlinear inverse model in section 4.3.

Discrete-time FMUs will serve for the investigation of parallel algorithms for automatic differentiation and numerical optimization in the PARADOM project.

Acknowledgements

This work was supported in parts by the Federal Ministry of Education and Research (BMBF) within the project PARADOM (PARAllel Algorithmic Differentiation in OpenModelica) – BMBF funding code: 01IH15002E.

References

- T. Blochwitz, M. Otter, J. Åkesson, M. Arnold, C. Clauß, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, A. Viel: Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models, 9th International Modelica Conference, Munich, 2012. <http://www.ep.liu.se/ecp/076/017/ecp12076017.pdf>
- H. Chen, A. Kremling, F. Allgöwer: Nonlinear Predictive Control of a Benchmark CSTR, Proceedings 3rd European Control Conference ECC'95, Rome, 1995.
- H. Elmqvist, M. Otter, S.E. Mattsson: Fundamentals of Synchronous Control in Modelica, 9th International Modelica Conference, Munich, 2012. <http://www.ep.liu.se/ecp/076/001/ecp12076001.pdf>
- H. Elmqvist, M. Otter, F. Cellier: Inline integration: A new mixed symbolic/numeric approach for solving differential-algebraic equation systems. In Proceedings ESM European Simulation Multiconference, Prague, 1995.
- S. Engell, K.-U. Klatt: Nonlinear control of a nonminimum phase CSTR. In American Control Conference, Los Angeles, 1993.
- R. Franke, E. Arnold: Applying new numerical algorithms to the solution of discrete-time optimal control problems. In: Computer Intensive Methods in Control and Signal Processing: The Curse of Dimensionality, Birkhäuser, Basel, 1997.
- R. Franke, M. Walther, N. Worschech, W. Braun, B. Bachmann: Model-based control with FMI and a C++ runtime for Modelica. Proceedings of 11th International Modelica Conference, Paris 2015. https://www.modelica.org/events/modelica2015/proceedings/html/submissions/ecp15118339_FrankeWaltherWorschechBraunBachmann.pdf
- R. Franke: Initialization of Clocked Discrete States, Modelica Change Proposal MCP-0024 2016. https://svn.modelica.org/projects/MCP/public/MCP-0024_InitializationClockedStates/MCP-0024_InitializationClockedStates.docx
- Functional Mock-up Interface for Model Exchange and Co-Simulation, Version 2.0, July 2014.
- G. Looye, M. Thümmel, M. Kurze, M. Otter, J. Bals: Nonlinear Inverse Models for Control. Proceedings of 4th International Modelica Conference, Hamburg, 2005. https://www.modelica.org/events/Conference2005/online_proceedings/Session3/Session3c3.pdf
- Modelica Association: Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification, Version 3.3. May 9, 2012.
- M. Otter, S.E. Mattsson, R. Franke, H. Elmqvist, T. Blochwitz: Discrete States and Time Events in FMI (#353), FMI Change Proposal FCP-001, 2016. https://svn.fmi-standard.org/fmi/trunk/FMI_ChangeProposals/FCP_001_SampledDataSystemsForModelExchange/FMI_Proposal_DiscreteStates_TimeEvents.docx
- M. Otter, B. Thiele, H. Elmqvist: A Library for Synchronous Control Systems in Modelica, 9th International Modelica Conference, Munich, 2012. <http://www.ep.liu.se/ecp/076/002/ecp12076002.pdf>

Model-based Embedded Control using Rosenbrock Integration Methods

Hans Olsson¹ Sven Erik Mattsson¹ Martin Otter² Andreas Pfeiffer² Christoff Bürger¹
Dan Henriksson¹

¹Dassault Systèmes AB, Lund, Sweden,

{Hans.Olsson, SvenErik.Mattsson, Christoff.Buerger, Dan.Henriksson}@3ds.com

²DLR, Institute of System Dynamics and Control, Germany,

{Martin.Otter, Andreas.Pfeiffer}@dlr.de

Abstract

Directly generating controller code from models is important for advanced model-based design. This paper describes how Dymola can generate embedded C-code from Modelica models, designed to be easy to embed, with care about minimal foot-print, traceability, and straightforward integration in embedded platforms and gives actual application examples.

The paper focuses on using Rosenbrock methods for index-1 problems (instead of the normal transformation to index 0) that allows Dymola to handle stiff systems in a way that both is theoretically sound and has an upper bound on the execution time per sample.

The stiff systems in the control system often occur due to using an inverse (simplified) model of the real plant in the controller. A nonlinear feedforward controller and a controller with feedback linearization, both applying an inverse model, demonstrate the proposed process by using Rosenbrock methods for embedded code generation.

Keywords: Modelica, inverse models, real-time, embedded, Rosenbrock methods, inline integration, feedforward controller, feedback linearization

1 Introduction

Modelica and Modelica tools such as Dymola are very well suited to model and simulate complex physical systems with primary focus on offline simulation for design and assessment, as well as on online simulation on special purpose hardware, e.g. for hardware-in-the-loop simulations. Modelica models have been used in controller applications where nonlinear Modelica models are part of the real-time control system, see for example (Looye *et al.*, 2005). The controller could be designed and assessed with Dymola, however, the actual real-time controller code had to be re-built manually either directly in C or with dedicated software for controller code generation.

There are several activities to extend the tool chains for Modelica models for real-time platforms, for example (Satabin *et al.*, 2015) for generation of certified code of simple Modelica models via the SCADE-suite (SCADE, 2017), or (Bertsch *et al.*, 2015)

for utilizing Modelica code on automotive electronic control units.

This paper describes the steps to generate embedded real-time code using a new prototype functionality of Dymola. The goals are (a) to generate code that can be certified for critical applications, (b) to guarantee an upper number of operations so that hard real-time constraints can be fulfilled, and (c) to support advanced controllers that can utilize nonlinear Modelica models in the feedback or feedforward path of the controller, which may require solving nonlinear differential-algebraic equation systems.

Numerical integration in real-time is a challenging task. Explicit integration, such as explicit Runge-Kutta methods or explicit multistep methods provide integration schemes with a deterministic number of numerical operations, but they may fail for stiff systems due to stability problems. Choosing a rather small step size can help to overcome this issue, but the sample rate and the computational power of real-time platforms are (strongly) limited. Standard implicit methods like implicit Runge-Kutta methods or BDF methods are designed for stiff systems with an acceptable step size, but nonlinear systems of equations have to be solved in each time step. Linearly implicit one-step methods, in particular Rosenbrock methods, provide a compromise. They can solve stiff problems using larger steps than explicit methods at the cost of having to solve linear systems.

The paper describes the new contributions in the following order: Section 2 gives an overview of the new code generator, Section 3 explains implementation of the Rosenbrock methods, Section 4 gives realistic application examples, and finally Section 5 gives a summary and outlines possible future extensions.

2 Model-based Embedded Controller Development in Dymola

Controller code intended to be executed in real-time on embedded devices is subject to special requirements. For example, (Bertsch *et al.*, 2015) discusses these challenges in the context of automotive embedded applications for the case of FMU source code

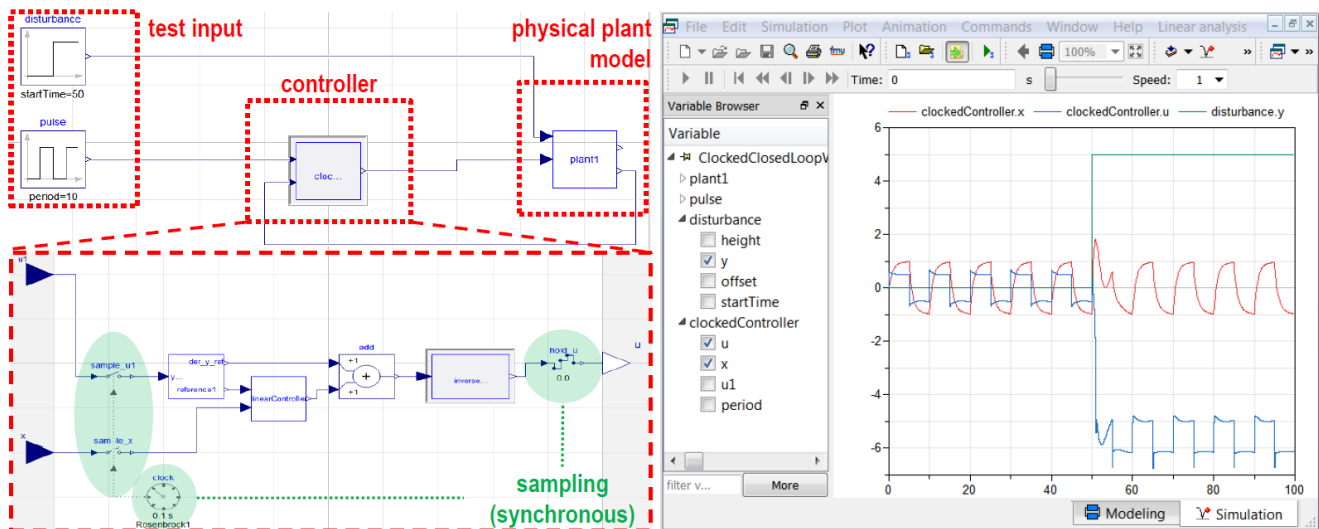


Figure 1. Embedded controller development scenario with Modelica/Dymola.

generation using Modelica tools. The standard C-code generated by Modelica tools is typically designed for desktop computer environments, where substantial hardware and software resources are available. Simulation is offline and without hard real-time constraints. Such standard code does not fulfill real-time system requirements, where code has to be deployed on embedded targets.

The standard C-code generated by Dymola from Modelica models is no exception; it is highly optimized to cope with several application scenarios including offline simulation and hardware-in-the-loop simulation of complex plant models on dedicated hardware platforms. However, this code includes many features not needed (*and* fails to fulfill constraints) for real-time controller code to be executed on embedded targets where minimalistic, self-contained, and human readable code is required.

On the other hand, Dymola provides convenient tooling for the development of full multi-domain system models and their simulation. It would be very convenient if embedded code for the controller parts also could be automatically generated and evaluated in software-in-the-loop simulations. The advantages of Modelica regarding complete system modeling and simulation are then leveraged also for real-time and embedded controller development.

Figure 1 summarizes the embedded development scenario we like to support. Physical plant models, controllers and test inputs for typical use cases can be fully modeled (left part) and simulated (right part) on system level. Throughout *iterative development* of *all* components, the *whole system* can be evaluated using standard simulation facilities. Embedded code can then be generated for the controller and co-simulated with the rest of the system. The results of such a software-in-the-loop co-simulation are shown on the right. The control signal (blue curve) is computed using the code generated by the embedded code generator. The red curve is the controlled plant output and the green signal

is a disturbance that becomes active midway through the simulation. The embedded code generator to support this process is described below.

2.1 Embedded Development Process

Given a physical system model in Modelica, the experimental Dymola embedded code generator considers the following four tasks for the design and implementation of controllers for an embedded target:

(1) Controller modeling: Implement controllers as Modelica models with continuous model equation parts as done in Modelica since many years.

(2) Model decomposition: Use the controller models in a synchronous environment as described in (*Otter et al., 2012*). Sample and hold blocks are used to incorporate the controller inputs and outputs. As integration scheme for the clocked blocks the Rosenbrock methods presented in Section 3 can be used. In Modelica terms, controllers are therefore just synchronously clocked sub-models. Their synchronous clock models the interval in which the embedded environment provides new real-time inputs and queries for respective control actions.

(3) Embedded code generation: To generate the code to be embedded for the controller parts, apply the embedded code generator on the total model. Dymola extracts the synchronously clocked parts and generates C-code which is a self-contained, real-time simulator of its clocked parts. The code is well-suited for embedded deployment.

(4) Embedded deployment: Adapt, integrate and test the generated controller code on a real-time platform, like a rapid prototyping platform or embedded device.

The four tasks can be iteratively performed, in interrelation with the development of the model of the controlled physical systems. Co-simulation of the generated controllers is achieved by binding the generated C-code of controllers as external C functions to Modelica and calling them at every sample point

throughout the system simulation. Examples of this procedure are given in Section 4.

2.2 Properties of Generated Code

In addition to the standard optimizations performed by Dymola's symbolic manipulation facilities (equation systems are automatically torn to solve as much symbolically as possible, constant expressions are folded and shared expressions are eliminated to be computed at most once), the controller source code generated by the embedded code generator complies with the following requirements for execution on embedded devices:

Code Integration

- All types (model variables, states and records) relevant for user code and further code integration are encapsulated in header files.
- Proper C data types are deduced. Substitutions are performed to reduce memory footprint.
- A clear interface (with separate C-functions for initialization, output calculations, etc.) enables easy integration within external embedded environments.
- A generic interface to a solver for linear equation systems enables the usage of solvers tailored for specific applications and targets. The code for a default LU-solver is provided.
- The generated code is self-contained, without dependencies on further libraries (including the C standard library), supporting embedded devices without operating system or restricted software availability.

Traceability

- Comments link the generated code to its Modelica model, enabling traceability of computations and declarations. An XML file describing all variables is generated.

Real-Time Execution

- No heap memory allocation or recursion enables deterministic static memory allocation and therefore memory requirement predictions.
- The Rosenbrock integration methods described in Section 3 are applied to achieve deterministic execution times and enable predictable response times by preventing iterative loops with unknown number of iterations.
- Equations and variables are only considered when relevant for *controller outputs*; irrelevant computations are removed from the code.

There are also some restrictions on the generated embedded code:

- It does not (yet) fulfill all requirements of the MISRA-C standard (MISRA, 2013), which is important for safety-critical systems.

- Simplified event handling is applied. Only state events can occur, since the models do not use time directly.
- Nonlinear systems of equations to be solved in real-time are currently not directly supported. By using linearly implicit integration methods with an index-1 formulation these systems are automatically avoided, see Section 3.2.

2.3 A Simple PI-Controller Example

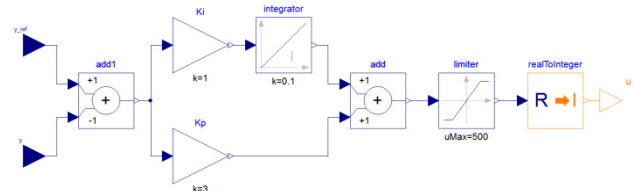


Figure 2. Simple PI-controller with output saturation and integer quantization.

Figure 2 shows a simple linear PI-controller. Since the synchronous model decomposition is only required to “mark” the controller for embedded code generation, but irrelevant for the actual embedded code generated, we can ignore the controller’s clock and in- and output samplings. Relevant for embedded code generation is that the output u of the controller model is declared with *min* and *max* attributes defining its saturation:

Modelica.Blocks.Interfaces.IntegerOutput

```
u (min = -500, max = 500) "Controller output";
```

The embedded code generator generates two C source code files: a header file defining the controller’s in- and output types (*dembedded.h*) and its actual implementation (*dembedded.c*).

The header file in Figure 3 defines a C struct that holds all relevant model variables, each annotated with a comment referring to its original Modelica declaration and description. Note, that the type of the output u is a signed 16-bit integer which Dymola has deduced from the *min* and *max* attributes declared in the controller’s Modelica model.

Figure 4 shows parts of the generated model implementation, in this case the start of the routine for the calculation of controller outputs. Each calculation is preceded by a comment that traces back to the original component, class and equation within the controller’s Modelica model responsible for the code generated. The comments also contain information about alias substitutions and deduced array sizes.

Similar code is generated for model update used to provide new sampling inputs and dynamics to compute complex simulation steps. Using the code and the types provided by the generated header file, the generated controller implementation can be integrated in the embedded software system actually deployed on some target device. More advanced application examples combining Rosenbrock integration methods and this embedded code generator are given in Section 4.

```

/* dseembedded.h
 * Model variables for Modelica model PIController */
#ifndef _dseembedded_h_
#define _dseembedded_h_
#include <dse_types.h>
#include <dseembedded_structs.h> /* structs for records */
#include <dseembedded_prototypes.h> /* function prototypes */

/* Model variables */
struct PIController_variables {
  /* input Modelica.Blocks.Interfaces.RealInput y
   "Measured variable" */
  real_t y;

  /* input Modelica.Blocks.Interfaces.RealInput y_ref
   "Reference signal" */
  real_t y_ref;

  /* output Modelica.Blocks.Interfaces.IntegerOutput
   u(min=-500, max=500) "Controller output" */
  integer16_t u;
  ...

  /* parameter Modelica.Blocks.Types.Init
   integrator.initType (min=1, max=4) =
   Modelica.Blocks.Types.Init.InitialState
   "Type of initialization (1: no init,
   2: steady state, 3,4: initial output)" */
  uinteger8_t integrator_initType;

  /* parameter Boolean limiter.strict = false
   "= true, if strict limits with noEvent(..)" */
  boolean_t limiter_strict;
  ...
};

```

Figure 3. C header file generated for the PI-controller.

```

/* dseembedded.c
 * Model equations for Modelica model PIController */
#include <dseembedded.h>
#include <dseembedded_codes.c> /* functions code */

/* Model outputs */
static int model_outputs(PIController_variables* v,
  PIController_states* s)
{
  /* Component add1 */
  /* Class Modelica.Blocks.Math.Add */
  /* y = k1*u1+k2*u2; */
  /* y = Ki.u; */
  /* u1 = y_ref; */
  /* u2 = y; */
  v->Ki_u = v->add1_k1*v->y_ref+v->add1_k2*v->y;

  /* Component Kp */
  /* Class Modelica.Blocks.Math.Gain */
  /* y = k*u; */
  /* u = Ki.u; */
  v->Kp_y = v->Kp_k*v->Ki_u;

  /* Component add */
  /* Class Modelica.Blocks.Math.Add */
  /* y = k1*u1+k2*u2; */
  /* u1 = integrator.y; */
  /* u2 = Kp.y; */
  v->add_y = v->add_k1*v->integrator_y+v->add_k2*v->Kp_y;
  ...
}

```

Figure 4. Generated PI-controller implementation.

3 Rosenbrock Methods

For real-time applications of stiff systems Dymola has historically reduced the model's equation system to index 0 (an ODE system) and used a nonlinear solver to handle the implicit Euler discretization by a limited number of Newton iterations, see e.g. (Elmqvist, Mattsson et al., 2004).

One main advantage of Rosenbrock methods is to directly solve stiff systems using only a *linear* solver.

A certain variant of the implicit Euler method doing only one Newton iteration per step is equivalent to the corresponding Rosenbrock method of order 1.

In the following subsection Rosenbrock methods are introduced for index-1 DAEs which are known from the literature. Further, the advantages of the index-1 formulation and their application on Modelica models are presented. Finally, some properties and details of their implementation in Dymola are discussed.

3.1 Rosenbrock Methods for Index-1 DAEs

The supported Rosenbrock methods consider non-autonomous DAE systems with index 1 of the form

$$E\dot{y} = f(t, y)$$

where E is a constant and possibly singular matrix.

The Rosenbrock methods (Hairer, Wanner, 1991) are defined by s stages for a single step from t_0 to $t_1 := t_0 + h$ with the initial state vector $y_0 = y(t_0)$ to get an approximation of the state vector $y(t_1)$:

$$\begin{aligned}
 y_1 &= y_0 + \sum_{i=1}^s m_i u_i, \\
 J_i &= \frac{1}{h\gamma_{ii}} E - f_y(t_0, y_0), \\
 J_i u_i &= f(t_0 + \alpha_i h, y_0 + \sum_{j=1}^{i-1} a_{ij} u_j) + E \sum_{j=1}^{i-1} \frac{c_{ij}}{h} u_j \\
 &\quad + \gamma_i h f_t(t_0, y_0) \quad (i = 1, \dots, s).
 \end{aligned}$$

Fixed method coefficients are γ_{ii} , a_{ij} , α_i , c_{ij} , γ_i and m_i . To compute the stage vectors u_i a linear system of equations has to be solved in each stage. Especially interesting are methods with $\gamma := \gamma_{ii}$ ($i = 1, \dots, s$), because then the iteration matrix J_i of the linear system is the same in each stage – and we can drop the index. So, only one decomposition of the iteration matrix J is required in each time step. Rosenbrock methods require the evaluation of the Jacobian f_y and the derivative with respect to time f_t .

For systems with input variables u (which must not be mixed up with the stage vectors u_i):

$$E\dot{y} = f(t, y) := \varphi(t, y, u(t))$$

this means $f_t = \varphi_t + \varphi_u \dot{u}$ with the derivatives \dot{u} of the external input signal u to be provided.

There exist coefficients of Rosenbrock methods with convergence orders from 1 to 4 with different stability properties. In (Lubich, Roche, 1990) an L-stable Rosenbrock method of order 3 with $s = 4$ stages is developed for index-1 systems. In (Rang, 2013) the coefficients of Rosenbrock methods are improved to get methods without order reduction for (very) stiff problems.

Rosenbrock methods are interesting for real-time simulation of stiff systems, because the computational procedure for a step includes the solution of linear

systems but not of nonlinear systems. For linear systems a fixed number of computations guarantee finding the numerical solution in contrast to the iteration process for solving nonlinear systems.

3.2 Linear and Nonlinear Systems of Equations

In comparison to the ODE representation of a Modelica model, the index-1 formulation in Section 3.1 has some advantages in combination with Rosenbrock methods. Consider the example system of index 1

$$\begin{aligned}\dot{x} &= f(t, x, y), \\ 0 &= g(t, x, y),\end{aligned}$$

where a possibly nonlinear function g couples states x and algebraic variables y . The typical transformation to ODE form would lead to

$$\begin{aligned}\dot{x} &= f(t, x, y(x, t)), \\ y &= g^{-1}(x, t).\end{aligned}$$

Here, maybe a nonlinear or at least a linear system of equations has to be solved when inverting the function g with respect to y . This can be avoided, if the index-1 formulation is used:

$$\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} f(t, x, y) \\ g(t, x, y) \end{pmatrix}.$$

When applying a Rosenbrock method only the right hand side and its derivatives are evaluated. The one step method provides an approximation of the solution vectors x , y just by solving linear systems in the stages of the method. So, no nonlinear or nested linear system has to be solved. This property is very helpful for real-time simulation, because nonlinear loops in the original Modelica model can be replaced by linear systems in this way – leading to predictable computation times, see Section 4.1.2 for an example.

3.3 Rosenbrock Methods in Dymola

The support of Rosenbrock methods has recently been implemented in Dymola. The integration schemes rely on the index-1 formulation of the manipulated Modelica model equations. By the symbolic manipulation algorithms of Dymola, it is structurally guaranteed, that the system $E\dot{y} = f(t, y)$ has index 1. Currently, in Dymola four different Rosenbrock methods with orders 1-4 are available. The method of order 1 is the linearly implicit Euler method. All the methods are available as global inline integration methods in Dymola, see the menu in Figure 5.

Further, a Rosenbrock method can be specified as a solver method for a clocked part, by setting the argument *solverMethod* of the Modelica Clock constructor *Clock(c, solverMethod)*. This functionality is then used in the *Modelica_Synchronous* library to define the integration method of clocked equations. In

the example in Figure 6 the first order Rosenbrock method “Rosenbrock1” is used.

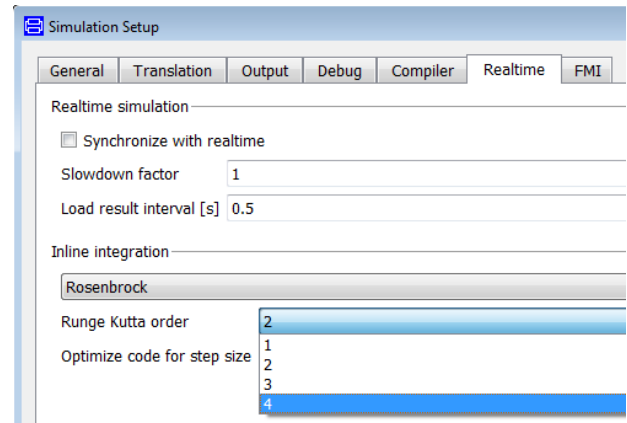


Figure 5. Menu to select a Rosenbrock integration method in Dymola.

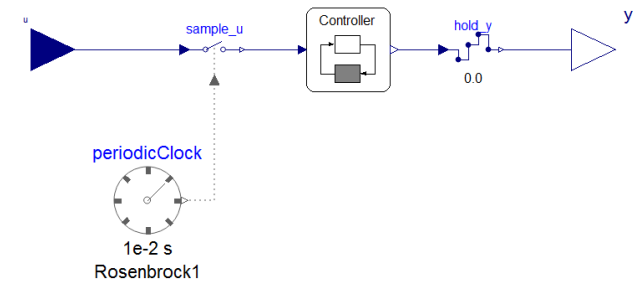


Figure 6. Inclusion of a continuous-time controller into a clocked environment using Rosenbrock integration.

To complete the tool chain, the Rosenbrock methods are also supported by Dymola’s embedded code generator. The symbolic machinery transforms the Modelica model equations after index reduction and fixed state selection to the form $E\dot{y} = f(t, y)$ and generates code for calculating E and f . Additionally the matrix f_y corresponding to the analytic Jacobian is straightforward to construct and generate code for.

It is more complicated to construct the vector f_t . The symbolic machinery normally deduces a total derivative with respect to time, but for Rosenbrock methods a partial derivative is needed. We will explain the difference with an example. If for example $f(t, y) = t^2 + y$, the total derivative with respect to time would be $d/dt f(t, y(t)) = 2t + \dot{y}$, but the partial derivative is $f_t(t, y) = 2t$. The symbolic machinery has also to deal with intermediate variables (e.g. z , if we rewrite the previous equation as $f(t, y) = z + y$; $z = t^2$; and the partial derivative with respect to time should differentiate those, but not the states).

Moreover, this time-derivative is not used by other standard numerical integration methods, and thus some Modelica functions do not provide the necessary derivatives (this can be handled either by assuming that the functions are smooth even if not specified or some minor modifications of libraries such as Modelica

Standard Library to specify this). This is especially the case, if the model follows some time-dependent trajectory $r(t)$ – because we need the derivative $\dot{r}(t)$, which is not needed for other methods. For input dependent models the derivative \dot{u} of the input is involved in f_t (as explained in Section 3.1) and could be approximated by a difference quotient or the influence of \dot{u} could be neglected in the method equations ($\dot{u} = 0$), when we assume that the input signal is piecewise constant. But this introduces some non-smoothness into the right hand side f , which could lead to numerical errors especially when applying Rosenbrock methods with orders greater than two. A more sophisticated solution for such input dependent models would require the additional input \dot{u} for the model. This approach has not been investigated so far.

The matrix E is generally sparse or even diagonal with just zeros and ones on the diagonal and it would be worth to exploit the structure of the matrix when generating tailored code for the application of a Rosenbrock method to a specific Modelica model – but this is not yet realized in the implementation.

Dymola's implementation of Rosenbrock is generic, and some method-specific optimizations are not yet included, e.g. some Rosenbrock methods have several rows of the matrix (a_{ij}) that are identical, and in those cases we could avoid re-evaluating the right hand side f . This can intuitively be explained as performing exactly two iterations of the nonlinear solver for that point.

There are variations of Rosenbrock methods (W-methods) that keep the factorized matrices for several steps. We have not considered them for real-time applications. The reason is that for real-time code the goal is to ensure a maximum computation time for each sampling point – not for the average one; and we will anyway need new factorized matrices after each event. If we do not explicitly detect events, the problem with W-methods would be more severe since the continuity assumptions are silently broken.

4 Application Examples

In this section two application examples are given to demonstrate how the embedded code generation and the Rosenbrock methods can be used to generate real-time code for nonlinear controller structures with guaranteed upper number of operations.

4.1 Nonlinear Feedforward Controllers

We consider a continuous-time controller with two structural degrees of freedom and an inverse plant model in the feedforward path. See (Looye et al., 2005) for details on this controller structure and its implementation in Modelica. In case the inverse plant and the plant model are identical, they start at the same initial values and the plant is stable, then the control

error is equal to zero, so the plant output follows the filtered reference input. The feedback controller is used to compensate for differences in the plant and inverse plant model, as well as for external disturbances, and it stabilizes a plant in case it is unstable.

4.1.1 Implementation in Modelica

In Modelica an inverse plant model can be constructed by using the model component

Modelica.Blocks.Math.InverseBlockConstraints

to exchange inputs and outputs and by connecting a filter to the input of the inverse model. As filter the model *Modelica.Blocks.Continuous.Filter* with parameters *filterType = LowPass* and *analogFilter = CriticalDamping* or *Bessel* can be used, see Figure 7. The minimum order of the filter results from the structural analysis of the inverse plant model resp. the corresponding DAE in order to only provide the input u but not derivatives of it. The derivatives of the smoothed input signal are computed inside the filter model.

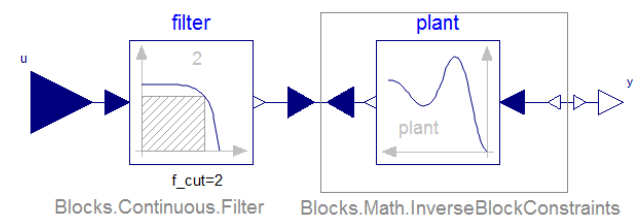


Figure 7. Definition of an inverse plant model.

In order that the controller can be used on a real time system, the process of Section 2 is applied. The continuous controller model is transformed to a clocked system with sample and hold blocks and an appropriate inline integrator needs to be selected for the clock. In simple cases, an *Explicit Euler* method might be enough. If the controller contains nonlinear algebraic equations or if the model is stiff, a *Rosenbrock* integrator has to be selected, see also Section 3.3. Note, that the filter might be stiff even if the inverse plant model might be non-stiff.

It follows an application example for the automatic construction of nonlinear feedforward controllers that can be used in an embedded system.

4.1.2 Example 1: Slider Crank Mechanism with Feedforward Controller

The following example is a slider-crank mechanism that is directed in vertical direction. At the top a spring-mass system is present. The goal is to move the revolute joint of the slider-crank mechanism, such that the mass follows a pre-defined path without vibrations.

When kinematically driving the revolute joint with constant velocity, then the vertical coordinate of the top mass moves as shown



in Figure 8. As can be seen, significant vibrations are present in the movement of the mass. The goal is to develop an embedded controller according to Section 2. The problem is rather challenging, because the slider crank mechanism introduces a nonlinear algebraic equation system in the plant, as well as in the plant inverse.

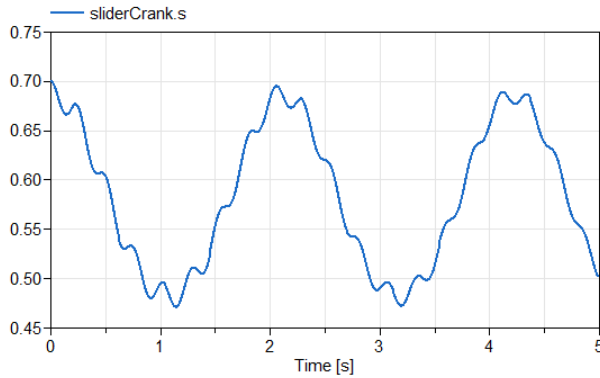


Figure 8. Vertical movement of top-mass of the slider-crank mechanism.

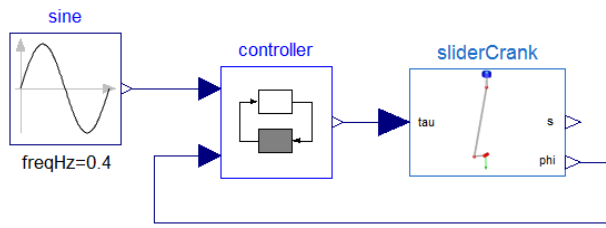


Figure 9. Controlled slider crank mechanism.

In Figure 9 the overall system including a controller is shown. The controller is detailed in Figure 10 where for the feedforward path of the controller an inverse model of the slider crank mechanism is present such that the input of the inverse model is the vertical position s of the top mass, and the outputs are (a) the reference torque τ for the revolute joint, and (b) the reference angle ϕ for the revolute joint. As filter a

third order critical damping filter is used. The control error is the difference between the reference angle ϕ computed by the inverse slider crank model and the measured angle ϕ from the plant. A simple P controller is used in the feedback loop.

Although a nonlinear system of equations appears in the model equations of the inverse slider-crank model, it is possible to generate embedded code for the sampled data controller according to Section 2 by using the newly supported Rosenbrock integrators of Section 3. The detailed explanation of this effect is found in Section 3.2. A proper step size of the tested Rosenbrock methods for the controller is 1 ms.

Some simulation results are shown in Figure 11. The Rosenbrock method of order 1 (the linearly implicit Euler method) leads to very accurate results with respect to the reference solution generated by a highly accurate DASSL simulation. The numerical solution of the Rosenbrock method with order 3 is also rather accurate, only in the torque signal some vibrations are visible. One reason could be neglecting the input derivatives of s_{ref} and ϕ in the integration scheme of Rosenbrock methods as described in Section 3.1.

Experiments show for the Explicit Euler method a maximum step size of 0.5 ms can be used; otherwise the numerical integration cannot be run due to difficulties with solving the nonlinear system. There remains still a nonlinear system of equations due to the index-0 formulation of the translated model equations. This also means that currently no embedded code can be generated for this controller example when using an Explicit Euler method as integrator.

There are two advantages of Rosenbrock methods: Because they are implicit methods, generally greater step sizes can be used than for explicit methods and nonlinear system of equations present in the model equations can be approximately solved by a Rosenbrock solver resulting in only linear systems.

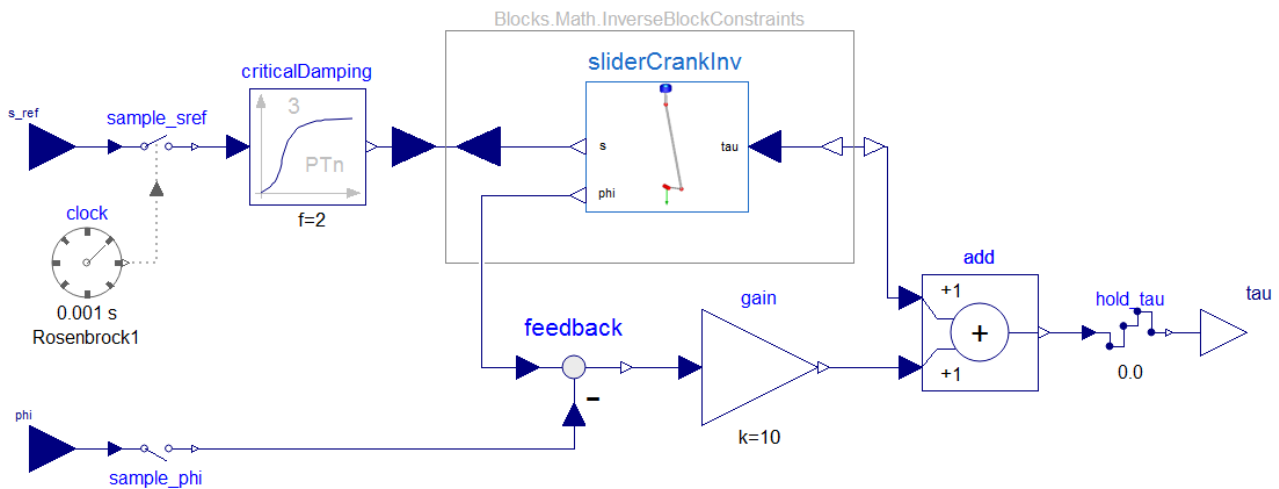


Figure 10. Sampled data controller of a slider crank mechanism consisting of inverse plant model in the feedforward path, a filter of order 3 and a P controller in the feedback loop.

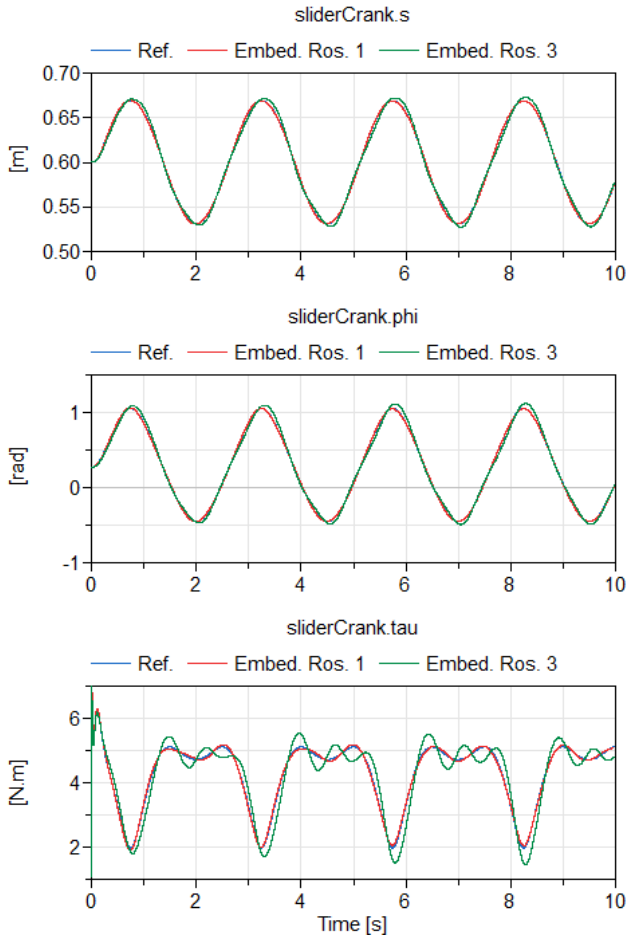


Figure 11. Simulation results of the slider crank mechanism with a nonlinear feedforward path and a P controller in the feedback loop. The reference solution using the continuous controller in Modelica is generated by highly accurate BDF-methods with DASSL (blue lines) whereas the embedded controllers contain Rosenbrock methods of order 1 (red lines) and order 3 (green lines) with a constant step size of 1 ms.

4.2 Feedback Linearization Controllers

A further important controller structure using nonlinear plant models is feedback linearization, see (Looye *et al.*, 2005) for more details including the implementation in Modelica.

4.2.1 Implementation in Modelica

The first part of this subsection is a summary of material provided in (Looye *et al.*, 2005). The principal differences between a controller with feedback linearization and a feedforward controller of Section 4.1 are that for the feedback linearization

- the inverse plant model is in the feedback part of the controller and
- the states in the inverse model are obtained from the actual plant via measurement and/or estimation and not via solving a DAE (but algebraic equations might need to be solved).

When deriving feedback linearizing control laws manually, the outputs to be controlled are differentiated

until an analytical relation with a control input is found. If the system model is available in Modelica, the derivation of the control laws can be automated using a similar procedure as described in Section 4.1.1. However, instead of a filter with a minimal order, a minimal set of integrators is added:

$$v := y^{(p)} := \frac{d^p}{dt^p} y \quad (1)$$

where v is the new model input corresponding to the output with relative degree p . We describe the procedure for a single output system with the controlled output y . The desired dynamic behavior of the closed-loop system is then imposed by application of an additional feedback law:

$$v = k_0(y_{Ref} - y) - \sum_{i=1}^{p-1} k_i y^{(i)} \quad (2)$$

with constant coefficients k_i . This feedback law requires availability of the $(p-1)$ -th derivative of the controlled output y . The derivatives may be obtained from measurements or from the computed values in the inverse model. In case the inverted model exactly represents the true system, the closed loop system becomes the single output case:

$$y^{(p)} + \sum_{i=1}^{p-1} k_i y^{(i)} + k_0(y - y_{Ref}) = 0. \quad (3)$$

A disadvantage of feedback linearization is that the state vector of the plant must be fully available from measurement and/or estimation.

In Modelica, the inverse model is built in a similar way as for a feedforward controller, see Figure 12:

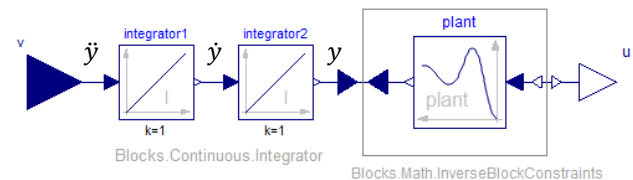


Figure 12. Definition of an inverse plant model for feedback linearization.

When translating this model with Dymola, typically an error message of the following kind is displayed: "The model requires derivatives of some inputs as listed below: ...". For example if derivatives of order 2 are required by v , then 2 more integrators have to be added.

This inverse model with the integrators of Figure 12 is placed in the feedback-loop of the controller. If the minimal number of integrators are added, then the model from $v \rightarrow y$ has the same number of states as the non-inverted plant. These states must be provided from measured and/or estimated values of the plant. To formulate this, Dymola has introduced an annotation to map a sampled input signal, say, $xs = \text{sample}(xc)$ to a state x , say:

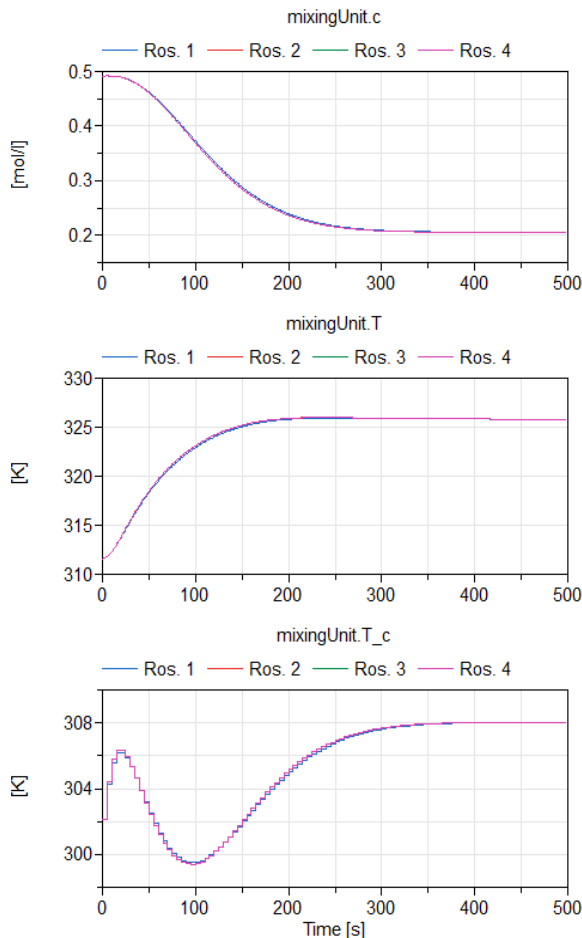


Figure 15. Step response of the mixing reactor controlled by a feedback linearization controller.

5 Summary and Outlook

Directly generating controller code from models is important for model-based design. This paper demonstrates how Dymola can generate embedded C-code for Modelica models (with several novel aspects) and demonstrates this for application examples.

By using Rosenbrock methods on the index-1 problem Dymola can handle stiff systems in a way that both is theoretically sound and has an upper bound on the execution time per sample. The stiff systems in the control system often occur due to using an inverse (simplified) model of the real plant in the controller.

Additionally Dymola's generated embedded C-code has been designed to be easy to embed, with care about minimal foot-print, traceability, and straightforward integration in embedded platforms.

Finally, a nonlinear feedforward controller and a feedback linearization controller have been implemented in Modelica for different application examples. They show the potential of the whole process by generating embedded real-time code for these nontrivial examples.

Future considerations include investigating how to handle inputs (piece-wise constant or extrapolation) for Rosenbrock methods, and nonlinear systems in

initialization and event code. Additionally, the choice of the specific Rosenbrock methods will be re-investigated. A possible future extension would be to use Rosenbrock methods with step-size control in off-line mode; one benefit would be to quickly get an estimate of the step-size needed for the model.

6 Acknowledgment

We would like to thank Gertjan Looye from DLR for his help regarding modeling of feedback linearizing controllers.

References

- C. Bertsch, J. Neudorfer, E. Ahle, S. S. Arumugham, K. Ramachandran, A. Thuy. FMI for Physical Models on Automotive Embedded Targets. *Proc. of 11th International Modelica Conference*, pp. 43-50. Versailles, France, 2015.
 - H. Elmqvist, S. E. Mattsson, H. Olsson, J. Andreasson, M. Otter, C. Schweiger, D. Brück. Realtime Simulation of Detailed Vehicle and Powertrain Dynamics. *Electronics Simulation and Optimization*. SAE 2004 World Congress, Detroit, USA, 2004.
 - E. Hairer, G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer, 1991.
 - G. Looye, M. Thümmel, M. Kurze, M. Otter, J. Bals. Nonlinear Inverse Models for Control. *Proceedings of 4th International Modelica Conference*, pp. 267-279, TU Hamburg-Harburg, Germany, 2005.
 - C. Lubich, M. Roche. Rosenbrock Methods for Differential-algebraic Systems with Solution-dependent Singular Matrix Multiplying the Derivative. *Computing* 43, 325-342, Springer, 1990.
 - MISRA Consortium. *Guidelines for the Use of the C Language in Critical Systems*. 2013.
 - M. Otter, B. Thiele, H. Elmqvist. A Library for Synchronous Control Systems in Modelica. *Proc. of 9th International Modelica Conference*, pp. 27-36. Munich, Germany, 2012.
 - J. Rang. *Improved traditional Rosenbrock Wanner methods for stiffodes and daes*. Technical report, Institute of Scientific Computing, Technical University Braunschweig, 2013.
 - L. Satabin, J.-L. Colaco, O. Andrieu, B. Pagano. Towards a Formalized Modelica Subset. *Proc. of 11th International Modelica Conference*, pp. 637-646. Versailles, France, 2015.
- SCADE:
www.esterel-technologies.com/products/scade-suite

Integration of complex Modelica-based physics models and discrete-time control systems: Approaches and observations of numerical performance

Kai Wang¹ Christopher Greiner¹ John Batteh² Lixiang Li²

¹ Ford Motor Company, USA, kwang37@ford.com, cgreiner@ford.com

² Modelon, Inc., USA, john.batteh@modelon.com, lixiang.li@modelon.com

Abstract

A Modelica-based air conditioning (A/C) system model has been integrated, closed-loop, with related S-function-based controls in the Simulink environment. The integration was performed with two different approaches, with a DymolaBlock-based S-function for the A/C model, and as a co-simulation FMU. The simulation performance of the integrated model needs to be sufficiently fast for the purpose of vehicle-level simulations and optimizations. This paper will discuss the integrated modeling of A/C system and associated control systems over a dynamic drive cycle, and the associated numerical performance issues discovered, as well as some approaches taken to increase said performance.

Keywords: *Modelica, discrete, variable, integration*

1 Introduction

As CAE simulations become more complex, the need for computational efficiency increases in order to provide timely solutions and analyses. One facet of this complexity is the integration of multiple software modeling tools and environments in order to utilize the most capable computational technologies for the different features of these complex system models. Physical plant models may be developed in Modelica and require variable step solvers to capture both fast and slow continuum dynamics while discrete time-based control systems may be developed in C-code or Simulink and require fixed time step solvers. Integrating these plant and control models into a single environment can result in computational inefficiencies due to conflicting solver time step requirements. This paper will discuss the integrated modeling of an automotive vapor compression air conditioning system and associated control systems over a dynamic drive cycle, and the associated numerical performance issues discovered, as well as some approaches taken to increase said performance.

2 Overview of the physical plant model - A/C refrigerant components, refrigerant system, and cabin

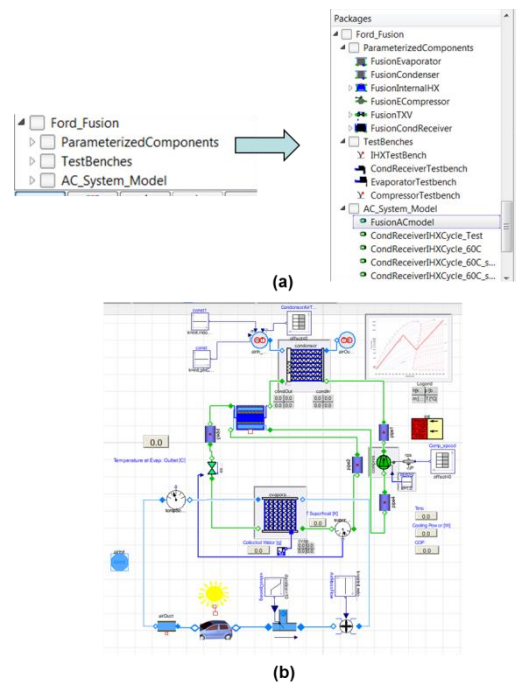


Figure 1 (a) Hierarchy structure of the A/C model (b) Modelon A/C model layout

For this study, only models of the complete refrigerant system and the vehicle interior (cabin) were required to simulate the physics of interest. The plant models are Modelica-based and developed in Dymola 2015FD01, utilizing component and refrigerant models from Modelon-supplied libraries. The structure and layout of the A/C model package are shown in Figure 1. The Dymola package browser in the upper-left of Figure 1(a) displays the package hierarchy and is shown with the A/C model selected. There are three sub-packages under the A/C model, namely, parameterized components, test benches, and A/C system model package. The parametrized components are specific, populated component models which are

used in the refrigerant circuit models. They include an evaporator, compressor, condenser, internal heat exchanger, and Thermostatic Expansion Valve (TXV), as well as piping and hoses. The test bench sub-package is a workspace for users to customize or calibrate individual component instances, and includes test benches for condenser, evaporator, and compressor models. The A/C simulation model sub-package consists of two main sub-models, the refrigerant circuit and the cabin model (including the air duct, blower model, etc.) as shown in Figure 1(b). Included in the refrigerant circuit model are the evaporator, compressor, condenser, internal heat exchanger, piping, and hoses. The cabin model consists of cabin interior, an air duct model, temperature blend door model, and the blower model.

The SC03 drive cycle is part of the EPA regulatory automotive 5-cycle fuel economy method. SC03 is a full vehicle chassis dynamometer test performed with the vehicle A/C unit operating with an ambient temperature of 95°F (35°C) and a solar loading on the vehicle of 850 W/m². The cycle represents a 3.6 mile (5.8 km) route with an average speed of 21.6 mph (34.8 kph), maximum speed 54.8 mph (88.2 kph), and duration of 596 seconds. The vehicle speed profile for the SC03 test is shown in Figure 2. The SC03 drive cycle was chosen for this study as it is the only cycle of the EPA 5-cycle method that requires the air conditioning system to be operating.

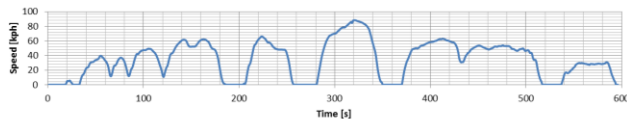


Figure 2 Vehicle speed race for SC03 cycle

3 Overview of the control systems models

The control systems required for the electrified automotive air conditioning system operation typically consist of the climate control head, compressor control, condenser fan control, active grille shutters, and electric water pump, as shown in Figure 3. The climate control head is the interface between the occupant and the climate control system. It controls the overall operation of the climate system, including cooling or heat request, blower airflow setting, airflow mode setting (location of discharge air), recirculation versus fresh air, cabin and evaporator temperature settings, and automatic or manual operational mode. The control head also controls the states of ancillary systems such as auxiliary heaters, glass fogging detection, heated/cooled seats, heated backlight/windshield, and heated steering wheel. Explicit modeling of a closed-loop control head is not included in this study, as the climate system settings are manually set at the start of the cycle and remain static throughout the drive cycle,

and the evaporator target temperature time trace comes from actual test data.

The compressor control modulates the compressor speed based on temperature request and refrigerant discharge pressure. The climate control head determines an appropriate target evaporator outlet temperature profile and the compressor control sets the compressor speed to achieve the target temperature using a PI control algorithm. If the compressor discharge pressure exceeds a specified limit, compressor speed is reduced and modulated using a PI controller to maintain a maximum allowed discharge pressure.

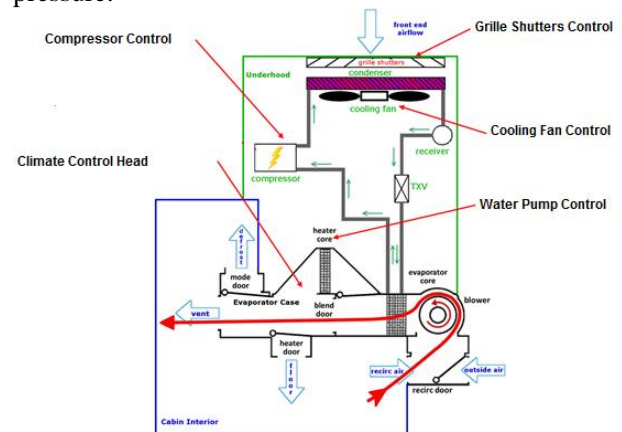


Figure 3 Automotive air conditioning control systems

The cooling fan control regulates the underhood front-end airflow fan speed in order to maintain the thermal management of systems requiring airflow through their associated heat exchangers. These systems typically include the A/C system, engine coolant, engine oil, and transmission oil systems. When the A/C system is operating the fan control calculates a desired fan speed/duty cycle based on compressor discharge pressure and ambient temperature. Fan speeds/duty cycles are also calculated for the other thermal systems and an arbitrator function determines the maximum required fan speed/duty cycle then commands the fan. Above certain vehicle speeds the fan speed/duty cycle is reduced to take advantage of ram air effect on front-end airflow.

Active grille shutters are used to balance aerodynamic drag and front end airflow/thermal management system requirements, closing down at higher vehicle speeds to reduce drag and opening more at lower speeds to enhance front end airflow. Similar to the cooling fan control, each thermal system has a desired shutter opening calculated and an arbitrator determines the maximum opening required and commands the shutters. For the A/C system, the desired grille shutter opening is calculated based on compressor discharge pressure and ambient temperature and then combined with a vehicle speed multiplier to account for aerodynamic effects.

Each of the control systems described above are implemented in the Matlab/Simulink (Version R2014a) environment as pre-compiled S-functions. The source of the S-functions is based on the C-language code implemented on the actual vehicle. While specific control strategies are hard-coded into the S-functions themselves, all control calibration parameters are user accessible at run time. Also, as each of the controls are discrete time-based systems operating with timing loops of 10 and 100 milliseconds, execution of the control models in Simulink require fixed step solvers.

4 Integrated Dymola/Simulink Model - Methods of model integration

In Dymola, the A/C plant model operates in an open loop fashion with the time varying inputs for a simulation prescribed in advance. While this is useful for a number of scenarios, such as plant model development and verification, substantially more value can be realized when the plant models are integrated with control systems in closed loop, thereby allowing more complex system performance and optimization studies to be conducted. Simulink is used here as the integration environment, and there are multiple methods for incorporating a Dymola model into Simulink.

The first integration approach is using the DymolaBlock S-function interface. This Dymola option allows models developed in Dymola to be compiled as S-functions incorporated directly into Simulink, enabling the powerful physical modeling capabilities of the Modelica language to be combined with the controls orientated approach of Simulink. Figure 4 depicts the A/C model integrated into Simulink as a DymolaBlock S-function.

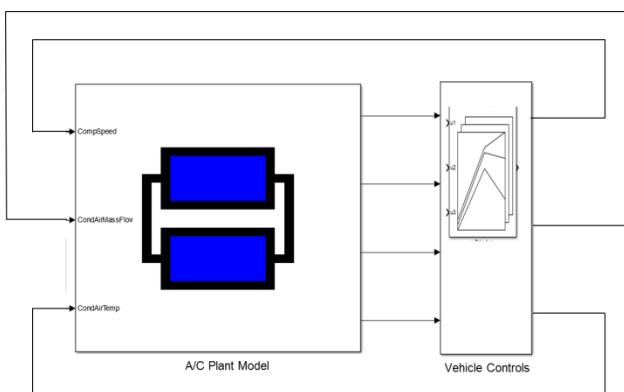


Figure 4 Integrated A/C model as a DymolaBlock S-function in Simulink

The second integration method utilizes the Functional Mock-up Interface (FMI) for model exchange or co-simulation. FMI defines a standardized modeling interface to be implemented by an executable module called a Functional Mock-up Unit (FMU). The FMI functions are used (called) by a simulation environment to create one or more instances of the

FMU and to simulate them, typically together with other model elements. An FMU may either have its own embedded numerical solvers (FMI for Co-Simulation) or utilize the simulation environment's own solvers (FMI for Model Exchange). [1].

In this application, we only consider the FMI for Co-Simulation option for two reasons. First, FMI for Model Exchange is very similar to utilizing an S-function function approach, like the DymolaBlock, as they both use the Simulink solver, and the DymolaBlock process is already incorporated into our modeling process. Second, due to the nature of the continuum behavior of the A/C system physics, and the need for a variable time step solver for the plant model, FMI for Co-Simulation allows the use of Dymola's solver for the physics in conjunction with Simulink's solver for the controls. Figure 5 shows an A/C model FMU in Simulink.

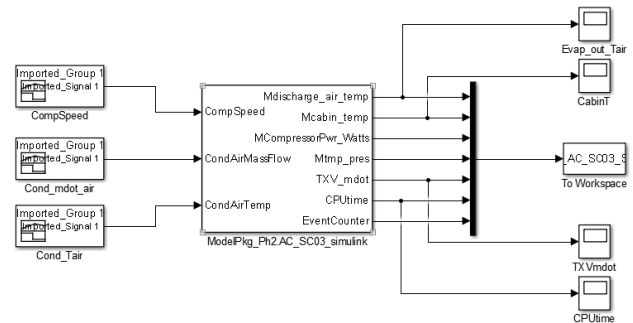


Figure 5 Integrated A/C model FMU in Simulink

Closed-loop control modeling requires the A/C system plant model to respond, minimally, at near real-time and produce physical and realistic outputs for the control systems to properly act upon [3]. In this study, there are three primary physical outputs from the A/C system plant model, specifically, the evaporator air out temperature, the vehicle cabin interior air temperature, and the compressor refrigerant discharge pressure. The evaporator air out temperature is the primary feedback signal used by the compressor controller to modulate the compressor speed, while the other two signals are used to a lesser degree. The input signals to the A/C system plant model from the control systems and Simulink-based physics models are the compressor speed, condenser airflow and air inlet temperature. The A/C system airflow through the evaporator and cabin interior is determined by the cabin blower fan and controlled by the Climate Control Head. In the SC03 drive cycle the blower is set to its maximum speed and is modeled as a constant input into the A/C model. Additional physics required by the integrated model are defined in Simulink, including the condenser airflow and air temperature. These physics models are coupled to both the control systems as well as the A/C model, as appropriate.

5 Model performance and solver strategies

The A/C system model, including the associated control systems, is one subsystem in a much larger and more complex total vehicle model. Because of this, it is desirable to optimize the computational performance of each subsystem in order to minimize the impact on the total vehicle model simulation time. The aforementioned A/C model integration approaches were selected to give the broadest range of solver and simulation settings to minimize the subsystem computational time.

The baseline performance for the A/C system model is defined as the physics-only Modelica model running the SC03 cycle open-loop in the Dymola environment, utilizing the DASSL variable time step with a solver tolerance of $1e-05$. Baseline simulation run time was 185 seconds, about one third of the real cycle time of 600 seconds. Next, the S-functions version of the Dymola model was run open-loop in the Simulink environment. Due to the numerical stiffness of the mathematics, only the Simulink ode15s variable time solver was capable of reliable solutions, and we used a solver tolerance of $1e-05$ to provide sufficient solution accuracy. The S-function simulation time averaged 183 seconds, essentially identical to the native-mode Dymola model. These results are shown as the first two bars in Figure 6. The actual SC03 cycle time comparator of 600 seconds is the right-most bar in the figure.

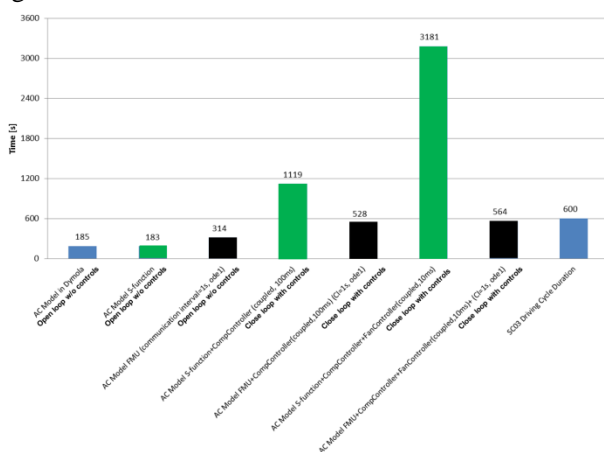


Figure 6. Comparison of the A/C system model computational performance

Next, we consider the closed-loop integration of the Dymola S-function with the 100ms fixed sampling rate compressor control and the 10ms sampling rate fan control. In Matlab R2014a, if any part of a Simulink model requires a variable step solver, the variable step solver must be the master solver for the simulation. However, Simulink does allow the user to define fixed sampling rates for parts of the model. For the integrated Dymola S-function model, we utilized the ode15s variable time step solver as the master

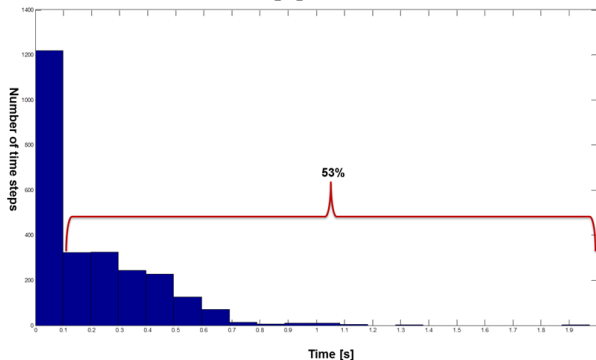
solver and specified a fixed sampling rate of 100ms for the compressor control, in order for the controls system to operate realistically. Likewise, when the cooling fan control was added to the integrated model, the fan control was set to run at a 10ms sampling rate, consistent with its actual operation.

The A/C model with the compressor control model completed the SC03 cycle in a time of 1119 seconds, as shown in the fourth bar of Figure 6, almost twice as slow as real time. When the fan control was added to the A/C model and compressor control the simulation time for the SC03 cycle increased dramatically to 3181 seconds, more than five times the actual cycle time.

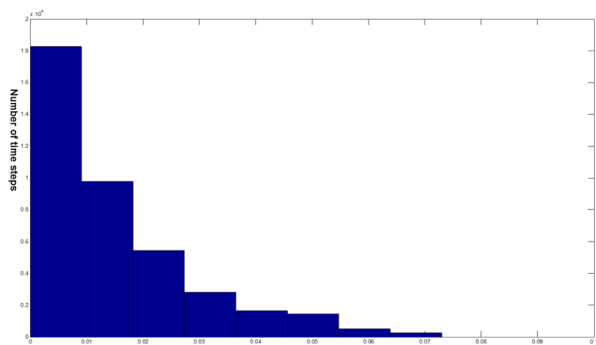
To rationalize the performance degradation we need to understand the interactions between the continuous, variable step solver used for the refrigerant system physics, and the discrete time-step solvers used by the control systems. With the integrated compressor control, the control module has to communicate 100ms. Additionally, when the fan control is also connected, it must communicate every 10ms. These sampling/exchange rates force the variable time step solver operating on the physical plant to synchronize the inputs and outputs of the Dymola S-function at the communication interval defined by the control system sampling rate. To illustrate this effect, the Simulink model was instrumented to record the timing rate of the variable time step solver. Figure 7a shows a histogram of the time step sizes used by the ode15s solver for the stand-alone, open-loop Dymola S-function model for the SC03 cycle. 53 percent of the time steps were larger than 100ms. Figure 7b shows the results of the A/C model with the integrated 100ms compressor control. The largest variable time step for this example was less than 80ms, and a large percentage 10ms or less. Finally, the combined A/C plant with compressor and cooling fan controls variable solver time steps are shown in Figure 7c, where the maximum step size is even less than the 10ms sampling rate of the fan controller. Combining variable and fixed step rates in Simulink models is referred to as a hybrid system in the Simulink documentation, as is detailed as follows:

“A hybrid system is a system that has both discrete and continuous states. Strictly speaking, any model that has both continuous and discrete sample times is treated as a hybrid model, presuming that the model has both continuous and discrete states. Solving such a model entails choosing a step size that satisfies both the precision constraint on the continuous state integration and the sample time hit constraint on the discrete states. The Simulink software meets this requirement by passing the next sample time hit, as determined by the discrete solver, as an additional

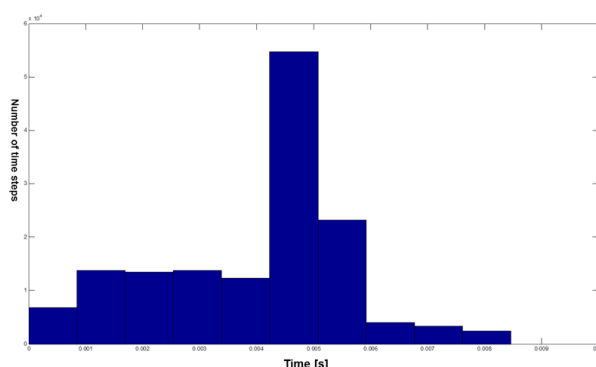
constraint on the continuous solver. The continuous solver must choose a step size that advances the simulation up to but not beyond the time of the next sample time hit. The continuous solver can take a time step short of the next sample time hit to meet its accuracy constraint but it cannot take a step beyond the next sample time hit even if its accuracy constraint allows it to.” [2]



(a). Standalone A/C Model S-function (183 seconds)



(b). A/C Model S-function+Compressor Control S-function (Coupled) (1119 seconds)



(c) A/C Model S-function+Compressor Control+Cooling Fan Control (Coupled) (3181 seconds)

Figure 7 Comparison of variable solver time step of the A/C model without/with compressor and fan control

As we attempted to optimize the computational performance of the combined A/C system and associated control systems, this approach of utilizing a Dymolablock-based S-function combined with the discrete time-based controls in Simulink was unable to deliver a sufficient level of performance, given the versions of the tools utilized, Dymola 2015FD01 and Matlab R2014a.

Next, the computational performance of the A/C model, as a co-simulation FMU, coupled with the control systems in Simulink, was evaluated. The primary benefit of FMI for co-simulation is that the FMU utilizes a native-mode solver from its parent tool, and the integrating environment uses its own appropriate solver, and the communication interval between the FMU and the integrating environment can be independently specified. After a series of tests, it was determined that a communication interval of one second, between the FMU and the Simulink-based controls, was sufficient to capture the required accuracy and dynamics of the of the A/C plant model to support vehicle-level cycle simulations. The following results discussed here are only for a communication interval of one second. The A/C model FMU utilized the Dymola DASSL solver, and the Simulink solver used was ODE1 with a fixed time step of 10ms, consistent with vehicle-level simulations.

Running the A/C model FMU coupled to open-loop inputs in Simulink resulted in a run time of 314 seconds, 70% slower than the stand-alone S-function, seen as the third bar in Figure 6. While this degradation was not expected, it was most likely due to the generation of time events in the FMU associated with the open-loop inputs, and not explored in depth here. Combining the A/C model FMU with the compressor controls resulted in a run time of 528 seconds, indicated by the fifth bar in Figure 6. Then, after adding the cooling fan controls, the simulation time only increased to 564 seconds, about a 7% increase in simulation time. Even with both controls systems integrated with the FMU, the model was able to execute faster than real time, as opposed to the large simulation times recorded utilizing the S-function-based plant model approach.

The benefit of the FMU co-simulation modeling approach is due to the variable time step solver for the physical plant model not having to synchronize lock-step with the discrete-time systems in the model, thus allowing the variable solver to run, on average, larger time steps permitted by the physics.

There have been recent performance enhancements and tool developments in the co-simulation software space, as well as enhancements to hybrid-solver simulations in more recent versions of Matlab, but those studies are not yet complete and could not be included in this paper.

6 Conclusions

A Modelica-based A/C system model has been integrated, closed-loop, with related S-function-based controls in the Simulink environment. The integration was performed with two different approaches, with a DymolaBlock-based S-function for the A/C model, and as a co-simulation FMU. The simulation performance

of the integrated model needs to be sufficiently fast for the purpose of vehicle-level simulations and optimizations. A study has been performed to evaluate and improve the simulation performance of the integrated model. The execution time of A/C model coupled with controls using Modelica FMI/FMU in closed-loop is faster than the real time of the SC03 cycle, and it is 5 times faster than the same A/C model using Dymola-Simulink interface S-function. Combining models that require both variable and fixed time step solvers can lead to serious numerical performance issues and care must be taken in evaluating potential solutions of these hybrid models. These performance issues are expected to grow as the complexity of physics and control models continue to increase, and the demand for faster model turnaround does, likewise.

References

- [1] T. Blochwitz., M. Otter, M. Arnold, C. Bausch and H. Elmqvist, "The Functional Mockup Interface for Tool independent Exchange of Simulation Models," in *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy, Dresden; Germany, 2011.*
- [2] MathWorks. [Online]. Available: <http://www.mathworks.com/help/simulink/ug/modeling-dynamic-systems.html?refresh=true>.

Improving Interoperability of FMI-supporting Tools with Reference FMUs

Christian Bertsch¹ Award Mukbil² Andreas Junghanns³

¹Corporate Research, Robert Bosch GmbH, Germany Christian.Bertsch@de.bosch.com

²Dept. of Informatics, Clausthal University of Technology, Germany, Awad.Mukbil@tu-clausthal.de

³QTronic GmbH, Germany, Andreas.Junghanns@QTronic.de

Abstract

The Functional Mockup Interface (FMI) is more and more adopted by industrial users, increasing the pressure for higher quality and standard compliance of FMI supporting tools. The FMI cross check infrastructure was created to support tool vendors in their quest for quality improvements and to give users some measure of confidence in the tool quality. Currently it is up to the tool vendors which FMUs to submit there. For this reason the features tested in the FMI cross check are incomplete and interpretation of failures is difficult. While for FMI export there is the FMU compliance checker to test a wide variety of FMI features, no means are available today to prove standard compliance for FMI import. This will be overcome by adding reference FMUs to the FMI cross check, testing specific features of the FMI standard for standard compliance and giving detailed feedback, if an importing tool violates the standard. The paper describes the realization and the importance of reference FMUs.

Keywords: FMI, Reference FMUs, Compliance, Testing

1 Introduction

The Functional Mock-up Interface (FMI) is a tool independent approach for model exchange (ME) and co-simulation (CS) (Blochwitz et al. 2011, 2012), and on the way to become the industry standard for exchange of models and cross-company collaboration (Bertsch et al. 2014). Its main purpose is to share and reuse simulation artifacts among a wide variety of tools and environments, by putting the model specifications into a simple compressed file called Functional Mockup Unit (FMU). The FMU contains a model description in XML format, source written in C and/or binaries ready to run and optional components such as documentation, model logo, etc.

Even before the release of the first version of FMI, several modeling and simulation tools started supporting the FMI standard. According to the official website of FMI project, more than 80 simulation tools support FMI version 1.0 and more than 40 support

version 2.0. Many automotive Original Equipment Manufacturers (OEM) have committed themselves to support FMI as exchange format for simulation models.

Because industrial users must rely on the results of FMI-based simulations, the maturity of FMI implementations comes into focus. For this reason, the FMI project has organized the FMI cross check (XC, 2014), where FMI exporting tools can upload test FMUs together with reference solutions as comma-separated values (CSV) files, and importing tools can run those FMUs and report the results. Once the results have been submitted, they are shown in the FMI cross check table at the FMI official website, which helps users to check which tools work well together and which vendors are serious in supporting FMI. In our experience, this has improved the quality and the maturity of FMI support of tools significantly.

The FMU compliance checker (FMU CC, 2016) is an open source software tool that was initiated by the FMI project and implemented by Modelon AB, contracted by the Modelica Association. Its intention is to check compliance of a given FMU with the FMI standard. Through this compliance checker, users can get reports about a wide range of problems that could arise from loading FMUs, which in turn play an important role in validating the tools that create (i.e. export) FMUs.

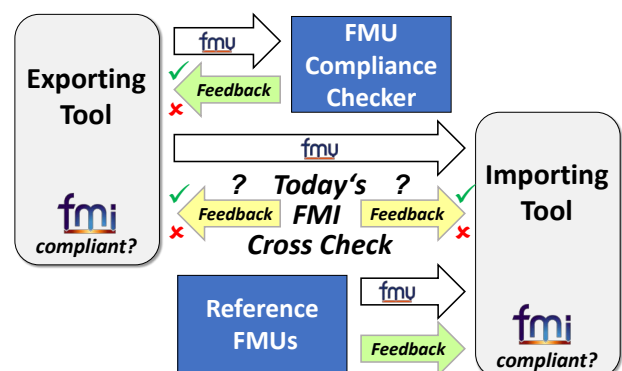


Figure 1: Three complementary ways of FMI compliance testing

According to the “rule #8” of the FMI cross check document (XC, 2014), vendors should test their FMUs using the FMU compliance checker before submitting

them, with or without reference results. This allows vendors to find problems in their implementations early. However, it is up to the exporting tool vendors which FMUs they submit to the FMI cross check, and (as depicted in Figure 1) in cases of problems it can be difficult to find out if it is a problem of the importing or exporting tool. This shows that special testing of FMI importing tools is no less important. We propose to realize this with the help of reference FMUs. Executing these in an importing tool can provide feedback on the FMI standard compliance of this tool as described below.

From the beginning, in the FMI cross check rules (XC, 2014) the important role of reference FMUs was foreseen. The contribution of this work is a step towards realizing such reference FMUs. In Sec. 2 we introduce the concepts, the requirements and the classifications of the reference FMUs. In Sec. 3 we present an initial implementation of reference FMUs. In Sec. 4 we present the means of testing the reference FMUs and first experience with FMI importing tools. Last but not least, we conclude our work and give an outlook to ongoing research in Sec. 5 respectively.

2 FMI and Reference FMUs

The FMI standard comes in textual form, supported by graphs, e.g., of the calling sequence, and XML schemata. (Blochwitz et al. 2012). The FMI 2.0 standard has added also a mathematical description of FMI, which clarifies a large number of concepts. However, the FMI standard is considered to be not fully formalized, which means the standard specifications are not written in formal description language. Formalizing the standard would help automatically generating test cases, and for validating FMUs statically or during runtime.

Formalizing the FMI standard has been partially addressed in some publications, e.g. (Hasanagić et al. 2016), but this seems far away from being realized for the whole standard in the next years. Thus, testing methods from software engineering come into the focus.

2.1 Reference FMUs and Software Testing

In order to test FMI standard compliance, we must test:

1. *Exporting tools*: these tools create FMUs.
2. *Importing tools*: these tools run FMUs.

The exporting tool should follow the FMI standard specifications for creating FMUs, such as providing a correct `modelDescription.xml` file, and use the correct naming and implementation for the functions as stated in the standard. The FMU compliance checker tests the validity of the FMUs and, implicitly, the exporting tools with respect to a large number of FMU properties. However, the FMU compliance checker can only check a finite number of FMU properties for correctness and is extended step by step. If the FMU

compliance checker does not find a problem, it is not guaranteed that the FMU is error free.

In software engineering, the three basic types of software testing are (Bruegge, B., Dutoit, A. H., 2009):

- *Black-Box testing*: testing done by giving inputs and analyzing outputs. Tester does not use source code.
- *White-Box testing*: testing done with knowledge of the internals of the software.
- *Grey-Box testing*: a combination of the black-box and white-box techniques.

Testing FMUs using the FMU compliance checker is grey-box testing because there are open aspects about an FMU (like the `modelDescription.xml`) and closed aspects of an FMU (compiled dynamic link libraries (DLLs) containing the model behavior of the FMU). If the FMU comes with reference CSVs, then the FMU compliance checker sets the inputs according to the input CSV file, runs the FMU and compares the resulting outputs with the reference outputs.

Dealing with the importing tools is different. Most of the simulation tools supporting FMI are commercial, with unknown import mechanisms. Therefore, those importing tools are considered black-boxes, and the only way to test them is to run special FMUs that can spot problems and log errors. These special FMUs are called “*Reference FMUs*”.

2.2 Definition of Reference FMUs

“A reference FMU is an FMU specifically implemented to test compliance with a certain aspect of the FMI standard of a simulation tool. It has the ability to detect and log errors and wrong practices according to the FMI standard specifications. A reference FMU shall be inspected and reviewed before being accepted and published; thus, they must be available in source code and all the creation tools must be freely available.”

This definition makes clear, that FMUs demonstrating a certain feature but exported by some commercial simulation tool cannot be considered as reference FMUs, because they might be too complicated, coming without the full source code and tools that are necessary to create them without having the needed licenses. However, they can also be very valuable. We encourage tool vendors also to export such “*Feature demonstration FMUs*” more often to the FMI cross check.

For the first set of reference FMUs we focus on testing “hard facts”, e.g., testing standard compliance aspects of the importing tools such as data type support and correct calling sequences. Other goals such as testing usability of FMUs with many parameters or large input/output sets, simulation performance with many states or simulation performance with many algebraic/discrete equations are currently not covered. We limit ourselves to “positive” test cases (that the importing tool should accept) and do not consider negative FMU cases (that should be rejected by the

importing tool), because we think that invalid or incorrect FMUs should be detected by the FMU compliance checker and in the FMI standard we have not seen requirements on an FMI importing tool to reject certain FMUs.

2.3 Requirements on Reference FMUs

Reference FMUs shall:

- test specific features of FMI importing tools, and the set of reference FMUs cover many features, and
- follow the FMI standard. A minimum requirement is, that the FMU compliance checker runs successfully (i.e., without warnings or errors) or that a trac issue has been created in case of limitations,
- be simple and well-documented,
- be of high quality; to this purpose they shall
- be reviewed according to defined rules
- be available with all source code and tools that are necessary to create them – in order that the creation process can be inspected and reproduced,
- detect and log the cause of a failure if possible, and
- fit into the FMI cross check infrastructure (if possible), e.g. by providing output signals.

The documentation of the reference FMUs is very important in order to reproduce the creation and interpret the results. It shall contain the following information:

- Authors, change history and review status of this reference FMU.
- The test purpose: What shall be tested with this FMU? Which potential errors of importing tools shall be detected with this FMU? Which capabilities of importing tools shall be tested?
- Implementation hints: How is this FMU created? (E.g. which libraries and tools are used?) Which steps or scripts have to be run to create the FMU?
- Test setup: What are inputs to this FMU (data type, values over time) and what are the expected outputs?
- Is this FMU suitable for the current FMI cross check infrastructure?

We have created a template for the documentation which will be made publicly available. The documentation will be contained within the reference FMUs as html documentation.

2.4 Sources of Reference FMUs and Coverage

There are several ways of deriving reference FMUs: One is to go systematically through the standard and trying to derive FMUs testing coverage and correct implementation of all features. Another is to implement FMUs based on (negative) experience with importing and simulating FMUs created by one and run in some

other (presumably buggy) tool. For creating a reference FMU based on this experience, one should abstract the missing feature or error of the importing tool to a simple example fulfilling the requirements listed above and triggering the erroneous behavior.

In the following we followed both concepts. In the current work, we did not intent creating a complete set of reference FMUs, but we consider this as a starting point that can be extended by developers and users once the reference FMUs will be released to the FMI project and to the public.

For certain aspects of the FMI standard, measures of coverage can be derived: E.g., we have created reference FMUs for all supported data types or we check the allowed function calls in all FMI states and have created reference FMUs that reach all of these states.

2.5 Classifications of Possible Reference FMUs

FMI standard has main features and specifications that should be followed, and from those features we propose this classification of reference FMUs:

- FMUs for testing data type's capability (one for each data type),
- FMUs having dependencies on binaries, e.g. DLLs, or other resources, e.g. CSV files,
- FMUs testing correct interpretation of the `modelDescription.xml` file (version string, GUID, model identifier... etc.),
- FMUs for testing access restrictions depending on variable attributes (i.e. causality/variability combination),
- FMUs for testing the calling sequence as specified in the finite-state machine of the standard document,
- FMUs testing correct event handling (e.g., plausible event localization),
- FMUs for testing optional capabilities, e.g., partial derivatives, and
- Complex FMUs enabling the testing of the interactions of different features (e.g., having continuous states, multiple variables with different attributes, different kind of events).

The first FMUs in this list can be considered as single-feature “*diagnostic FMUs*”, i.e., they are designed to test for a specific feature. A failure of them is very easy to interpret. It is intended that the features to be tested by different diagnostic FMUs are “orthogonal” in the sense that the feedback is as clear as possible.

On the other hand, the more complex “*multi-feature*” FMUs enable the detection of more subtle errors that only occur due the interplay of different effects or due to high complexity of the FMUs.

While in principle there could be completely different reference FMUs for ME and CS, for our first set of

reference FMUs we have created all our reference FMUs for both ME and CS.

2.6 Feedback of Reference FMUs

If an importing tool does not support a feature (e.g., specific data types) it should check this during FMU import based on the information in the `modelDescription.xml` file, and give a meaningful feedback: this is an “announced incompatibility”.

During runtime – especially for diagnostic FMUs – an internal check of the reference FMU will trigger an FMI error, so that the simulation is stopped and a meaningful log message is created. Another possibility is, that the FMU runs without an error, but the outputs of the FMU are wrong. This can be detected, e.g. by the FMI cross check infrastructure.

In the best case the reference FMU is simulated by the importing tool without any error and produces the correct outputs (within specified tolerances).

3 Implementation of Reference FMUs

3.1 Tools to Create our Reference FMUs

The FMU Software Development Kit (FMUSDK, 2014) is considered a good starting point for supporting FMI and implementing reference FMUs. The FMUSDK demonstrates the basic use of Functional Mockup Units (FMUs) as defined by the FMI version 1.0 and 2.0 specifications (FMUSDK, 2014) and implemented by QTronic and freely available in open source. The FMUSDK is suitable to create source code FMUs in a quite simple manner, and already contains many checking mechanism for the functions calling sequence. Therefore, reusing and adding to this implementation helps in creating the first reference FMUs. Furthermore, we used the FMUSDK scripts and libraries for building our FMUs.

For the first reference FMUs we concentrated on FMI 2.0 FMUs for Windows 64-bit binaries. Extending this to other platforms is discussed in Sec. 4.2.

3.2 Preliminary Set of Reference FMUs

The basic structure of our reference FMUs is the same as proposed by the FMUSDK (Figure 2).

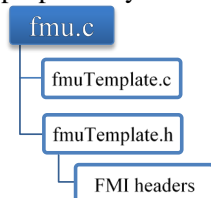


Figure 2: Reference FMU source code structure

The `fmuTemplate.c/.h` source files contain all necessary FMI function implementations and are included by the main FMU source file “`fmu.c`”. We consider this structure versatile, because each FMU can

reuse the template and just modify small code parts for the realization of specific features. As a starting point, we used the original template files from the FMUSDK. For advanced checks for the calling sequence, we modified the templates (see Sec. 3.2.5).

All of the source code is included in the FMU in order to enable inspection and debugging. Except for the FMU mentioned in 3.2.3, all FMUs are separately created as ME or CS FMUs. We will briefly describe our first set of reference FMUs:

3.2.1 FMUs for Testing Datatype Support

While the support for real variables is standard for FMI importing tools, the support for other data types is limited. String inputs/outputs are not standard in many simulation tools; however, it is expected that an importing tool gives a meaningful error message in such a case.

We created an FMU for testing of support for Boolean inputs/outputs (**bool.fmu**): We implemented a simple test of the Boolean data type support. This model demonstrates a simple AND gate logical operation, with two Boolean inputs, executing AND operation and a Boolean output with the result. This FMU uses three model variables: two Boolean inputs and one Boolean output.

Additionally, we implemented an FMU for testing of support for Integer inputs/outputs (**integer.fmu**): It implements the addition of two Integer inputs written to an output.

Further on, an FMU for testing String capability (**string.fmu**) was created: It gets a string input, concatenates it with a locally defined string and writes it to a String output. As stated in the FMI standard, the importer should provide his own allocating and freeing functions (e.g. `calloc`, `free`) along with the logger function. This property gives the importer the ability to manage memory also for the FMU. We use this allocation function to initialize strings, and problems of the importing tool arising from the allocation will be detected.

3.2.2 FMU for Testing FMI Version Number for Future Bugfix Release FMI 2.0.1 (ver.fmu)

This FMU tests if FMI 2.0 importing tools accept FMUs with a version string “2.0.1”. A future version 2.0.1 of the FMI standard will have only clarifications about ambiguities in the FMI 2.0 standard. FMI 2.0.1 FMUs shall be valid FMI 2.0 FMUs (i.e., FMI 2.0 shall be “forward compatible”) as mandated by the FMI development process (FMI DEV, 2015). This reference FMU contains no calculations.

3.2.3 FMU for Testing Events

We created an FMU with internal time events **tEvents.fmu**: it increments an internal integer variable every second; for ME, time events are defined for this

purpose; for CS the events are handled internally in the FMU.

State events are present by the bouncingBall example described in 3.2.6. Event handling shall be tested by more reference FMUs to be developed in the future, see Sec. 5.4.

3.2.4 FMU for Testing the Support of both ME and CS Support in One FMU (*mecs.fmu*)

This FMU contains both ME and CS binaries – without any calculations. The FMI 2.0 standard allows for having both ME and CS in one FMU. It is expected, that importing tools supporting only one of these FMI flavors, nevertheless accept such an FMU. This FMU was inspired by negative experience with one simulation tool only supporting ME import and rejecting FMUs containing ME and CS. This FMU is created with modified build scripts compared to the original FMUSDK, which can create only FMUs supporting either ME or CS.

3.2.5 Testing the Handling Additional Resources

We implemented an FMU testing the calling of an additional dynamic link library (DLL) in the binaries folder (*dll.fmu*): The FMI 2.0 standard allows the exporting tools to include additional binaries to be shipped along with the FMU. These libraries should be placed in the same folder of the compiled FMU binary (or binaries) and to test this capability we have created a simple FMU that contains and uses an additional DLL. This DLL is compiled for each specific target platform (i.e. 32-bit or 64-bit for Windows) with /MT option to include a run-time environment, which is also mentioned by the FMI standard when compiling the FMU source code. In our example, we included “square.dll”, which includes a function that returns the square of a given real value. We use this function to calculate the square of an input and to write it to an output.

Additionally, we implemented an FMU shipped using a CSV file resource (*csv.fmu*): The FMI 2.0 standard enforced the importing tool to provide a clear, IETF RFC3986 compliant URI of resources location during FMU instantiation. According to the standard, this URI could be used for a local resources folder (prefixed by ‘file:///’) or for remote ones (prefixed by ‘http://’, ‘https://’ or ‘ftp://’). The resources folder is intended to be used *only* during FMU instantiation. To test this feature, we created a simple FMU that is shipped with a csv file in resources folder, which is accessed during instantiation. This csv file contains an integer value, and during instantiation we load this file, set the value stored in the csv file to a local variable and calculate the square of this value as an output.

3.2.6 FMUs for Testing the Calling Sequence

The FMUSDK has already implemented many checks regarding the calling sequence. However, we

have gone through all states again and re-considered the allowed function calls. The FMI standard describes the calling sequences for ME and CS using finite-state machines and textual representations (Blochwitz et al., 2012). The finite-state machines and their legends also describe which functions are allowed in which state, including which categories of the variables are allowed to be accessed in each state, regarding the causality-variability-initial attributes.

The FMUSDK functions knows which state the FMU is in by an indicator and a table of allowed functions calls in each state is defined. The following features are already checked by the FMUSDK:

1. Whenever an FMI interface function is called, it checks whether the current state is among the allowed states, otherwise it returns *fmi2Error*. This ensures the detection of erroneous calling sequences.
2. *fmi2Instantiate* \neq NULL. This can happen if:
 - a. there is no valid logger function,
 - b. no allocate/free function provided by importer,
 - c. the GUID is inconsistent, or
 - d. model variables did not initialize successfully.

Those features are clearly described in the state machine graphs. However, there are a few rules mentioned in the textual description of the FMI standard, that should also be checked, which are not yet handled by the FMUSDK. Those features are:

1. *Fmi2SetupExperiment* should be called at least once before *fmi2EnterInitializationMode*, although they can be called in the same state: instantiated.
2. *stopTime* and *Tolerance* are optional, but should be handled if set. If *stopTimeDefined* = *fmi2True*, then the independent variable time must not be set to a value greater than *stopTime*.
3. In case of CS, after an *fmi2SetXXX* call, there must be an *fmi2DoStep* before an *fmi2GetXXX* is allowed. In other words, the order *fmi2SetXXX* – *fmi2DoStep* – *fmi2GetXXX* must be followed.

Checks for these features are implemented in a modified version of the *fmuTemplate.h/.c* files. Several FMUs with an increasing difficulty using these modified template files have been created:

An FMU testing the correct calling sequence for an algebraic calculation for real variables has been implemented: (*real.fmu*): It sums two real inputs and writes them to the output.

An FMU testing the correct calling sequence for one continuous state was created (*dq.fmu*): This is a simple FMU with a continuous state, we used the Dahlquist’s example from the FMUSDK.

Another FMU tests the correct calling sequence for continuous states and state events (*bb.fmu*): this is the BouncingBall example from the FMUSDK, which

contains two continuous states and state events. The simulation time for running this FMU shall be 2s, so that the phase of minimal amplitudes of the bouncing ball including the Zeno effect (Fritzson 2004) is currently excluded from the evaluation.

We used modified `fmuTemplate.c/.h` files with the additional checks to create these FMUs.

3.2.7 Testing Variables Access Restrictions

One of the enhancements brought by the FMI 2.0 standard is the clear definition of variables access restrictions. The standard added more categories to the causality/variability attributes, and the “initial” attribute was added. According to the standard, a specific set of combinations are allowed to be used in describing model variables. Furthermore, there are restrictions in accessing model variables in each state according to the attribute combinations of the variables. These restrictions are considered to be part of the state-machine (Figure 3 and Figure 4), because not only a specific set of functions are allowed to be called in a state, but also a specific set of variables are allowed to be accessed in each state. E.g., discrete variables are only allowed to be accessed when an event is triggered. Another example is that the simulator should never set constant variables. A last example is, that continuous states may not be set via `fmi2SetReal`, but with `fmi2SetContinuousStates`. This is a basic idea of this kind of reference FMU and the authors are still working on these when submitting this paper.

Compared to the current implementation of the FMUSDK, the `template.c/.h` and the specific `fmu.c` files have to be enriched by additional information regarding the variable attributes, as the FMUSDK does not parse the `modelDescription.xml` during FMU creation and this information is currently not available to the implementation of the FMU during simulation.

4 Testing and Using Reference FMUs

We validated our reference FMUs with the FMU compliance checker, and tested them with several tools. This led to three tickets for clarification of the FMI standard version 2.0.1, three tickets for extension to the FMU compliance checker, several tickets for improvements of the FMUSDK and several bug reports regarding errors or improvements the tested tools.

4.1 Implementing an Erroneous Simulator

Two simple open source simulators come with the FMUSDK that import and run FMUs, one for ME and one for CS. We used these simulators to perform first test of the reference FMUs. Then we injected some faults (or wrong practices) in these simulators to check that these errors are detected by the reference FMUs and meaningful feedback is provided. Those faults are

chosen carefully from our experience and from most frequent errors that occur. Examples of these faults are to initialize FMUs before instantiating, or to exit initialization mode before entering it. Another example for CS to call `fmi2SetXXX` and directly call `fmi2GetXXX` afterwards without an `fmi2DoStep` call between them.

4.2 Tests with Importing Tools - Overview

We tested the reference FMUs with 10 different tools for Windows 64-bit binaries. Most of these tools cope very well with normal FMUs generated by other simulation tools. However, with our reference FMUs we detect some limitations and bugs in the involved tools, which are communicated to the tool vendors and implementers.

In Table 1 and Table 2, we depict the result of our checks of the ME and CS Reference FMUs:

Table 1: Results for ME:

	bool	integer	string	tEvent	version	mecs	csv	dll	real	dq	bB
Tool A											
Tool B											
Tool C											
Tool D											
Tool E											
Tool F											
Tool G											
Tool H											
Tool I											
Tool J											

Table 2 Results for CS:

	bool	integer	string	tEvent	ver	mecs	csv	dll	real	dq	bB
Tool A											
Tool B											
Tool C											
Tool D											
Tool E											
Tool F											
Tool G											
Tool H											
Tool I											
Tool J											

■ OK

■ “Announced limitation” of the tool

■ Error or missing feedback for limitations

■ Error; possibly standard clarification needed

■ ME or CS not supported by the tool

Alone for testing the Windows 64-bit combinations of 10 tools with 11 ME and 11 CS FMUs, it took the effort of setting up and running more than 200 simulation models. Additionally, we tested some 32-bit tools, with no significant differences to the 64-bit results. The effort of running the tests and diagnosing the results will be shifted in the future to the tool vendors by including the reference FMUs in the FMI cross check (see Sec. 5.1).

In most tools the FMI import support has a quite mature quality level, and the problems encountered in our tests mostly are mainly due to our very strict diagnostics.

4.3 Problems Detected in Importing Tools

The reference FMUs for data type support revealed problems of several tools for non-real data types: For the FMUs testing for Integer and Boolean input data support (**bool.fmu** and **integer.fmu**), five tools show errors in ME due to violations of the calling sequence: They want to set discrete values in continuous time mode, which is forbidden. (Remark: this is not an incompatibility between hybrid modeling in Modelica and limitations of the Modelica standard; several Modelica-based tools do not have a problem with these FMUs).

The results for the **string.fmu** reflect, that String inputs and outputs are not supported by typical block-oriented simulation tools. However, it is expected that in this case, the tools give a meaningful diagnostic message during FMU import and not just ignore the string in/outputs.

The time events FMU **tEvents.fmu** was successfully run by all except one tool (violating the calling sequence).

A “2.0.1” version string in the **ver.fmu** as foreseen in the FMI 2.0 standard for a future FMI 2.0.1 FMU is problematic for all but one tested importing tools. FMUs following a future FMI 2.0.1 bugfix release, shall be valid “FMI 2.0” FMUs, see (FMI DEV, 2015). Thus, we suggest not to use the “2.0.1” version string in FMI 2.0.1, but “2.0”. FMUs could contain the information that they follow the FMI 2.0.1 standard in an annotation in the `modelDescription.xml` file. This shall be discussed in the FMI project and clarified for FMI 2.0.1.

Only one tool is not able to import an FMU with both ME and CS support contained (**mecs.fmu**), but gives a meaningful feedback.

The **csv.fmu** crashes for one tool both in ME and CS.

The **dll.fmu** detects in one tool a violation of the calling sequence which is not related to DLL-access.

In ME, in none of the tools, except one, problems due to the additional checks in the calling sequence (**real.fmu**, **dq.fmu** and **bb.fmu**) have been detected. For CS, two tools violate the rule, that there may not be a call to `fmi2GetXXX` directly after an `fmi2SetXXX` without an `fmi2DoStep` call in between for **real.fmu**.

5 Outlook

The implemented reference FMUs will first be internally shared within the FMI project, e.g. within the “sandbox” of the FMI cross check infrastructure. The intention is to gather feedback both on the concept and the reference FMUs, to fix bugs and extend the documentation and to give tool vendors the opportunity to fix their implementations. The cross check working

group of the FMI project will review the FMUs and discuss the proposed requirements on reference FMUs.

5.1 Usage within the FMI Cross Check

After the feedback and review phase within the FMI project we plan to publish the reference FMUs on the public part of the FMI project’s resources on GitHub.

After acceptance by the FMI project, the part of the reference FMUs that fit into the FMI Cross Check infrastructure (e.g., w.r.t. to real inputs/outputs) shall be committed there and treated as an exporting tool and extensions to the infrastructure shall be considered.

5.2 Versioning and Indexing

When releasing the reference to the public within the FMI cross check, we will use a version number to refer to this release of the reference FMUs.

With the first release, we will propose an indexing of the FMUs by a naming convention enabling for a serial execution of the reference FMUs in a meaningful order. E.g. single-feature diagnostic FMUs should be performed before complex FMUs, so that the localization of errors is simplified. In this ordering, as few features as possible shall be added from one FMU to the next.

5.3 Extension of Supported Platforms

With the current solution, it is very easy to create reference FMUs for Windows 32-bit and 64-bit binaries. In order to support other platforms like Linux (32-bit/64-bit) or MAC OS X, the port of the FMUSDK to Linux (FMUSDK Linux, 2015) could be used. However, this version is not up to date with the latest version for the FMUSDK. Linux and OS X versions of the FMUSDK would be beneficial.

5.4 Increasing the Coverage

The first set of reference FMUs shall be extended by additional diagnostic and complex FMUs, e.g.:

- for systematically testing all kinds of events (state, time, externally triggered, zero crossings),
- dealing with a larger number of states (e.g. ≥ 10 states with multiple events), and
- testing for optional capabilities of FMUs (e.g., partial derivatives).

Additionally, reference FMUs shall be implemented as proof of concept of new features of a future FMI standard from FMI Change Proposals (FCPs).

5.5 Connected FMUs and Parameter Sets

We also propose to extend the FMI cross check and reference FMUs to connected FMUs: for this purpose one could use connected FMUs inspired by the FMI 2.0 test FMUs (Test FMUs FMI 2.0 ME). However, these FMUs are implemented in Modelica, and need a Modelica tool for the generation of the C-code. Thus, it

would have to be clarified, if this can be done with a publicly available tool chain fulfilling our requirements listed in 2.4 or a re-implementation in C-code is needed. The definition of connected FMUs should be realized using the future System Structure and Parameterization (SSP) standard for the definition of connected FMUs (Köhler et al., 2016). Additionally, the SSP standard could be used to test the correct setting of parameter values to an FMU by the importing tool. For this purpose, the FMI cross check will have to be extended for connected FMUs.

5.6 Additional Benefit of Reference FMUs

Reference FMUs can also provide additional example implementations of FMUs to the FMI community that can serve as a starting point to implement FMI features in a good way; in other words, they give hints to the exporting tools of how typical FMUs could be implemented. This could help, e.g., for the handling of additional resources (binaries or other files), where we have observed many problems of tools in the past.

Reference FMUs can also contribute to clarifying unclear points of the FMI standard, as demonstrated e.g. for the version string reference FMU.

Additionally, with reference FMUs we can provide test FMUs for features that are not yet supported by (many) exporting tools, e.g. string inputs, provision of partial derivatives, and serialization of states.

6 Summary

In the current paper, we present the concept for the creation and usage of reference FMUs. As a starting point, first reference FMUs have been implemented and tests with importing tools have been performed, which led to the detection of several bugs in importing tools. This is seen as a proof of concept for the idea of reference FMUs. They shall be made publicly available first within the FMI project and then publicly by including them in the regular FMI cross check. With FMI community effort, the set of reference FMUs and thus feature coverage shall be increased. This will contribute to the improvement of quality of FMI importing tools.

Acknowledgements

The authors want to thank the members of the FMI cross check working group for their valuable input to our work.

Many thanks also to Torsten Blochwitz (ESI group), Umut Durak (DLR Braunschweig), Dan Henriksson (Dassault Systems), Adrian Tirea (QTronic), Karl Wernersson (Dassault Systems) for their valuable input and fruitful discussions.

Referenced Tools and Online Documents

- (XC, 2014) FMI Cross Check Rules, v3.1, Modelica Association Project FMI: How to Improve FMI Compliance, June 2015. [Accessed online on Dec 16th 2016] <https://www.fmi-standard.org/tools>
- (FMI DEV, 2015) FMI Development Process and Communication Policy, [Accessed online on Jan 22nd 2017] <https://www.fmi-standard.org/development>
- (FMI 2.0 Standard, 2014) Functional Mock-up Interface for Model Exchange and Co-Simulation, Version 2.0, [accessed on Jan 21st 2017] <https://www.fmi-standard.org/downloads>
- (FMU CC, 2016) FMU Compliance Checker, Modelon AB, released by Modelica Association Project FMI [accessed on Dec 16th 2016] <https://www.fmi-standard.org/downloads>
- (FMUSDK, 2014) FMUSDK 2.0.4, QTronic GmbH, July 2014. [Accessed online on Dec 16th 2016] <https://resources.qtronic.de/fmusdk/>
- (FMUSDK Linux, 2015) FMUSDK port to Linux and OS X, [accessed on Jan 6th 2017] <https://github.com/cxbrooks/fmusdk2>
- (Test FMUs FMI 2.0 ME) Testing FMI 2.0 Model Exchange features of connected FMUs, Martin Otter, DLR, [Accessed on Jan 13th 2017] <https://www.fmi-standard.org/downloads>

References

- Bertsch, C., Ahle, E., Schulmeister, U., The Functional Mockup Interface - seen from an industrial perspective, In: Proceedings of the 10th International Modelica Conference 2014, Lund, Sweden
- Blochitz, T., Otter M., Arnold, M., Bausch, C., Clauß, C., Elmquist, H., Junghanns, A., Mauss, J., Monteiro, M., Neidhold, T., Neumerkel, D., Olsson, H., Peetz, J.-V., Wolf, S., The Functional Mockup Interface for Tool independent Exchange of Simulation Models, In: Proceedings of the 8th International Modelica Conference 2011, Dresden, Germany
- Blochitz, T., Otter, M., Akesson, J., Arnold, M., Clauß, C., Elmquist, H., Friedrich, M., Junghanns, A., Mauss, J., Neumerkel, D., Olsson, H., Viel, A., The Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models, In: Proceedings of the 9th Modelica Conference 2012, Munich, Germany
- Bruegge, B., Dutoit, A. H., Object-Oriented Software Engineering Using UML, Patterns, and Java, 3rd edition, Prentice Hall Press, 2009, Upper Saddle River, USA
- Fritzson, P., Principles of Object-Oriented Modeling and Simulation with *Modelica* 2.1, Wiley, 2004, Hoboken, USA
- Hasanagić, M., Tran-Jørgensen, P. W. V., Lausdahl, K., Larsen, P. G., Formalising and Validating the Interface Description in the FMI Standard, FM 2016: Formal Methods, 2016, Springer, Heidelberg, Germany
- Köhler, J., Heinkel, H.-M., Mai, P., Krasser, J., Deppe, M., Nagasawa, M., Modelica-Association-Project "System Structure and Parameterization" – Early Insights, Modelica Conference Japan, 2016
- Pressman, R. S., Software engineering: a practitioner's approach, seventh edition. Publisher McGraw-Hill Higher Education, (2010), New York, USA

The Embedded Simulation via FMI and its Application to the Simulation of Lifetime Tests Including Wear

Julia Gundermann¹ Matthias Thiele² Sebastian Fraulob³ Susanne Walther¹ Karsten
Todtermuschke¹ Uwe Schnabel¹

¹ESI ITI GmbH, julia.gundermann@esi-group.com

²Institute of Electromechanical and Electronic Design, Technische Universität Dresden

³Johnson Electric Germany GmbH & Co. KG Dresden

Abstract

The "Embedded Simulation via FMI" is a new modeling approach which allows for efficient and fast computation of systems with a clear separation of time axis or time scale. For its application the so-called "inner model" is wrapped into an FMU and embedded into an outer model, whose dynamics control the integration. The computation of the embedded model is only utilized on demand. In this way the "Embedded Simulation via FMI" uses the Functional Mock-up Interface for Co-Simulation in a different way than it was provided for. The functionality has been realized within SimulationX prototypically. It is applied to the simulation of the lifetime test of a linear stepper motor including wear in the screw drive, for which the axial play after several months' runtime shall be determined. A significant reduction of computing time while preserving considerable accuracy can be shown.

Keywords: *FMI, wear, SimulationX, simulation, mechatronics*

1 Introduction

The functional mock-up interface (FMI) defines a tool independent standard interface used for the coupling of different simulation tools (FMI, 2014). It allows for the simulation of multi-component systems. Components of the system are packed into an functional mock-up unit (FMU), which is a .zip-file. It contains an xml-file with a model description and C-functions covering the functionality and behavior of the model by functions defined by the FMI-standard. The acceptance and application of the FMI-standard is increasing, there are about 90 tools which support FMI. Depending on the simulation task a model can be exported with (FMI for Co-Simulation) or without its solver (FMI for Model exchange).

The FMI-standard for Co-simulation is used for the coupling of component models, which run independently besides the communication after predefined time-intervals. There are examples for which the simulation of all components is not necessary throughout the whole simulation time. This is the case, if the system's topology

can be divided into an inner and an outer model, which have a separated time axis or time scale. In such cases the pure FMI for Co-Simulation enforces unnecessary computations of the inner model. The "Embedded Simulation via FMI" avoids this by calling the computation of the inner model only on demand.

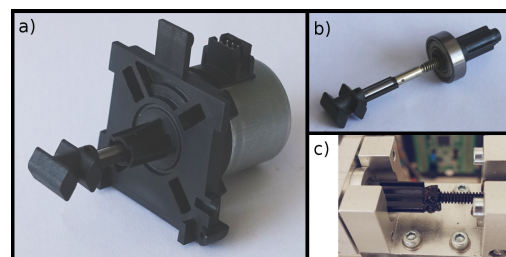


Figure 1. A linear stepper motor based on a screw drive. a) Complete assembly of the motor to be used as adjusting drive. b) Dismantled nut thread with ball bearing. These two components are susceptible to mechanical wear. c) Screw drive without motor in a setup for lifetime testing. The manufacturer of this and other mechatronic components is interested in the simulation of these lifetime tests in order to save time during the development of new components.

One application of this concept is the simulation of lifetime tests of mechatronic (and other) components. Figure 1 shows an linear stepper motor, which underlies wear effects due to abrasion in the screw drive. The manufacturer is faced with the following problem: the development period of a new component is comparable to the duration of lifetime tests. The latter are necessary since they assess the compliance of predefined criteria during the lifetime of the component. One way out of this problem is the virtual simulation of these lifetime tests. Such a simulation consists of a high number of repetitions of nearly identical cycles, i.e. rotating a screw drive back and forth. During the lifetime the component is subject to wear and aging effects. These alter the behavior of the component slowly, i.e. on a long time scale, however barely within the duration of one cycle. However, the current behavior of a component and the wear phenomena cannot be considered separately, since wear is dependent on

the usage of the component, but vice versa wear and aging modify the response of a system, e.g. debris increases friction. The manufacturer is interested in a fast extrapolation of the wear effects, yet not in the computation of each cycle. The virtual simulation of a whole lifetime test including wear and aging can help to derive information about the accumulated effects of loads, temperature, etc., which in turn can be used for the re-planning, shortening or even circumvention of real lifetime tests.

The simulation of the wear effect in a lifetime test consisting of many nearly identical cycles is slowed down by the calculation of the fast degrees of freedom such as the position of the screw drive. Wrapping these fast degrees of freedom into an inner model and embedding this inner model into an outer extrapolation enables to substantially speed up the simulation. In the "Embedded Simulation via FMI" the inner model containing the simulation of one cycle is wrapped into a functional mock-up unit, and then embedded into an outer integration, which extrapolates the wear quantities such as the abraded volume.

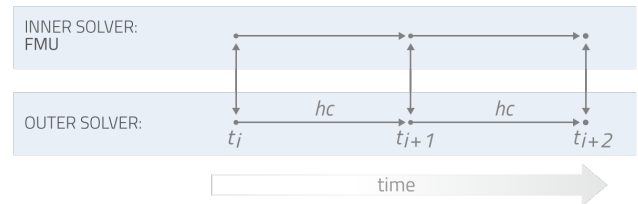
The remainder of this paper is structured as follows: In section 2 the concept of the Embedded Simulation via FMI is illustrated. Section 3 introduces the linear stepper motor. The motor is modeled with components from SimulationX libraries. The screwDrive - component is extended by a wear model, which is also presented here. The preparation of the model for the Embedded Simulation is described in section 4, and its performance is assessed in section 5. The article closes with a summary.

2 Embedded Simulation via FMI

The Embedded Simulation via FMI allows for an accelerated computation of certain types of coupled simulations, where one model's time scale or time axis is separated from the time scale or axis of the other models. The following small examples shall illustrate the idea and what is meant by time scale or time axis separation. One of the examples - the linear stepper - will be illustrated explicitly in the following section.

- Equation-free modeling / molecular dynamics (Kevrekidis and Samaey, 2009) - time scale separation: Given is a system with many fast degrees of freedom (such as molecules), for which the time-like evolution is defined by equations. However, one is interested in the long-term dynamics of averaged quantities such as the evolution of the energy or temperature of the system.
- Simulation of lifetime tests - time scale separation: The example of lifetime tests including wear and aging was introduced above. These tests - and hence its simulation - contain a high number of repetitions of nearly identical cycles. However, one is only interested in the slowly varying quantities such as the amount of wear debris and the increase of axial backlash, but not in the fast degrees of freedom.

FMI for Co-Simulation



Embedded Simulation via FMI

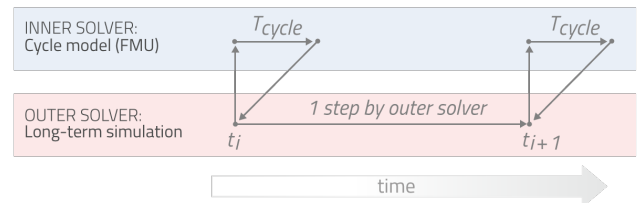


Figure 2. Reuse of the Functional Mock-up Interface (FMI). The upper figure illustrates the usage of FMI in the way it was designed for. Two models are connected and are simulated in parallel independently of each other. They exchange variables at previously defined communication times. The Embedded Simulation via FMI is shown in the lower figure. In contrast to the Co-Simulation only the outer solver runs for the whole simulation. It calls the inner solver at each of its time steps but not at predefined communication steps. The inner solver simulates for the duration of one cycle only. It returns output variables to the outer solver afterwards. Since the time steps of the outer solver are expected to be much longer than T_{cycle} , the outer solver continues its calculation at t_i , but not at $t_i + T_{cycle}$.

- Model predictive control (Dittmar and Pfeiffer, 2004) - time axis separation: The model predictive control uses a time-discrete dynamical model of the process, which is to be controlled. This model shall compute the future behavior of the process depending of the input signals based on an iterative, finite-horizon optimization of the underlying model.

Common to all three examples is the possibility to split the system into an inner model - the fast molecules, one cycle in the lifetime test, or the model predictive control - and an outer model - the evolution of energy or temperature, the extrapolation of accumulated wear debris, or the outer system which the controller is part of. The crucial point is that the inner system needs only to be computed on demand, but not for the whole simulation time of the outer system. For the Embedded Simulation the inner model is wrapped with its own solver into a FMU and embedded into an outer model. The latter determines the time-integration of the overall system. In contrast to the common usage of the FMI for Co-Simulation the inner model does not run independent from the outer model and communicates only at previously fixed points in time. Instead it is run for a predefined (short) time interval, and only on demand given by the outer model. The number of calls of the inner model depends on the simulation task. For systems with time scale separation this number will be

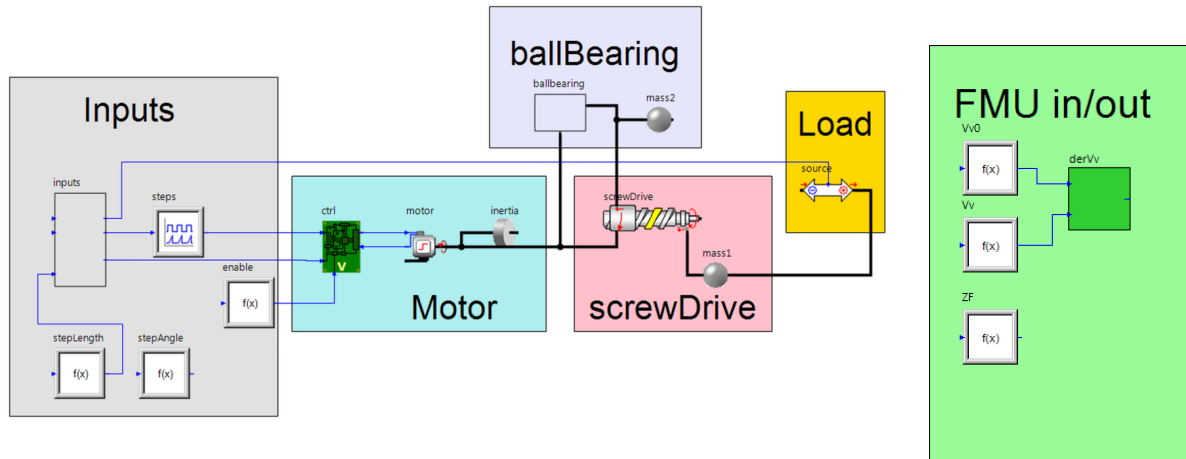


Figure 3. SimulationX model of the linear stepper motor. The components affected by wear are the screwDrive due to abraded volume and the ballBearing due to an increasing backlash. The latter effect is negligibly small and hence not considered further. The FMUin/out block on the right is added in preparation of the FMU-Export.

considerably low compared to the ratio between the simulation times of the outer and inner model. In contrast, for systems with time axis separation no general statement can be made about the number of calls of the inner model besides that it depends on the variables exchanged between the embedded inner and the outer model.

Every time it is initiated, it receives input values from the outer model (such as previous wear debris), while some other variables have default start values (such as spindle position). The inner model is reset, initialized, run, and returns output values to the outer model, and is reset (Technically it is reset at the beginning of the FMU-call). Figure 2 illustrates the difference to the standard of FMI for Co-Simulation.

With the Embedded Simulation via FMI it is possible to speed up simulations preserving considerable accuracy. Furthermore the inner model is replaceable more easily. The presented method was realized and tested in SimulationX for an example with time scale separation. The following section shall illustrate the application of the Embedded Simulation via FMI to the lifetime test of a linear stepper motor.

3 The Linear Stepper

3.1 The SimulationX model

Figure 1 shows a picture of the linear stepper model. In preparation of the lifetime-simulation it was modeled in SimulationX, as shown in Figure 3. It is parametrized such that during one simulation the motor drives the nut to rotate by a predefined angle forth and back. This rotation leads to a translation of the spindle a few millimeters forward and backward. The translation is acting against a constant force (Load) (in the example, $F = 1$ N). By construction the angle of rotation increases stepwise, but not continuously. Each jump triggers an event, which slows down the simulation. The step frequency of 50 Hz on av-

erage for 1000 steps in one cycle leads to a computation time of 1.6 s, versus 20 s of real cycle length.

3.2 The wear in the screw drive

The centerpiece of wear processes is the screw drive, which is modeled by the screwDrive element from the PowerTransmission library in SimulationX. The linear stepper model does not contain other wear or aging phenomena such as the increase of backlash in the ball bearing, since the increasing amount of wear debris in the screw drive is considered as the major effect leading to system failure.

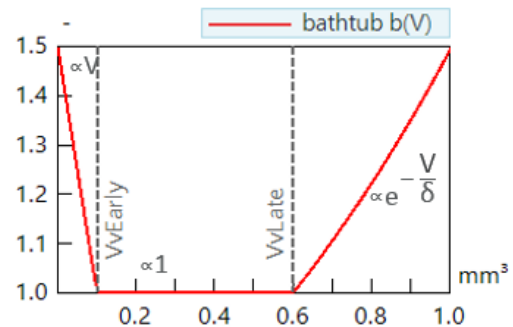


Figure 4. Bathtub curve determining the value of the wear coefficient k (Eq. (2)) and the friction coefficient $\mu = \mu_0 b(V)$. The parameter b linearly decreases for $V < V_{vEarly}$, stays constant until $V = V_{vLate}$ and increases exponentially afterwards. The values for the regime-changes and the exponential increase are optimally derived from experiments, but in general they have to be estimated.

The underlying wear model bases on Archard's law,

$$dV = \frac{k F s}{h}, \quad (1)$$

where V is the volume of wear debris, dV its increase, F the total normal load, s the sliding distance. The material

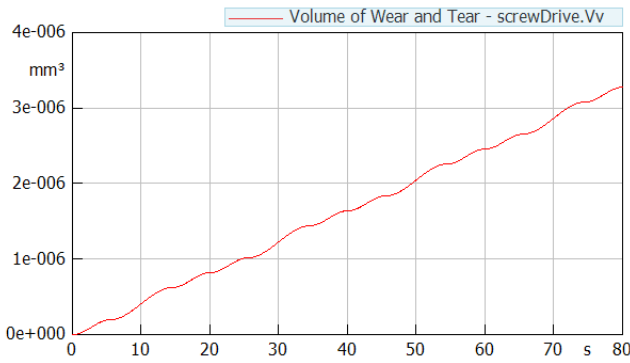


Figure 5. Simulation results of the linear stepper motor, original model. Abraded volume in the screw drive vs. simulation time. The overlaying oscillation is easy to see. Computation step size was $\ll 1$ s.

quantities h and k are the Vicker's hardness of the plastics and the wear coefficient, respectively. In the model here the wear coefficient is assumed to be dependent on the wear debris V , and to run through a bathtub curve (Stachowiak, 2006),

$$k = k_0 b(V). \quad (2)$$

The bathtub is shown in Figure 4.

The axial play in the screw drive increases due to the abraded layer on the flanks. It can be calculated by geometric construction. The wear debris also feeds back on the current wear process, since the friction coefficient is not constant, instead, $\mu = \mu_0 b(V)$.

Technically the wear is modeled by an extension of the ScrewDrive component.

4 Preparation for the Embedded Simulation

Per cycle the linear stepper model computes a change in wear volume. Figure 5 shows the wear debris calculated during four cycles. The increasing curve is superimposed by an oscillation with period equal to the cycle length. For a simulation of approximately 60 days, i.e. rotating forth and back 259.200 times, one is not interested in this oscillation, but more in the long term dynamics. Therefore it suffices to calculate the average change in wear debris per cycle, i.e. $\text{derVv} = (\text{screwDrive.Vv0} - \text{screwDrive.Vv}) / \text{TCycle}$, where Vv0 and Vv are the values of wear debris at the beginning and end of the cycle.

For the preparation of the Embedded Simulation the averaged derivative derVv is calculated in an additional element, cf. Figure 3. Furthermore, a function element ZF is added, which contains the regime changes of the bathtub, i.e., $\text{ZF.y} = \{\text{screwDrive.Vv} - \text{screwDrive.VvEarly}, \text{screwDrive.Vv} - \text{screwDrive.VvLate}\}$. Triggering events in the Embedded Simulation by adding if-conditions containing ZF.y forces the solver to

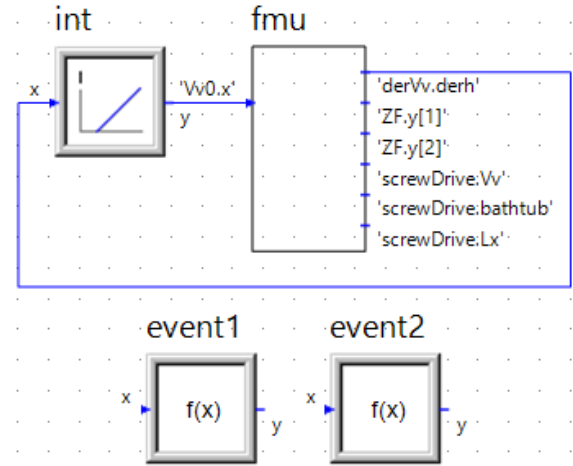


Figure 6. Embedded Simulation via FMI of the linear stepper motor. The wear debris Vv is integrated by the outer solver. The function blocks event1 and event2 are used to let the outer solver catch the transition points in the bathtub curve correctly (cf. section 4).

integrate more accurately at the regime changes.

The linear stepper model is exported as FMU for Co-Simulation 2.0. The wear debris Vv0 at the beginning of the cycle is set as input, the derivative derVv and the bathtub-differences ZF as output. Further output variables or parameters are optional.

The FMU is imported into a new SimulationX model and the modelica-component containing it is modified to make the FMU run in the embedded mode (cf. Figure 2). The main modifications are the replacement of the communication step size h_c by the cycle length TCycle , and the altered calling of the fmu in order to be re-set, re-initialized and run for one cycle everytime it is called. Technically the latter was realized in SimulationX by wrapping all these steps into a single function call.

The output of the variable of interest, i.e. derVv.derh is connected to an integral element, whose output, in turn, is connected to the FMU's input. For each component of ZF a function block is added in order to trigger an event, e.g. $\text{event1.y} = \text{if}(\text{fmu.ZF.y}[1] > 0) \text{ then } 1 \text{ else } 0$, cf. Figure 6.

5 Performance and Validation

To validate the approach, the Embedded Simulation via FMI of the linear stepper is compared with a longterm-simulation of the original model. The simulation time is set to 60 days.

Using the wear coefficient from experiment, $k_0 = 2 \cdot 10^{-10}$, there is little wear and tear within 60 days. The error tolerances of the outer solver have been chosen such that a reduction would not improve the accuracy significantly, but would slow down the computation speed. The deviation of calculated wear debris between original model and Embedded simulation is smaller than 0.05%. The computation time in SimulationX was reduced from

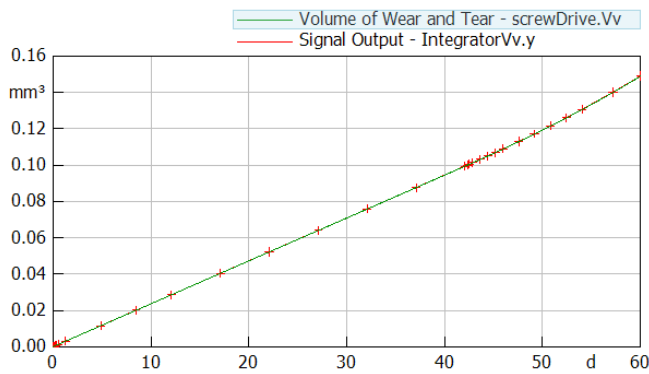


Figure 7. Comparison of Embedded Simulation with FMI (pluses, red) and original model (solid line, green) with accelerated effect of wear and tear. The wear behavior switches between regions after some hours, where $V_v = 10^{-3} \text{ mm}^3$, and after 42 d, where $V_v = 0.1 \text{ mm}^3$, respectively. These behavior changes are passed to the outer solver via triggering events. In the second region, where the wear coefficient stays constant, the time step size of the outer model reaches its maximal value of 5d.

about 2.5 days to 1 minute or by the factor 4000, respectively.

To reveal the dynamics within the region of exponential increase longer simulation times (with the original model) are needed. On the other hand it is also possible to accelerate the wear by increasing the wear coefficient k_0 to $6 \cdot 10^{-8}$. Then the simulation time of 60 days is sufficient to reach all regions of the bathtub curve. The maximum difference between the result values of both simulations is smaller than 0.1%, cf. Figure 7, which also reveals that the solver is able to catch the regime changes in the bathtub. In this case the computation time was reduced from about 2.3 days to 2.5 minutes or by the factor 1300, respectively.

6 Summary and Outlook

The Embedded Simulation via FMI can speed up certain types of simulation tasks by preserving considerable accuracy. The new simulation approach was introduced and motivated by three potential applications. Its successful implementation was proven exemplary for the lifetime simulation of a mechatronic component including wear. The application of the Embedded Simulation to the other two of the named examples or further systems is future work. It remains to be examined whether other integration approaches such as Quantized State Systems methods (Cellier and Kofman, 2006; Casella, 2015) could serve as an alternative.

Acknowledgements

The project Robustness and Reliability Simulation of Mechatronic Systems including Aging and Wear (ROMESA (BMBF, 2015)) runs in cooperation of ESI ITI GmbH Dresden, Institute of Electromechanical and Electronic Design from Technische Universität Dresden,

Dynardo GmbH Weimar and Johnson Electric Germany GmbH & Co. KG Dresden (associated partner). Especially the authors but also the involved companies and institutes like to thank the German Federal Ministry of Education and Research (BMBF) represented by the project coordinator German Aerospace Center (DLR) for supporting financially.

References

- Federal Ministry of Education and Research BMBF. *Projektblatt Romesa*, 2015. URL http://www.pt-sw.de/media/content/Projektblatt_ROMESA.pdf.
- Francesco Casella. Simulation of large-scale models in modelica: State of the art and future perspectives. In *Proceedings of the 11th International Modelica Conference*, pages 459–468. The Modelica Association, September 21–23 2015. doi:10.3384/ecp15118459.
- François E Cellier and Ernesto Kofman. *Continuous system simulation*. Springer Science & Business Media, 2006. ISBN 0387261028.
- Rainer Dittmar and Bernd-Markus Pfeiffer. *Modellbasierte prädiktive Regelung: Eine Einführung für Ingenieure*. Walter de Gruyter, 2004. ISBN 3486275232.
- Functional Mock-up Interface FMI. *FMI 2.0*, 2014. URL <https://www.fmi-standard.org/>.
- Ioannis G. Kevrekidis and Giovanni Samaey. Equation-free multiscale computation: Algorithms and applications. *Annual review of physical chemistry*, 60:321–344, 2009.
- G. W. Stachowiak. *Wear: materials, mechanisms and practice*. John Wiley & Sons, 2006. ISBN 0-262-16209-1.

Integration Modelica with Digital Mockup Tool using the FMI

Shinji Matsuda¹ Hiroshi Toriya¹ Hiromasa Suzuki² Koichi Ohtomi²

¹Lattice Technology Co.,Ltd., Japan, {matsuda, toriya}@lattice.co.jp

²Department of Precision Engg., The University of Tokyo, Japan, suzuki@den.t.u-tokyo.ac.jp
koichi.ohtomi@delight.t.u-tokyo.ac.jp

Abstract

The *Delight Design Platform Project* is managed by *The University of Tokyo* as part of the *Cross-ministerial Strategic Innovation Program (SIP)* which is organized by *Cabinet Office, Government of Japan*. This project recommends using 1DCAE design tools in the concept phase of the product development. In the *Delight Design Platform*, new product concepts are simulated and evaluated as 1DCAE models. One of the objectives of this project is to prototype a tool for translating product concepts to 3D models. This paper describes a method of integrating Modelica with a 3D digital mockup (DMU) tool. The prototype is implemented as an extension of *XVL Studio*, which is a popular DMU tool provided by *Lattice Technology Co.,Ltd.* The integration is implemented using the *FMI (Functional Mockup Interface)*.

Keywords: Modelica, 1DCAE, Functional Mockup Interface, XVL, Digital Mockup

1 Introduction

For many years, Japanese manufacturers placed top priority on developing high quality products at low cost. However, nowadays their primary focus is to develop more attractive products (Ohtomi K, 2015). When thinking about attractive product design, we focused on the fact that there are many potentially interesting product ideas left unattended in the backyard of the companies without being commercialized. In the *Delight Design Platform Project*, we are trying to support the development of attractive products by visualizing the attractive qualities of new ideas. The technology of Model Based Design (MBD), represented by Modelica, is one of the key technologies in our project. In this paper, we propose a method to visualize the MBD simulation results in a DMU tool using FMI. By integrating with the DMU tool, the simulation results can be represented with realistic 3D graphics which makes the presentation more impressive and persuasive. Also, the DMU tool has another useful feature – it supports the inclusion of a 3D human model, which is a very effective way to perform human work analyses. For example, product usability can be evaluated using estimates for product

characteristics such as weight and size, but with an MBD human model it is possible to directly evaluate the load on the human body. In our prototype, we tried to perform a realistic evaluation by modeling the product along with a human model.

This paper describes the integration of Modelica and DMU tools and some of the resulting outputs.

Figure 1 shows the outline of the integration.

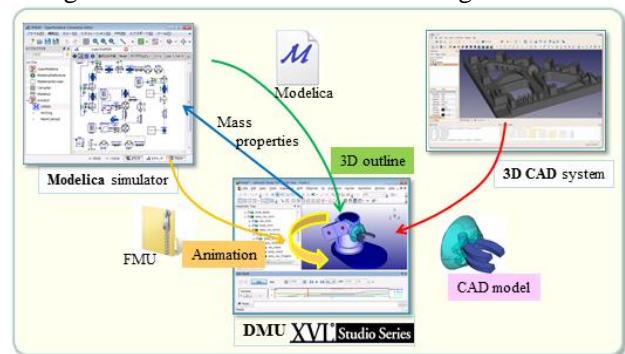


Figure 1 Outline of the integration Modelica with XVL Studio.

2 XVL and DMU Tool

XVL® (eXtensible Virtual reality description Language) is a lightweight 3D file format developed by Lattice Technology Co.,Ltd. And XVL Studio is a digital mockup tool based on the XVL technology (Lattice Technology, 2016). It is widely used for design review, digital assembly and generating technical illustrations (Toriya H. 2008).

The file size of a complete model of an automobile, a piece of construction equipment, an agricultural machine, a train or a ship can easily reach several GB, and it is difficult for CAD systems to handle such large assemblies. In these cases a lightweight 3D file format such as XVL is more effective (Toriya H. 2014; Toriya H., Jablonski M., 2017).

Digital Assembly is the concept of performing a virtual assembly validation without using an actual prototype. It requires managing multiple structures for each product. Using XVL it is possible to define multiple structures in a single file -- for design, for manufacturing and for service. This makes it possible

for the user to easily change the design structure to the desired structure, such as the manufacturing structure.

XVL is also capable of handling the large point cloud data generated by laser scanners. 3D point cloud data is a new technology that makes it easy to convert existing facilities to 3D models. For example, a 3D point cloud scan of a facility makes it possible to check whether new equipment will fit in the existing space. The following chapters will include an example of using 3D point cloud data.

3 3D Model Generation

In the libraries provided by Modelica we want to highlight here is MultiBody library. Since MultiBody library is targeting to 3D mechanical system, we integrate the library with a DMU tool for visualizing mechanical simulation.

3.1 Geometry

Some of the Commercial products for Modelica modeling support 3D visualization (Figure 2). But it is limited only for visualization purpose. In most cases, reuse of the visualization data is not considered.

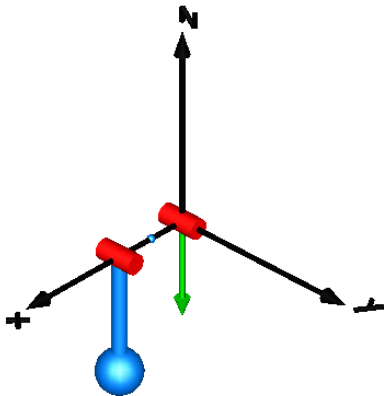


Figure 2 3D visualization in Dymola (Dassault Systèmes, 2016).

Each component in the Multibody library has the parameters of the visualization such as the sphere diameter or the length of the cylinder. We have implemented the parser, which parses the information of the visualization, and a generator which generates B-Spline surface using *XVL Kernel*. Most of the Modelica tools support only polygon data as 3D model. However, NURBS models are more common than polygon models in 3D CAD system (Figure 3). The 3D model can be exported as IGES file format with the standard command of *XVL Studio*. The IGES files can be imported to most of the CAD systems.

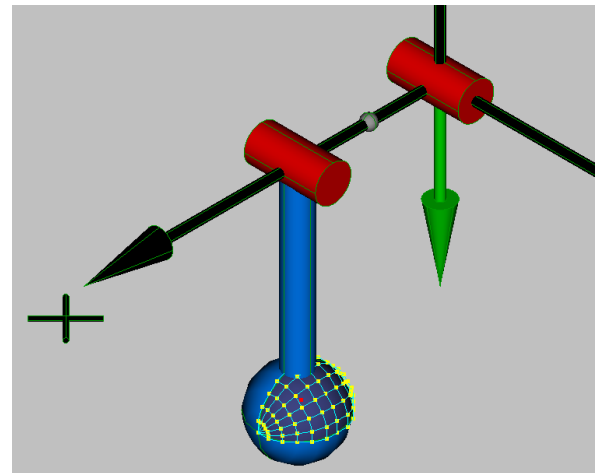


Figure 3 3D model (with NURBS surface)

Our prototype generates 3D models from the Modelica components listed in the Table 1.

Table 1 Components supported in the prototype.

Package	Model
MultiBody	World
MultiBody.Joints	Prismatic
MultiBody.Joints	Revolute
MultiBody.Joints	Cylindrical
MultiBody.Joints	Universal
MultiBody.Joints	Spherical
MultiBody.Joints	SphericalSpherical
MultiBody.Joints	UniversalSpherical
MultiBody.Joints	JointUPS
MultiBody.Parts	Fixed
MultiBody.Parts	FixedTranslation
MultiBody.Parts	Body
MultiBody.Parts	BodyShape
MultiBody.Parts	BodyBox
MultiBody.Parts	BodyCylinder
MultiBody.Parts	PointMass

3.2 Assembly Structure

The CAD system or the DMU tool has an assembly structure which is defined as a tree structure (Figure 4).

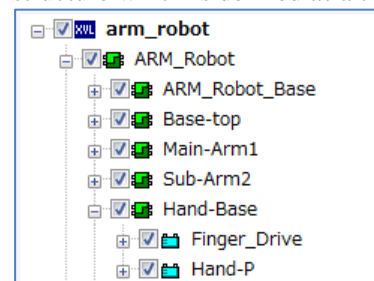


Figure 4 An assembly tree in the *XVL Studio*.

The assembly structure has an important functionality in CAD system (also in DMU Tool), that is locating its sub-components in the model coordinate system. An assembly has the information of its sub-

components and the transformation matrix for each sub-component. User can move a sub-component by changing its translation matrix. This functionality can be used to animating the simulation result in the DMU Tool. The appropriate structure should be created while generating the 3D model from Modelica. For example a Body component connected to a Revolute joint, has to be moved synchronizing with the rotation of the joint (Figure 5). To do this, the 3D model of the joint and the model of Body should belong to one assembly group (Figure 6).

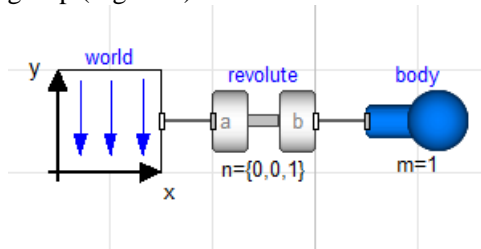


Figure 5 Example model with a Revolute and a Body.



Figure 6 Assembly tree of the example, which shows the shape of the body “prt_body” and the shape of revolute “revolute” are included in one assembly “body”.

4 Kinematics

The MultiBody Library of Modelica is designed to simulate both kinematic and dynamic mechanical models (Martin Otter, et al. 2003). The analysis of kinematics is one of the most popular features of 3D CAD systems. *XVL Studio* (A DMU Tool) also has the functionality of the kinematic analysis. The revolute joint, the linear sliding mechanism and some other types of kinematic objects can be defined in *XVL Studio* (Figure 7).

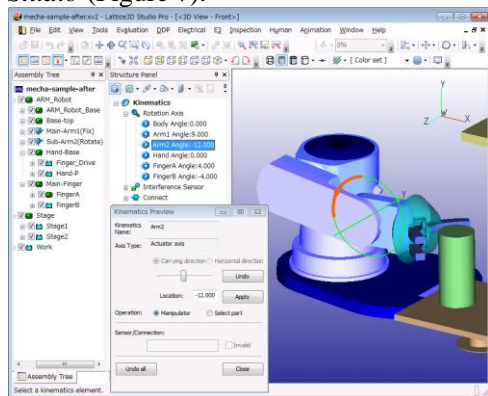


Figure 7 GUI of the kinematic analysis in *XVL Studio*.

While reading the Modelica model, the kinematic information is translated to the kinematic objects of *XVL Studio*. For example the information of the revolute joint of the Modelica is translated to the information of the Rotation Axis object in *XVL Studio*. It has following information as the kinematic object (Table 2).

Table 2 The kinematic information of the Rotation Axis in *XVL Studio*.

information	data type
Joint Name	text
Point of the origin	vector
Rotational Axis	vector
Rotational part	text
Fixed part	text

An instance of the revolute joint defined as “revolute1” in a Modelica file is translated as follows.

The instance name of the Modelica is translated to the name of the Rotation Axis of *XVL*. The Point of the Origin of *XVL* is calculated by traversing the connection of the Modelica model. The parameter “n” of the revolute joint is translated to the vector of the rotational axis in *XVL*. The Rotational part and the fixed part in the *XVL* are also found by traversing the connection of the Modelica model.

This information is used for the 3D animation in *XVL Studio*.

5 Running Simulation

The DMU Tool does not have the functionality to generate the executable module for simulation. The prototype runs the simulation using FMI. We used FMI Library by JModelica.org in our prototype for running the simulation. FMU module is generated by using a small batch command. Following is an example batch command for JModelica.

```
@echo off
call C:\JModelica.org-1.17\setenv.bat
if %errorlevel% neq 0 exit /b 1
echo from pymodelica import compile_fmu
>>_t3.py
echo mn = '%1'>>_t3.py
echo mf = '%2'>>_t3.py
echo my = compile_fmu(mn, mf, target='cs',
version='2.0')>>_t3.py
"C:\Python27\python.exe" "_t3.py"
if %errorlevel% neq 0 exit /b 2
```

This batch file takes 2 arguments. One is the name of the model, and the other is the name of the Modelica file. It generates a python script as _t3.py and call python.exe.

5.1 FMI Interface

The FMI (Functional Mockup Interface) is the standard which enables running the simulation from any tools (FMI-Standard.org, 2014). There are publicly available

FMI implementations. For example, FMI Library (JModelica.org, 2016) is a library developed by JModelica.org can be downloaded in source code or binaries for Windows. Using the library, all of the steps for running the simulation can be executed by simple function call, like unzipping the FMU module or parsing the modelDescription.xml. Our prototype is implemented to use the FMI Library.

5.2 3D Matrix

The connector of the MultiBody Library is defined as the Frame. Frame model is defined in the package Interfaces as following.

```
connector Frame
  SI.Position r_0[3]
  Frames.Orientation R
  flow SI.Force f[3]
  flow SI.Torque t[3]
end Frame;
```

The position vector r_0 is directed from the origin of the world coordinate system to the origin of the Frame. The orientation object R describes the relative orientation between the world frame and the Frame. From these parameters, we can generate a transformation matrix which is used to animate the 3D object in DMU Tool.

$$|A| = \begin{vmatrix} R.T[1,1] & R.T[2,1] & R.T[3,1] & r_0[1] \\ R.T[1,2] & R.T[2,2] & R.T[3,2] & r_0[2] \\ R.T[1,3] & R.T[2,3] & R.T[3,3] & r_0[3] \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (1)$$

This translation matrix moves the 3D object in the DMU model. The values r_0 and $R.T$ of the components in the Modelica model are referred while running the simulation and are used for the animation of 3D model.

5.3 Using CAD Model

Using the 3D CAD model of the existing products, the presentation of the simulation result becomes more realistic. The DMU Tool has the functionality to import CAD model. Most of the DMU Tools support many types of the CAD format. Figure 8 is the list of the supported formats in *XVL Studio*.

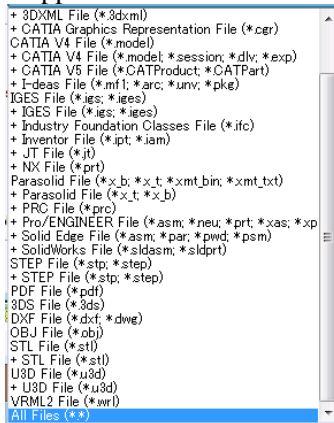


Figure 8 CAD format supported by XVL Studio.

To use the existing CAD data in our prototype, just drag and drop the CAD file to *XVL Studio*. Following instructions show how to use the CAD data in the prototype.

- Import Modelica model to the prototype.
- The 3D visualization model is generated. (Figure 9)
- Import CAD data to DMU Tool. (Figure 10)
- Move parts of the CAD model to the group under the visualization model. (Figure 11)
- Run simulation. (Figure 12)

Using the integration Modelica and DMU Tool the simulation can be visualized with CAD model with simple operations like this.

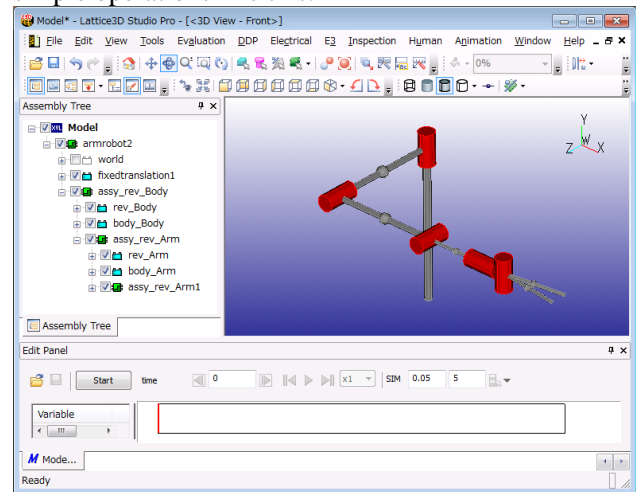


Figure 9 3D visualization model generated with the prototype.

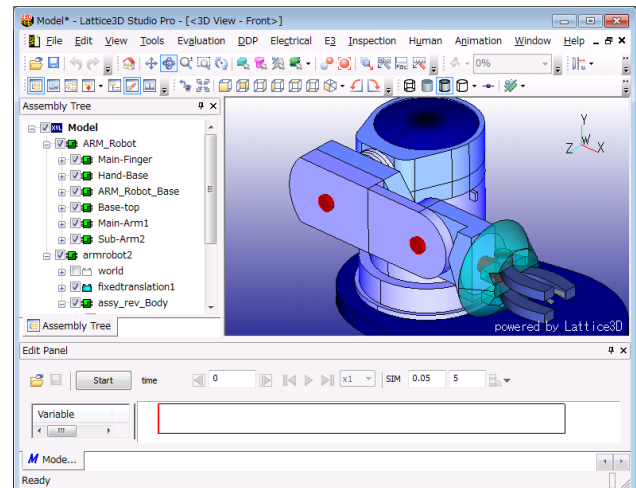


Figure 10 CAD model is imported over the visualization model.

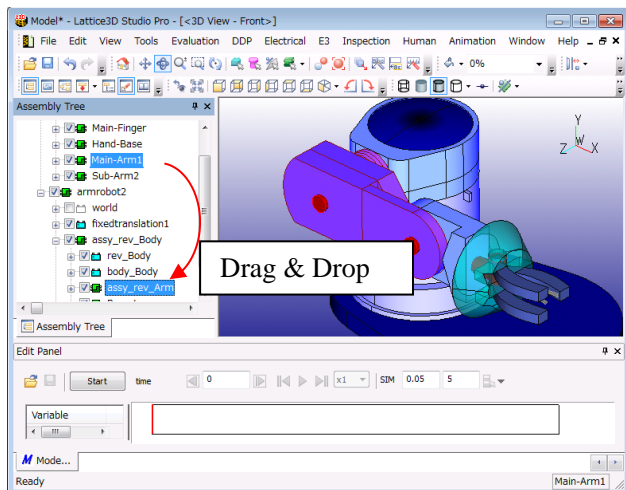


Figure 11 Move CAD model to the group generated with the 3D visualization model.

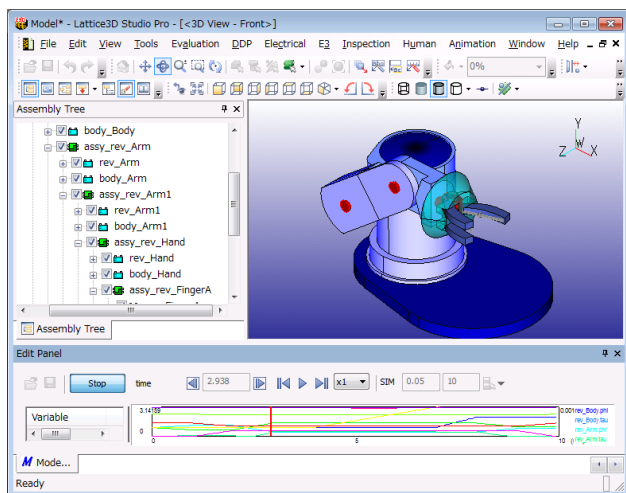


Figure 12 Simulation is visualized with the CAD model.

6 Example

We create a crane model in our project. The CAD data of a crane is provided by a Japanese manufacturer of mobile cranes.

6.1 Crane Model

Figure 13 is a crane model created with *Dymola*. It consists of 2 revolute joints for simulating the fall over problem. 2 revolute joints and 4 prismatic joints are for simulating the movement of the boom. One prismatic joint held in the revolute joint and a universal joint are for the extending of the wire. Figure 14 is a screenshot of the prototype showing the 3D model generated from the Modelica file. As instructed in chapter 5.3, CAD model of the product can be imported (Figure 15). Figure 16 is the screenshot of the final 3D model, which includes the 3D skeleton model generated from the visualization information defined in Modelica, CAD data of the product design and point cloud of the construction field. The point cloud is measured with the laser scanner and imported to *XVL Studio*.

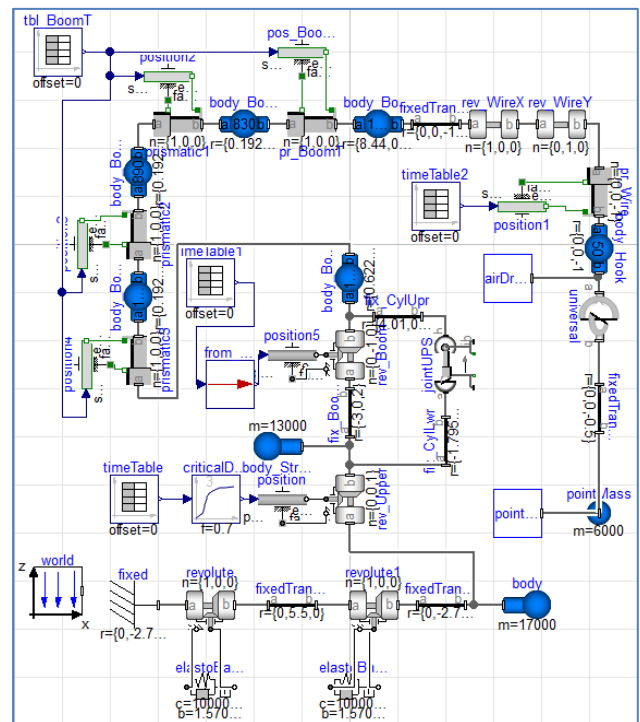


Figure 13 Modelica model of a mobile crane.

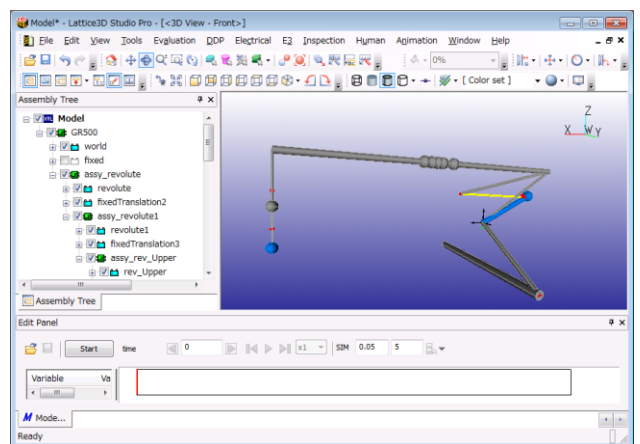


Figure 14 3D skeleton data generated by the prototype

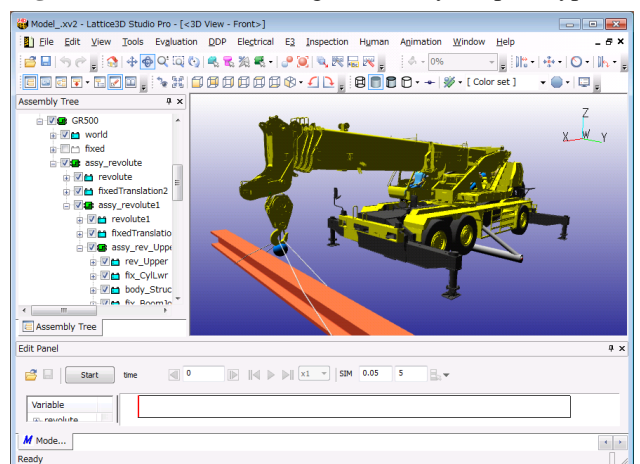


Figure 15 A screen shot after imported the CAD model.

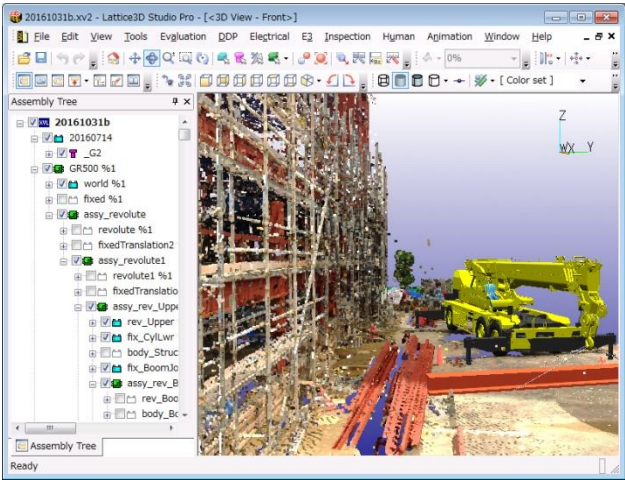


Figure 16 A screen shot after imported point cloud.

Using this model, we have simulated the swing of the load while turning the Boom (Figure 17). Figure 18 shows the simulation of the problem of fall over of the full vehicle.

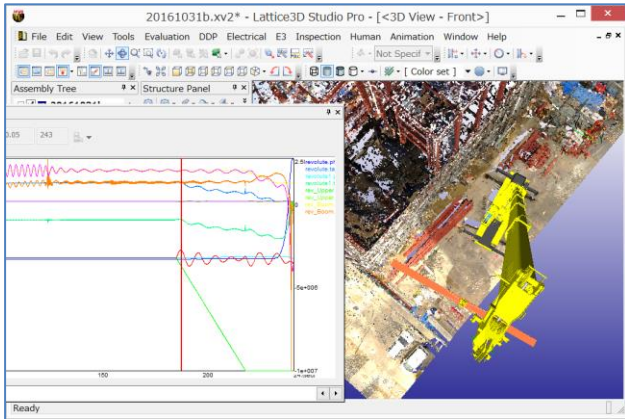


Figure 17 A screenshot simulating the swing of the load while turning the Boom.

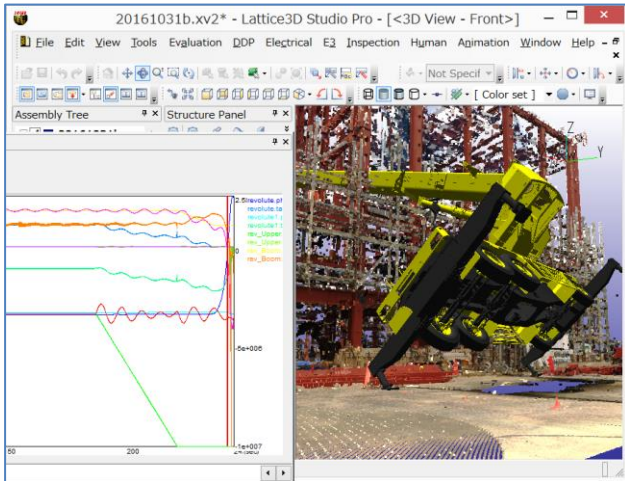


Figure 18 Simulating the problem of fall over when the boom was tilted.

7 Human Model

Analyzing the human interaction, during the production in the factory is one of the important use-case of the DMU tool. Some of the DMU Tools has the functionality of evaluate the posture of the worker using 3D human model. On the other hand there are some biomechanics software enables to analyze the human muscle bone model. For example *DhaibaWorks* developed by *National Institute of Advanced Industrial Science and Technology* (AIST, 2016) is well known in Japan. These bio-mechanics software are for the expert users in the laboratory of the universities or the enterprise. Our target in the *Delight Design Platform* is to develop a human model easy to use for engineers in the manufacturer using the Modelica technology.

7.1 Muscle-Bone Model Prototype

Our first prototype was a generator, which generate a simple 3D muscle-bone model using the kinematics of *XVL*. The generator reads a *BVH* file, and generates a *XVL* model referring the structure data in the *BVH* file. The *BVH* file format is originally developed by *Biovision* as a motion capture data file format (Autodesk, 2016).

In this model, there are 16 skeletal joints. We defined degree-of-freedom for each joint, and associated prime mover muscle (Table 3).

Table 3 Mapping of the skeletal joint and muscle.

skelatal joint	axis	associated muscle
Left/Right Thigh	x	psoas l/r
	y	gluteus l/r
	z	piriformis l/r
Left/Right Leg	x	femoris l/r
Left/Right Foot	x	surae l/r
	y	peroneus l/r
Chest	x	abdominis
	y	ex oblique
	z	in oblique
Left/Right Shoulder	y	trapezius l/r
	z	pectoralis min l/r
Left/Right Arm	x	pectoralis maj l/r
	y	deltoid l/r
	z	spinatus l/r
Left/Right Forearm	x	biceps l/r
	y	pronator l/r
Left/Right Hand	x	digitorum l/r
	y	carpi l/r
Head	x	sternocleido
	y	splenius

The basic model of a set of joint and muscle is shown in Figure 19.

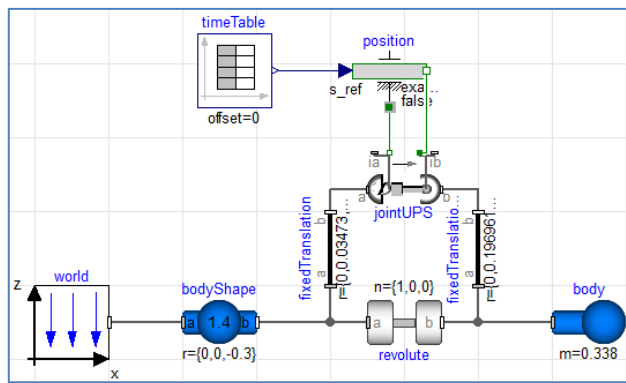


Figure 19 The basic model with one rotational joint and one JointUPS.

The Revolute component is associated to one degree of freedom of a skeletal joint. The JointUPS component is associated to the prime mover muscle. Similarly, we generated instance for all joints (Figure 20).

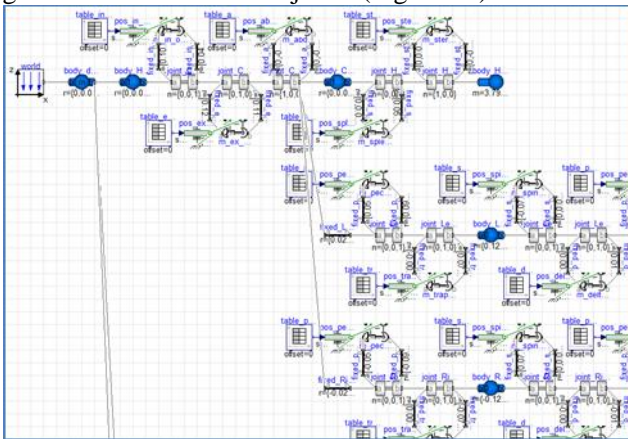


Figure 20 A Modelica model of the muscle bone model.

We prepared a 3D model with kinematic definitions referring a 3D human model for anatomy (Figure 21). The prototype generates the Modelica model from the geometric and kinematic information in the 3d model.

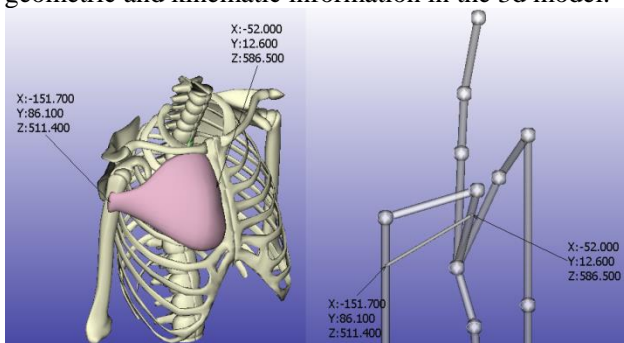


Figure 21 Defining the fixing position of the muscle to the bone. The left is an anatomy model and the right is a kinematic model.

A skin model created with CG software is added just like the CAD model written in the section 5.3 (Figure 22).

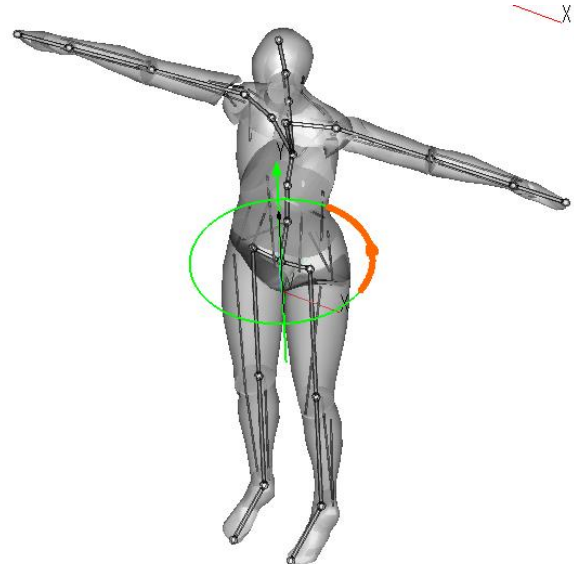


Figure 22 A 3D muscle bone model with skin.

In this model the properties of mass and the inertia tensor are calculated by the skin data of the 3D model. This concept is as same as the one in the paper *Redundancies in Multibody System and Automatic Coupling of CATIA and Modelica* (Hilding Elmqvist et al., 2009). Also the value in the TimeTable which is the input of the JointUPS is embedded from the value in the BVH motion data. Figure 23 is a table plot of the motion data in *Dymola*. The input of the JointUPS is the relative distance and it is calculated from the Euler angles of the joint contained in the BVH file.

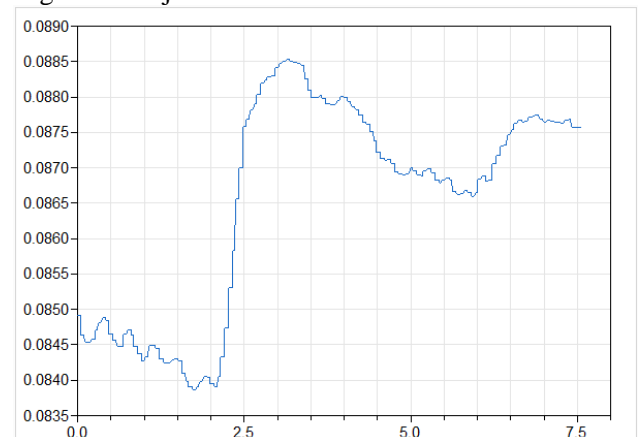


Figure 23 A table plot of the motion data embedded in the Modelica model.

We have visualized the force of the JointUPS as color mapping. Figure 24 is a sample showing the color mapping of the human model while using the hair-dryer. Our prototype generates the color mapping as a key frame animation of *XVL Studio*.



Figure 24 The visualization with the color mapping.

Following charts are samples of the simulation result. Figure 25 is a chart showing the length of the JointUPS placed at the position of the biceps. Figure 26 is the force of the JointUPS. Since the sampling rate of the motion capture is not high, the force of the JointUPS is filtered with *Blocks.Continuous.Filter* component.

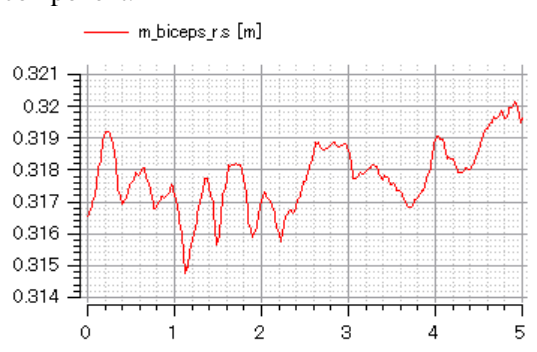


Figure 25 The length of the biceps.

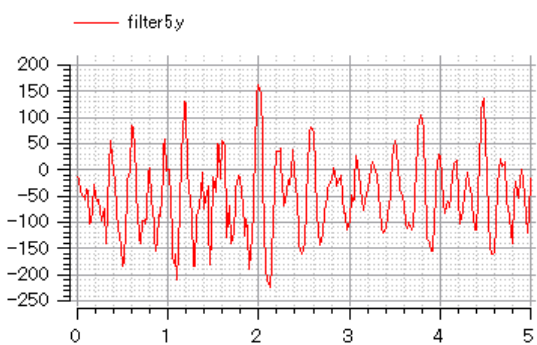


Figure 26 The force of the biceps.

7.2 Reaction Force on the Ground

Our first prototype described in the previous section has some problems. For example it does not simulate the reaction force from the ground. There is a contact library for the MultiBody Library, proposed as the IdealizedContact (Oestersötebier et al., 2014). Also a simple point contact model is proposed in the paper

Kinematic and Dynamic Analysis for Biped Robots Design (David M., 2012).

We have used a modified one point contact model in the following example. This model is a passive dynamic walking model, which is intended to simulate the human gait (Figure 27, Figure 28).

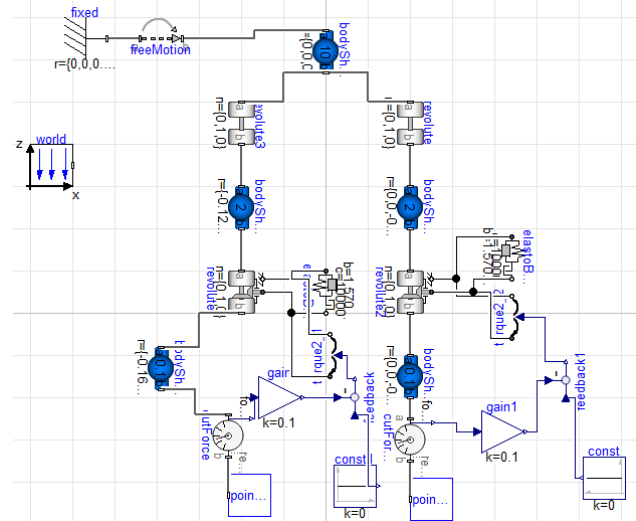


Figure 27 The passive dynamic walking model.

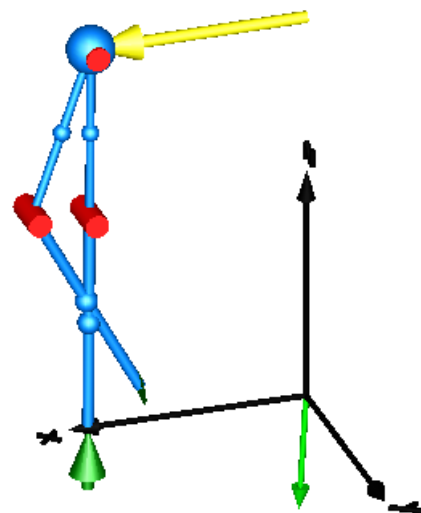


Figure 28 Animation view of the passive dynamic walking model.

In this model, 2 *BodyShape* components corresponding to the legs are connected to the *PointContact* component. The torque of the revolute joint corresponding to the knee is controlled as zero when the reaction force from the ground is zero. It holds the knee angle while the reaction force from the ground is above zero.

The gravity vector is tilted from Z axis. With the gravity the walking motion is continued. Since the legs of the model are placed at the same position in Y direction, the model fell down sideways after 10 steps.

The simulation can be animated with the skin of the human model installed with *XVL Studio* (Figure 29).



Figure 29 The walking model with the skin in *XVL Studio*.

8 Future Work

In the future work, we plan to extend the functionality of the prototype, and evaluate in the actual product design.

In the next prototype of the human model we will not use the muscle bone model. The forces of the muscle will be calculated from the torque of the joint inside the DMU Tool. Because of the detailed evaluation of each force of the muscles are not required in the use case of the DMU Tool. The reaction force on the ground described in the section 7.2 will be included in the next prototype. And the calculation of the gravity center of the whole body will be included. It helps to evaluate the working posture which is the main use case of the human model of the DMU Tool. In the Figure 30 left image is a typical human model holding a box which is created with *XVL Studio*. Since the gravity balance is not considered, the simulation model fall down foreword like the image right side.

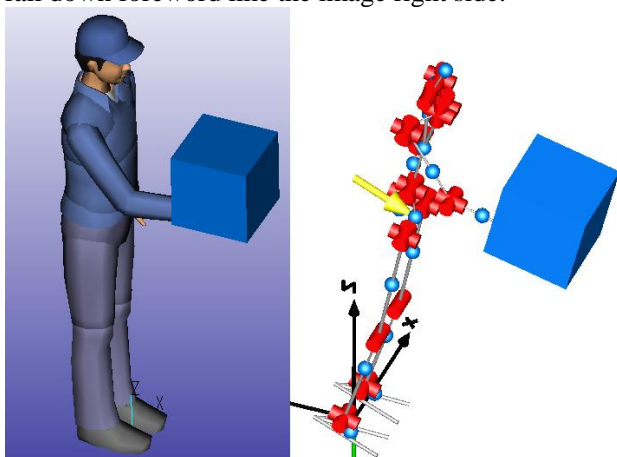


Figure 30 A typical human model holding a box.

Figure 31 shows the posture considered gravity balance. In this way, by using Modelica simulation in DMU Tool more natural posture can be created.

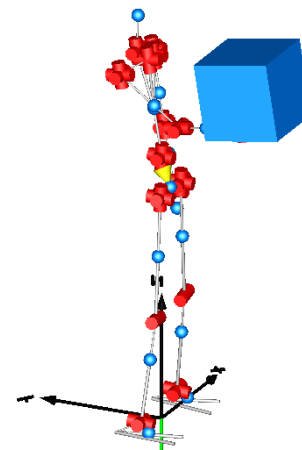


Figure 31

Also, the automatic generation of the models corresponding to human bodies of various physiques is planned.

9 Conclusion

This project demonstrated significant advantages using a DMU tool as a Modelica front-end. The advantages are as follows.

- Enables visualization of simulation results with 3D CAD models and/or 3D scan data.
- Enables easier 1D modeling by using the 3D user interface of the DMU Tool.
- Better visualization of results will promote the use of Model based design.

Further advantage can be expected with the future work described in the previous section.

Acknowledgements

The authors wish to thank Takayuki Kosaka (TADANO LTD.) and Marc Jablonski for their contributions and feedbacks. This research and the prototype were supported by New Energy and Industrial Technology Development Organization (NEDO) of Japan, and we would like to thank them for their assistance.

References

- Autodesk (2016): BVH File Specification.
<http://www.autodesk.com>
- Dassault Systèmes (2016): Dymola 2016
<http://www.Dymola.com>
- David Mauricio Alba Lucero. (2012): Kinematic and Dynamic Analysis for Biped Robots Design.
- Felix Oestersotebier, Peng Wang and Ansgar Trachtler. (2014): A Modelica Contact Library for Idealized Simulation of Independently Defined Contact Surfaces.
- FMI-Standard.org (2014): Functional Mockup Interface for Model Exchange and Co-Simulation Version 2.0, July 25, 2014. <https://www.fmi-standard.org>

Hilding Elmqvist, Sven Erik Mattsson, Christophe Chapuis.
(2009): Redundancies in Multibody Systems and Automatic Coupling of CATIA and Modelica. In: Proceedings of the 7th International Modelica Conference.

JModelica.org (2016): FMI Library 2.0.2

<http://jmodelica.org>

Lattice Technology (2016): XVL

<http://www.lattice3d.com/>

Martin Otter, Hilding Elmqvist and Sven Erik Mattsson

DLR; Dynasim: (2003): The New Modelica MultiBody Library. In: Proceedings of the 3rd International Modelica Conference, Linköping, November 3-4, 2003.

National Institute of Advanced Industrial Science and Technology (AIST) (2016): DhaibaWorks

<http://www.dhaibaworks.com>

Ohtomi, K. (2015): Kansei Modeling for Delight Design based on 1DCAE Concept. In: Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015. doi: 10.3384/ecp15118

Toriya, H. (2008): 3D Manufacturing Innovation. doi: 10.1007/978-1-84800-038-4

Toriya, H. (2014): Manufacturing Innovation Based On Lightweight 3D Technology. In: The 4th IEEEJ International Workshop on Image Electronics and Visual Computing 2014.

Toriya H., Jablonski M. (2017): 3D Manufacturing Evolution: Evolutionary Change in Global Manufacturing with Digital Data. ASIN: B01N29ZFZM

Solving Large-scale Modelica Models: New Approaches and Experimental Results using OpenModelica

Willi Braun¹ Francesco Casella² Bernhard Bachmann¹

¹FH Bielefeld, Bielefeld, Germany, {willi.braun, bernhard.bachmann}@fh-bielefeld.org

²Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy,
francesco.casella@polimi.it

Abstract

Modelica-based modeling and simulation is becoming increasingly important for the development of high quality engineering products. Therefore, the system size of interest in a Modelica-based simulation is continuously increasing and the traditional way of generating simulation code, e.g. involving symbolic transformations like matching, sorting, and tearing, must be adapted to this situation. This paper describes recently implemented sparse solver techniques in OpenModelica in order to efficiently compile and simulate large-scale Modelica models. A proof of concept is given by evaluating the performance of selected benchmark problems.

Keywords: Modelica, large-scale, sparse solver techniques

1 Introduction

The design and safe operation of modern large-scale cyber-physical systems requires the ability to model and simulate them efficiently. The Modelica language is optimally suited for the modelling task, thanks to the high-level declarative modelling approach and to the powerful object-oriented features such as inheritance and replaceable objects. On the other hand, as noted in (Casella, 2015), until recently the development of Modelica tools has been focused on the modelling of moderate-sized models, optimizing the simulation code as much as possible by means of structural analysis and symbolic processing of the system of equations.

Large system models are usually characterized by a high degree of sparsity, since each component interacts only with a few neighbours, so that each differential-algebraic equation in the model only depends on a handful of variables. The availability of reliable open-source sparse solvers (Hindmarsh et al., 2005; Davis and Natarajan, 2010) and of cheap computing power and memory even on low-end workstations opens up the possibility of tackling much large system models, featuring hundreds of thousands or possibly millions of equations, exploiting the sparsity of such models for their solution.

In particular, the interest in the use of Modelica for the modelling and simulation of national- and continental-sized power generation and transmission systems recently

motivated a first exploratory effort in this direction, using OpenModelica as a development platform, see (Casella et al., 2016). The methods implemented for the power system studies also allowed to efficiently simulate the cooling blanket of the future DEMO nuclear fusion reactor, which requires the modelling of thousands of individual heat-exchanging pipes, see (Froio et al., 2016).

The goal of this paper is threefold: to discuss different strategies for the simulation of large-scale Modelica models using sparse solvers; to describe an implementation of such strategies in the OpenModelica Compiler (OMC), using open-source solvers; finally, to present and discuss the performance obtained in a number of benchmark cases. The numerical methods are discussed in Section 2. The simulation performance is analyzed on three sets of benchmarks: the ScalableTestSuite library (Casella, 2015; Casella and Sezginer, 2016), some large power system models (Casella et al., 2016), and large high-fidelity models of the cooling system of the future DEMO nuclear fusion plant (Froio et al., 2017); results are reported in Section 3. Finally, Section 4 concludes the paper and gives an outlook to future work.

2 Solving Modelica Models

2.1 ODE mode

2.1.1 Symbolic Transformation Steps

In common Modelica tools the compile process can be summarized with the following steps, which are also explained in (Cellier and Kofman, 2006):

Flattening The Modelica model is transformed by the front-end into a flat representation, consisting essentially of lists of variables, functions, equations and algorithms.

Pre-Optimization In this phase a basic structural analysis of the differential-algebraic equations (DAE) is performed, e.g. detecting the potential states and discrete variables, eliminating alias variables.

Causalization This is a basic step in a Modelica Compiler, the so-called BLT-Transformation. Matching, sorting, and index reduction algorithms are applied

in order to causalize the DAE and transform it to a system of ordinary differential equations (ODE).

Post-Optimization In this phase further optimization processes are applied on the equation system, e.g. the optimization of algebraic loops, like tearing, or the generation of corresponding symbolic Jacobians.

Code-Generation The final step after the symbolic manipulation is the target code generation for the optimized system in order to perform the simulation.

For this description and without lack of generality, and for clarity of the presentation, only the continuous part of the DAE is considered in the following. The result of the **Flattening** is the equation system:

$$\begin{aligned} F(t, \dot{x}(t), x(t), u(t), y(t), p) &= 0, \quad t \in [t_0, t_f] \\ x(t_0) &= x_0 \end{aligned} \quad (1)$$

where $\dot{x}(t) \in \mathbb{R}^{n_x}$ are the potential state derivatives, $x(t) \in \mathbb{R}^{n_x}$ are the potential states, $u(t) \in \mathbb{R}^{n_u}$ are the inputs and $y(t) \in \mathbb{R}^{n_y}$ are the algebraic variables. For simplicity, the initial conditions of the DAE states are given by x_0 . Introducing $z = (\dot{x} \ y)$, denoting the unknowns of the DAE, and $v = (x \ u)$, denoting the known variables, the DAE can be re-written as

$$F(z, v) = 0 \quad (2)$$

that is basically the result of the **Pre-Optimization**.

The conceptual idea of the DAE **Causalization** commonly used in Modelica tools is to get an ordering of the unknowns $z(t)$, which enables to solve them sequentially

$$z = G(v) \in \mathbb{R}^{n_x + n_y} \quad (3)$$

If index reduction is necessary, some of the potential states and state derivatives become algebraic and the number of equations might change. The general form of the causalized system consists of a sequence of assignment statements including implicit systems of equations (algebraic loops)

$$0 = g_i(z_i, z_1, \dots, z_{i-1}, x, u), \quad i = 1, \dots, k \quad (4)$$

where

$$z = (z_1, \dots, z_k), \quad z_i \in \mathbb{R}^{n_i}, \quad \sum_{i=1}^k n_i = n_x + n_y.$$

Finally, the ODE may be re-written

$$\dot{x} = f(x, u, p, t) \quad (5)$$

$$\hat{y} = h(x, u, p, t) \quad (6)$$

where \hat{y} are the outputs of the system. Note that the other algebraic variables of y are considered to be internal to the ODE in this representation.

In the **Post-Optimization** mainly algebraic loops are torn down (Täuber et al., 2014) and the symbolical Jacobians are determined where applicable. Also the sparsity pattern of equation (5) is detected, which can be employed for the numerical jacobian calculation of the integration method (see also (Braun et al., 2012)).

2.1.2 Numerical Solving Process

For the simulation of the generated ODE equation (5) a numerical integration method for solving the differential equations as well as linear and non-linear system solvers for the implicit equations (4) are needed. In the next section the utilized methods and the exploitation scope for sparsity are described.

The numerical integration can be performed with explicit or implicit methods, whereby the implicit approaches are used in a Modelica environment more often, since most problems arising in practice are stiff. For explicit methods the next step can be calculated by

$$x(t + \delta t) = \Phi(x(t), u, p, t, \delta t, f), \quad (7)$$

whereby Φ is calculated by explicitly evaluating the function f in formula (5). Therefore, sparse methods can only be applied for calculating the solution of algebraic loops with respect to equation (4). The handling of sparse algebraic loops is described below.

For implicit methods the next step has to be calculated by

$$x(t + \delta t) = \Psi(x(t + \delta t), u, p, t, \delta t, f), \quad (8)$$

whereby the evaluation of Ψ involves the solution of a non-linear system using equation (5). The most widely used method for solving such non-linear systems is Newton's method and the core of it is to solve consecutive a linear system of the form

$$J \cdot (x(t + \delta t) - x(t)) = -F \quad (9)$$

where F denotes the residual form of equation (8) and J is the corresponding Jacobian matrix. The solution of this linear system offers some potential to gain performance for large-scale systems. Firstly, the matrix J can be calculated by exploiting the sparsity of the system, both numerically and symbolically. Naturally, this includes the storage of the matrix in a suitable sparse format to reduce the memory consumption. Secondly, in order to solve equation (9) sparse linear solvers (e.g. sparse LU factorization) can be utilized. For that purpose several methods have been developed and made publicly available (Davis and Natarajan, 2010; Davis, 2004).

For calculating the solution of algebraic loops with respect to equation (4) the same sparse solution methods can be utilized to gain some performance.

2.2 Simulation in DAE mode

An other way to go is to pass-through the whole system of equation (2) directly to an DAE solver, instead of using the ODE solver for integration and solve the implicit parts of equation (5) explicitly by algebraic solvers. Due to the fact that the index reduction is an important step for better convergence to the solution (Brenan et al., 1996), it is preferable to pass the system with index 1 (eq. (3)). Also, if the simulation is performed with equation (3), some time

consuming steps in the **Post-Optimization** compiling process, which deal with algebraic loops, namely tearing and the generation of symbolic Jacobians, can be skipped. A DAE solver is always an implicit solver and has to solve a non-linear system, which is usually solved using some variants of the Newton's method (Brenan et al., 1996). Thus, all the local implicit algebraic loops are solved all together by the global routine. One effect of using equation (3) instead of solving equation (5) is that the Jacobian matrix gets bigger, since the integration method needs to solve for the variables x , \dot{x} and y instead only for x . But this also preserves the sparse structure of the equation system with respect to Modelica models. In the ODE mode the corresponding Jacobian matrix is more dense due to the fact that the algebraic variables y are considered as internal variables.

Note, that in the current status of the DAE mode implementation it is still mandatory to generate the causalized code for proper handling of synchronous events and discrete variables. Therefore, OpenModelica generates currently an additional system in DAE mode. However, it is possible to skip unnecessary compiling steps by some specific compiler flags, which are documented in the OpenModelica User's Guide (Open Source Modelica Consortium).

2.3 Implementation in OpenModelica

The default simulation in OpenModelica is performed by solving system (5) using DASSL as a pure ODE solver. Hereby, the implicit parts (algebraic loops) are solved explicitly with algebraic equation solvers, the linear parts with lapack and the non-linear parts with a newton-based solver implemented in OpenModelica (Bachmann et al., 2015).

For the simulation of large scale Modelica models the most important part is a suitable sparse linear solver as depicted in section 2.1.2. Currently, one of the best under public domain available direct sparse linear solver for unsymmetric problems is the KLU solver (Davis and Natarajan, 2010) from the sparse matrix suite SuiteSparse. This solver is designed for solving sequences of unsymmetric sparse linear systems that arise from differential-algebraic equations, occurring when simulating electronic circuits. In fact, the linear systems arisen when simulating Modelica models are in general unsymmetric and often sparse, both in ODE and DAE mode. The open-source software family called SUNDIALS offers as a DAE solver the IDA solver (Hindmarsh et al., 2005). The IDA solver stands for Implicit Differential-Algebraic solver and is based on DASSL, but is written in ANSI-standard C. Further, for the solution of the underlying non-linear system at each time step, the IDA solver offers an interface to the sparse linear solver KLU. Furthermore, the SUNDIALS suite includes also a newton-based non-linear solver KINSOL, which is also able to use the KLU solver for the underlying linear system. Through the connection of SUNDIALS and SuiteSparse suite to the OpenModelica environment it is

now possible to rely on sparse methods at every step of the numerical simulation process.

3 Performance Results

3.1 Benchmarks from the ScalableTestSuite

3.1.1 Test set-up

The ScalableTestSuite (Casella, 2015; Casella and Sezginer, 2016) contains a number of different benchmark models, whose size can easily be chosen by setting one or more Integer parameters. The benchmarks are designed to stress some aspect of the code generation and execution, e.g. by possessing large implicit systems of algebraic equations, large number of states, large number of event-generating functions, etc. Please refer to the library documentation for further details.

This section reports the performance of a selection of nine benchmark models, each one coming in three different sizes. The results obtained with four different numerical solution strategies are presented and compared. Note that the current set of benchmarks does not include systems with large implicit systems of nonlinear equations – these will be added in the final version of the paper.

The first solution strategy, labelled *OD* in the result table, is the default approach to solving Modelica models implemented in the OpenModelica tool (see section 2). The DAEs are turned into ODEs by solving them for the derivatives, using the BLT transformation to do so efficiently, applying symbolic index reduction if the system has index greater than one. The implicit equations in the BLT corresponding to strong components in the dependency graph are solved with dense linear and nonlinear equation solvers, using tearing to reduce the size of the implicit part of the problem and thus somehow exploiting sparsity. The ODEs are then solved by the DASSL BDF integrator, using a dense linear solver for its internal operations.

The second strategy, labelled *OS*, still resorts to causalization; however, the implicit equations corresponding to the strong components in the BLT are solved by the KinSol/KLU sparse solvers, while the ODEs are solved by the IDA BDF integrator, relying on the KLU sparse linear solver internally.

In this case, tearing is not applied to solve the implicit equations corresponding to strong components in the BLT. The rationale behind this decision is that on the one hand, the sparse solver already vastly reduces the computational complexity, if the system is highly sparse. On the other hand, tearing very large systems might take a disproportionately large amount of time by the compiler back-end, so that the time savings at run time are likely to be more than offset by the much longer code generation time. In fact, this trade-off would itself deserve to be studied, but that goes beyond the scope of the present paper.

The third strategy, labelled *DA*, is to only apply symbolic index reduction (if needed) to the DAEs, and then

use the sparse IDA solver directly to solve them over time.

The fourth strategy, labelled *DD*, is a variant of the former one, in which the subset of the DAEs that is strictly required to be solved in order to compute the state variables at the next time step is identified and passed to the sparse IDA solver. Once the new time step has been computed and accepted as valid by the error estimation routine, the remaining equations are solved for the remaining variables by OpenModelica-generated code, exploiting the usual BLT decomposition to solve them efficiently.

This strategy can be advantageous because it avoids computing unnecessary variables during the internal solver iterations, particularly when tentatively computing a step that may then be rejected by the error estimation routine. Also, it is often the case that the dependencies in the systems are such that, once the variables required to advance the states have been computed, the remaining ones can be computed by explicit assignments. Furthermore, these are only computed once instead of getting unnecessarily involved many times in the iterative solution of the implicit sparse nonlinear DAEs.

As to the initialization problem, with the first strategy the standard dense linear and nonlinear solvers with tearing are used; with the other three, the sparse solvers KinSol/KLU without tearing are used instead.

All tests were carried out on the Open Source Modelica Consortium continuous testing infrastructure, using the development version 1.12.0 of OpenModelica. The computer used to run the tests is a 16-core Intel i7-6900K CPU @ 3.20 GHz, with 132 GB RAM.

3.1.2 Results and discussion

Table 1 reports some selected results, showing the number of equations *NE*, number of states *NS* and the running times of the simulations in seconds, including the time spent for initialization, for the four above-mentioned strategies. The full online report for each strategy can be retrieved by clicking on the corresponding label in the table headings of the PDF file.

Note that all the employed solvers are stiff and equipped with automatic order and step-size adaptation, with relative tolerance set to 10^{-6} , so that accuracy of the simulation results is comparable and the comparison among simulation times is fair and meaningful.

First of all, it is apparent how the adoption of sparse solvers turns out to be beneficial for 7 out of 9 benchmark models, reducing the simulation times by factors ranging from about 2 (SteamPipe and OneDHeatTransferTT_Modelica) to about 60 (TransmissionLineEquations_N_1280). It is also not significantly harmful in the remaining two.

Although the models in the ScalableTestSuite might be somewhat artificial and thus possibly bring higher benefits than real-life models, in the author's opinion this result is a clear indication that sparse solvers are the recommended option to simulate large-scale Modelica models.

The improvement in performance can be ascribed both

to the more efficient solution of the large implicit systems of equations involved in the solution process, and probably also to the lower number of time-consuming memory cache misses, due to the much smaller memory footprint of the simulation executable.

For some models, a large fraction of the simulation time is spent computing the right-hand-sides of the equations, rather than solving them, as in the case of the SteamPipe, where most of the time is spent computing the steam properties. In these cases, the adoption of a sparse solver cannot change the situation dramatically. On the contrary, sparse methods can bring huge benefits to models like TransmissionLineEquations, which have a large number of state variables, and an easy-to-compute right-hand side of the ODEs, with a very sparse Jacobian.

The advantage of using a sparse DAE solver over a sparse ODE solver is instead much less clear, and depends a lot on the specific case.

The multi-body models StringModelica, a suspended string modelled as a chain of rigid bodies and free rotational joints, and FlexibleBeamModelica, a cantilevered beam modelled as a chain of rigid bodies with elastic rotational joints, perform much better with the DAE solver, for reasons currently under investigation; also the SimpleAdvection models show a factor 2 improvement when using the DAE solver.

In other cases, such as TransmissionLineEquations and PowerSystemStepLoad, the advantage is more limited. The TransmissionLineModelica model turns out to be five times faster with the sparse ODE solver than with the full DAE solver (*DA* strategy). The penalty is reduced to about a factor 2 when using the more advanced *DD* strategy, which is understandable, as the model is built with basic Resistor and Capacitor models from the Modelica Standard Library and thus has a lot of redundant equations.

Finally, it seems that the *DA* strategy never turns out to provide any substantial advantage over the second best choice.

3.2 Large-scale power generation and transmission system models

The interest in Modelica modelling of national- and continental-size power generation and transmission systems is growing. A first feasibility study in this field was reported in (Casella et al., 2016). The relevant features of the benchmark models are reported here for convenience; the interested reader is referred to the above-mentioned reference for background information and more details.

Three benchmark test cases from that study are considered in this paper, whose main features are reported in Table 3. Note that the size of these models is much larger than the typical size of the ScalableTestSuite examples reported in the previous section.

RETE_C is a model of the Irish power generation and high-voltage power transmission system, while RETE_E and RETE_G are a medium- and a high-fidelity model

Table 1. Simulation times of ScalableTestSuite benchmarks in seconds.

<i>Benchmark</i>	<i>NE</i>	<i>NS</i>	<i>OD</i>	<i>OS</i>	<i>DA</i>	<i>DD</i>
SimpleAdvection_N_3200	6402	3199	20.81	4.851	3.087	2.561
SimpleAdvection_N_6400	12802	6399	104.9	13.27	6.107	6.781
SimpleAdvection_N_12800	25602	12799	642.2	41.15	19.17	18.38
SteamPipe_N_640	8966	1280	169.2	148.4	158.7	139.3
SteamPipe_N_1280	17926	2560	395.8	316.8	357.8	302.9
SteamPipe_N_2560	35846	5120	1165	651.0	801.9	679.9
TransmissionLineEquations_N_320	642	640	4.344	0.5742	0.2626	0.3563
TransmissionLineEquations_N_640	1282	1280	23.52	1.133	0.8848	0.7923
TransmissionLineEquations_N_1280	2562	2560	241.1	6.099	4.973	4.621
TransmissionLineModelica_N_320	6755	642	3.677	1.100	3.337	1.937
TransmissionLineModelica_N_640	13475	1282	29.15	2.090	11.63	7.59
TransmissionLineModelica_N_1280	26915	2562	235.0	9.012	47.80	20.96
FlexibleBeamModelica_N_16	5949	32	26.74	21.65	14.4	9.611
FlexibleBeamModelica_N_32	10877	64	111.9	64.87	38.12	28.30
FlexibleBeamModelica_N_64	20733	128	1819	393.8	n.a.	65.47
StringModelica_N_16	5887	34	1.801	1.410	0.4385	1.012
StringModelica_N_32	10783	66	9.710	10.02	1.541	1.897
StringModelica_N_64	20575	130	86.48	25.91	3.756	n.a.
PowerSystemStepLoad_N_16_M_4	1059	193	0.2272	0.1477	0.1329	0.4610
PowerSystemStepLoad_N_32_M_4	3139	385	0.7197	0.632	0.4116	0.5558
PowerSystemStepLoad_N_64_M_4	10371	769	2.713	2.961	2.277	2.867
OneDHeatTransferTT_Modelica_N_320	3190	318	0.322	0.2358	0.1794	0.3176
OneDHeatTransferTT_Modelica_N_640	6390	638	0.9237	0.3579	0.4711	0.4736
OneDHeatTransferTT_Modelica_N_1280	12790	1278	1.822	1.038	0.9342	1.058
HeatingSystem_N_20	103	41	16.76	20.26	n.a.	n.a.
HeatingSystem_N_40	203	81	113.7	155.6	n.a.	n.a.
HeatingSystem_N_80	403	161	827.2	831.5	n.a.	n.a.

Table 2. Number of synchronous generators, transmission lines, transformers and equations of the benchmark models

Network	Gen's	Lines	Trafo's	Equations
RETE_C	74	369	583	56386
RETE_E	267	1458	1202	157022
RETE_G	407	6833	2824	593886

of the Italian high-voltage power generation and transmission system, with an equivalent simplified representation of the interconnection to the pan-European grid.

These models have a peculiar feature, i.e., their DAE representation is highly sparse, but their ODE representation is dense, because all the synchronous generators interact instantaneously with each other, due to the phasor-based algebraic description of the transmission network. As a consequence, the use of implicit ODE solvers is not recommended, because the corresponding Jacobian is very large and dense.

At the time of the writing of (Casella et al., 2016), the sparse DAE solver only worked on the smallest test case, so for the larger ones a variant of the *OD* strategy was employed, using an explicit Runge-Kutta solver to avoid computing the dense Jacobian. Linearized load models were required in order to use the linear sparse solver KLU to compute the causalized equations. However, this approach was clearly sub-optimal, because a) realistic load models are non-linear and b) the system models are significantly stiff. Using fully implicit sparse DAE solvers with variable step size is clearly preferable from a performance point of view.

In this paper, we can now report the simulation performance obtained with the *DA* strategy, using an Intel Xeon CPU E5-2650 server running at 2.30GHz with 72 GB of RAM installed. All simulations start with the system in steady-state, then at time $t = 1$ s a big load is disconnected from the grid, causing an imbalance between generated and consumed power. The system undergoes a transient with some voltage and frequency oscillations, until the voltage and frequency controllers re-establish a new equilibrium in about 10-15 seconds. The simulation time span is 20 seconds, in order to check that the system actually returns to steady-state.

Performance results are reported in Table 2. It is worth noting that these results were obtained with a first implementation of the *DA* strategy; the authors are confident that the optimization of the IDA solver parameters and a more thorough scaling of the problem, which is badly scaled due to the use of SI units, could further improve the performance significantly.

3.3 Large-scale models of nuclear fusion reactor components

The development of a conceptual design of the European Demonstration Fusion Power Reactor (EU DEMO) is one of the goals defined in the EU fusion roadmap Horizon

Table 3. Simulation performance with *DA* strategy

Network	Rel. tol.	No. of steps	Sim. time [s]
RETE_C	10^{-4}	39	0.96
RETE_C	10^{-6}	146	3.18
RETE_E	10^{-4}	140	8.80
RETE_E	10^{-6}	364	15.22
RETE_G	10^{-4}	221	59.95
RETE_G	10^{-6}	615	123.19

2020. The future DEMO reactor aims at demonstrating industrial-scale electrical power production from nuclear fusion processes.

Politecnico di Torino, in cooperation with Politecnico di Milano, is developing a global thermal-hydraulic model of the entire system, using Modelica. One important part of that is the breeding blanket cooling system, in which pressurized water flows through a very complex and large system of tubes, collecting the heat generated from the nuclear fusion process and delivering it to a standard steam generator and turbine system, similar to those used for traditional PWR nuclear power plants. The breeding blanket cooling system is highly modular and has a repetitive structure, but its sub-components have different geometric features, so that it necessary to simulate each and every tube individually. As a result, models of this system can have a very large size. The interested reader can refer to (Froio et al., 2017) for more details.

The model reported in the above-mentioned reference has 289126 equations and 20772 states. The simulation of a transient of interest for the study of such system requires 64 s with the *DD* strategy and 146 s with the *DA* strategy.

The model has been benchmarked and validated against more detailed 3D CFD models. Given the simulation times shown above, which are obviously much faster than those of the CFD simulation, the model is suitable for use in parametric optimization studies, aimed at the optimal design of the coolant flow distribution.

4 Conclusion

This paper introduces methods and strategies to solve large-scale Modelica models and reports the performance of their implementation in OpenModelica on selected benchmark problems.

The main result of this study is that the use of sparse solvers is almost always beneficial, sometimes very substantially, over the traditional use of dense solvers supported by thorough symbolic manipulation. The comparison between sparse DAE solvers and sparse ODE solvers has many different outcomes, depending on the specific problems at hand.

Another interesting result is that we have demonstrated the feasibility of using such sparse solvers to successfully simulate Modelica models of industrially relevant systems with size up to over half a million DAEs.

Further developments of this work are already planned. First of all, it would be interesting to use the DAE solver to simultaneously handle the differential equations and the nonlinear algebraic implicit equations corresponding to strong components of the BLT, while still exploiting the BLT to compute the residuals of the DAEs by sequences of explicit assignments. This would combine the advantages of the sparse ODE and sparse DAE approaches discussed in this paper, avoiding the nested iterations of the nonlinear strong components solver and of the implicit ODE solver, possibly further improving the results reported in this paper on some classes of models.

It will also be necessary to further optimize the code generation for use with sparse solvers, as the current implementation is such that the code generation time is typically much larger than the simulation time, particularly for very large models. Radically new approaches to the code generation process are needed to break the one million equation barrier with reasonable executable code sizes and code generation times.

Last, but not least, although the handling of hybrid models with DAE sparse solvers is already implemented in OpenModelica, it has not been specifically optimized for efficient handling of large-size models. Such optimization would be another interesting research direction.

5 Acknowledgments

The presented work is partly financed by the PARADOM project, that is funded by the Federal Ministry of Education and Research (BMBF) under the support code 01IH15002B.

CESI S.p.A. is gratefully acknowledged for making the power system models available for this study.

References

- B. Bachmann, W. Braun, L. Ochel, and V. Ruge. Symbolical and numerical approaches for solving nonlinear systems. Annual OpenModelica Workshop 2015, 2015. URL <https://www.openmodelica.org/images/docs/openmodelica2015/OpenModelica2015-talk04-Bernhard-Bachmann-NLSinOpenModelica.pdf>.
- W. Braun, S. Gallardo Yances, K. Link, and B. Bachmann. Fast simulation of fluid models with colored jacobians. In *Proceedings of the 9th International Modelica Conference*, pages 247–252, Munich, Germany, Sep. 3–5 2012. Modelica Association. doi:10.3384/ecp12076247.
- K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, 1996. doi:10.1137/1.9781611971224.fm.
- F. Casella and K. Sezginer. The ScalableTestSuite Modelica Library, 2016. URL <https://github.com/casella/ScalableTestSuite>.
- Francesco Casella. Simulation of large-scale models in Modelica: State of the art and future perspectives. In Peter Fritzson and Hilding Elmqvist, editors, *Proceedings 11th International Modelica Conference*, pages 459–468, Versailles, France, Sep 21–23 2015. The Modelica Association. ISBN 978-91-7685-955-1. doi:10.3384/ecp15118459.
- Francesco Casella, Andrea Bartolini, Simone Pasquini, and Luca Bonuglia. Object-oriented modelling and simulation of large-scale electrical power systems using Modelica: a first feasibility study. In *Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society IECON 2016*, pages 0–6, Firenze, Italy, Oct. 24–27 2016. IEEE, IEEE. ISBN 978-1-5090-3474-1.
- F. E. Cellier and E. Kofman. *Continuous System Simulation*. Springer-Verlag, 2006.
- T. A. Davis. Algorithm 832: UMFPACK v4.3—an unsymmetric-pattern multifrontal method. *ACM Transactions On Mathematical Software*, 30(2):196–199, June 2004. ISSN 0098-3500. doi:10.1145/992200.992206. URL <http://dx.doi.org/10.1145/992200.992206>.
- T. A. Davis and E. Palamadai Natarajan. Algorithm 907: Klu, a direct sparse solver for circuit simulation problems. *ACM Trans. Math. Softw.*, 37(3):36:1–36:17, September 2010. ISSN 0098-3500. doi:10.1145/1824801.1824814. URL <http://doi.acm.org/10.1145/1824801.1824814>.
- Antonio Froio, Francesco Casella, Fabio Cismondi, Alessandro Del Nevo, Laura Savoldi, and Roberto Zanino. Dynamic thermal-hydraulic modelling of the eu demo well breeding blanket cooling loops. *Fusion Engineering and Design*, in press, available online:1–5, 2017. doi:10.1016/j.fusengdes.2017.01.062.
- C. Froio, F. Casella, F. Cismondi, A. Del Nevo, L. Savoldi, and R. Zanino. Dynamic thermal-hydraulic modelling of the eu demo well breeding blanket cooling loops. In *Proc. 29th Symposium on Fusion Technology (abstract)*, Prague, Czech Republic, 2016.
- A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.
- Open Source Modelica Consortium. OpenModelica User's Guide. Online. URL <https://openmodelica.org/doc/OpenModelicaUsersGuide/latest/>.
- P. Täuber, L. Ochel, W. Braun, and B. Bachmann. Practical realization and adaptation of cellier's tearing method. In *Proceedings of the 6th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, EOOLT '14*, pages 11–19, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2953-8. doi:10.1145/2666202.2666204. URL <http://doi.acm.org/10.1145/2666202.2666204>.

Transformation of Differential Algebraic Array Equations to Index One Form

Martin Otter¹ Hilding Elmqvist²

¹Institute of System Dynamics and Control, DLR, Germany, Martin.Otter@dlr.de

²Mogram AB, Sweden, Hilding.Elmqvist@Mogram.net

Abstract

Several new algorithms are proposed that in effect transform DAEs (Differential Algebraic Equations) to a special index one form that can be simulated with standard DAE integrators. The transformation to this form is performed without solving linear and/or nonlinear equation systems, the sparsity of the equations is kept, and array equations remain array equations or differentiated versions of them. Furthermore, certain DAEs can be handled where structural index reduction methods fail. It is expected that these new algorithms will help to treat large Modelica models of any index in a better way as it is currently possible. The algorithms have been evaluated and tested in the experimental simulation environment Modia that is implemented with the Julia programming language.

Keywords: *Modelica, Modia, Julia, DAE, sparse DAE, large DAE, Pantelides algorithm, Dummy Derivative Method.*

1 Introduction

The objective is to handle larger Modelica models as it is practically possible today. For this purpose new algorithms have been developed: equations as well as variables are not scalarized but keep their original array types, even if differentiated. This gives a more compact code which is a benefit with regards to code cache behavior. Furthermore, there is a possibility to utilize vector instructions of modern processors. With respect to current Modelica tools, equation systems are not solved locally in the model code but by a DAE solver where the sparsity of the Jacobian is taken into account and in the model code single (array) equations are either explicitly solved if this is possible or residues for implicit equations are computed to be solved by the DAE solver. The new algorithms have been evaluated and tested in the prototype Modia (Elmqvist *et al.*, 2016, Elmqvist *et al.*, 2017) which is implemented with the Julia programming language¹ (Bezanson *et al.*, 2017) and takes advantage of this very promising language effort with focus on scientific computing. Modia is available from <https://github.com/ModiaSim>.

¹ <http://julialang.org/>

2 Special Index One DAE Form

The goal is to simulate physical models that are described by a modeling language such as Modelica and mapped to a DAE in the form

$$\mathbf{f}_0(\dot{\mathbf{x}}_{d0}, \mathbf{x}_{d0}, \mathbf{x}_{a0}, t) = \mathbf{0} \quad (1)$$

$$\mathbf{f}_0 \in \mathbb{R}^{nd_0} \times \mathbb{R}^{nd_0} \times \mathbb{R}^{na_0} \times \mathbb{R} \rightarrow \mathbb{R}^{nd_0+na_0}$$

where $\mathbf{x}_{d0}(t)$ are variables that appear differentiated and $\mathbf{x}_{a0}(t)$ are variables that do not appear differentiated. Furthermore, it is assumed that a unique solution of this DAE exists if consistent initial conditions of $\dot{\mathbf{x}}_{d0}, \mathbf{x}_{d0}, \mathbf{x}_{a0}$ are given, and that the equations and variables are smoothly differentiable sufficiently often. Typically, Modelica tools transform (1) into ODE (Ordinary Differential Equation) form and use ODE or DAE integration methods for the solution. In this paper, this is not done because (a) a transformation to ODE form may destroy the sparsity structure of the equations and (b) requires in general solving linear and/or nonlinear algebraic equation systems and an implicit integration method will in turn solve nonlinear algebraic equation systems as well (so a nonlinear solver is called within a nonlinear solver). For certain classes of physical models, such as large 3-dimensional mechanical systems, this approach might not be efficient and not reliable. Instead a new approach is proposed to transform (1) to the following *special index one DAE*

$$\begin{aligned} \mathbf{f}_d(\dot{\mathbf{x}}, \mathbf{x}, t) &= \mathbf{0} \\ \mathbf{f}_c(\mathbf{x}, t) &= \mathbf{0} \end{aligned} \quad (2a) \quad \mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}_d}{\partial \dot{\mathbf{x}}} \\ \frac{\partial \mathbf{f}_c}{\partial \mathbf{x}} \end{bmatrix} \text{ is regular} \quad (2b)$$

without solving equation systems. DAE (2) shall have an identical solution space as DAE (1) and $\mathbf{x}_{d0}, \mathbf{x}_{a0}$ shall be part of \mathbf{x} . Note, when differentiating $\mathbf{f}_c(\cdot)$,

$$\begin{aligned} \mathbf{f}_d(\dot{\mathbf{x}}, \mathbf{x}, t) &= \mathbf{0} \\ \frac{\partial \mathbf{f}_c}{\partial \dot{\mathbf{x}}} \dot{\mathbf{x}} + \frac{\partial \mathbf{f}_c}{\partial t} &= \mathbf{0} \end{aligned} \quad (3)$$

can be solved for $\dot{\mathbf{x}}$ because the matrix of partial derivatives of (3) with respect to $\dot{\mathbf{x}}$ is the Jacobian \mathbf{J} of (2b) which is regular. This shows that (2) has index 1.

A number of methods exist for solving system (2) numerically. In particular, under mild conditions BDF

(Backward Differentiation Formula) methods with order k ($k < 7$) and fixed or variable step-size h converge with $O(h^k)$, see (Brenan et al. 1996) page 51-54. This means that a DAE integrator like Sundials IDA (Hindmarsh et al. 2005) can solve such systems.

On the other hand, solving (2) with a BDF-method requires to solve a nonlinear equation system where the inverse of the iteration matrix becomes singular for $h \rightarrow 0$. (Petzold and Lötstedt, 1986) point out that step size selection is difficult if reducing the step size makes the iteration matrix ill-conditioned and it is proposed to scale elements of the iteration matrix with h so that this effect does not occur. In (Arnold, 2016) it is shown how this technique can be applied for multibody systems. For system (2) scaling with h is particularly simple, resulting in two possible ways:

$$\begin{aligned} h\mathbf{f}_d(\dot{\mathbf{x}}, \mathbf{x}, t) = \mathbf{0} \\ \mathbf{f}_c(\mathbf{x}, t) = \mathbf{0} \end{aligned} \quad (4a) \quad \begin{aligned} \mathbf{f}_d(\dot{\mathbf{x}}, \mathbf{x}, t) = \mathbf{0} \\ \frac{1}{h}\mathbf{f}_c(\mathbf{x}, t) = \mathbf{0} \end{aligned} \quad (4b)$$

Assume that these systems are solved with a BDF method of order k . This means that the derivatives $\dot{\mathbf{x}}$ at step i are approximated as:

$$\dot{\mathbf{x}}_i \approx \frac{\alpha_{k0}}{h}\mathbf{x}_i + \frac{1}{h}\sum_{j=1}^{j=k} \alpha_{kj}\mathbf{x}_{i-j} \quad (5)$$

where α_{kj} are constant coefficients depending on the order of the method, $h = t_i - t_{i-1}$ is the step size and the sum $\sum \alpha_{kj}\mathbf{x}_{i-j}$ is a known term computed from values of \mathbf{x} at previous time instants. Inserting (5) in (4a) results in a nonlinear system of equations for \mathbf{x}_i :

$$\begin{aligned} h\mathbf{f}_d\left(\frac{\alpha_{k0}}{h}\mathbf{x}_i, \mathbf{x}_i, t_i\right) = \mathbf{0} \\ \mathbf{f}_c(\mathbf{x}_i, t_i) = \mathbf{0} \end{aligned} \quad (6)$$

Assume that $\mathbf{x}, \mathbf{f}_d, \mathbf{f}_c$ are sufficiently smooth and bounded and that \mathbf{x}_{i-1} solves (6) at the previous time instant t_{i-1} . According to the implicit function theorem, (6) has a *unique solution* at time instant $t_{i-1} + h$ if the inverse of the Jacobian of this system

$$\mathbf{J}_i = \begin{bmatrix} \frac{\partial h\mathbf{f}_d}{\partial \mathbf{x}_i} \\ \frac{\partial \mathbf{f}_c}{\partial \mathbf{x}_i} \end{bmatrix} = \begin{bmatrix} \alpha_{k0} \frac{\partial \mathbf{f}_d}{\partial \dot{\mathbf{x}}} + h \frac{\partial \mathbf{f}_d}{\partial \mathbf{x}_i} \\ \frac{\partial \mathbf{f}_c}{\partial \mathbf{x}_i} \end{bmatrix} \quad (7)$$

exists for small h . This is indeed the case, since for $h \rightarrow 0$ the Jacobian (7) converges to the regular Jacobian (2b), when dividing the upper equation by the constant α_{k0} . A similar result can be derived for (4b).

To summarize, DAE (2) can be solved by standard (index one) DAE integrators and a reliable numerical solution with a BDF method can be expected even for small step sizes when one of the h scaling methods of (4) is used. In the remaining part of this paper it is shown, how a large class of DAEs in the form (1) can be transformed to (2). This transformation is performed in several steps that are discussed now in sequence.

3 Index Reduction of DAEs that have Array Equations

3.1 Algorithms for Index Reduction

In order to reduce a DAE (1) to ODE or index one form, equations of (1) might need to be differentiated. There are in principle many algorithms to perform this index reduction based on the *structure of the equations*, that is by the information which variable is present in which equation. The essential idea and the key algorithm are from (Pantelides, 1988): The structure of the equations is described by a bipartite graph of equations and variables. The equations are differentiated until a *complete assignment* of the highest derivative equations is possible with respect to the highest derivative variables (which include algebraic variables that are not differentiated).

Since the goal is to achieve complete assignment in a bipartite graph, *any* matching algorithm for a bipartite graph can be used as basis. In (Pantelides, 1988), the matching algorithm of (Duff, 1981) is utilized which results in a very simple and elegant implementation. It results in a worst time complexity of $O(n \cdot m)$ where n is the number of equations in the final system (= original and all differentiated equations) and m is the number of entries (incidences) in the final bipartite graph.

In (Duff et al., 2011) eight matching algorithms and various additional heuristics are compared. Most of them have the same worst time complexity as (Duff, 1981), a few have $O(\sqrt{n} \cdot m)$. On average the $O(n \cdot m)$ algorithm PF+ described in this article had the best performance on the test matrices. In (Frenkel et al., 2012) nine matching algorithms and different implementations for structural index reductions are compared for multibody examples with varying number of bodies. This evaluation indicates that PF+ has on average the best performance for index reduction with the Pantelides algorithm.

It is well-known that complete matching in a bipartite graph is equivalent to the network flow problem where the maximum amount of flow shall be determined that can be sent between two given vertices of a graph, see for example (Skiena, 2008, page 217). It is also well-known that both the network flow problem and the bipartite matching problem can be formulated as a special linear programming problem, see for example (Edmond, 1965; Cook and Rohe, 1999; Skiena, 2008, pp. 509-510). For all these problem classes solution algorithms are available and can be used for index reduction. For example, (Pryce, 2001) describes an index reduction method based on the special linear programming problem.

All above algorithms for index reduction are iterative and the question is when the iteration stops. (Pantelides, 1988) provides an elegant method to test beforehand whether the (structural) index is finite:

Adding the relationship $0 = h_i(\dot{x}_{d0,i}, x_{d0,i})$ structurally for all differentiated variables in (1) gives an *extended system* which can be interpreted as solving (1) with an implicit integration method. If this system has a *complete matching*, the (structural) index is finite and structural index reduction algorithms will converge.

All the algorithms mentioned above have the disadvantage that only structural properties are utilized and therefore will fail if state constraints are not structurally visible. In (Chowdhry *et al.*, 2004) a symbolic/numeric index reduction procedure is proposed. It is based on a symbolic numerical algebra for pattern manipulations of the DAE (1) and reduces the index of linear constant coefficient DAEs *numerically* by LU decompositions and the index of nonlinear parts by a *structural* index reduction algorithm.

In section 5 a new method is proposed to exactly handle singularities in the connection graph of a model, both to treat (consistently) underdetermined and overdetermined equation systems, as well as state constraints that cannot be handled by a structural index reduction algorithm. This technique transforms a DAE (1) in a DAE (1) and is therefore a pre-processing step for the transformations of section 3 and 4.

3.2 Index Reduction on Array Equations

The Pantelides algorithm and other structural index reduction algorithms are designed for *scalar* variables and equations. So Modelica tools typically symbolically expand array equations into a set of scalar equations involving the array elements and the description with array equations is lost. This has significant drawbacks for large array equations. In (Schuchart, *et al.*, 2015) it is shown how special for-loops can be handled so that they need not to be expanded for the Pantelides algorithm and are retained in the generated code.

Below, a new technique is proposed to handle any kind of array equations. Hereby the (conceptual) expansion of array equations is performed *only* in the bipartite graph to perform structural index reduction and in a BLT (Block Lower Triangular) transformation on the highest derivative equations. In order to illustrate this technique, a tiny multibody example will be used.

Consider the following model of a *sliding mass*. It is a one degree-of-freedom model, with scalar parameters c, d, m , vector parameters \mathbf{n}, \mathbf{g} , scalar unknown s and vector unknowns $\mathbf{r}, \mathbf{v}, \mathbf{f}, \mathbf{u}$, that is described by the following equations:

$$\begin{aligned} \mathbf{r} &= \mathbf{n}s \\ \mathbf{v} &= \dot{\mathbf{r}} \\ m\dot{\mathbf{v}} &= \mathbf{f} + m\mathbf{g} + \mathbf{u} \\ 0 &= \mathbf{n} \cdot \mathbf{f} \\ \mathbf{u} &= -(cs + d\dot{s})\mathbf{n} \end{aligned} \quad (8)$$

(8) is a DAE (1) with $4 \cdot 3 + 1 = 13$ equations in the 13 variables $\mathbf{x}_{d0} = [s; \mathbf{r}; \mathbf{v}]$, $\mathbf{x}_{a0} = [\mathbf{f}; \mathbf{u}]$.

With the Pantelides algorithm it is determined how often every equation would have to be differentiated until the highest derivatives variables can be uniquely assigned to the highest derivative equations. Since we want to keep array equations intact, it is natural to assign array variables to array equations, provided they have the same type and the same dimensions. See also (Stavåker, 2015) chapter 9. However, this does not work for the sliding mass example above and any other multibody system where bodies are connected by joints.

The scalar variable s appears only in vector equations, so s or a higher derivative of it can only be assigned to an element of these equation (or a derivative of them), which means that a vector equation must be expanded in scalar equations. Furthermore, the vector variable \mathbf{f} appears as only variable in a scalar equation ($0 = \mathbf{n} \cdot \mathbf{f}$) and therefore one element of \mathbf{f} must be assigned to this scalar equation. As a result, the two other elements of \mathbf{f} have to be assigned in other equations, which then must be expanded as well. In the end, all equations must be expanded to scalar equations in order that an assignment of all variables is possible.

After expanding all equation graphs, the Pantelides algorithm determines that the first vector equations must be differentiated twice and the second one time leading to the following assignments:

assigned	highest derivative equations	diff. order
\ddot{s}	$\ddot{r}_1 = n_1 \ddot{s}$	2
\ddot{r}_2	$\ddot{r}_2 = n_2 \ddot{s}$	2
\ddot{r}_3	$\ddot{r}_3 = n_3 \ddot{s}$	2
\dot{r}_1	$\dot{v}_1 = \dot{r}_1$	1
\dot{v}_2	$\dot{v}_2 = \dot{r}_2$	1
\dot{v}_3	$\dot{v}_3 = \dot{r}_3$	1
\dot{v}_1	$m\dot{v}_1 = f_1 + mg_1 + u_1$	0
f_2	$m\dot{v}_2 = f_2 + mg_2 + u_2$	0
f_3	$m\dot{v}_3 = f_3 + mg_3 + u_3$	0
f_1	$0 = n_1 f_1 + n_2 f_2 + n_3 f_3$	0
u_1	$u_1 = -(cs + d\dot{s})n_1$	0
u_2	$u_2 = -(cs + d\dot{s})n_2$	0
u_3	$u_3 = -(cs + d\dot{s})n_3$	0

As will become clear in section 4, the set of highest derivative equations must be sorted, so a BLT (Block Lower Triangular) transformation must be applied that identifies the order of evaluation, as well as the algebraic loops under the assumption that the lower-order derivative variables are known. Furthermore, the assumption is used that array equations have full incidence (see Assumption 1 below).

highest derivative equations	solve for
$u_1 = -(cs + d\dot{s}) n_1$ $u_2 = -(cs + d\dot{s}) n_2$ $u_3 = -(cs + d\dot{s}) n_3$	u_1, u_2, u_3
$\ddot{r}_1 = n_1 \ddot{s}$ $\ddot{r}_2 = n_2 \ddot{s}$ $\ddot{r}_3 = n_3 \ddot{s}$ $\dot{v}_1 = \dot{r}_1$ $\dot{v}_2 = \dot{r}_2$ $\dot{v}_3 = \dot{r}_3$ $m\dot{v}_1 = f_1 + mg_1 + u_1$ $m\dot{v}_2 = f_2 + mg_2 + u_2$ $m\dot{v}_3 = f_3 + mg_3 + u_3$ $0 = n_1 f_1 + n_2 f_2 + n_3 f_3$	$\ddot{s}, \ddot{r}_1, \ddot{r}_2, \ddot{r}_3,$ $\dot{v}_1, \dot{v}_2, \dot{v}_3,$ f_1, f_2, f_3

The result is that there is an algebraic system with 3 equations that has to be solved for 3 unknowns. Afterwards an algebraic equation system with 10 equations has to be solved for 10 unknowns. Note, whenever an algebraic loop is encountered, the previous assignment information *on equation level* is no longer relevant (which is anyway not unique in such a case) but only the *set of unknown variables* of the respective algebraic loop.

Although the Pantelides algorithm must be performed on (conceptually) expanded scalar equations and scalar variables, the BLT transformed highest derivative equations can be contracted to array equations again:

highest derivative equations	solve for
$\mathbf{u} = -(cs + d\dot{s})\mathbf{n}$	\mathbf{u}
$\ddot{\mathbf{r}} = \mathbf{n}\ddot{s}$ $\dot{\mathbf{v}} = \dot{\mathbf{r}}$ $m\dot{\mathbf{v}} = \mathbf{f} + m\mathbf{g} + \mathbf{u}$ $0 = \mathbf{n} \cdot \mathbf{f}$	$\ddot{s}, \ddot{\mathbf{r}}, \dot{\mathbf{v}}, \mathbf{f}$

The original Pantelides algorithm and the BLT transformation are graph based algorithms that assume nodes and vertices correspond to scalar real variables and equations. These algorithms can be generalized to work directly on the array variables and equations.

3.3 Properties of Array Equations

The presented approach relies on the fact that all scalarized equations/variables appear in the same algebraic equation system (or more precisely in the same strongly connected component of a directed graph). This sub-section contains the assumption under which this property holds and the proof of the property. Multi-dimensional arrays are treated as vectors with length being the total number of elements.

Assumption 1: When expanding the bipartite graph of DAE (1) regarding array variables and array equations, it is assumed that they have full incidence.

Example: For the equation $\mathbf{e}: \mathbf{w} = \mathbf{f}(\mathbf{u}, \mathbf{v})$ with all variables being vectors of length 2, the incidence structure is assumed to be:

	w_1	w_2	u_1	u_2	v_1	v_2
e_1	x	x	x	x	x	x
e_2	x	x	x	x	x	x

Theorem 1: If one element of an array equation needs to be differentiated, all elements of all time varying variables in the equation need to be differentiated.

Example: For the equation $\mathbf{e}: \mathbf{w} = \mathbf{f}(\mathbf{v})$ with all variables being vectors of length 2, the incidence structure of the differentiated equation is:

	w_1	w_2	\dot{w}_1	\dot{w}_2	v_1	v_2	\dot{v}_1	\dot{v}_2
\dot{e}_1	x	x	x	x	x	x	x	x
\dot{e}_2	x	x	x	x	x	x	x	x

Proof: This follows since if an array variable has full incidence, so has its time derivative. This means that derivatives of all time varying array variable elements will appear in the differentiated element of the array equation. ■

This is consistent with the Pantelides algorithm because, if the function `augmentPath` returns false, all variable elements with incidence are marked as colored. All colored V-nodes to be differentiated are then marked in the A vector.

Theorem 2: If one element of an array equation needs to be differentiated, all other elements of the equation need to be differentiated.

Proof: According to Theorem 3, all elements of an array equation will appear in the same strongly connected component. It means that there is a mutual dependency between all array equation elements. This means that there is also a mutual dependency between all differentiated array variable elements. In order to be able to solve for all derivatives, an equal number of array equation elements are needed, that is all array equation elements must be differentiated. ■

Function `augmentPath` colors all equations visited. If assignment is not possible, `augmentPath` tries to reassign. Due to the mutual dependency, all of the elements of an array equation are visited. In the B-vector of the Pantelides algorithm it is marked that all colored equation nodes should be differentiated.

Theorem 3: If the highest derivative equations are structurally non-singular with respect to the highest derivative variables, all elements of an array equation will appear in the same strongly connected component.

Proof: The incidence matrix of an array equation consists of n identical rows with n being the number of scalar elements of the left and right hand side. Assume that these elements of the array equation (incidence matrix rows) appear in different strongly connected components. This would mean that some of these elements of the array equation could be solved without the others. Since they have the same incidence, it would mean that the other elements of the array

equation would be structurally over-constrained. This contradicts the assumption of structural non-singularity. ■

Limitations

It is worth noting that there are cases when symbolic expansion of array equations is beneficial. For example, (Elmqvist and Mattsson, 2016) discusses detection and handling of planar loops of multibody systems. In such cases, certain elements of position vectors are overdetermined and certain elements of force vectors are underdetermined. In order to reveal this, the zeros in axis of rotation and translation vectors must be utilized, that is, certain equations must be expanded.

When discretized partial differential equations are handled, the boundary elements may give rise to issues with Assumption 1. In such a case, array equations for the inner elements might be formulated and scalar equations for the boundary elements added separately.

3.4 Implementation Notes

The Pantelides and BLT algorithms have as input the incidence graph for the array variables and equations, that is, *non-expanded* equation and array structure. Additionally, a vector of lengths of each variable, that is the number of scalar elements, and a vector of the lengths of equations are given as inputs.

All for-loops over variables and equations in the original algorithms are replaced with nested for loops also looping over the lengths.

The usual indices in the assignment, lowlink and number vectors and stack are replaced by tuples denoting which array and which array element is referred to. However, due to Theorem 1, the **A** vector (see below) which tells which variables are differentiated is still just a vector over array variables. Similarly, due to Theorem 2, the **B** vector (see below) for differentiated equations does not need the tuple indexing. The strongly connected component representation is only referring to array equations due to Theorem 3.

3.5 Result of Structural Index Reduction and BLT

For the further processing, the result of the structural index reduction and BLT transformation of array equations is summarized formally. For this, the following notation is used

- All symbols are collected in a variable vector **v** and v_j is symbol j . A symbol may represent a scalar or an array. $V_{length,j}$ gives the number of elements of symbol j .
- All equations are collected in an equation vector **e** and e_i is equation i . An equation may be a scalar or an array equation. $E_{length,i}$ gives the number of elements of equation i .

- The relationship between the symbols is defined by the variable association vector **A**, such that:

$$A_j = \text{if } \dot{v}_j = v_k \text{ then } k \text{ else } 0.$$

Also the inverse relationship is needed below:

$$A_{inv,k} = \text{if } \dot{v}_j = v_k \text{ then } j \text{ else } 0.$$

- The relationship between the equations is defined by the equation association vector **B**, such that:

$$B_i = \text{if } \dot{e}_i = e_k \text{ then } k \text{ else } 0.$$

Also the inverse relationship is needed below:

$$B_{inv,k} = \text{if } \dot{e}_i = e_k \text{ then } i \text{ else } 0.$$

Starting point is DAE (1) that can be defined with $\mathbf{v}_0 = [\dot{\mathbf{x}}_{d0}; \mathbf{x}_{d0}; \mathbf{x}_{a0}]$ and the n_{e0} array equations:

$$\mathbf{e}_0(\mathbf{v}_0, t) = \mathbf{0} \quad (9)$$

Structural index reduction determines the minimal number of differentiations of (9) such that the following conditions are fulfilled by the final system:

$$e_i(v_j, t) = 0; \quad B_i = 0; \quad A_j \geq 0 \quad (10a)$$

$$e_k(v_j, t) = 0; \quad B_k > 0; \quad A_j > 0 \quad (10b)$$

$$\frac{\partial e_i}{\partial v_j} \text{ structurally regular for } A_j = 0 \quad (10c)$$

(10a) are n_{e0} array equations (the highest derivative equations) in n_{e0} unknown arrays (the highest derivative variables v_j with $A_j = 0$). The matrix (10c) of partial derivatives of the highest derivative equations with respect to the highest derivative variables is structurally regular. (10b) are $n_e - n_{e0}$ array equations describing the constraints between the variables appearing differentiated. The highest derivative variables (that is v_j with $A_j = 0$), do *not* appear in these equations.

4 Transformation to Index One Form

4.1 Overview

The transformation from (1) to index one form (2) using (10) is made in three steps: In a first step, the solution is sketched for multibody system equations. In a second step this approach is generalized and in a final step the transformation is made more efficient by partial static state selection.

In the field of multibody systems, constraints appear in nearly every model and hence multibody programs need to inherently cope with the special constraints appearing in 3-dimensional mechanical systems. It is therefore natural to inspect the many solution methods developed for multibody systems and try to generalize one or more of them to general DAEs (1). In the recent report (Arnold, 2016), a very broad and nice overview of the current state of the art for simulation of multibody systems is given and used as basis of this section.

One solution method is to integrate the highest derivative equations (10a) and when the violation of the constraints (10b) becomes too large project on the

constraint manifold, see for example (Ascher and Petzold, 1991; Eich, 1993). Such methods could be implemented by utilizing an implicit one-step DAE integrator and after every step project the solution on the constraint manifold.

Many industrial applications of Modelica models are best solved with an implicit multistep method. Unfortunately, multistep methods require non-trivial modifications to embed a projection method because the solution is interpolated smoothly over several steps and (potentially) in every step the solution is modified in a discontinuous way by a projection. It seems that the stabilized index-2 formulation according to Gear, Gupta, Leimkuhler (Gear et al., 1985) is a more attractive starting point especially for multistep methods. This method is analyzed in the sequel in some detail. The presentation is made in such a way that a generalization is straightforward.

4.2 Transformation of Multibody Equations

Starting point is the following set of equations for a multibody system:

$$\dot{\mathbf{q}} = \mathbf{v} \quad (11a)$$

$$\mathbf{M}(\mathbf{q}, t)\dot{\mathbf{v}} + \mathbf{G}^T(\mathbf{q}, t)\boldsymbol{\lambda} = \mathbf{h}(\mathbf{q}, \mathbf{v}, t) \quad (11b)$$

$$\mathbf{0} = \mathbf{g}(\mathbf{q}, t) \quad (11c)$$

with

$$\mathbf{G} = \frac{\partial \mathbf{g}}{\partial \mathbf{q}}, \quad \mathbf{M} = \mathbf{M}^T > \mathbf{0} \quad (11d)$$

It is assumed that \mathbf{G} has full row rank that is the constraints equations (11c) are not redundant. (11) has $n_q + n_v + n_\lambda$ real unknowns $\dot{\mathbf{q}}, \dot{\mathbf{v}}, \boldsymbol{\lambda}$ for the same number of equations. With the new array version of the Pantelides algorithm, equation (11a) is differentiated once and equation (11c) twice in order to arrive at the following equation system that needs to be fulfilled by consistent initial values $\mathbf{q}_0, \mathbf{v}_0, \boldsymbol{\lambda}_0, \dot{\mathbf{q}}_0, \ddot{\mathbf{q}}_0, \dot{\mathbf{v}}_0$:

$$\dot{\mathbf{q}} = \mathbf{v} \quad (12a)$$

$$\mathbf{M}\dot{\mathbf{v}} + \mathbf{G}^T\boldsymbol{\lambda} = \mathbf{h}(\mathbf{q}, \mathbf{v}, t) \quad (12b)$$

$$\mathbf{0} = \mathbf{g}(\mathbf{q}, t) \quad (12c)$$

$$\mathbf{0} = \mathbf{G}\dot{\mathbf{q}} + \mathbf{g}^{(1)}(\mathbf{q}, t) \quad (12d)$$

$$\mathbf{0} = \mathbf{G}\ddot{\mathbf{q}} + \mathbf{g}^{(2)}(\mathbf{q}, \dot{\mathbf{q}}, t) \quad (12e)$$

$$\ddot{\mathbf{q}} = \dot{\mathbf{v}} \quad (12f)$$

with

$$\mathbf{g}^{(1)} = \frac{\partial \mathbf{g}}{\partial t}, \quad \mathbf{g}^{(2)} = \dot{\mathbf{G}}\dot{\mathbf{q}} + \dot{\mathbf{g}}^{(1)} \quad (12g)$$

This is an ODAE (Overdetermined Differential Algebraic Equation) with $2n_q + n_v + 3n_\lambda$ equations for the $2n_q + n_v + n_\lambda$ unknowns $\dot{\mathbf{q}}, \ddot{\mathbf{q}}, \dot{\mathbf{v}}, \boldsymbol{\lambda}$. In order to arrive at an equation system with the same number of unknowns and equations that are consistent (so locally a unique solution exists), the following approach of (Gear et al., 1985) is used:

$$\mathbf{0} = \dot{\mathbf{q}} - \mathbf{v} + \mathbf{G}^T\boldsymbol{\mu} \quad (13a)$$

$$\mathbf{0} = \mathbf{M}\dot{\mathbf{v}} + \mathbf{G}^T\boldsymbol{\lambda} - \mathbf{h}(\mathbf{q}, \mathbf{v}, t) \quad (13b)$$

$$\mathbf{0} = \mathbf{g}(\mathbf{q}, t) \quad (13c)$$

$$\mathbf{0} = \mathbf{G}\mathbf{v} + \mathbf{g}^{(1)}(\mathbf{q}, t) \quad (13d)$$

These are $n_q + n_v + 2n_\lambda$ residue equations for the $n_q + n_v + 2n_\lambda$ unknowns $\dot{\mathbf{q}}, \dot{\mathbf{v}}, \boldsymbol{\lambda}, \boldsymbol{\mu}$, so the number of equations and number of unknowns is the same. (13) has a *differential index of two* and has the same solution as (12) because it can be shown that $\boldsymbol{\mu} = \mathbf{0}$: Inserting equation (13a) in equation (13d):

$$\mathbf{0} = \mathbf{G}(\dot{\mathbf{q}} + \mathbf{G}^T\boldsymbol{\mu}) + \mathbf{g}^{(1)}(\mathbf{q}, t)$$

and subtracting the derivative of (13c) results in the equation $\mathbf{0} = \mathbf{G}\mathbf{G}^T\boldsymbol{\mu}$. Provided \mathbf{G} has full row rank, $\mathbf{G}\mathbf{G}^T$ is regular and therefore $\boldsymbol{\mu} = \mathbf{0}$. ■

This scheme can be easily generalized. For example assume that (say due to an inverse model) the third derivative of (12c) is needed. Then, new $\boldsymbol{\mu}_2$ variables and corresponding dummy derivatives are introduced in combination with the second derivatives of the constraints:

$$\begin{aligned} \mathbf{w} &= \dot{\mathbf{v}} + \mathbf{G}^T\boldsymbol{\mu}_2 \\ \mathbf{0} &= \mathbf{G}\mathbf{w} + \mathbf{g}^{(2)}(\mathbf{q}, \mathbf{v}, t) \end{aligned}$$

With the same argument as before it can be shown that $\boldsymbol{\mu}_2 = \mathbf{0}$: Inserting \mathbf{w} in the second equation and subtracting the differentiated equation (13d) results in $\mathbf{0} = \mathbf{G}\mathbf{G}^T\boldsymbol{\mu}_2$ and therefore $\boldsymbol{\mu}_2 = \mathbf{0}$. ■

In (Gear et al., 1985) it is shown that variable-step and variable order BDF (Backward Differentiation Formula) methods converge for this index-2 DAE. However, (13) is not yet in the desired form (2). In particular, the BDF iteration matrix becomes singular for a small step size. (13) can be transformed to (2) by using the substitution (Gear, 1988):

$$\boldsymbol{\mu} = \dot{\boldsymbol{\mu}}_{int}, \quad \boldsymbol{\lambda} = \dot{\boldsymbol{\lambda}}_{int} \quad (14)$$

as well as $\mathbf{x} = [\mathbf{q}; \mathbf{v}; \boldsymbol{\lambda}_{int}; \boldsymbol{\mu}_{int}]$:

$$\mathbf{0} = \begin{bmatrix} \mathbf{f}_d(\dot{\mathbf{x}}, \mathbf{x}, t) \\ \mathbf{f}_c(\mathbf{x}, t) \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}} - \mathbf{v} + \mathbf{G}^T\dot{\boldsymbol{\mu}}_{int} \\ \mathbf{M}\dot{\mathbf{v}} + \mathbf{G}^T\dot{\boldsymbol{\lambda}}_{int} - \mathbf{h}(\mathbf{q}, \mathbf{v}, t) \\ \mathbf{g}(\mathbf{q}, t) \\ \mathbf{G}\mathbf{v} + \mathbf{g}^{(1)}(\mathbf{q}, t) \end{bmatrix} \quad (15)$$

Note, the Jacobian (2b) of (15) is regular (\mathbf{P} in (16) is a permutation matrix to exchange the third and the fourth equation of (15) in order that the regularity is at once visible):

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}_d}{\partial \dot{\mathbf{x}}} \\ \frac{\partial \mathbf{f}_c}{\partial \mathbf{x}} \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{G}^T \\ \mathbf{0} & \mathbf{M} & \mathbf{G}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{G} & \mathbf{0} & \mathbf{0} \\ \mathbf{G} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \text{ is regular} \quad (16)$$

4.3 Transformation of general DAEs

The goal is to transform the ODAE (10) to (2). In a first step the elements of vector \mathbf{x} are identified:

1. All variables v_j that *appear differentiated* are collected together with their derivatives in vector \mathbf{x}_d with exception of their highest derivatives (so $x_{d,j}$ are all variables v_j with $A_j > 0$).
2. All variables v_j that do *not appear differentiated* (so v_j with $A_j = 0$ and $A_{inv,j} = 0$) are collected either in vector \mathbf{x}_a or vector $\boldsymbol{\lambda} = \dot{\boldsymbol{\lambda}}_{int}$ in such a way that (a) all assigned variables of every BLT block are either only differential variables ($\dot{\mathbf{x}}_d; \dot{\boldsymbol{\lambda}}_{int}$) or only algebraic (\mathbf{x}_a) variables and (b) a BLT block with assigned \mathbf{x}_a variables does not contain any differential variables ($\dot{\mathbf{x}}_d; \dot{\boldsymbol{\lambda}}_{int}$).
3. New n_μ unknown variables $\boldsymbol{\mu}_{int}$ are introduced (n_μ is defined below).

The index-one DAE (2) can now be defined as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_d \\ \mathbf{x}_a \\ \boldsymbol{\lambda}_{int} \\ \boldsymbol{\mu}_{int} \end{bmatrix}, \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{x}}_d \\ \dot{\mathbf{x}}_a \\ \dot{\boldsymbol{\lambda}}_{int} \\ \dot{\boldsymbol{\mu}}_{int} \end{bmatrix} \quad (17a)$$

$$\mathbf{0} = \begin{bmatrix} \mathbf{f}_d(\dot{\mathbf{x}}, \mathbf{x}, t) \\ \mathbf{f}_c(\mathbf{x}, t) \end{bmatrix}$$

$$= \begin{bmatrix} \dot{\mathbf{x}}_{der(0:n-2)} - \mathbf{x}_{der(1:n-1)} + \mathbf{G}^T \dot{\boldsymbol{\mu}}_{int} \\ \mathbf{r}_{0,d} \\ \mathbf{r}_{0,a} \\ \mathbf{r}_{DER(0)} \\ \mathbf{r}_{DER(1:n-1)} = \mathbf{G}\mathbf{x}_{der(1:n-1)} + \mathbf{g} \end{bmatrix} \quad (17b)$$

The variables in **gray color**, that is $\boldsymbol{\lambda}_{int}, \boldsymbol{\mu}_{int}, \dot{\mathbf{x}}_a$, are not used in equations (17b) and are variables needed by the integrator. The different parts of the equations are:

1. The *index vectors* $der(0:n-2), der(1:n-1)$ are defined in such a form that (for $\dot{\boldsymbol{\mu}}_{int} = \mathbf{0}$):

$$\frac{d(\mathbf{x}_{der(0:n-2)})}{dt} = \mathbf{x}_{der(1:n-1)}$$
2. $\mathbf{r}_{0,d}$ are *non-differentiated* equations of (10a), so equations e_i with $B_i = 0$ and $B_{inv,i} = 0$, that have only differentiated variables ($\dot{\mathbf{x}}_d; \dot{\boldsymbol{\lambda}}_{int}$) as assigned variables of the respective BLT block.
3. $\mathbf{r}_{0,a}$ are *non-differentiated* equations of (10a), so equations e_i with $B_i = 0$ and $B_{inv,i} = 0$, that have only algebraic variables (\mathbf{x}_a) as assigned variables of the respective BLT block.
4. $\mathbf{r}_{DER(0)}$ are constraint equations (10b) that are not differentiated, so equations e_i with $B_i = 0$ and $B_{inv,i} = 0$.
5. $\mathbf{r}_{DER(1:n-1)} = \mathbf{G}(\mathbf{x}_{der(0:n-2)}, t)\mathbf{x}_{der(1:n-1)} + \mathbf{g}(\mathbf{x}_{der(0:n-2)}, t)$ are constraint equations (10b) that are differentiated at least once, but not the highest derivative equations, so equations e_i with $B_i > 0$ and $B_{inv,i} > 0$. The number of additionally introduces variables n_μ is equal to the number of equations of $\mathbf{r}_{DER(1:n-1)}$.

6. Matrix \mathbf{G} collects the linear factors of the equations $r_{DER(1:n-1),i}$ with respect to the highest derivatives $\mathbf{x}_{der(1:n-1),i}$ appearing in the resp. equation e_i . Note, as recognized in (Führer, 1988) in a similar context, \mathbf{G} is part of the iteration matrix (Jacobian) of a BDF integrator and therefore if the iteration matrix is computed numerically, \mathbf{G} is determined *without additional effort*, see also (Arnold, 2016).

With this structuring we can now prove the following theorems:

Theorem 4: (17) is a DAE (2) under the assumption that $\frac{\partial e_i}{\partial v_j}$ for $A_j = 0$ in (10a) is regular (and not just structurally regular) and \mathbf{G} has full row rank (= the constraints are not redundant).

Proof: (a) Due to the construction, the upper two equations of (17b) are a function of $\dot{\mathbf{x}}, \mathbf{x}, t$ and the lower three equations are not functions of $\dot{\mathbf{x}}$, so (17b) has the functional dependency as required by (2a).

$$(b) \quad \frac{d\mathbf{r}_{DER(0:n-2)}}{dt} = \mathbf{r}_{DER(1:n-1)} \rightarrow$$

$$\mathbf{0} = \mathbf{G}(\dot{\mathbf{x}}_{der(0:n-2)} - \mathbf{x}_{der(1:n-1)})$$

$$= \mathbf{G}\mathbf{G}^T \dot{\boldsymbol{\mu}}_{int} \rightarrow \dot{\boldsymbol{\mu}}_{int} = \mathbf{0}$$

(c) If the highest order constraint equations in the lower part of (17b) are differentiated once, then these differentiated equations, together with the second and third equation of (17b) are the highest order derivative equations of (10a) which can be solved for the highest order derivatives (so for $\dot{\mathbf{x}}$, since $\dot{\boldsymbol{\mu}}_{int} = \mathbf{0}$) due to the assumption and therefore (2b) holds. ■

Theorem 5: (17) and (9) have the same solution space.

Proof: Since $\dot{\boldsymbol{\mu}}_{int} = \mathbf{0}$, (17) are equations (9) and differentiated equations of (9). ■

To summarize, every DAE (1) can be transformed to DAE (17) *without* solving linear or nonlinear algebraic equation systems provided the Pantelides algorithm or an equivalent structural index reduction algorithm can be applied to it. (17) is an index one DAE (2).

4.4 Example

(17) is demonstrated with the following example from (Mattsson and Söderlind, 1993) that has been extended with additional equations and unknowns to include several special cases on the basis of a simple DAE:

$$0 = u_1(t) + x_1 - x_2 \quad (18a)$$

$$0 = u_2(t) + x_1 + x_2 - x_3 + \dot{x}_6 \quad (18b)$$

$$0 = u_3(t) + x_1 + \dot{x}_3 - x_4 \quad (18c)$$

$$0 = u_4(t) + 2\ddot{x}_1 + \ddot{x}_2 + \ddot{x}_3 + \dot{x}_4 + \ddot{x}_6 \quad (18d)$$

$$0 = u_5(t) + 3\ddot{x}_1 + 2\ddot{x}_2 + x_5 + 0.1x_8 \quad (18e)$$

$$0 = u_6(t) + 2x_6 + x_7 \quad (18f)$$

$$0 = u_7(t) + 3x_6 + 4x_7 \quad (18g)$$

$$0 = u_8(t) + x_8 - \sin(x_8) \quad (18h)$$

The $u_i(t)$ are known forcing functions. Applying the Pantelides algorithm (Pantelides, 1988)² and sorting the equations on highest derivative level results in the following three BLT components:

BLT component 1 (unknowns: x_8)

$$0 = u_8(t) + x_8 - \sin(x_8) \quad (19h)$$

BLT component 2 (unknowns: \ddot{x}_6, \ddot{x}_7)

$$0 = \ddot{u}_6(t) + 2\ddot{x}_6 + \ddot{x}_7 \quad (19f)$$

$$0 = \ddot{u}_7(t) + 3\ddot{x}_6 + 4\ddot{x}_7 \quad (19g)$$

BLT component 3 (unknowns: $\dot{x}_1, \dot{x}_2, \dot{x}_3, \dot{x}_4, x_5$)

$$0 = \dot{u}_1(t) + \dot{x}_1 - \dot{x}_2 \quad (19a)$$

$$0 = \dot{u}_2(t) + \dot{x}_1 + \dot{x}_2 - \dot{x}_3 + \ddot{x}_6 \quad (19b)$$

$$0 = \dot{u}_3(t) + \dot{x}_1 + \dot{x}_3 - \dot{x}_4 \quad (19c)$$

$$0 = u_4(t) + 2\dot{x}_1 + \dot{x}_2 + \dot{x}_3 + \dot{x}_4 + \ddot{x}_6 \quad (19d)$$

$$0 = u_5(t) + 3\dot{x}_1 + 2\dot{x}_2 + x_5 + 0.1x_8 \quad (19e)$$

Transformation to the index one DAE (17) results in:

$$\mathbf{x}_d = [x_1; x_2; x_3; x_4; x_6; x_7; \dot{x}_1; \dot{x}_2; \dot{x}_3; \dot{x}_6; \dot{x}_7; \ddot{x}_6; \ddot{x}_7]$$

$$\mathbf{x}_a = [x_8]$$

$$\dot{\lambda}_{int} = [x_5]$$

$$\mathbf{r}_{0,d} = \begin{bmatrix} u_4(t) + 2\dot{x}_1 + \dot{x}_2 + \dot{x}_3 + \dot{x}_4 + \ddot{x}_6 \\ u_5(t) + 3\dot{x}_1 + 2\dot{x}_2 + x_5 + 0.1x_8 \end{bmatrix}$$

$$\mathbf{r}_{0,a} = [u_8(t) + x_8 - \sin(x_8)]$$

$$\mathbf{r}_{DER(0)} = \begin{bmatrix} u_1(t) + x_1 - x_2 \\ u_2(t) + x_1 + x_2 - x_3 + \dot{x}_6 \\ u_3(t) + x_1 + \dot{x}_3 - x_4 \\ u_6(t) + 2x_6 + x_7 \\ u_7(t) + 3x_6 + 4x_7 \end{bmatrix}$$

$$\mathbf{r}_{DER(1:n-1)} = \begin{bmatrix} \dot{u}_1(t) + \dot{x}_1 - \dot{x}_2 \\ \dot{u}_2(t) + \dot{x}_1 + \dot{x}_2 - \dot{x}_3 + \ddot{x}_6 \\ \dot{u}_6(t) + 2\dot{x}_6 + \dot{x}_7 \\ \dot{u}_7(t) + 3\dot{x}_6 + 4\dot{x}_7 \\ \ddot{u}_6(t) + 2\ddot{x}_6 + \ddot{x}_7 \\ \ddot{u}_7(t) + 3\ddot{x}_6 + 4\ddot{x}_7 \end{bmatrix}'$$

$$\mathbf{G} = \begin{bmatrix} 1 & -2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 3 & 4 \end{bmatrix}$$

$$der(0:n-2) = [1; 2; 3; 5; 6; 10; 11]$$

$$der(1:n-1) = [7; 8; 9; 10; 11; 12; 13]$$

The result is a DAE with 21 equations that can be solved with an index one DAE integrator.

4.5 Structuring the Constraint Sets

The direct mapping to (17) results often in unnecessarily large DAEs. We will therefore now improve the mapping by utilizing (partial) *static state selection*. As a first step, the inherent structure of the constraint set (10b) is (algorithmically) determined. This can be performed elegantly by utilizing results from the dummy derivative method of (Mattsson and Söderlind, 1993). The "lower derivative" equations (10b) determined by the Pantelides algorithm are ignored in the sequel and they are instead newly derived from the sorted highest derivative equations (10a) by inspecting every BLT component in sequence and for component k the follow actions are performed:

Step 1: The differentiation order of an equation in component k is reduced by one if it is a differentiated equation. The resulting set of equations forms an independent constraint set.

Step 2: The differentiation order of an unknown (= assigned variable) in component k is reduced by one if it is a differentiated variable. The resulting set of variables contains the unknowns of the derived constraint set.

Step 3: Goto Step 1, if there are still differentiated equations in the derived constraint set and apply Step 1-3 on it. Otherwise, go to Step 4.

Step 4: Order the components in such a way that within a BLT component first the lowest order derivative constraints are placed, then the constraints with one differentiation order higher and so on. The order of the BLT components is not changed.

Applying this procedure to (19) results in the following sorted ODAE:

BLT component 1 (unknowns: x_8)

$$0 = u_8(t) + x_8 - \sin(x_8)$$

BLT component 2

BLT component 2.1 (unknowns: x_6, x_7)

$$0 = u_6(t) + 2x_6 + x_7$$

$$0 = u_7(t) + 3x_6 + 4x_7$$

BLT component 2.2 (unknowns: \dot{x}_6, \dot{x}_7)

$$0 = \dot{u}_6(t) + 2\dot{x}_6 + \dot{x}_7$$

$$0 = \dot{u}_7(t) + 3\dot{x}_6 + 4\dot{x}_7$$

BLT component 2.3 (unknowns: \ddot{x}_6, \ddot{x}_7)

$$0 = \ddot{u}_6(t) + 2\ddot{x}_6 + \ddot{x}_7$$

$$0 = \ddot{u}_7(t) + 3\ddot{x}_6 + 4\ddot{x}_7$$

BLT component 2.4 (unknowns: \ddot{x}_6, \ddot{x}_7)

$$0 = \ddot{u}_6(t) + 2\ddot{x}_6 + \ddot{x}_7$$

$$0 = \ddot{u}_7(t) + 3\ddot{x}_6 + 4\ddot{x}_7$$

BLT component 3

BLT component 3.1 (unknowns: x_1, x_2, x_3)

$$0 = u_1(t) + x_1 - x_2$$

$$0 = u_2(t) + x_1 + x_2 - x_3 + \dot{x}_6$$

BLT component 3.2 (unknowns: $\dot{x}_1, \dot{x}_2, \dot{x}_3, x_4$)

² For simplicity of the example, the equations contain higher derivatives. The Pantelides algorithm can be easily generalized to this case by providing a corresponding \mathbf{A} vector.

$$\begin{aligned}
0 &= \dot{u}_1(t) + \dot{x}_1 - \dot{x}_2 \\
0 &= \dot{u}_2(t) + \dot{x}_1 + \dot{x}_2 - \dot{x}_3 + \ddot{x}_6 \\
0 &= u_3(t) + x_1 + \dot{x}_3 - x_4 \\
\text{BLT component 3.3 (unknowns: } \ddot{x}_1, \ddot{x}_2, \ddot{x}_3, \dot{x}_4, x_5) \\
0 &= \ddot{u}_1(t) + \ddot{x}_1 - \ddot{x}_2 \\
0 &= \ddot{u}_2(t) + \ddot{x}_1 + \ddot{x}_2 - \ddot{x}_3 + \ddot{x}_6 \\
0 &= \dot{u}_3(t) + \dot{x}_1 + \dot{x}_3 - \dot{x}_4 \\
0 &= u_4(t) + 2\ddot{x}_1 + \ddot{x}_2 + \ddot{x}_3 + \dot{x}_4 + \ddot{x}_6 \\
0 &= u_5(t) + 3\ddot{x}_1 + 2\ddot{x}_2 + x_5 + 0.1x_8
\end{aligned}$$

Due to the construction, a variable like \ddot{x}_6 is computed in a BLT sub-component (here: BLT component 2.3) and used only in later BLT components (here BLT component 3.2).

In the next step the number of constraint equations and unknowns shall be reduced statically. In principal this is just a variant of the dummy derivative method. However, (Mattsson and Söderlind, 1993) describe a (very useful) conceptual algorithm, but not how to implement it practically for nonlinear systems which requires non-trivial extensions. In order to do this, an auxiliary algorithm is needed that is described in the next section.

4.6 Tearing with retained solution space

Starting point is a nonlinear algebraic equation system

$$0 = \mathbf{g}(\mathbf{z}), \quad \mathbf{z} \in \mathbb{R}^{nz}, \mathbf{g} \in \mathbb{R}^{ng}, ng \leq nz \quad (21)$$

where the number of equations ng is at most the same as the number of unknowns nz , but it may be less. The goal is to split this equation system in an explicitly solvable part and an implicit part:

$$\mathbf{z}_e := \mathbf{g}_e(\mathbf{z}_e, \mathbf{z}_t) \quad (22a)$$

$$0 = \mathbf{g}_r(\mathbf{z}_e, \mathbf{z}_t) \quad (22b)$$

where \mathbf{z}_e can be solved recursively from (22a) by utilizing only already computed elements of \mathbf{z}_e when calculating a new element of \mathbf{z}_e . \mathbf{z}_e are called the explicitly solvable variables of \mathbf{z} , \mathbf{z}_t the tearing variables of \mathbf{z} and \mathbf{g}_r the residue equations. The interpretation is that when \mathbf{z}_t is given, \mathbf{z}_e can be explicitly computed and \mathbf{z}_t must be provided in such a way that the residues equations are fulfilled.

In order that this transformation is practically useful, the solution space of (22) must be identical to the solution space of (21). In the following an algorithm is derived to automatically deduce (22) from (21) such that (22) has the same solution space as (21).

Tearing is a well-known technique and was probably introduced by (Kron, 1962). A recent extensive literature survey is given in (Bahainv et al., 2016a). In (Bahainv et al., 2016b) a novel tearing technique is proposed based on an integer programming formulation with a custom branch and bound algorithm. Tearing was used in object-oriented modeling, for example in (Elmqvist and Otter, 1994) and in (Carpanzano et al., 1997). The following algorithm sketch for automatic tearing is due to a

development of the authors of this paper in 1999. Some results of it have been reported in (Otter, 1999):

Equation (22a) can be interpreted as a DAG (Directed Acyclic Graph) where the nodes are equations $g_{e,i}$ together with the explicitly solved variables $z_{e,i}$ of $g_{e,i}$, and the edges of a node i are directed to the nodes of the remaining variables $z_{e,i}$ appearing in $g_{e,i}$. The goal of the algorithm is to construct such a DAG using equations and unknowns from (21). Initially, the DAG is empty and is constructed with the following steps:

Step 1: Select an array equation i from (21) that is not yet in the DAG (in a first step equations are selected according to their initial ordering; later, heuristics for the selection are added based on additional information).

Step 2: Select an array variable j from equation i that is (a) not assigned to equation i , (b) not yet selected before in this equation, and (c) can be explicitly solved from equation i (so the array size of variable j and of the array equation i must agree) without changing the solution space of this equation (e.g. using only variables z_j as candidates that are within linear factors $c \cdot z_j$ where c is a constant with $c \neq 0$; see also the discussion about heuristics below and (Otter, 1999)).

Step 3: Add equation i and the selected variable j from Step 2 as node to the potential DAG.

Step 4: Traverse the potential DAG *starting* from the added equation node and use a standard DFS (Depth First Search) to determine if there is a cycle for this equation node. If there is a cycle, remove the last added equation from it and if not all variables of equation i have been inspected, go to Step 2. If no cycle is present, continue with the next step.

Step 5: If not all equations have been inspected, go to Step 1. Otherwise, stop (equations (22a) are the equations in the DAG, \mathbf{z}_e are the assigned variables in the DAG).

With m the number of variable incidences in the system of equations, the worst time complexity of this algorithm to find the tearing variables and residue equations is $O(m^2)$ because every DFS from every inserted node has a worst-time complexity of $O(m)$ and this operation is executed potentially m times (since in the worst case a DFS is performed on every variable in every equation).

In the last decade, several new algorithms have been developed to perform *incremental cycle detection* when inserting vertices and edges to an existing DAG. The algorithm of (Bender et al., 2016) has the currently best worst case performance for sparse DAGs with $O(\min(m^{1/2}, n^{2/3}) \cdot m)$, where n is the number of vertices and m the number of edges. For comparison of such algorithms, see (Sigurdsson, 2016).

The implementation in Modia/Julia uses currently the simple *Algorithm N* of (Bender et al., 2016) that has a worst time complexity of $O(n \cdot m)$. For example, when an equation system of the following form is present

$$\begin{aligned} 0 &= f_1(z_1, z_n) \\ 0 &= f_2(z_2, z_1) \\ 0 &= f_3(z_3, z_2) \\ &\dots \\ 0 &= f_n(z_n, z_{n-1}) \end{aligned} \quad (23)$$

and the equations are added in the order 1,2,..., n (or also in the order n,n-1,...,1), this tearing algorithm has a complexity of $O(n)$ to find the single tearing variable. On a standard notebook, this takes about 2 s for $n = 10^6$.

The result of the tearing algorithm depends on the order in which equations are added to the DAG. There are at least two useful heuristics: (a) If the user explicitly requires to solve equations for particular variables (for example using the operator "!=" instead of "=" in the Modia prototype, or the algorithm section or a function call in Modelica), then these equations are *inspected first*. All equations belonging to the connection graph (especially all linear equations with only Integer coefficients, see section 5) are inspected *last*, because it seems most natural for a physical system to cut an algebraic loop along the connection graph, and not within a component.

Tearing can have a significant influence on the reliability of the numerical solution and therefore it is not always clear whether it is useful to apply tearing to solve algebraic loops. For this reason, more heuristics need to be added, for example, arrays might be solved in Step 2 above only, if they appear as linear term and the linear factors are the scalars +1 or -1, in order to avoid a potential division by a small value, or if the linear term is an orthogonal matrix so that inversion is reliable.

4.7 Partial state selection

The tearing algorithm from the last section shall now be used to partially solve the constraint equations and thereby identify states and dummy states. The constraint equation sets are derived with the approach of section 4.5 and all these sets have the following structure:

$$\begin{aligned} \mathbf{0} &= \mathbf{g}(\mathbf{x}_1, \mathbf{x}_2) \\ \mathbf{x}_1 &\in \mathbb{R}^{nx1}, \mathbf{x}_2 \in \mathbb{R}^{nx2}, \mathbf{g} \in \mathbb{R}^{ng}, ng \leq nx1 \end{aligned} \quad (24)$$

where \mathbf{x}_1 are potential states, \mathbf{x}_2 are potential states that have been already handled in a previous BLT sub-component (so can be treated as known variables) and $\mathbf{g}(\cdot)$ is a set of algebraic constraint equations on \mathbf{x}_1 . Via tearing the constraint equation set (24) can be split

in an explicitly solvable part $\mathbf{g}_e(\cdot)$ and in an implicit part $\mathbf{g}_r(\cdot)$:

$$\begin{aligned} \mathbf{x}_{1e} &:= \mathbf{g}_e(\mathbf{x}_{1e}, \mathbf{x}_{1t}, \mathbf{x}_2) \\ \mathbf{0} &= \mathbf{g}_r(\mathbf{x}_{1e}, \mathbf{x}_{1t}, \mathbf{x}_2) \end{aligned} \quad (25)$$

The explicitly solvable variables \mathbf{x}_{1e} are dummy states according to the dummy derivative method of (Mattsson and Söderlind, 1993). The tearing variables \mathbf{x}_{1t} remain potential states. If no residue equations $\mathbf{g}_r(\cdot)$ are present, the full set of states has been identified. If the equations are *linear* in $\mathbf{x}_{1e}, \mathbf{x}_{1t}$, then the residue equations can be transformed to a linear equation in the tearing variables, see for example (Elmqvist and Otter, 1994). For *constant coefficient linear systems*, this equation system can be at once solved. For *variable coefficient linear systems*, an inline solution of the linear system might be used, at least for systems with up to three unknowns. In all these cases the full set of states has been identified as well.

The state selection with tearing is applied on all constraint sets starting from the lower to the higher derivative constraint sets $\text{ec}[1] \dots \text{ec}[\text{end}]$ of every BLT component. Since by construction $\text{ec}[i]$ is a superset of $\text{ec}[i-1]$, and the unknowns $\dot{\mathbf{v}}_i$ of $\text{ec}[i]$ are a differentiated superset of the unknowns \mathbf{v}_{i-1} , all explicitly solvable equations of $\text{ec}[i-1]$ are also explicitly solvable equations of $\text{ec}[i]$ and therefore tearing need only to be performed *additionally* on equations that are *added* at a higher level.

Applying the partial state selection for example on BLT sub-component 3.1 of (20) identifies x_2 as state and computes the dummy states from:

$$\begin{aligned} x_1 &:= -u_1(t) + x_2 \\ x_3 &:= -u_2(t) - x_1 + x_2 - \dot{x}_6 \end{aligned}$$

The same analysis holds for BLT sub-component 3.2, so \dot{x}_2 is also a state and the following equations are directly deduced from the previous equations:

$$\begin{aligned} \dot{x}_1 &:= -\dot{u}_1(t) + \dot{x}_2 \\ \dot{x}_3 &:= -\dot{u}_2(t) - \dot{x}_1 + \dot{x}_2 - \ddot{x}_6 \end{aligned}$$

Tearing must therefore only be applied for the additional equation of this sub-component leading to:

$$x_4 := u_3(t) + x_1 + \dot{x}_3$$

Applying the partial state selection on BLT sub-component 2.1 of (20) identifies x_6 as state and computes the dummy state from

$$x_7 := -u_6(t) - 2x_6$$

The residue equation is linear in x_6 and can then be solved for:

$$x_6 := (-u_7(t) - 4u_6(t))/5$$

Once partial state selection has been applied on all constraint sets, all the dummy states and their derivatives, *up to the highest derivatives of the dummy states*, are removed from \mathbf{x} and $\dot{\mathbf{x}}$ and are computed

locally from the remaining \mathbf{x} and $\dot{\mathbf{x}}$. The remaining equations can be transformed to DAE (17). Hereby, the sorting order of the locally solved equations matters and therefore the ordering has to be performed according to the BLT ordering and the corresponding ordering of the constraint sets.

Applying partial state selection with tearing on example (20) results in the following DAE (17):

$$\begin{aligned}
 \mathbf{x}_d &= [x_2; \dot{x}_2] \\
 \mathbf{x}_a &= [x_8] \\
 \dot{\lambda}_{int} &= [] \\
 \dot{\mu}_{int} &= [] \\
 x_6 &:= (-u_7(t) - 4u_6(t))/5 \\
 x_7 &:= -u_6(t) - 2x_6 \\
 \dot{x}_6 &:= (-\dot{u}_7(t) - 4\dot{u}_6(t))/5 \\
 \dot{x}_7 &:= -\dot{u}_6(t) - 2\dot{x}_6 \\
 \ddot{x}_6 &:= (-\ddot{u}_7(t) - 4\ddot{u}_6(t))/5 \\
 \ddot{x}_7 &:= -\ddot{u}_6(t) - 2\ddot{x}_6 \\
 \ddot{x}_6 &:= (-\ddot{u}_7(t) - 4\ddot{u}_6(t))/5 \\
 \ddot{x}_7 &:= -\ddot{u}_6(t) - 2\ddot{x}_6 \\
 x_1 &:= -u_1(t) + x_2 \\
 x_3 &:= -u_2(t) - x_1 + x_2 - \dot{x}_6 \\
 \dot{x}_1 &:= -\dot{u}_1(t) + \dot{x}_2 \\
 \dot{x}_3 &:= -\dot{u}_2(t) - \dot{x}_1 + \dot{x}_2 - \dot{x}_6 \\
 x_4 &:= u_3(t) + x_1 + \dot{x}_3 \\
 \ddot{x}_1 &:= -\ddot{u}_1(t) + \ddot{x}_2 \\
 \ddot{x}_3 &:= -\ddot{u}_2(t) - \ddot{x}_1 + \ddot{x}_2 - \ddot{x}_6 \\
 \dot{x}_4 &:= \dot{u}_3(t) + \dot{x}_1 + \dot{x}_3 \\
 x_5 &:= -u_5(t) - 3\dot{x}_1 - 2\dot{x}_2 - 0.1x_8 \\
 \mathbf{r}_{0,d} &= [u_4(t) + 2\ddot{x}_1 + \ddot{x}_2 + \ddot{x}_3 + \ddot{x}_4 + \ddot{x}_6] \\
 \mathbf{r}_{0,a} &= [u_8(t) + x_8 - \sin(x_8)] \\
 \mathbf{G} &= [] \\
 \text{der}(0:n-2) &= [1] \\
 \text{der}(1:n-1) &= [2] \quad (\rightarrow \dot{x}_{d,1} = x_{d,2})
 \end{aligned}$$

The result is a DAE with 3 equations (without partial state selection, it had been 21 equations).

Using tearing for the constraint equations seems to be always a useful approach because this can significantly reduce the number of variables that need to be discretized by the integrator. For example, assume a tree-structured multi-body system is modeled with the Modelica.Mechanics.MultiBody library and has n bodies that are connected together by revolute joints. Without partial state selection, DAE (17) consists of $(6 \cdot 3 + 4 \cdot 9 + 2)n = 56n$ equations³. After partial state selection with tearing the DAE consists of the $2n$ equations (15), provided the simple heuristics from section 4.6 are used.

However, it is less clear whether tearing is also useful when applied on the highest order derivative equations that are no derivatives of constraints. For example, discretized partial differential equations typically lead to structures where tearing cannot reduce

the equation size much but will completely destroy the sparseness and may be numerically less reliable. In such cases it is much better to not apply tearing and rely on the sparse matrix handling used by the integrator. More investigations are needed here.

5 Exact Removal of Singularities

5.1 Overview

In this section a new method is proposed to exactly remove certain types of singularities of a physical system model provided as DAE (1). The result is again a DAE in form (1). A typical example is shown in Figure 1.

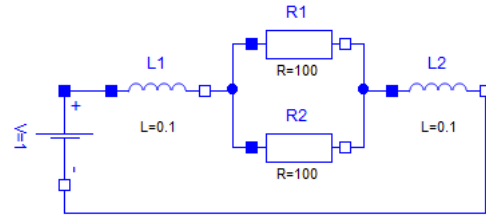


Figure 1. Modelica model of an electrical circuit that is difficult to simulate. It can be automatically handled with the method of this section.

Modelica tools transform DAEs (1) with structural symbolic algorithms. These algorithms fail for the circuit in Figure 1 (as well as other useful application models). Since this electrical circuit is not grounded, the potentials of the electrical pins can float, that is, the system equations are underdetermined. On the other hand, the equations are overdetermined regarding currents. An analysis, literature survey, and a solution based on exploitation of the connection graph is presented in (Elmqvist and Mattsson, 2016).

Additionally, the currents i_{L1}, i_{L2} appear differentiated in the inductors L_1, L_2 and are therefore assumed to be known. The two inductors are connected by two resistors in parallel leading to the following connection equations for the currents:

$$\begin{aligned}
 i_{L1} &= i_{R1} + i_{R2} \\
 i_{L2} &= i_{R1} + i_{R2}
 \end{aligned} \tag{26}$$

where i_{R1}, i_{R2} are the currents through the resistances R_1, R_2 . Structurally, (26) are two equations for two unknown algebraic variables i_{R1}, i_{R2} since the potential states i_{L1}, i_{L2} are assumed to be known. Therefore, structural algorithms assume that i_{R1}, i_{R2} can be determined from (26). However, when subtracting the two equations $i_{L1} + i_{L2} = 0$, that is an equation with only known variables is obtained, which means that one of the two variables cannot be a state. As a result structural index reduction algorithms, as discussed in section 3, will fail on this circuit.

The method below is based on the observation that object-oriented models have a particular structure: Zero-sum equations of flow variables i in connectors

³ $\mathbf{x} = [\mathbf{r}_i; \dot{\mathbf{r}}_i; \mathbf{T}_i; \dot{\mathbf{T}}_i; \boldsymbol{\omega}_i; \mathbf{r}_i^{CM}; \dot{\mathbf{r}}_i^{CM}; \mathbf{T}_i^{CM}; \dot{\mathbf{T}}_i^{CM}; \boldsymbol{\omega}_i^{CM}; \varphi_i; \dot{\varphi}_i]$
 $i = 1, 2, \dots, n$

c_k have the form $\sum c_k \cdot i = 0$. After alias elimination, relative potential variables in a component have the form $u_{rel} = c_k \cdot v - c_j \cdot v$. All these equations have the common property that they are linear and the coefficients are integer (even +1 or -1). Due to the integer coefficients *exact analysis* is possible. In particular, singularities and state constraints in linear equations with integer coefficients are identified and if possible removed. The latter *cannot* be achieved with methods based on connection graphs, such as (Elmqvist and Mattsson, 2016) or similar techniques.

When applied to the circuit in Figure 1, the method in section 5.3 gives the result that the following equation shall be removed since redundant:

$$-L2.n.i - V.n.i = 0$$

In order to make all potentials well-defined, the following equation is added:

$$L2.n.v = 0$$

In order to make the state constraints structurally visible, the equation

$$-R1.p.i - R2.p.i - L1.n.i = 0$$

is replaced by

$$-L1.p.i + L2.p.i = 0$$

5.2 Transformation to upper trapezoidal form

The new approach is based on a utility algorithm to perform a fraction-free Gaussian elimination of linear algebraic equations with integer coefficients. The algorithm is a slight generalization of (Bareiss, 1968), see also (Turner 1995). Starting point is a linear algebraic equation system

$$\mathbf{A} \cdot \mathbf{X} = \mathbf{B}, \quad \mathbf{A} \in \mathbb{Z}^{na1 \times na2}, \mathbf{B} \in \mathbb{Z}^{na1 \times nb2} \quad (27)$$

where \mathbf{A} and \mathbf{B} are sparse, rectangular integer matrices. The goal is to use fraction-free Gaussian elimination to transform (27) to upper trapezoidal form:

$$\begin{bmatrix} \mathbf{A}_{u11} & \mathbf{A}_{u12} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X}_{u1} \\ \mathbf{X}_{u2} \end{bmatrix} = \begin{bmatrix} \mathbf{B}_{u1} \\ \mathbf{B}_{u2} \end{bmatrix} \quad (28)$$

where $\mathbf{A}_{u11}, \mathbf{A}_{u12}, \mathbf{B}_{u1}, \mathbf{B}_{u2}$ are integer matrices and \mathbf{A}_{u11} is quadratic, regular, and upper triangular with non-zeros on the diagonal, that is $\text{rank}(\mathbf{A}) = \text{size}(\mathbf{A}_{u11}, 1)$. Additionally, permutation vector p_1 describes the accumulated row interchanges of \mathbf{A}, \mathbf{B} and permutation vector p_2 describes the accumulated column interchanges of \mathbf{A} and row interchanges of \mathbf{X} such that $\mathbf{X}_u = \mathbf{X}[p_2, :]$. Permutation vector p_2 is selected such that if possible the "upper" part of \mathbf{X} is utilized in \mathbf{X}_{u1} (this is used in section 5.3).

Algorithm 1 is a straightforward implementation of Gaussian elimination with full pivoting for sparse matrices. The key point are the two equations at the end with the "**div**(a,b)" operator where equation i is subtracted from equation k with a fraction free operation. Here, the non-trivial to derive property is used that the integer division $\text{div}(a,b)=a/b$ has no remainder in this case. For details see (Bareiss, 1968).

Algorithm 1

```
(Au,Bu,rk,p1,p2) = upperTrapezoidal(A,B)
# Transform the rectangular linear system A*X=B
# to upper trapezoidal form Au*X[p2,:]=Bu
# A,B,Au,Bu are integer matrices
# initialize variables
(na1,na2) = size(A); nb2 = size(B,2); p1=1:na1; p2=1:na2
Au = copy(A); Bu = copy(B);
oldPivot = 1
# inspect all rows of Au
for k = 1:na1
  # search column wise for a pivot in Au[k,min(k,na2):]
  pivotFound = false
  for k2 = k:na2
    for (k1,pivot) in < row indices k1 and values pivot of
                          non-zero entries of column k2 >
      if k1 >= k && pivot != 0
        pivotFound = true
        p1k = k1
        p2k = k2
        break
      end
    end
  end
  if pivotFound; break; end
end
# exchange rows/columns such that Au[k,min(k,na2)] != 0
if pivotFound
  <exchange rows k and p1k of Au, Bu, p1, and
  exchange columns k and p2k of Au and row p2k of p2 >
else # submatrix Au[k:na1,:] has only zeros
  rk = k-1
  return (Au, Bu, rk, p1, p2)
end
# Subtract row k from rows k+1:na1
k1 = k+1
j = k1:na2
for (i,val) in < row indices i and values val of non-zero
               entries of column k >
  if i >= k1
    Bu[i,:] = div(pivot*Bu[i,:]-val*Bu[k,:], oldPivot)
    Au[i,j] = div(pivot*Au[i,j]-val*Au[k,j], oldPivot)
    Au[i,k] = 0
  end
end
oldPivot = pivot
end
return (Au, Bu, rk, p1, p2)
```

5.3 Identifying singularities in the model

Starting point is the largest subset of equations of DAE (1) that is described by a linear algebraic system

$$\mathbf{A}_x \mathbf{v}_x + \mathbf{A}_y \mathbf{v}_y + \mathbf{A}_c \mathbf{v}_c + \mathbf{A}_r \mathbf{v}_r = \mathbf{0} \quad (29)$$

where $\mathbf{A}_x, \mathbf{A}_y, \mathbf{A}_c, \mathbf{A}_r$ are sparse matrices with (scalar) integer elements of appropriate dimensions and $\mathbf{v}_x, \mathbf{v}_y, \mathbf{v}_c, \mathbf{v}_r$ are vectors of variables of (1). An element of these vectors may be a scalar, an array, or an instance of any data structure for which the operators "+", "-", "*" are defined (overloaded) as operations between instances of the same type and between an instance of the type and a scalar integer. Furthermore, it is assumed that *within one equation* the

variables are all of the same type and arrays have the same dimension sizes (so, one equation may state a relationship between $[3,3]$ matrices, whereas another equation between $[6]$ vectors, and yet another one between scalars). Since an equation can only depend on variables of the same type, equations (29) are basically a disjunct set of equations for different types that are analyzed conceptually in an independent way from each other. The various vectors have the following meaning:

\mathbf{v}_x	Variables used in the derivative operator, so $\mathbf{der}(\mathbf{v}_{x,i})$ appears in the model.
\mathbf{v}_y	After removing equations (29) from (1), variables \mathbf{v}_y are no longer present in (1). Therefore, these variables <i>must be computed from</i> (29). For example, if $v_{rel} = p.v - n.v$ and the connector variables $p.v$ and $n.v$ are not used otherwise in the model, they are part of \mathbf{v}_y .
\mathbf{v}_c	Variables defined by a parameter expression. For example, if $v_0 = 5$ and v_0 is utilized in other linear equations with integer coefficients, then v_0 is part of \mathbf{v}_c .
\mathbf{v}_r	All remaining variables that do not belong to one of the other three categories above.

In a first step, equations (29) are restructured to:

$$\mathbf{A}_{yr} \mathbf{v}_{yr} = \mathbf{B}_{xc} (-\mathbf{v}_{xc}) \quad (30)$$

with

$$\begin{aligned} \mathbf{A}_{yr} &= [\mathbf{A}_y \quad \mathbf{A}_r], \quad \mathbf{v}_{yr} = \begin{bmatrix} \mathbf{v}_y \\ \mathbf{v}_r \end{bmatrix} \\ \mathbf{B}_{xc} &= [\mathbf{A}_x \quad \mathbf{A}_c], \quad \mathbf{v}_{xc} = \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_c \end{bmatrix} \end{aligned} \quad (31)$$

With Algorithm 1 from section 5.2

$$(\mathbf{A}_u, \mathbf{B}_u, rk, p_1, p_2) = \text{upperTrapezoidal}(\mathbf{A}_{yr}, \mathbf{B}_{xc})$$

(30) can be transformed to upper trapezoidal form:

$$\begin{bmatrix} \mathbf{A}_{u,11} & \mathbf{A}_{u,12} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v}_{u1} \\ \mathbf{v}_{u2} \end{bmatrix} = \begin{bmatrix} \mathbf{B}_{u1} \\ \mathbf{B}_{u2} \end{bmatrix} \cdot (-\mathbf{v}_{xc}) \quad (32)$$

where $\mathbf{A}_{u,11}$ is a quadratic, regular, upper triangular integer matrix of size $[rk, rk]$ with non-zeros on the diagonal, $\mathbf{v}_{u1} = \mathbf{v}_{yr}[p_2[1:rk]]$ are rk elements of \mathbf{v}_{yr} and $\mathbf{v}_{u2} = \mathbf{v}_{yr}[p_2[rk+1:]]$ are the remaining elements of \mathbf{v}_{yr} . With $\mathbf{B}_{u2} = [\mathbf{B}_{u2,1} \quad \mathbf{B}_{u2,2}]$, the lower part of (32) can be stated as:

$$\mathbf{B}_{u2,1} \mathbf{v}_x = \mathbf{B}_{u2,2} (-\mathbf{v}_c) \quad (33)$$

Using Algorithm 1 again:

$$(\mathbf{A}_{ux}, \mathbf{B}_{ux}, rk_x, p_{x1}, p_{x2}) = \text{upperTrapezoidal}(\mathbf{B}_{u2,1}, \mathbf{B}_{u2,2})$$

(33) can be transformed to upper trapezoidal form:

$$\begin{bmatrix} \mathbf{A}_{ux,11} & \mathbf{A}_{ux,12} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v}_{ux1} \\ \mathbf{v}_{ux2} \end{bmatrix} = \begin{bmatrix} \mathbf{B}_{ux1} \\ \mathbf{B}_{ux2} \end{bmatrix} \cdot (-\mathbf{v}_c) \quad (34)$$

where $\mathbf{A}_{ux,11}$ is a quadratic, regular, upper triangular integer matrix of size $[rk_x, rk_x]$ with non-zeros on the diagonal, $\mathbf{v}_{ux1} = \mathbf{v}_x[p_{x2}[1:rk_x]]$ are rk_x elements of \mathbf{v}_x and $\mathbf{v}_{ux2} = \mathbf{v}_x[p_{x2}[rk_x+1:]]$ are the remaining elements of \mathbf{v}_x .

From (32) and (34) the following conclusions can be drawn regarding singularities in the model equations:

- If \mathbf{B}_{ux2} is not the zero matrix, then there are constraints $\mathbf{B}_{ux2} \mathbf{v}_c = \mathbf{0}$ between the parameter expressions \mathbf{v}_c . A tool may reject such a model. In the following it is assumed that $\mathbf{B}_{ux2} = \mathbf{0}$.
- If $rk + rk_x < \text{length}(\mathbf{v}_{yr})$, then the lower part of (34) are zero-equations and represent redundant equations. As a result, the original equations with row indices $p_1[rk + p_{1x}[rk_x + 1, :]]$ can be removed since they can be expressed as a linear combination of the other integer equations. A tool may just remove these equations and print an information message that it removed them.
- If \mathbf{v}_{u2} contains elements of \mathbf{v}_y , then these variables can have an arbitrary value. For example assume that \mathbf{v}_{yr} are scalar real variables, then (32) can be solved for \mathbf{v}_{u1} :

$$\mathbf{v}_{u1} = -\mathbf{A}_{u,11}^{-1} (\mathbf{A}_{u,12} \mathbf{v}_{u2} + \mathbf{B}_{u1} \mathbf{v}_{xc})$$

Therefore, for given states \mathbf{v}_{xc} and given values of \mathbf{v}_{u2} , variables \mathbf{v}_{u1} can be uniquely computed. Note, since variables \mathbf{v}_y *must* be computed from (29) they can be arbitrarily set, if part of vector \mathbf{v}_{u2} . Since variables \mathbf{v}_r appear also in the remaining model equations, they need to be computed in these remaining model equations. A tool may either reject a model where \mathbf{v}_{u2} contains elements of \mathbf{v}_y , or may set arbitrary values (say the null-element of the respective type) and print an information message.

- The upper part of (34) can be formulated as:

$$\mathbf{A}_{ux,11} \mathbf{v}_{ux1} = -\mathbf{A}_{ux,12} \mathbf{v}_{ux2} - \mathbf{B}_{ux1} \mathbf{v}_c \quad (35)$$

Since $\mathbf{A}_{ux,11}$ is regular this means that \mathbf{v}_{ux1} can be computed from \mathbf{v}_{ux2} and \mathbf{v}_c and therefore \mathbf{v}_{ux1} are (dependent) dummy states. A tool can either utilize this information directly and transform the model equations with this information, or the equations leading to (35), that is the equations of (29) with row indices $p_1[rk + p_{1x}[1:rk_x]]$, are replaced by equations (35). Since $\mathbf{A}_{ux,11}$ is upper triangular, the constraints between the states are *structurally* visible and therefore index reduction with structural algorithms (see section 3) is possible.

To summarize, with the transformation of the integer equations (29) of DAE (1) to upper trapezoidal form (32),(34) an important set of singularities can be exactly identified and *if possible and desired removed*.

6 Outlook

With this paper new algorithms are provided to start from a high level modeling language like Modelica or Modia and generate code for standard Index-1 DAE integrators. The algorithms are designed to keep array data structures intact from the model language until the generated code. Furthermore, no equation systems are solved to transform to index 1 form and therefore the sparsity of the model equations is kept. As a consequence, sparse matrix methods can be utilized in the DAE integrator.

In the paper it was not discussed how to initialize the index-1 DAEs. In principal similar techniques can be used as for Modelica models. Currently, in Modia it is experimented with a new form of initialization where start values $\mathbf{x}_0(t_0^-)$ can be provided that do not fulfill the constraints of (2), so $\mathbf{f}_c(\mathbf{x}_0(t_0^-), t_0^-) \neq 0$. Via Dirac impulses of the derivatives of the discontinuous start values, consistent start values $\mathbf{x}_0(t_0^+)$ are computed with the new technique of impulse handling for DAEs (2), developed in (Benveniste et al., 2017).

In industrial applications often steady-state initialization is required. This is still a difficult topic and not yet satisfactorily solved. Typically, reliable steady-state initialization requires the use of a *probability one* homotopy method; see for example (Melville et al., 1993; Sielemann, 2012). It is an open question how to restrict the special index-one DAEs (2) so that probability one homotopy methods can be applied.

The goal is to further extend the algorithms and the Modia prototype in order to be able to simulate multi-mode systems, where the number of equations and unknowns can change during simulation (for example to simulate drastic failure cases or perform end-to-end simulations of complicated scenarios).

Acknowledgements

The authors would like to thank Martin Arnold from University Halle-Wittenberg for discussions to understand the fine details of multibody integrators.

References

- M. Arnold (2016): *DAE aspects of multibody systems*. In A. Ilchmann, T. Reis (eds.): *Surveys in Differential-Algebraic Equations IV*. - Springer, 2017 (in print). - A preliminary version of this material was published as Technical Report 01-2016, Martin Luther University Halle-Wittenberg, Report No. 01, 2016. <http://sim.mathematik.uni-halle.de/reports/sources/2016/01-2016.pdf>
- U.M. Ascher, L.R. Petzold (1991): *Projected Implicit Runge-Kutta Methods for Differential-Algebraic Equations*. SIAM J. Numer. Anal., 28(4), pp. 1097–1120.
- A. Bahainv, H. Schichl, A. Neumaier (2016a): *Tearing systems of nonlinear equations. I. A survey*. http://www.mat.univie.ac.at/~neum/ms/tearing_survey.pdf
- A. Bahainv, H. Schichl, A. Neumaier (2016b): *Tearing systems of nonlinear equations. II. A practical exact algorithm*. http://www.mat.univie.ac.at/~neum/ms/tearing_exact_algorithm.pdf
- E.H. Bareiss (1968): *Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination*. Math. Comp. 22, pp. 565-578.
- M.A. Bender, J.T. Fineman, S. Gilbert, R.E. Tarjan (2016): *A New Approach to Incremental Cycle Detection and Related Problems*. ACM Transactions on Algorithms, Volume 12, Issue 2.
- A. Benveniste, B. Caillaud, H. Elmqvist, K. Ghorbal, M. Otter, and Marc Pouzet (2017): *Multi-Mode DAE Models Challenges, Theory and Implementation*. Lecture Notes on Computer Science, submitted for review.
- J. Bezanson, A. Edelman, S. Karpinski and V.B. Shah (2017): *Julia: A Fresh Approach to Numerical Computing*. SIAM Review, Vol. 59, No. 1, pp. 65-98. <http://julialang.org/publications/julia-fresh-approach-BEKS.pdf>; see also: <http://julialang.org/>
- K.E. Brenan, S.L. Campbell, and L.R. Petzold (1996): *Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*. SIAM.
- E. Carpanzano, R. Girelli (1997): *The Tearing Problem: Definition, Algorithm and Application to Generate Efficient Computational Code from DAE Systems*. Proceedings of 2nd Mathmod Vienna, IMACS Symposium on Mathematical Modelling, Wien.
- S. Chowdhry, H. Krendl, and A.A. Linninger (2004): *Symbolic numeric index analysis algorithm for differential algebraic equations*. Industrial and Engineering Chemistry Research. Vol. 43, Issue 14, pp. 3886-3894.
- W. Cook, and A. Rohe (1999): *Computing Minimum-Weight Perfect Matchings*. INFORMS Journal of Computing, Vol. 11. www.math.uwaterloo.ca/~bico/papers/match_ijoc.pdf
- I.S. Duff (1981): *On algorithms for obtaining a maximum transversal*. ACM Trans. Math. Software, Vol. 7, Issue 3.
- I.S. Duff, K. Kaya, and B. Ucar (2011): *Design, Implementation, and Analysis of Maximum Transversal Algorithms*. ACM Trans. Math. Software, Vol. 38, Issue 2.
- J. Edmonds (1965): *Paths, Trees, and Flowers*. Canadian Journal of Mathematics. Vol. 17, pp. 449-467. <https://cms.math.ca/openaccess/cjm/v17/cjm1965v17.0449-0467.pdf>
- E. Eich (1993): *Convergence Results for a Coordinate Projection Method Applied To Mechanical Systems with Algebraic Constraints*. SIAM J. Numer. Anal. Vol. 30, No. 5, pp. 1467-1482.
- H. Elmqvist, M. Otter (1994): *Methods for Tearing Systems of Equations in Object-Oriented Modeling*. Proceedings ESM'94, European Simulation Multiconference, Barcelona, Spain, June 1–3, pp. 326–332.
- H. Elmqvist, T. Henningsson, M. Otter (2016): *System Modeling and Programming in a Unified Environment based on Julia*. Proceedings of ISOla 2016 Conference Oct. 10-14, T. Margaria and B. Steffen (Eds.), Part II, LNCS 9953, pp. 198-217. <http://www.isola-conference.org/isola2016/proceedings.html>

- H. Elmqvist, S.E. Mattsson (2016): *Exploiting Model Graph Analysis for Simplified Modeling and Improved Diagnostics*. Proceedings EOOLT '16, April 18, Milano, Italy.
- H. Elmqvist, T. Henningsson, M. Otter (2017): *Innovations for future Modelica*. Modelica Conference 2017, Prague.
- J. Frenkel, G. Kunze, P. Fritzson (2012): *Survey of appropriate matching algorithms for large scale systems of differential algebraic equations*. Proceedings of the 9th International Modelica Conference, Munich.
<http://www.ep.liu.se/ecp/076/045/ecp12076045.pdf>
- C. Führer (1988): *Differential-algebraische Gleichungssysteme in mechanischen Mehrkörpersystemen. Theorie, numerische Ansätze und Anwendungen*. PhD thesis, TU München, Mathematisches Institut und Institut für Informatik.
- C.W. Gear (1988): *Differential-Algebraic Equation Index Transformations*. SIAM J. Sci. Stat. Comput., Vol. 9, No. 1, pp. 39-47.
- C.W. Gear, G.K. Gupta and B. Leimkuhler (1985): *Automatic integration of Euler-Lagrange equations with constraints*. J. Comp. Appl. Math., 12&13, pp. 77-90.
- A.C. Hindmarsh, P.N. Brown, K.E. Grant, S.L. Lee, R. Serban, D.E. Shumaker, and C. S. Woodward (2005): *SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers*. ACM Transactions on Mathematical Software, Vol. 31, No. 3, pp. 363-396.
http://computation.llnl.gov/projects/sundials/toms_sundials.pdf
- G. Kron (1962): *Diakoptics – The piecewise Solution of Large-Scale Systems*. MacDonald & Co., London.
- S.E. Mattsson and G. Söderlind (1993): *Index Reduction in Differential-Algebraic Equations using Dummy Derivatives*. SIAM Journal of Scientific Computing. 14(3), pp. 677-692.
- S.E. Mattsson, H. Olsson and H. Elmqvist (2000): *Dynamic Selection of States in Dymola*. Modelica Workshop 2000, Lund, Sweden, pp. 61-67.
<https://www.modelica.org/events/workshop2000/proceedings/old/Mattsson.pdf>
- R.C. Melville, L. Trajkovic, S.-C. Fang, L.T. Watson (1993): *Artificial parameter homotopy methods for the DC operating point problem*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, pp. 861-877.
- M. Otter (1999): *Objektorientierte Modellierung Physikalischer Systeme, Teil 4: Transformationsalgorithmen*. at Automatisierungstechnik, 47, 3, pp. A9-A12.
- C.C. Pantelides (1988): *The Consistent Initialization of Differential-Algebraic Systems*. SIAM Journal on Scientific and Statistical Computing, 9(2):213-231.
- L. Petzold and P. Lötstedt (1986): *Numerical Solution of Nonlinear Differential Equations with Algebraic Constraints II: Practical Implications*. SIAM J. Sci. Stat. Comput. Vol. 7, No. 3, pp. 720-733.
- J.D. Pryce (2001). *A Simple Structural Analysis Method for DAEs*. BIT Numerical Mathematics, Vol. 41, Issue 2, pp. 364-394.
- J. Schuchart, V. Waurich, M. Flehmig, M. Walther, W.E. Nagel, I. Gubsch (2015): *Exploiting Repeated Structures and Vectorization in Modelica*. Proc. of the 11th Int. Modelica Conference, Versailles.
www.ep.liu.se/ecp/118/028/ecp15118265.pdf
- M. Sielemann (2012): *Device-Oriented Modeling and Simulation in Aircraft Energy Systems Design*. PhD-Dissertation. Technische Universität Hamburg-Harburg.
<http://www.dr.hut-verlag.de/9783843905046.html>
- R.L. Sigurðsson (2016): *Practical performance of incremental topological sorting and cycle detection algorithms*. Chalmers University of Technology. Master Thesis.
<http://publications.lib.chalmers.se/records/fulltext/248308/248308.pdf>
- S.S. Skiena (2008): *The Algorithm Design Manual*. Second edition. Springer.
- K. Stavaker: *Contributions to Simulation of Modelica Models on Data-Parallel Multi-Core Architectures*. PhD thesis. Linköping University. <http://liu.diva-portal.org/smash/get/diva2:806837/FULLTEXT01.pdf>
- P.R. Turner (1995): *A simplified fraction-free Integer Gauss Elimination Algorithm*. REPORT NO: NAWCADPAX-96-196-TR. Office of Naval Research.
<http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA313755>

Smart Processing of Function Calls to Achieve Efficient Simulation Code

Jan Hagemann Patrick Täuber Lennart Ochel Bernhard Bachmann

Department of Engineering and Mathematics, University of Applied Sciences Bielefeld, Germany,
{jan.hagemann,patrick_marcel.taeuber,lennart.ochel,bernhard.bachmann}@fh-bielefeld.de

Abstract

This paper introduces a new algorithm to increase the simulation performance of algebraic equation systems by encapsulating function calls. This avoids unnecessary evaluations of function calls and leads to positive structural effects, such as code motion. To enable the reader to reconstruct the algorithm, all four phases of the algorithm are described in detail and the complexity of them is analyzed. The overall complexity for practical models is $O(n)$, where n is the number of equations. It is shown that the algorithm significantly decreases the simulation time for a wide range of physical based models.

Keywords: function calls, backend, simulation

1 Introduction

Symbolic transformation and simplification methods (e.g. alias elimination, tearing methods, and index reduction) are essential within a Modelica compiler in order to achieve efficient simulation of the underlying differential-algebraic equation system. In this paper a proper handling of time-consuming function calls is discussed. It will be shown that the corresponding implementation of the module *WrapFunctionCalls* results in a tremendous decrease of simulation time for many models of specific libraries including the *MSL 3.2.1*. The expected performance increase has been already discussed in earlier publications (Jorissen et al., 2015).

The module *WrapFunctionCalls* traverses the equation system for occurring function calls. Those are stored in temporary variables, which are inserted at the according places in the equation system. This elimination is similar to the *Common Subexpression Elimination* (Jakubowski, 2002), hence the auxiliary variables for the function calls are called CSE-variables. However, the difference is that in the case of *WrapFunctionCalls* also single occurring function calls are stored.

There are some requirements on an efficient and reliable algorithm to encapsulate function calls, which are discussed in the next section before the algorithm is presented in Section 3. Such or similar symbolic transformations of function calls could not be found in literature or are not accessible in other simulation environments.

At the end of the paper a verification section proves the effects on models with a practical orientation and there-

fore the significance of the algorithm.

2 Requirements on the Algorithm

To achieve efficient simulation code by processing function calls first and foremost all function calls must be found regardless whether they occur in simple equations or if the function itself is an argument to another function. Consequently, nested function calls (e.g. $\exp(\cos(\text{time}))$) must be analyzed in detail to guarantee that function calls in arguments of other functions are replaced as well.

At this point it should be mentioned that only equations are traversed for functions. Algorithms are not handled by the module yet. Special Modelica constructs like impure functions are not traversed for nested calls as well and functions which are called conditionally (i.e. functions in bodies of when- and if-equations) are not encapsulated if they do not appear in the rest of the equation system.

All the found functions must be stored in CSE-variables. Considering the Modelica modelling language, it must be noted that for calls in equations of the form

$\text{variable} = \text{call}$ (e.g. $x = \cos(\text{time})$)

or

$\text{tuple} = \text{call}$ (e.g. $(a, b, _) = f(\text{time}, x)$)

no new CSE-variables should be introduced, because in those cases the variables on the left-hand side already encapsulate the corresponding functions and can therefore be used as CSE-variables.

Additionally, identical function calls, which do not necessarily have to look similar at the first sight, have to be recognized. The following example shows three equations with mathematically identical function calls in the second and third equation, due to the first equation:

$x = \cos(\text{time})$

$a = \exp(x)$

$y = \exp(\cos(\text{time}))$

The main objective of the algorithm is to reduce the simulation time by encapsulating function calls. Thus, duplicated calls only need to be evaluated once instead of every time they occur in the equation system. Especially for models with more complex functions, this will lead to vast

time savings during simulation.

Nevertheless, this is not the only way to reduce expensive function evaluations. As already mentioned above, one difference to the well known *Common Subexpression Elimination* methods is that function calls that only occur once in the whole model are encapsulated as well, because this can lead to a desirable code motion effect. Code motion has its origin in the optimization of standard compilers (Aho et al., 2008; Muchnick, 1997; Vogt, 2004). The code expressions, which appear in loops (e.g. for, while, ...) and are independent of the iteration variables and their dependencies can be moved out of the loop and be evaluated beforehand. In Modelica code motion can be seen as the extraction of subexpressions, i.e. function calls, out of algebraic loops. The following call illustrates the advantage of code motion in an acausal environment by encapsulating all function calls:

$$f_1(f_2(x), f_3(time))$$

Assuming this call to be part of an algebraic loop, where x may be the iteration variable, by encapsulating the calls in the following way, the calculation of function f_3 can be moved out of the loop because it is not dependent on variables, that are changing its value during the iteration process:

$$\begin{aligned} cse_1 &= f_1(cse_2, cse_3) \\ cse_2 &= f_2(x) \\ cse_3 &= f_3(time) \end{aligned}$$

Without the encapsulation f_3 would be evaluated in each iteration step, which could be expensive.

The verification of the expected effects mentioned above is done in Section 4.

3 Algorithm

The algorithm *WrapFunctionCalls* is divided into four parts:

- I. Analysis
- II. Dependencies
- III. Substitution
- IV. Creation of CSE-Equations

For a better understanding of the functionality the algorithm is introduced section by section with an example and the corresponding pseudo code. After each part the complexity is assessed.

Listing 1 shows the abstract example model that is used to illustrate the requirements on a reliable algorithm. Thus, not only the sine and cosine function calls must be stored in CSE-variables but also the more complex function foo , which has more than one return value, that must be handled properly. The function foo occurs in two equations. In the first one, only the second return value is accessed. In

the second equation, however, only the first output is used. Additionally, both calls of foo are identical because of the third equation. Lastly, it must be considered that the top-level functions in the first and third equation do not need to be stored in CSE-variables, because the left-hand side is already a simple variable.

```

1  model wfc
2    function foo
3      input Real x1;
4      input Real x2;
5      output Real y1;
6      output Real y2;
7      output Real y3;
8      [...]
9    end foo;
10   Real a, b, x;
11   equation
12     (, b,) = foo(x, sin(cos(time)));
13     a = sin(foo(x, x)) + 5.0;
14     x = sin(cos(time));
15   end wfc;

```

Listing 1. Abstract example model to introduce *WrapFunctionCalls*

The algorithm works with three data structures, which need to be created at the beginning. First, a hash table is needed to store and access the found function calls efficiently. Due to experiences with the *MSL* the size of the hashtable is chosen accordingly to avoid collisions. The keys of the hash table are the function calls and the values are integers representing indices of an expandable array, which is the second data structure. The expandable array contains entries consisting of a CSE-variable, a function call and an integer list, which represents the dependencies between function calls. As third data structure, another array is needed to store the manipulated equations and the new CSE-equations (Listing 2).

```

1  // Array of equations: eqArray
2  // Equation: eq
3  // List of variables of eqSystem: varList
4  // Expression: exp
5  // Subexpression: subExp
6  // CSE variable: cse_var
7  // Hash table: ht
8  // Expandable array: array+
9
10 create ht :=
11   <key=call(exp), value=index(int)>;
12
13 create array+ :=
14   [cse_var(exp), call(exp), dependencies(list<
15     int>)];
16
17 create eqArray := [];

```

Listing 2. Pre-phase of *WrapFunctionCalls* algorithm: Create data structures

3.1 Analysis

In the first part of the algorithm all equations of the equation system are traversed top-down to collect all function calls. A detected function call is stored within the

hash table, together with an index referring to the correspondent entry of the expandable array, which is basically an incremented integer for each call starting by 0. The corresponding entry of the expandable array contains the CSE-variable, the function call itself and an initially empty dependency list. In the first part of the algorithm all function calls are stored, even those which might turn out to be identical to other ones later on. To avoid the introduction of unnecessary CSE-variables an additional conditional branch is needed to detect equations of the form $var = call$ and $tuple = call$. In that case the already existing variables are stored as CSE-variables in the expandable array. If in the expandable array a CSE-variable has been entered for a call that turns out to be equal to an existing variable or tuple, the CSE-variable entry is overwritten with the already existing variable (cf. Listing 3).

```

17 index = 0;
18 // I. Analysis of eqSystem
19 for each eq in eqSystem
20   if eq as (cref=call) or (tuple=call) then
21     if call has not an entry in ht then
22       create ht entry := <call, index>;
23       create array+ entry := [cref/tuple, call, {}];
24       index = index + 1;
25     else
26       update/merge array+ entry = [Cref/Tuple, Call, {}];
27   end if;
28 end for;
29
30 for each call in eq [TopDown]
31   if call has not an entry in ht then
32     create ht entry := <call, index>;
33     create a cse_var for call;
34     create array+ entry := [cse_var, call, {}];
35     index = index + 1;
36   end if;
37 end for;
38 end for;
39
40 if ht is empty then
41   // algorithm terminates
42   return;
43 end if;

```

Listing 3. Phase I of *WrapFunctionCalls* algorithm: Analysis

Since the operations depend on the number of equations and each equation is addressed exactly one time, the complexity of the first part of the algorithm is $O(n)$, where n is the number of equations in the equation system.

If the analysis is performed on the example model from Listing 1 the hash table from Table 1 is generated. Because of the top-down traversal, the first call detected is $foo(x, \sin(\cos(time)))$. After that $\sin(\cos(time))$ and $\cos(time)$ are found. The same applies to the second equation. In the last equation a call is found, which already exists in the hash table, so no new hash table entry is created.

The detected function calls are also stored in the expandable array at the corresponding position determined

Table 1. Hash table after the first part of the algorithm (Analysis)

Hash Table (ht)	
Function Call	Integer
$foo(x, \sin(\cos(time)))$	1
$\sin(\cos(time))$	2
$\cos(time)$	3
$\sin(foo(x,x))$	4
$foo(x,x)$	5

by the index from the hash table. For the first call no additional CSE-variable is necessary because the call is equal to a tuple, so the tuple is stored as CSE-variable. For all other calls CSE-variables are introduced. Since the call in the last equation is equal to x and the call already exists in the hash table, the CSE-variable for $\sin(\cos(time))$ is overwritten with x again. This leads to the expandable array from Table 2, where the integer dependency lists are still empty.

Table 2. Expandable array after the first part of the algorithm (Analysis)

Array+		
CSE-Variable	Function Call	Integer List
$(-, b, -)$	$foo(x, \sin(\cos(time)))$	{ }
x	$\sin(\cos(time))$	{ }
cse_2	$\cos(time)$	{ }
cse_3	$\sin(foo(x,x))$	{ }
$(cse_4, -, -)$	$foo(x,x)$	{ }

3.2 Dependencies

At the beginning of the second phase all the occurring function calls are already located in the expandable array and in the hash table. In the second part the analysis continues by considering each array entry to find the dependencies between the different function calls. To do so, the function calls must be analyzed successively whether there are other function calls in their arguments. If another function call is detected within an argument, the index (value) of that call has to be determined from the hash table. Now, at the position of that index in the expandable array, the dependency list is appended by the index of the outer function call (Listing 4). So, at the end the dependency list for each call contains the indices of the calls in which the current call occurs. An empty dependency list after the second phase signifies a function call independent of all other function calls.

```

44 // II. Dependencies
45 for each entry of array+
46   traverse arguments of call
47   add index of call to each argument entry (in list <int>);
48 end for;

```

Listing 4. Phase II of *WrapFunctionCalls* algorithm: Dependencies

If no nested function calls occur in the equation system, the complexity of the second phase is $O(k)$, where k is the number of supposedly different function calls, i.e. the number of array elements because each array entry is addressed only once.

If there are nested function calls, the complexity is smaller than or equal to $O(k + \sum_{i=1}^a i) = O(k + \frac{a^2+a}{2}) = O(k + a^2)$, where a is the number of function calls in the arguments of other function calls in the model, because each call has to be addressed again for each occurrence in the arguments of another call. The worst case of $k + \frac{a^2+a}{2}$ operations pertains if there is exactly one nested function call in the equation system because this would lead, for instance, to the following entries in the array, where a is $k - 1$:

$$\begin{aligned} &f_1(f_2(\dots(f_{k-1}(f_k(x))))\dots)) \\ &f_2(\dots(f_{k-1}(f_k(x))))\dots) \\ &\vdots \\ &f_{k-1}(f_k(x)) \\ &f_k(x) \end{aligned}$$

Since $\sum_{i=1}^{a_1} i + \sum_{i=1}^{a_2} i < \sum_{i=1}^{a_1+a_2} i$ less operations have to be performed if all the inner calls are not in only one call but distributed in different calls. In practical and realistic models the worst case would not occur for a big k , because a high level of nesting in functions is not used. An analysis of the models of the *Modelica Standard Library 3.2.1* shows that the average maximum dependency list length is about 0.5, where the fluid and media models have the largest nesting. Thus, with respect to the number of function calls the calls in arguments of other calls are negligible. This assumptions lead to a nearly linear complexity of $O(k)$.

Table 3. Expandable array after the second part of the algorithm (Dependencies)

Array+		
CSE-Variable	Function Call	Integer List
$(-, b, -)$	$foo(x, \sin(\cos(time)))$	$\{ \}$
x	$\sin(\cos(time))$	$\{1\}$
cse_2	$\cos(time)$	$\{1, 2\}$
cse_3	$\sin(foo(x, x))$	$\{ \}$
$(cse_4, -, -)$	$foo(x, x)$	$\{4\}$

Applying the second phase on the example leads to the dependency list from Table 3. Since $\sin(\cos(time))$ occurs in the first entry index 1 is entered in its dependency list. The cosine occurs in the first and second entry, so its

dependency list is $\{1, 2\}$, while $foo(x, x)$ appears in entry four and therefore has the dependency list entry 4. For the other function calls the dependency lists remain empty since they are the top-level functions and do not appear in any other function arguments.

3.3 Substitution

In the third part of the algorithm the equations in the equation system are traversed again but this time bottom-up. The encountered function calls are replaced with the CSE-variables. If the corresponding dependency list has entries, the function calls in the expandable array at which the indices point to are replaced as well and the dependency list is deleted afterwards. Additionally, a new entry in the hash table is created, which contains the new function call and its original index in the array, to guarantee that occurring calls of that kind can be found as well. If this entry is already existent in the hash table, then the CSE-variables have to be merged in the expandable array. After the processing of each equation the substituted equation is added to the equation array if it is not redundant. If the substituted equation is redundant, such as $x = x$, it is discarded (cf. Listing 5).

```

49 // III. Substitution
50 for each eq in eqSystem
51     for each call in eq [BottomUp]
52         get cse_var from array+ and substitute
           call;
53         substitute the arguments with the cse_var
           in the calls referenced by the
           dependency list;
54         delete the dependency list of the current
           cse_var;
55         if not already in ht then
56             add the new call to ht with the original
               index;
57         else
58             merge cse variables in the expandable
               array
59         end if
60         if the substituted eq is not redundant
           then
61             add eq to eqArray;
62         end if;
63     end for;
64 end for;
    
```

Listing 5. Phase III of *WrapFunctionCalls* algorithm: Substitution

Since the number of operations in the equation system depends on the number of equations, and the number of additional operations in the expandable array depends on the total number of all dependency list entries, which is negligible relative to the number of equations in models with a practical orientation, the complexity for the third phase is nearly linear regarding n ($O(n)$).

If the third phase is performed on the example model, the equations are traversed again. This time $\cos(time)$ is the first call that is detected because now the algorithm works bottom-up. So $\cos(time)$ is substituted by its CSE-variable cse_2 . Additionally, the calls in the array the dependency list of $\cos(time)$ points to are substituted as well,

so the first call is $foo(x, \sin(cse_2))$ and the second call is $\sin(cse_2)$ now. The dependency list is deleted because this substitution only needs to be performed once. Finally, new hash table entries are created for $foo(x, \sin(cse_2))$ and $\sin(cse_2)$ with the values 1 and 2, respectively.

The next call that is found is $\sin(cse_2)$, which is replaced by x in the equation and in the first array entry and its dependency list is deleted. The first array entry now contains $foo(x, x)$. Since this call already exists in the hash table, this call is now equal to another call and the introduced CSE-variables have to be merged. Therefore, the CSE-variables in the fifth array entry are changed to $(cse_4, b, _)$. The function call in the first array entry is also overwritten with $(cse_4, b, _)$.

After that, the call $foo(x, x)$ is found in the substituted equation. It is replaced by $(cse_4, b, _)$ in the equation and by cse_4 in the forth array entry because there the first output is needed. The call $\sin(cse_4)$ is added to the hash table with value 4 and the dependency list is deleted. At the end of the processing of the first equation the substituted equation is $(_, b, _) = (cse_4, b, _)$. Since this equation is redundant it is not added to the equation array.

While processing the next equations, no more manipulations on the hash table and on the expandable array have to be performed. At the end of the substitution of the second equation it reads $a = cse_3 + 5.0$. It is added to the equation array. The third equation reads $x = x$ and is not added to the equation list because it is redundant.

The hash table and the expandable array after the third phase are illustrated in Table 4 and Table 5, respectively.

Table 4. Hash table after the third part of the algorithm (Substitution)

Hash Table (ht)	
Function Call	Integer
$foo(x, \sin(\cos(time)))$	1
$\sin(\cos(time))$	2
$\cos(time)$	3
$\sin(foo(x, x))$	4
$foo(x, x)$	5
$foo(x, \sin(cse_2))$	1
$\sin(cse_2)$	2
$\sin(cse_4)$	4

Table 5. Expandable array after the third part of the algorithm (Substitution)

Array+		
CSE-Variable	Function Call	Integer List
$(_, b, _)$	$(cse_4, b, _)$	{ }
x	$\sin(cse_2)$	{ }
cse_2	$\cos(time)$	{ }
cse_3	$\sin(cse_4)$	{ }
$(cse_4, b, _)$	$foo(x, x)$	{ }

3.4 Create CSE-Equations

In the fourth and last part of the algorithm each entry of the expandable array is considered and an equation with the CSE-variable and the corresponding expression is generated. If that equation is not redundant it is added to the equation array and the CSE-variable is added to the variable list (Listing 6).

```

65 // IV. Create cse equations
66 for each entry of array+
67     generate eq (cse_var=call);
68     if eq is not redundant then
69         add eq to eqArray;
70         add cse_var to varList;
71     end if;
72 end for;
73
74 add eqArray and varList to eqSystem;

```

Listing 6. Phase IV of *WrapFunctionCalls* algorithm: Create CSE-equations

Since each array entry is addressed one time, the complexity of this phase is $O(k)$.

In the example, all equations derived from the expandable array, except the one from the first entry, are added to the equation array. So the generated CSE-equations in the example are the following:

$$\begin{aligned}
 x &= \sin(cse_2) \\
 cse_2 &= \cos(time) \\
 cse_3 &= \sin(cse_4) \\
 (cse_4, b, _) &= foo(x, x)
 \end{aligned}$$

The processed example model at the end of the *WrapFunctionCalls* algorithm is shown in Listing 7.

```

1 model wfc_result
2   function foo
3     input Real x1,
4     input Real x2;
5     output Real y1;
6     output Real y2;
7     output Real y3;
8     [...]
9   end foo;
10  Real a, b, x, cse2, cse3, cse4;
11  equation
12    a = cse3 + 5.0;
13    cse3 = sin(cse4);
14    (cse4, b, \_) = foo(x, x);
15    x = sin(cse2);
16    cse2 = cos(time);
17  end wfc_result;

```

Listing 7. Example model after processing with *WrapFunctionCalls*

To estimate the costs of the complete algorithm the complexities of the single phases have to be considered together, so it adds up to $O(n + k + n + k)$. Since in practical models the number of different function calls is negligible relative to the number of equations, the complexity of the *WrapFunctionCalls* algorithm can be estimated with $O(n)$. The actual complexity is assessed in the next section.

4 Verification

4.1 Complexity

The complexity of the optimization module *WrapFunctionCalls* depends on the number of given function calls and equations as described above. The estimation of a complexity of $O(n)$ is confirmed by a test of the *SteamPipe* models of the *ScalableTestSuite* library (Casella, 2015a). The *ScalableTestSuite* library is developed by Francesco Casella and Kaan Sezginer, Politecnico di Milano (Casella and Sezginer, 2016). This includes different models among others of the electrical engineering, mechanical science and thermodynamics. The models are characterized by the fact that they are scalable by parameters, so as to test the ability of the Modelica tools in terms of an efficient compilation and simulation time as the size increases. To do so the execution time of the algorithm for these models was measured during compilation for different parameters N which is a scalable parameter depending linearly on the number of equations n .

The results of the tests are shown in Table 6 and Figure 1. These tests confirm that the execution time is linear proportionate to the parameter N and thus also linear to the number of equations n . Due to experiences with a variety of different other libraries it is assumed that this $O(n)$ behaviour is not only valid for a certain structure of equations.

Table 6. Execution time of *WrapFunctionCalls* for *SteamPipe* models

Model	N	n	Calls	Time [s]
SteamPipe_N_10	10	262	40	0.0074
SteamPipe_N_20	20	522	80	0.0221
SteamPipe_N_40	40	1042	160	0.0298
SteamPipe_N_80	80	2082	320	0.0607
SteamPipe_N_160	160	4162	640	0.1233
SteamPipe_N_320	320	8322	1280	0.253
SteamPipe_N_640	640	16642	2560	0.552

4.2 Code Motion

Below it is shown that the expected effect of code motion actually happens in practice, i.e. function calls are moved out of algebraic loops, so that these are calculated beforehand and not in each iteration step. To do so the simulation time of different libraries are compared with the optimization module *WrapFunctionCalls* deactivated and activated, respectively. The results of the test are shown in the next section. The analysis shows that among others in the model *WaterIF97* of the *MSL 3.2.1* code motion occurs in practical relevant examples. This model has a nonlinear algebraic loop (Table 7), whose residual equation contains a function call that can be stored in a CSE-variable (*cse4*) and thus can be moved out of the loop (Table 8).

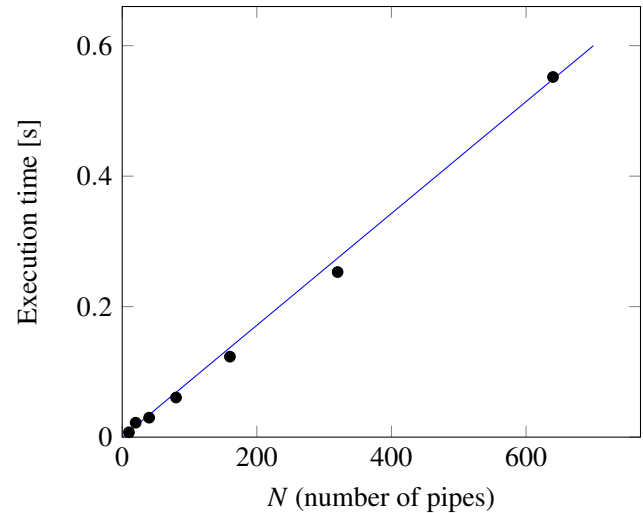


Figure 1. Graphic representation of the execution time of *WrapFunctionCalls* for *SteamPipe* models

Table 7. Code Motion example without *WrapFunctionCalls*

```
Nonlinear loop.
Iteration variable: DER.medium.p
-----
DER.medium.h = DER.medium.u -
(medium.p * DER.medium.d -
der(medium.p) * medium.d)/(medium.d ^ 2.0)
-----
Residual equation:
Modelica.Media.Water.IF97_Uutilities.rho_ph_der
(medium.p, medium.h, Modelica.Media.Water.
IF97_Uutilities.waterBaseProp_ph(medium.p,
medium.h, medium.phase, 0), DER.medium.p,
DER.medium.h) - DER.medium.d = 0.0
```

Table 8. Code Motion example with *WrapFunctionCalls*

```
cse4 := Modelica.Media.Water.IF97_Uutilities.
waterBaseProp_ph(medium.p, medium.h,
medium.phase, 0);
-----
Nonlinear loop.
Iteration variable: DER.medium.p
-----
DER.medium.h = DER.medium.u -
(medium.p * DER.medium.d -
der(medium.p) * medium.d)/(medium.d ^ 2.0)
-----
Residual equation:
Modelica.Media.Water.IF97_Uutilities.rho_ph_der
(medium.p, medium.h, cse4, DER.medium.p,
DER.medium.h) - DER.medium.d = 0.0
```

4.3 Improvement in Simulation Time

This section shows the improvement in simulation time of many models in specific libraries. For this purpose, Table 9 presents some extracts of single libraries with improvements of over 40 percent. The reasons for the significant upgrade are especially the described effects of code motion and the encapsulation of expensive function calls.

The analysis shows that the *ThermoPower* library achieves the best results with an improvement of over 90 percent, followed by the *SteamPipe* models of the *ScalableTestSuite* with more than 80%. The *MSL 3.2.1* also demonstrates enhancements far above 40 percent. For example, the already mentioned model *WaterIF97* gains a speed-up of 53%.

These absolutely positive results are confirmed by several OpenModelica power users, which also demonstrate a magnificent improvement in runtime performance of their applications by encapsulating function calls. (Franke et al., 2015; Franke, 2016; Casella, 2014, 2015b).

Table 9. Improvement in simulation time shown on an extract of expressive libraries

Model (Library)	– WFC [s]	+ WFC [s]	Imp. [%]
MSL 3.2.1			
DrumBoiler	4.52	1.73	62
MomentumBalanceFittings	4.01	1.63	60
HeatExchangerSimulation	36.31	12.89	65
InverseParameterization	78.81	41.03	48
NonCircularPipes	8.71	3.91	55
R134a1	16.77	5.19	69
DryAir2	21.85	5.17	76
TestTwoPhaseStates	2.38	0.93	61
WaterIF97	1.90	0.90	53
PlanarMechanics			
KinematicLoop_			
DynamicStateSelection	19.26	6.12	68
PowerSystems			
PowerWorld	7.59	1.73	77
ThermoPower			
CISESim180504	284.6	20.24	93
TestMixerSlowFastSteam	8.80	0.69	92
TestWaterFlow1DFV_F	45.73	15.82	65
ScalableTestSuite			
SteamPipe_N_10	23.84	4.35	81
SteamPipe_N_20	45.56	8.60	81
SteamPipe_N_40	98.24	17.61	82
SteamPipe_N_80	197.93	35.39	82
SteamPipe_N_160	412.87	73.8	82

5 Conclusions

This paper shows an optimization algorithm which encapsulates function calls of the given equation system in temporary variables. Above all, the idea is that expensive function calls should be evaluated only once. Furthermore, the effect of code motion of standard compilers motivates to find this effect in Modelica models. The extraction of function calls out of algebraic loops promises a huge improvement in simulation time since the iteration does not have to be performed over a function call in each step. Thus, also single occurring function calls are stored

in contrast to the original *Common Subexpression Elimination*. Other peculiarities, e.g. nested function calls, functions with multiple outputs or simple equations with a special form such as *variable = call*, are considered as well.

Additionally, the algorithm works very efficient, which is confirmed by a complexity assessment of $O(n)$. A test with the help of the *ScalableTestSuite* approves the estimation. The desired effect of code motion also can be found in practical models and is depicted by an example of the *MSL 3.2.1*.

Moreover, single extracts of different libraries are listed, which show vast improvements in simulation time due to code motion and the encapsulation of function calls.

Acknowledgments

This work is supported by the Ministry of Innovation, Science and Research of the German State of North Rhine-Westphalia (MIWF NRW) as part of the research cooperation “MoRitS - Model-based Realization of intelligent Systems in Nano- and Biotechnologies” (grant no. 321 - 8.03.04.03 - 2012/02).

References

- Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers - Principles, Techniques, and Tools*; 2. Edition. Pearson, 2008. ISBN 978-0321486813.
- Francesco Casella. Thermopower library simulation. OpenModelica Workshop, Linköping, Sweden, Februar 3rd, 2014, 2014. URL <http://www.modprod.liu.se/openmodelica2014-talks/1.544931/OpenModelica2014-talk03-Francesco-Casella-ThermoPowerLibrarySimulation.pdf>.
- Francesco Casella. Simulation of large-scale models in modelica: State of the art and future perspectives. In *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*, number 118, pages 459–468. Linköping University Electronic Press, Linköpings universitet, 2015a.
- Francesco Casella. Modelling of energy systems with OpenModelica. OpenModelica Workshop, Linköping, Sweden, Februar 2nd, 2015, 2015b. URL <http://www.modprod.liu.se/openmodelica-2015/1.620217/OpenModelica2015-talk03-Francesco-Casella.pdf>.
- Francesco Casella and Kaan Sezginer. *ScalableTestSuite homepage*. <https://github.com/casella/ScalableTestSuite>, December 2016.
- Rüdiger Franke. Embedded optimizing control using the OpenModelica C++ runtime. OpenModelica Workshop, Linköping, Sweden, Februar 1st, 2016, 2016. URL <http://www.modprod.liu.se/filarkiv/1.672872/OpenModelica2016-talk08-RudigerFranke-EmbeddedOptimizingControl.pdf>.

Rüdiger Franke, Marcus Walther, Niklas Worschech, Willi Braun, and Bernhard Bachmann. Model-based control with FMI and a C++ runtime for Modelica. In *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*, pages 339–347, 2015.

Jacek Jakubowski. Architekturunabhängige Quellcodeoptimierung durch Mustererkennung. Dissertation, Universität Dortmund, 2002.

Filip Jorissen, Michael Wetter, and Lieve Helsen. Simulation speed analysis and improvements of modelica models for building energy simulation. In *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*, pages 59–69, 2015.

Steven S. Muchnick. *Advanced Compiler Design and Implementation*. Morgan Kaufmann Publishers, San Francisco, 1997. ISBN 978-1-55860-320-2.

Michael Vogt. Plattformunabhängige Eliminierung gemeinsamer Teilausdrücke auf Quellcode-Ebene. Dissertation, Universität Dortmund, 2004.

Integrative Physiology in Modelica

Jiří Kofránek¹, Tomáš Kulhánek¹, Marek Mateják¹, Filip Ježek¹, Jan Šilar¹

¹Department of Pathophysiology, 1st Faculty of Medicine, Charles University
{kofranek, tmkulhanek, matejak.marek, jezekf, janeksilar}@gmail.com

Abstract

The integrative model of human physiology connects individual physiological subsystems into a single unit. They are enormous (contain thousands of variables) and represent a formalized description of interconnected physiological regulations. The issue of formalization of physiological systems became part of a series of international projects (e.g. the worldwide program “PHYSIOME”, or the European program “VIRTUAL PHYSIOLOGICAL HUMAN”). The development of large-scale models of human physiology was facilitated by a new generation (i.e. equation-based) of simulation environments, especially by the Modelica language. These models can be used to explain the causal relations of the pathogenesis of many diseases. They can be applied in the evaluation of clinical trials and they can also be used in the core of sophisticated medical simulators.

Keywords: Simulation, Physiology, Integrative models

1 Introduction

In 1972, Arthur Guyton published an article (A. C. Guyton, Coleman, and Granger 1972) in the journal *Annual Review of Physiology*, whose form quite surpassed the usual forms of physiological articles of those times at the very first sight. An extensive diagram, slightly resembling a complex electronic circuit, enclosed as an attachment, was used as introduction, showing interconnection of essential subsystems, that have an effect on circulation, by means of special symbols expressing mathematical operations (addition, subtraction, multiplication, division, integration and functional dependencies). The connections of the elements thus represented mathematical equations (Fig. 1) and the entire scheme provided a visual graphic representation of a set of equations describing the then available ideas of bloodstream regulation including essential connections with other physiological systems of the human organism (Fig. 2). This graphic scheme of Guyton was thus one of the first mathematical descriptions of mutually connected physiological systems in the human organism and it initiated development of physiological research, today sometimes described as integrative physiology.

2 Integrative physiology modelling

Modelling is closely related to formalization – i.e. replacement of verbal description of physiological systems with the exact language of mathematics in the form of a mathematical model. Given the complexity of

biological systems, the process of formalization is delayed in biological and medical sciences compared to physics, chemistry or technical sciences.

While in physics, the formalization process began at some point during the 17th century, in medical and biological sciences it occurred only with the onset of cybernetics and computer science. Computer models developed based on mathematical description of biological reality are used as the methodological tool in latter sciences.

In physiology, formalized descriptions have been used since the 1940s; at that time, McCulloch and Pitts

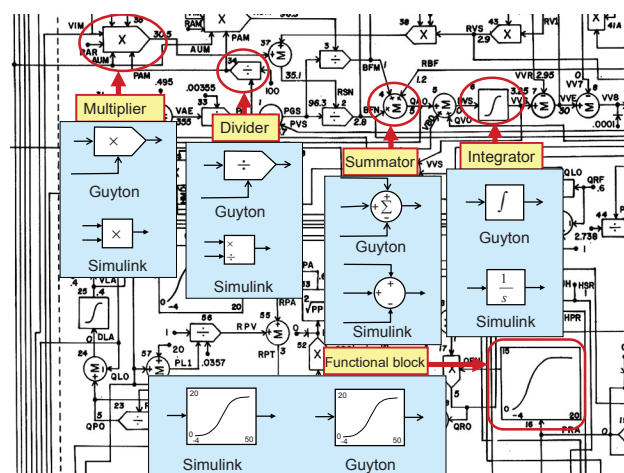


Figure 1. Individual elements in the scheme of Guyton's model display mathematical operations whose connections represent graphically expressed mathematical equations. Blocks in the original Guyton notation (1972), and the same blocks in Simulink (1990).

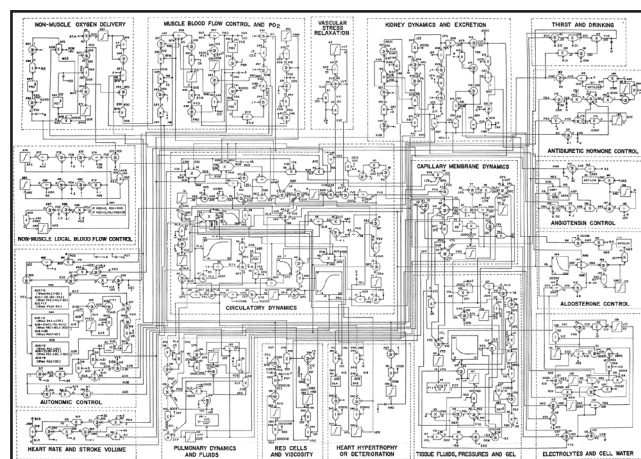


Figure 2. Interconnected physiological subsystems in the Guyton model (Guyton et al., 1972).

(McCulloch and Pitts 1943; Pitts and McCulloch 1947), to mention some examples, designed a simplified neuron model and Sheppard (Sheppard 1948) introduced his compartment approach, that found quick application in pharmacokinetics. In the 1950s, Hodgkin and Huxley (Hodgkin and Huxley 1952) published their groundbreaking model of the neuronal excitation membrane. In the 1960s, the development of computers supported another wave of publications related to formalized description of physiological reality, for example, Milhorn's monograph on the use of Automated Control Theory in physiological systems (Milhorn 1966) or the pioneering work of Grodins regarding the modelling of respiration (Grodins, Buell, and Bart 1967). At the end of the 1960s and at the beginning of the 1970s of the past century, multicompartment systems found broad application in biology and medicine (Atkins 1969), and computer-based methods were developed to determine the parameters of biological systems (Potůček et al. 1977).

2.1 Integrative physiology

The above mentioned model of Guyton and his collaborators from 1972 (A. C. Guyton, Coleman, and Granger 1972) was one of the first extensive mathematical descriptions of physiological functions of mutually connected subsystems in the human organism, which established an area of physiological research, today described as “integrative physiology” (Coleman and Summers 1997). Similarly as theoretical physics seeks to describe physical reality and explain the results of experimental research using formal means, “integrative physiology” also seeks to create a formalized description of mutual connections among physiological regulation systems and to explain their role in the development of various diseases based on experimental results. From this point of view, Guyton's model was a certain milestone aimed at capturing the dynamics of relationships among the regulation systems of the cardiovascular circulation, kidneys, respiration, volume and ion composition of body fluids using a graphically depicted network while applying a system view of physiological regulation systems.

Guyton's graphic notation of the formalized description of physiological relationships, inspired by then commonly used analogue computers, provides a highly visual representation of mathematical correlations – blocks at the network nodes represent graphic symbols for individual mathematical operations, while the connecting lines represent individual variables. Guyton's graphic notation was soon adopted also by other authors – for example, Ikeda et al. (Ikeda et al. 1979) in Japan or the Amosov research group in Kiev (Amosov et al. 1977).

With his research and teaching, Guyton changed physiology from a science of verbal descriptions to one of quantitative analysis. He brought mathematics and physics into the discipline. He was a pioneer in the use

of computers to study of body function and has taught scientists all over the world computer simulation.

Guyton's model also served as an inspiration and a resource for the development of complex models of physiological regulation systems used to explain causal chains of reactions in the human organism to various stimuli and to understand the development of various pathological conditions. Among others, the modified Guyton model became part of the foundations for an extensive model of physiological functions in the NASA program “Digital Astronauts” (White and McPhee 2007).

Currently, the international project **PHYSIOME** (<http://www.physiome.org>) is focused on the formalized description of physiological systems; this project is the successor of the “GENOME” project, which resulted in a detailed description of the human genome. The aim of the “PHYSIOME” project is to develop a formalized description of physiological functions. Computer models are used as the methodical tool (Bassingthwaite 2000; P. Hunter, Robbins, and Noble 2002; P. J. Hunter et al. 2006; Peter J. Hunter, Crampin, and Nielsen 2008; Omholt and Hunter 2016). “VIRTUAL PHYSIOLOGICAL HUMAN” (<http://www.vph-institute.org>) is a European initiative in this field, focused, among other things, on applications of the formalized approach to human physiology in clinical medicine (P. Hunter 2016). Descendants of Guyton's original computer model of the cardiovascular system are some of the resources for the development of present complex models of physiological regulation systems in

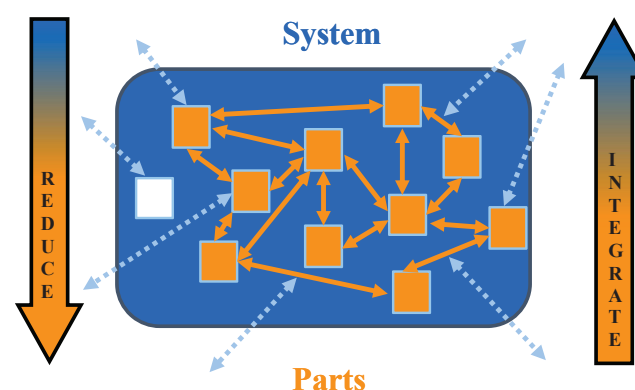


Figure 3. A system as an entity that maintains its existence through mutual interactions of its individual parts (system elements). In the system analysis, the system should include only those elements that enter in mutual interactions with each other (orange squares), while elements that may be structurally and functionally similar to other system elements but interact only with the surroundings of the system (an empty square) should be excluded from the system. The surroundings of the system interact with individual system elements or modulate their mutual bonds (dashed arrows). When studying a system (the transfer between individual hierarchical levels), reductionist and integration tools and methods need to be combined. Adapted from (Peter Kohl and Noble 2009).

this European project (Thomas et al. 2008).

Besides integrative models of human physiology, integrative models of laboratory animals have also been developed recently. The project “VIRTUAL RAT” is aimed at designing a complex model of a laboratory rat that can be validated more easily by comparing to experimental data from laboratory animals (<http://www.virtualrat.org>).

2.2 The human organism as a hierarchical system

The task of exploring a living organism as a system unit poses a key problem of how (with respect to the explored problem) the system structure should be defined in the biological object, what parts should be understood as system elements, how to define the subsystems, etc.

According to the definition of Bertalanffy (Von Bertalanffy 1973), the *system* is an *entity that maintains its existence through mutual interactions of its individual parts (system elements)*. Therefore in the system analysis, a system defined on any given real object should include only those parts that primarily interact with each other (see Fig. 3).

System research must include (Peter Kohl and Noble 2009):

- Identification of individual parts of the entity;
- Detailed characteristics of mutually interacting parts of the entity to be included among the system elements (while parts interacting only with the surroundings of the system will not be included);
- Exploration and subsequent description of mutual interactions among individual elements;
- Exploration and subsequent description of interactions with the surroundings of the system (the system surroundings have direct or indirect effects on the system elements, by influencing mutual interactions among the system elements);
- Combinations of reductionist and integration tools and methods in exploration of any system entity on various hierarchical levels.

A system as a set of elements and integration bonds is thus defined on a real object. Based on more detailed exploration of the system entity, an ever a more complex system can be defined, which may be composed of a number of mutually integrating subsystems. However, this is not a purely mechanical process. When passing to a more detailed level, a number of included functions and bonds of the higher hierarchical level must be reduced, and on the contrary, when passing to a higher level a number of elements and bonds must be integrated (Fig. 4). Every model is a simplified notion of representation of reality on various hierarchical levels.

The approach of classical molecular biology goes “from below upwards”. It starts from “bottom elements” of the organisms – genes and proteins. Molecular biology models provide formalized descriptions of interactions of gene and protein cell structures that can be used to understand their functions.

The approach of classical physiology is the opposite – “from above downwards”, somewhat resembling reverse engineering. First, the system is studied on higher levels and subsequently, the process goes down in an effort to find inverse solutions. The system behaviour is used to try to derive the functions of its individual parts.

Integrative models start “from the middle”. They combine both approaches – downward to the cellular and molecular level and upward to integration and deriving of functionalities of the human organism as a whole (P. Kohl et al. 2010).

The circulatory system model of the Japanese authors Shim et al. (Shim et al. 2006) can be given as an example of *connecting models of various hierarchical levels*; these authors combined a simple model of cardiovascular haemodynamics of the vascular system with the model of ventricles of the heart (Fig. 5), acting as a heart pump. The ventricles were modelled, in a simplified manner, as spherical elastic compartments with variable tension of their wall. This tension was obtained from the model of actomyosin cross-bridges of the myocyte (formation of these bridges determines the strength of the stretched

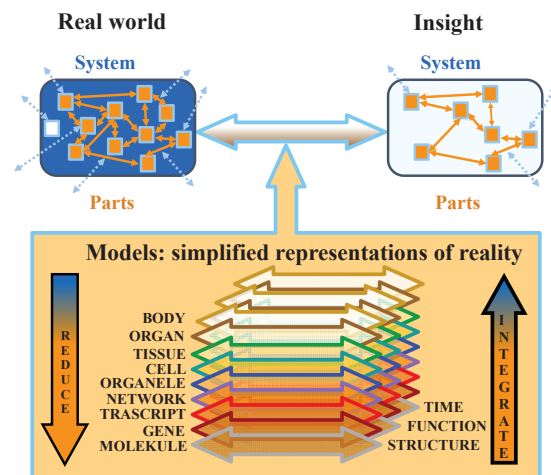


Figure 4. Our understanding of the “real world system” usually provides only a simplified representation of reality. The gradual evolution of our understanding of the real biological world is based on the use and analysis of experimental and theoretical (mathematical) models on all hierarchical levels. The result is reflected in ever more detailed knowledge of the structure of functional relationships and their changes in time, gradually integrated in higher hierarchical levels. System biology provides a framework for targeted interconnection of various aspects of applications of models in biomedical research and development. Adapted from (Peter Kohl and Noble 2009).

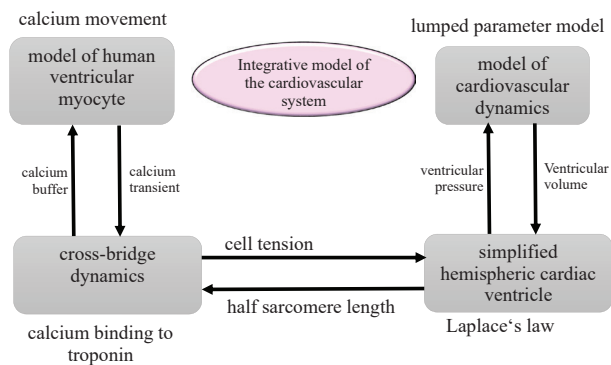


Figure 5. The integrative model of the cardiovascular system as a combination of models on various hierarchical levels according to (Shim et al. 2006).

muscle cell). Their formation is affected by calcium crossing the cell membrane and the sarcoplasmic reticulum membrane where calcium is cyclically released and uptaken. In the sarcoplasmic bridge model, calcium binds to troponin. This binding of calcium causes actomyosin cross-bridges to form, resulting in subsequent tension of the muscle cell. The model of actomyosin cross-bridges was therefore connected to the model of calcium passing between the cytoplasm and sarcoplasmic reticulum.

The cardiovascular haemodynamics model is based on a considerable simplification of reality; it is designed as an RLC model with lumped parameters. Ventricular pressure is found at the input; it is generated by the cardiac ventricular model depicted as a sphere with the wall of variable rigidity. The tension value of the myocyte muscle fibre, being the output of the actomyosin cross-bridges (validated according to experimental results), is the starting value for calculating the ventricular wall rigidity. The actomyosin cross-bridge model depends on the output of the myocyte calcium dynamics model (validated according to experimental results). The connection of models of different hierarchical levels integrates important outputs of lower hierarchical level models (for example, behaviour of the myocardium as a whole is derived from the actomyosin cross-bridge model of one cell). Although the models of every hierarchical level represent considerable simplification of reality, the model outputs indicate, for example, the effect of calcium levels in the muscle cell cytoplasm on pressure-volume curves of the left ventricle, thus illustrating, e.g., the clinically verified effect of pharmaceuticals acting on the potassium pump in myocytes.

2.3 Hummod

Today, the most extensive model of integrated physiological systems of human physiology is apparently the *HumMod* model created based on international cooperation of a group of collaborators and disciples of A. Guyton, at the Mississippi University Medical Center, USA (R. L. Hester et al. 2011; R. Hester et al. 2011; Lerant

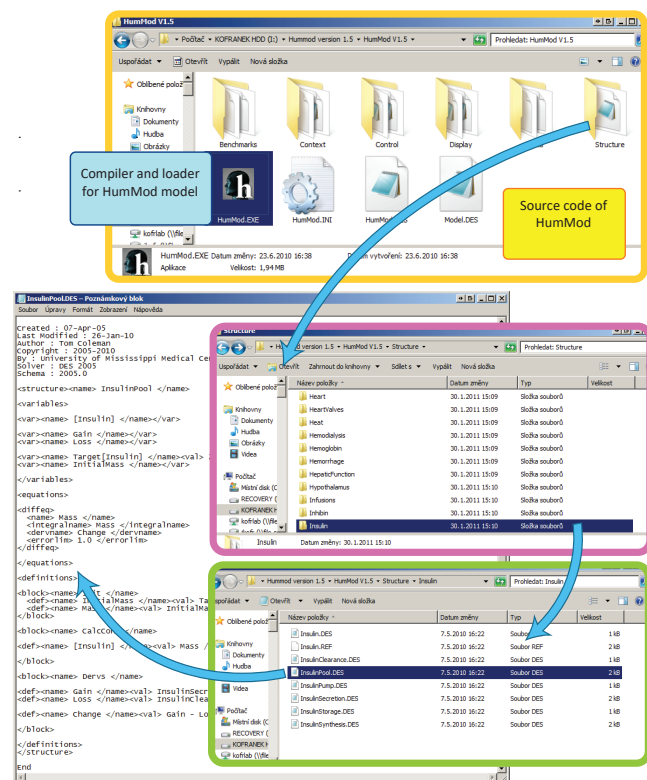


Figure 6. The HumMod simulator has been distributed with a compiler, loader and the source code written in thousands of XML files.

et al. 2015; W. A. Pruetz, Clemmer, and Hester 2016).

The authors do not seek to keep its structure a secret; the source text of the model (containing more than 5,000 variables) can be downloaded from the website of the model, <http://hummod.org>. The source text was written in the special markup language XML. The entire mathematical model is offered as “open source”; users can use the website to download and install on their computer the source code as well as the compiler and run the model on their own machine (Figs. 6 and 7). Users can thus adapt and modify the model. The problem is that the XML source texts of the entire model are written in several thousands of files located in hundreds of folders, and it is very difficult to orient oneself in the mathematical relationships by studying more than a thousand of mutually connected XML files.

In the development of models in the field of integrative physiology, many research teams actually preferred to use older models of complex physiological regulations – for example, the old models of Guyton (A. C. Guyton, Coleman, and Granger 1972; A. C. Guyton et al. 1986; A. C. Guyton, Hall, and Montani 1988; J. P. Montani, Mizelle, et al. 1989; J. P. Montani, Adair, et al. 1989; J.-P. Montani and Van Vliet 2009) or the old model of Ikeda (Ikeda et al. 1979). For example, this path was taken by the international research team of the SAPHIR (System Approach for Physiological Integration of Renal, cardiac and respiratory control) project in 2008 after deciding that

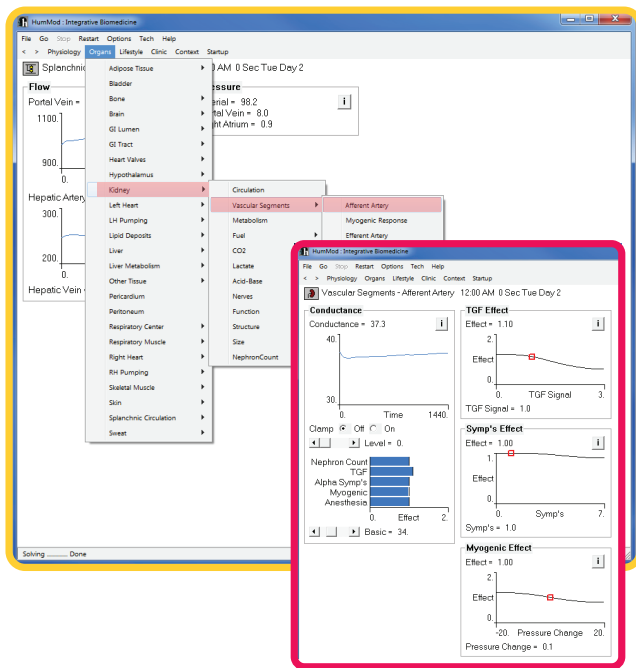


Figure 7. The user can compile and run the HumMod model. Using a widely branched menu, hundreds of variables can be monitored during simulation experiments.

the open source mathematical model of integrative human physiology containing over 3000 variables from Guyton's laboratory (R. L. Hester, Coleman, and Summers 2008) at that time seemed very poorly readable and difficult to understand to the project participants. Therefore two legacy integrated models have served as a starting point for integrative model development (Thomas et al. 2008), namely the classic model of Guyton et al. (1972), which focused on blood pressure regulation, and the model of Ikeda et al. (1979), based on Guyton's models but extended to focus on the overall regulation of body fluid. The model of Ikeda was recently reimplemented in the modern simulation environment (Fontecave-Jallon and Thomas 2015). Similarly, in 2011 Mangourova et al. (Mangourova, Ringwood, and Van Vliet 2011) implemented in Simulink an older model of Guyton of 1992 written in C instead of the then most recent (but difficult to understand for them) version of the large integrative model - **QHP** (**Quantitative Human Physiology** - the predecessor of the **HumMod** model) from Guyton's laboratory.

It is apparent that *comprehensibility of descriptions of complex integrative models is one of their limiting factors for their acceptance by the scientific community*. If the creators are the only ones to understand their models, their potential of factual communication with other scientists is thus hindered. This also limits the possibilities of designing integrative models within a broader scientific community. This is an important reason why the development of methodologies has been gaining importance; such methodologies would make the descriptions of structures of complex hierarchical models

clearer in a way so that a wider spectrum of users can understand them.

Special viewers have been created for better understanding of the **HumMod** model that enable the user to go through individual relationships within the model (Xu et al. 2011; Wu et al. 2013; Chen et al. 2013). In spite of this, the equations of the model and their relationships still remain difficult to understand for the user.

2.4 Our results – Modelica libraries and PHYSIOMODEL

One of the ways to facilitate the understanding of complex hierarchical models consists in using the new object-oriented modelling language Modelica. Therefore we have decided to reimplement the entire complex model of the American authors in this language.

We were not frightened by the complex structure of the HumMod model (called **QHP** in the previous version) and established closer cooperation with the American authors.

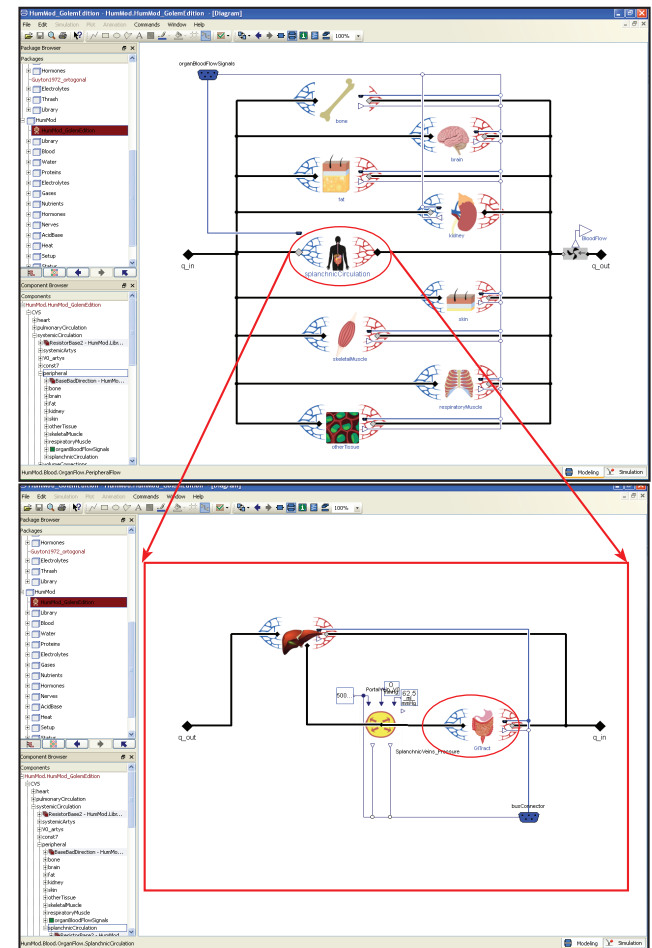


Figure 8. An illustration of a part of the source text of our HumMod implementation in Modelica. The source text resembles hierarchical physiological schemes. The content of the component of splanchnic circulation (from the upper figure) shows blood flow through the gastrointestinal tract component, the elastic compartment of the portal vein, and flow through the liver.

We developed a special software tool called **QHPView** (J. Kofránek, Mateják, and Privitzer 2010) to create a clear visualization of the used mathematical relationships from the thousands of the source text files. This enabled us to orient ourselves in the large model.

The model reimplementation in Modelica resulted in a substantially better visualization of the model structure (see Fig. 8) and among other things, it also helped to discover some mistakes in the original American implementation of *HumMod*. We modified and expanded the model especially regarding the modelling of the transfer of blood gases and homeostasis of the inner environment, especially the acid-base balance (Jiri Kofranek, Matejak, and Privitzer 2011; Jiri Kofránek et al. 2013; Marek Mateják 2015; M. Mateják and Kofránek 2015).

Our version of *HumMod* called **PHYSIOMODEL** has been developed as an open source model. Source texts of the model (i.e. the equations, values of all constants, etc.) representing formalized expressions of physiological relationships are available to the public at <http://www.physiomodel.org>. The development of the integrative model of human physiology has also resulted in designing application libraries for the modelling of physiological and chemical systems in Modelica called “*Physiolibrary*” and “*Chemical*” (see <http://www.physiolibrary.org>) (Marek Mateják et al. 2014; Marek Mateják 2014; Marek Mateják et al. 2015; Matejak et al. 2015).

A more detailed description of the libraries and of our implementation of the integrative model of human physiology is the subject of a PhD thesis (Marek Mateják 2015).

3 Importance of integrative models

A relatively logical question emerges in a connection with the relatively demanding activities of developing integrated models – what can these models, created while exerting such great efforts, be used for?

3.1 Understanding of the context

The main benefit of these models consists in *understanding how the human organism works as a whole, being a hierarchical system with complex regulations*; how individual disturbances are manifested representing the bases of various diseases; and how an appropriate therapy is applied.

The reason why actually Guyton with his collaborators created the model cited in the introduction can also be given as an example (A. C. Guyton, Coleman, and Granger 1972). It was for the study of regulatory disorders resulting in high blood pressure, for the study of effects that control the heart pump activity, and for exploration of adaptive responses to a heart failure (A. C. Guyton, Granger, and Coleman 1971; A. C. Guyton 1981). The

model has helped to understand the mechanisms of these actions.

In the past, when the physiologists focused only on the study of the dynamics of blood circulation, a simple mechanistic notion existed saying that high blood pressure was caused by an elevated peripheral resistance of blood vessels. Clinical findings in hypertonic patients corresponded to this notion – some of them actually did have increased peripheral resistance. However, we can ask why in some diseases associated with increased peripheral resistance (for example, hypothyroidism or in conditions after amputation of multiple limbs) the blood pressure is normal? Also, blood pressure remains unchanged in some diseases where peripheral resistance is decreased – for example, hyperthyroidism, beriberi, anaemia or arteriovenous shunts. It has shown, that exploration of regulation in the circulatory system alone is insufficient to explain these phenomena; we need to take into account also the regulation of volume and osmolarity of body fluids and the regulation of water and salts intake and output. Namely arterial blood pressure depends, among others, not only on peripheral vascular resistance but also on the contents of the vascular bloodstream, i.e. on the overall volume of circulating blood and also on the cardiac output. Blood pressure rises together with the volume of circulating blood. Kidneys promptly respond to this situation, excrete the excessive volume, and the blood pressure is adjusted. When the heart starts pumping more blood in a time unit – i.e. when the output increases, while peripheral resistance does not decrease at the same time, blood pressure rises, as well. On the other hand, the heart is a special pump that is controlled also by the pressure at its output – heart output changes also when pressure increases in large veins at the input of the atria of the heart. When the heart output remains increased in the long term, it gradually leads to a regulatory response in the periphery where peripheral resistance rises in order to reduce chronic hyperperfusion of internal organs. As shown by the research of Guyton using simulation models, the pathogenesis of the hypertension disease consists in disorders of these regulatory mechanisms – the kidneys are wrongly set to regulating a larger volume of circulating blood; increased contents of the bloodstream leads to an increase of the contents of large veins; this causes an increase in the pressure in large veins; and the increased pressure at the input of the atria of the heart causes increased heart output responded to in the periphery by increased peripheral resistance after some time (in order to reduce the hyperperfusion of peripheral organs), and thereby the increased blood pressure becomes fixed (A. C. Guyton, Hall, and Montani 1988).

In his models, Guyton also showed the mechanism of adaptive response to heart failure where, again, mechanisms related to circulatory and volume regulation are applied (J.-P. Montani and Van Vliet 2009). The

results of these simulation studies have found their way to medical textbooks.

Guyton himself paid great attention to teaching of physicians and wrote a generally recognized textbook of physiology that provides a logical explanation of the mechanism of physiological regulatory actions. Guyton died in 2003 in a car accident, but his collaborators and students continue his work – they not only elaborated the original Guyton's model creating the above mentioned extensive model HumMod, but they also continue publishing his textbook complemented with new knowledge – currently, the 13th edition of this textbook is available (Arthur C. Guyton and Hall 2015).

The extended integrated model has also found application in cosmic medicine. For example, the disciples of Guyton succeeded in using the model to explain why the adaptation to the gravitational force of the Earth after returning from an orbit takes about 5 times longer in female than male astronauts. Model simulations showed the cause of this phenomenon. In females, the centre of gravity is found lower than in males due to anatomical differences. The extracellular space is dewatered in the weightless condition and rewatered again after returning to the atmosphere – in females, the volume of fluids moved back from the blood to the interstitium is larger than in males due to the shifted centre of gravity, resulting in prolonged adaptation to the force of gravity – for details see (Summers et al. 2010).

Simulation games with an integrated model can also contribute to the guidelines for some procedures in acute medicine. For example, HumMod showed why (and for how long) it is important to preoxygenate the patient with 100% inhalatory oxygen before intubation (during anaesthesia) (this is a guideline for anaesthesia) – the patient namely does not breathe for a certain period of time during the intubation procedure. Furthermore, as shown by the model, it is needless to apply preventive hyperventilation after intubation and after connecting the patient to artificial pulmonary ventilation (which used to be the routine approach of some anaesthesiologists) – for details see (Lerant et al. 2015).

One of the body's most critical tasks is water homeostasis. Physical challenges to the body, including exercise and surgery, almost always coordinate with some change in water handling reflecting the changing needs of the body. The HumMod integrative mathematical model of human physiology was also validated against six different challenges to water homeostasis with special attention to the secretion of vasopressin and maintenance of electrolyte balance. HumMod successfully replicated the experimental results, remaining within 1 standard deviation of the experimental means in 138 of 161 measurements. Only three measurements lay outside of the second standard deviation. This validation suggests that

HumMod can be used to understand water homeostasis under a variety of conditions (W. A. Pruett, Clemmer, and Hester 2016).

As shown by the examples above, the use of integrated models can help to *explain causal relationships of a number of physiological actions*.

3.2 Populations of virtual patients for clinical studies

In order to explain the course of pathogenesis of various diseases and responses of people to administered therapy, it is important to ensure that the integrated model represents more than a kind of an average person. Sensitivity analysis can show how the changes in values of individual parameters are manifested in the overall behaviour of the model. For the purpose of studying individual responses, the integrated model representing a “normal” patient is used to create a population of models representing the population of various patients by variation of parameter values (approx. by $\pm 10\%$). Precisely this approach then makes it possible to observe individual variability of behaviour of the model and compare the same to individual variability of the population of real patients.

Thus, for example, in the study of individual responses to bleeding (Zhang, Pruett, and Hester 2015) the population of 395 patients was first created using this method. About 85% of the thus created population

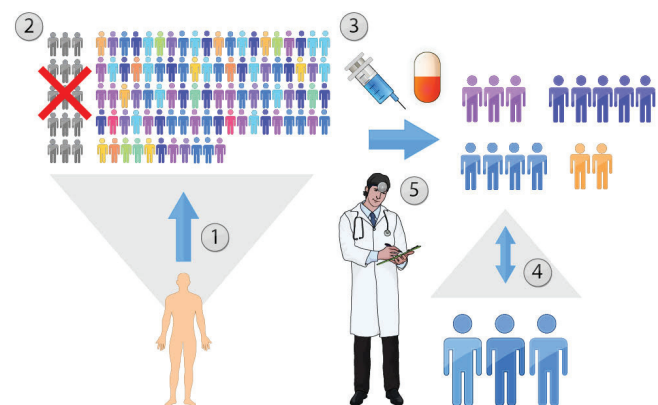


Figure 9. A possible way of using integrative models for interpretation of clinical study results. (1) A variation of parameter values is used to create a population of virtual patients. (2) Patients whose variable values exceed normal ranges are excluded from the thus created population of patients. (3) The remaining “healthy” heterogeneous population of virtual patients is used to perform the clinical experiment (simulated administration of pharmaceuticals). (4) The virtual patients are sorted in groups with similar responses to the virtual therapy. (5) For the given groups of virtual patients, we try to find matching groups of real patients with similar responses in the clinical study. Analysis of behaviour of the simulation model during the simulated therapy is used to seek explanations of individual differences in responses to the administered therapy.

of virtual patients showed normal physiological values – and only 15% showed abnormal values, which were removed from the population. This is how a single integrated model of the “average” patient was used to create the population of models representing a set of individual (virtual) patients. And this heterogeneous population was then used for research aimed at revealing the causes of individual deviations in patient responses to a pathogenic noxa (bleeding in the given case) or to administered therapy. Results of the study (behaviour of virtual patients in haemorrhage) were then classified using cluster analysis in order to sort patients with similar behavioural patterns; these groups of virtual patients were then compared to similar behavioural patterns in real patients. Subsequently, qualitative analysis of the model behaviour could be done in order to find the causes of the individually different responses.

The analysis of sensitivity of parameters affecting the blood pressure value was performed similarly, resulting in the population of individual models with similar behavioural patterns based on an older model of Guyton (Moss et al. 2012).

It thus seems that the path leading to future application of integrative models in clinical situations (especially in clinical studies) consists in generating a population of models representing the population of virtual patients, subsequent modelling of the given pathology or effect of medications using this heterogeneous population of models, and sorting of the simulated virtual patients to groups according to similar responses. Based on comparison with a group of similarly responding patients of the clinical study, analyses of the model behaviour can reveal the causes of different responses in the patient groups to the given pathogenic noxa or to the therapy (see Fig. 9).

The pathology or the effect of the therapy in integrated models is usually modelled by changing some parameters that cause an appropriate (pathogenic or therapeutic) response.

As follows from simulation studies, the cause of differences of some individual responses need not be based on a difference in only one parameter – but in combined changes of several parameters.

Let us explain this problem using an illustrative example (Fig. 10) (P. Kohl et al. 2010). For a clearer idea, only the parametric state space of two parameters, P1 and P2 will be considered. The value of the hypothetic biological function is the axis z differentiated by its height and colour. We shall consider a patient whose biological profile is located at the position A. The required action (simulating the effect of the therapy or the effect of a “side effect of some other medication”) consists in reducing the parameter P1 to its target value. A direct change of parameter P1 (the path from A to C) leads to a serious

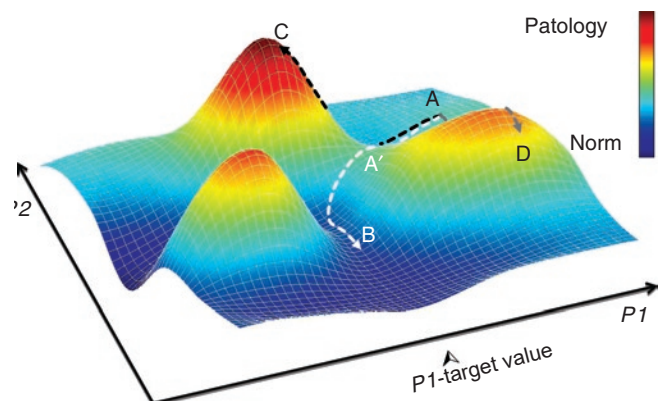


Figure 10. The state space of parameters P1 and P2. The vertical axis represents the size of the biological response with the given values of parameters P1 and P2. The purpose of a targeted intervention is to change the value of parameter P1 to its target value. No biological response is induced by a concurrent change of parameters P1 and P2 (path from A to B). The change of parameter P2 to the target value results in a biological effect (path from A to C). Similarly, if only parameter P2 changes (while parameter P1 remains unchanged), a biological response is induced. This illustrative example demonstrates that isolated changes of individual parameters may result in a biological effect while a covariant, concurrent change of two parameters may not cause any effect at all.

biological (pathological) response. The covariance of both parameters P1 and P2 (the path A – A' – B) makes it possible to move to the required level P1 without any harmful consequences. An isolated reduction of the parameter P2 in the same range as at the point B (without changing P1) would also be harmful; as can be seen intuitively, precisely the path of gradual modification of both parameters (P1 and P2) from A to B causes no biological response.

And vice versa – only a concurrent change of several parameters causes an unfavourable biological effect, while changes in individual parameters cause no adverse biological effect – and frequently, this is also the core of the robustness and ultrastability of physiological regulations that can be revealed precisely and only using integrative models.

Thus for example, variations of parameters of the HumMod model were used to monitor the sensitivity of blood pressure changes after a salt intake (W. Pruett, Husband, and Hester 2014). It was shown that no single parameter which would lead to an increased blood pressure after an increased salt intake existed – this is the core of high stability of physiological regulations. Only the change of several parameters resulted in a pathological response.

By comparing classified groups of virtual patients with the same behaviour and groups of real patients in clinical studies – and subsequent qualitative analysis of the courses of the modelled actions, the cause of individual deviations

in response to an appropriate stimulus can be revealed – no matter whether the stimulus is a pathological noxa or the effect of a medication.

This is why integrated models will also find their future application in clinical studies. The project of the European Union called “*AVICENNA – A Strategy for in silico Clinical Trials*” (see <http://avicenna-iscrt.org>), currently under preparation, will focus on the topic of using simulation models in clinical studies.

3.3 Medical simulators

Medical simulators represent another extensive field for the application of integrative models; similarly as flight simulators, medical simulators provide quite a new mode of teaching where students can train diagnostic and therapeutic tasks in virtual reality without any risk for the patient. In sophisticated medical simulators, students can also observe in detail the course of values of various quantities that are commonly not available for clinical examination in real patients, which supports deeper understanding of the pathophysiological core of the development of the clinical condition and its affection by therapeutic interventions.

The important thing is that unlike the real world, mistakes are reversible in virtual reality. When a flight simulator is used to train landing we can crash many times in a row, while in the real world an airplane crashes only once as a rule. In acute care medicine, diagnostic and therapeutic procedures can be trained on a virtual patient who can be brought back to life at any time. However, patients in the real life have no “reset” button and, as expressed by one harsh proverb, “the mistakes of rescuers are covered by soil”.

Similarly as a sophisticated airplane model is the core of flight simulators, an integrative patient model is the key component of current top medical simulators (for example, in CAE Healthcare simulators – see <http://www.caehealthcare.com>).

4 Development tools for integrative models

Formerly, dynamic systems were often programmed using *analogue computers*, while later they were combined with a digital computer in the so-called *hybrid computers*. The program was created by connecting individual computing elements (integrators, summaters ...) using connection cables. The computer processed analogue (continuous) electrical signals whose changes were responded to immediately, and therefore it remained a suitable tool for solving sets of differential equations of simulation models until the increasing power of digital computers removed this advantage of analogue solutions.

4.1 Classical programming languages for the

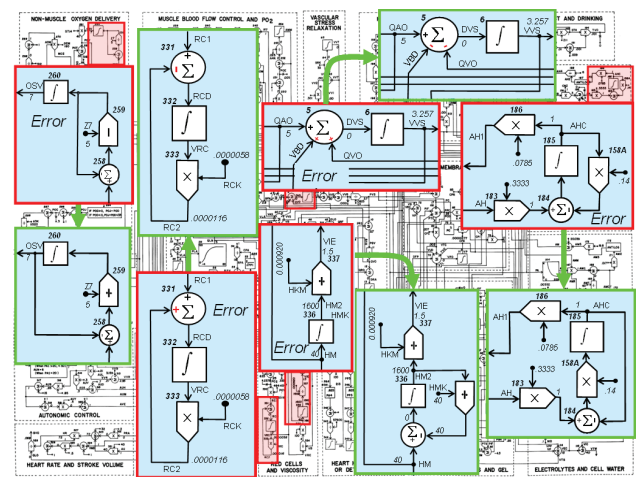


Figure 11. Mistakes in the Guyton's graphic scheme and their corrections.

development of simulation models

The era of analogue computers inspired also the graphic notation of Guyton used to write physiological models using a network of mutually connected computing blocks (integrators, summaters, dividers, multipliers and function blocks). However, in 1972, at the time when the groundbreaking paper of Guyton (A. C. Guyton, Coleman, and Granger 1972) was published, models were implemented predominantly on digital computers using *classical programming languages* (e.g. Fortran, C, C++ etc.). The graphic scheme in the paper served only as an illustrative picture used to provide a compact description of the model structure. The model itself was programmed in the programming language Fortran for digital computers.

However, this scheme was not flawless (Jiří Kofranek and Rusz 2010) – some errors were apparent at first sight (for example, a wrongly connected integrator that would soon result in its overloading with an infinitely rising value due to the feedback), while others required a deeper analysis, understanding of the text of the article and knowledge of physiology (Fig. 11). Actually, these were easily detectable “graphic typing errors” (switched signs, shifted connectors) without any effect on the model functionality because the entire scheme was created only as an illustration and not as the source code of the model (programmed in Fortran). The picture itself was a part of the PhD thesis of a co-author of the Guyton's article, Thomas Coleman, and today, it can be find as a certain scientific relic in a display case of the Guyton's research centre at the University of Mississippi.

4.2 Simulation chips in block-oriented languages

At the beginning of the 1990s, specialized modelling tools emerged. These tools used computing blocks (very similar to those used by Guyton in his graphic notation); these blocks are connected on the computer screen using the mouse to create a simulation network.

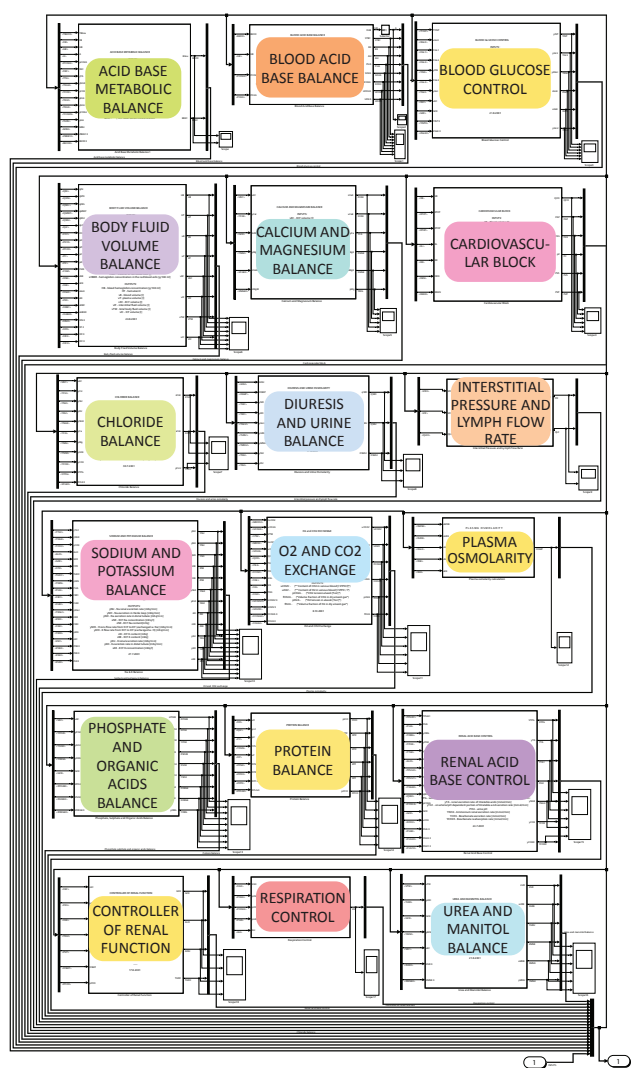


Figure 12. Structure of the connected blocks that implement the model for the Golem simulator in the block-oriented language Simulink. The inputs and outputs of 18 blocks modelling individual physiological subsystems are connected through a common bus.

These so-called *block-oriented simulation languages* utilize connected blocks. Signals “flow” through the connecting lines between individual blocks and transfer the values of individual variables from the output of one block to the inputs of other blocks. The inputted information is processed in the blocks to obtain the output information. The connections among individual blocks show how the values of individual variables are calculated – i.e. the algorithm of the computation.

Blocks can be grouped in individual subsystems that communicate with their surroundings through input and output “pins”, thus representing certain “simulation chips”. These subsystem blocks hide the simulation network structure from the user, similarly as electronic chips hide the connection structure of individual transistors and other electronic elements so that the user need not take care of the internal structure and of the computational algorithm used to obtain output variables from input

variables. “Simulation chips” in block-oriented languages have a hierarchical structure – they may contain a network of mutually connected subsystem blocks of a lower hierarchical level. “Simulation chips” can be grouped in libraries and their individual instances can be created using the mouse; their inputs and outputs are connected using connecting lines, through which information (i.e. the values of variables) “flows”. The entire complex model can be thus depicted as interconnected simulation blocks, while the structure of their connection provides clear information about what values are calculated and how.

This facilitates interdisciplinary cooperation in the development of integrative models where experimental physiologists do not have to explore in detail what mathematical relationships are hidden inside the connected subsystem blocks, and from the connections among individual subsystem blocks the physiologists can understand the model structure and verify the model behaviour in an appropriate simulation visualization environment of the block-oriented simulation language.

Block-oriented simulation languages provided a considerable simplification of implementation of simulation models. The most widely used block-oriented languages include, for example, Mathworks Simulink (<http://www.mathworks.com/products/simulink>) or Visual Solution VisSim (<http://www.vissim.com>).

In the past, we used Simulink to create a freely available library of blocks for the modelling of physiological systems (<http://www.physiome.cz/simchips>), which also included the source code of an integrated model of physiological systems used as a source for our teaching simulator Golem (Fig. 12). The teaching simulator Golem was developed by us at the end of the 1990s and at the turn of the millennium it was intended for teaching the homeostasis of the inner environment in clinical physiology. The simulator was used at some national as well as foreign faculties of medicine (Jiří Kofránek et al. 2001).

4.3 Disadvantages of block-oriented simulation languages

Blocks in block-oriented languages have a hierarchical structure. On the lowest level, the blocks are created as a network of interconnected numeric blocks that use input values to calculate output values. The connections among the numeric blocks represent a solution of mathematical equations of the model designed so that output values are calculated from input values.

However, the connection of blocks in the network of relationships cannot be arbitrary. No algebraic loops may occur in the connected elements – i.e. cyclic structures where an input value brought to the input of a computing block depends in the same time step (through several

intermediators) on the output value of the same block.

Development environments of block-oriented languages provide tools to avoid algebraic loops; however, their use often results in transformations that make the model structure less clear.

The main problem of block-oriented languages is that the simulation network composed of hierarchically connected blocks shows a graphic representation of a chain of transformations of input values to output values, meaning that when the model is designed, an exact computational algorithm must be defined from input to output values of the model.

The requirement of a fixed direction of connections from inputs to outputs means that the *connections of the blocks reflect the computation procedure and not the very structure of the modelled reality.*

For example, when the direction of the computation is reversed (by replacing inputs with outputs), the algorithm will be different although the model equations remain the same. Thus for example, in the model of an electrical RLC circuit (or its hydraulic analogy) it will make a difference if the voltage (pressure in the hydraulic domain) or (electrical or hydraulic) current is used as the input for the circuit although the electrical (hydraulic) scheme itself does not change. The Simulink network representing the computational process will be different.

In complex models, it is usually not simple to derive the computation causality (i.e. to derive the algorithm of computing output variables from input variables).

4.4 Modelica – the best tool for development models of integrative physiology

At the turn of the millennium, a completely new category

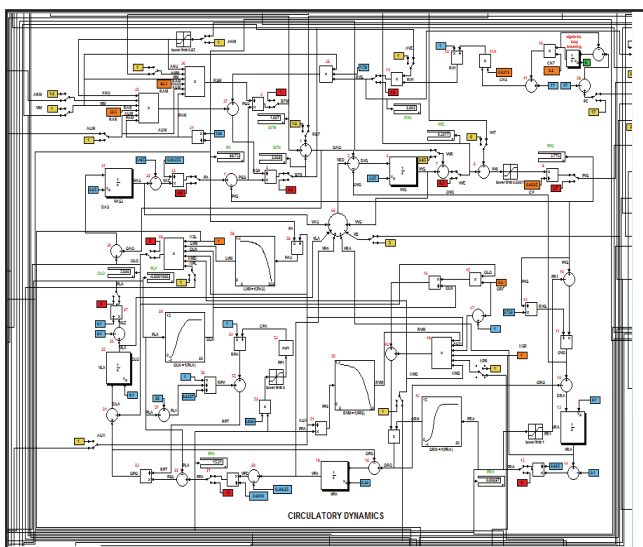


Figure 13. Circulatory dynamics - more detailed central structures of the Simulink implementation of Guyton's model, representing flows through aggregated parts of the circulatory system and the activity of the heart as a pump..

of modelling tools emerged, which makes it possible to leave the computation aside and *describe directly equations in the modelling blocks*. A special object-oriented equation-based language called *Modelica* was developed.

Modelica, originally developed as an academic project in cooperation with small development companies associated with Lund and Linköping universities, soon showed to be a very efficient tool for the modelling of complex models applied particularly in mechanical engineering, in the automobile and airline industries. The development of Modelica therefore gradually gained support of the commercial sector.

The speed at which the new simulation language Modelica spread in various industries and at which it was embraced by various commercial development environments is astonishing. Today, several commercial and non-commercial development tools exist that use this language (see <https://www.modelica.org>).

In Modelica, the connection of individual components results in the connection of sets of equations with each other. The component connections thus define the modelled reality instead of the computation process. The way of resolving the equations is thus “left up to the machines”.

Unlike block-oriented languages where the structure of connections among hierarchical blocks represents rather the computation process instead of the modelled reality, the structure of models in Modelica shows the structure of the modelled reality (see fig. 13 and 14). This is why even complex models are sufficiently transparent and comprehensible in Modelica (Kulhánek, Kofránek,

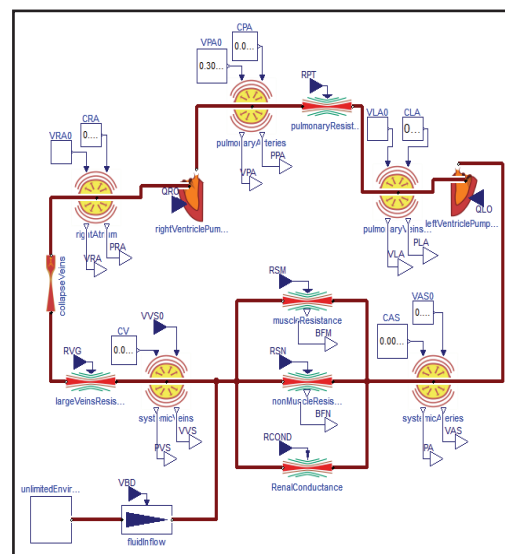


Figure 14. The same model structure as is shown in figure 13 implemented in Modelica. The structure of the model in Simulink corresponds to the structure of computational steps, while the Structure of Modelica model reflects the structure of the modeled physiological reality.

and Matejak 2014).

This is very important precisely for the development of complex integrated models. The task of unifying and designing complex models faces another problem due to the complicatedness. Usually, only the authors are able to understand and use complex models. Modelica partially resolves this problem thanks to its characteristics and a complex model of human physiology designed in Modelica may lead to a wider use of the model in the scientific community. The source text of our integrated model of human physiology **PHYSIOMODEL** in Modelica (see <http://www.physiomodel.org>) resembles hierarchical physiological schemes (see Fig. 8). **PHYSIOMODEL** is an implementation of **HumMod** (modified and expanded, particularly in the field of acid-base balance and the transfer of blood gases) (Jiri Kofranek, Matejak, and Privitzer 2011; Jiri Kofranek et al. 2013; Marek Matejak 2015; M. Matejak and Kofranek 2015).

5 Prospects of integrative models of human physiology

5.1 Prospects of sharing and publishing integrative models

The development of integrative models in physiology exhibits an exclusively *interdisciplinary nature*. The team needs to have broad knowledge of physiology as well as knowledge of computer sciences, mathematics, the theory of control, and cybernetics. In addition, the team members of various professions must dispose of a considerable intersection of their knowledge.

This is also why there are not many scientific teams that develop large integrative models in physiology.

The developed integrative models should be *comprehensible* not only within the development team, but also externally – if a model is comprehensible only to its authors, it will hardly receive the necessary feedback and new impulses from specialized scientific community.

The issue of a *suitable form of publishing* the achieved results is also related to this issue. *Reproducibility* is the main attribute of any scientific result. Leaving aside certain acts of deception not discovered by reviewers, the principle of reproducibility is a key for the gradual discovery of the secrets of nature. This principle is often violated in the field of scientific publications related to biomedical models (both small and large). It is not always the fault of the authors – many times the reason is that a sign or an index is omitted in equations while the paper is prepared for printing, which causes a lot of problems to readers who seek not only to understand, but also to implement the described model.

In addition, in many cases biomedical models are as complex that the limited space for the article is sufficient only for fundamental equations of the model (and often

not even all of them), while no space remains for more detailed information (initial values of state variables, values of all parameters, etc.), necessary to set up the model at a different department. Therefore the classical form of publications of models in journals is insufficient. A specialized article that describes a model should include, as a minimum, a digital (available through the Internet) appendix giving a detailed description of the model structure including the values of all parameters (preferably in the form designed in some modelling language), sufficient for the reader to be able to reproduce the model (and perhaps follow up in his or her own work). This solution has already been approached by a number of journals that publish specialized articles on computer models.

If the model is published in a modelling language that requires a commercial licence (for example, in MathWorks Matlab&Simulink), a problem arises because the reader needs to have an appropriate licence to be able to run the given model in the licensed development environment.

Considerable efforts were thus developed in the international project **PHYSIOME** to create databases – repositories of models that, besides storing the source text of the model in the defined format, offer publicly available tools for their simulation. Given that Modelica is a standardized language – and not a corporate proprietary product (such as MathWorks Matlab&Simulink) and given that open source development tools exist today for this language (for example, OpenModelica – see <https://openmodelica.org>) – **Modelica** seems to be a highly promising tool for *publishing and sharing biomedical models*.

So far, no other open source alternative besides Modelica exists that could be used for publishing extensive models. For example, Guyton's model version of 1992, implemented by Montani in C using the C-MODSIM environment (J. P. Montani, Adair, et al. 1989), is divided in the cellML repository in 22 modules in the open source cellML language. However, attempts at running these modules connected in one unit were not successful (<https://models.cellml.org/exposure>), while the Simulink version of the model works without problems (Mangourova, Ringwood, and Van Vliet 2011) (however, it requires the commercial environment of Matlab@Simulink).

For the sake of completeness, we should note that theoretically, also the environment used to publish HumMod is an open source environment for implementation of large models – the source code of the model is saved in a number of XML files. This is sufficient for simple models; however, complex models are difficult to be understood by users – the reader can compare for themselves the HumMod model structure in the original form (<http://hummod.org>) and its implementation in

Modelica (<http://www.physiomodel.org>).

Modelica thus seems to be a **promising publication tool for extensive integrated models**.

5.2 Prospects of commercial application of integrative models

The potential of **commercial application** especially in the two areas below will provide a powerful stimulus for further development of integrative models of human physiology:

- in medical teaching simulators;
- in the development of new therapeutic methods and in clinical testing of new pharmaceuticals.

Medical simulators provide a very efficient teaching aid. They enable the students to train basic examination and therapeutic techniques and also the process of decision-making in medicine. Sophisticated medical simulators utilize a robotized patient mannequin as the user interface. A model of interconnected physiological systems of the body is the core of modern medical simulators. Integrative physiology and integrated models of physiological systems thus become the technological know-how for the development of products with a high added value in the form of medical information and robotic knowledge that can find applications on the rapidly developing market.

Integrative models of human physiology will allow detailed monitoring of causal chains of application of various therapeutic or pathogenic stimuli, thus providing a wide potential for application of integrative models of human physiology especially in **clinical testing of pharmaceuticals** and in the development and **testing of modern medical instruments** (see Section 3.2).

The pressure of possible commercial applications leads to the fact that formalized descriptions of physiological regulations expressed as an integrated model often become carefully protected information, which limits the sharing of the results of scientific physiological research and undermines the possibilities of scientific cooperation.

5.3 Prospects of combining commercial and academic development

However, international cooperation and openness to sharing the results are the driving force of scientific development in today's globalized world. For example, as shown by experience, a community of users and developers as wide as possible is important for the development of complex software systems, thus a community that can provide feedback and ensure further innovations of a complex product through cooperative development, while subsequently, further entrepreneurial opportunities open up in the connection with this product – this is why such a great spreading of the development of projects with the open source code has been seen in recent years.

In order to ensure the development of complex integrated models in physiology, it will probably be suitable to seek such forms that will combine entrepreneurial opportunities and financing by the commercial sector with open scientific development.

One of the possibilities is to utilize a similar form in which the product OpenModelica has been developed in the open community (see <https://openmodelica.org>). The development of products is ensured by the consortium of 23 universities and 23 companies and institutions as well as a number of individual developers (**Open Source Modelica Consortium** – see <https://openmodelica.org/home/consortium>). Research is financed using member contributions whose amount is determined based on the size of the company and based on the number of sold products in whose development OpenModelica licences have been used. OpenModelica has created a circle of a relatively large community of users as well as a high number of cooperating developers; the result is a functional open source product equivalent in terms of functionality to competitive expensive commercial implementations of Modelica such as Dymola from Dassault Systèmes), MapleSim from MapleSoft), Wolfram SystemModeler from Wolfram, etc. Commercial companies may use and further develop any part of the OpenModelica environment in their own commercial applications, also in the development of competitive commercial implementations of Modelica (this is why companies such as Wolfram Math Core or MapleSoft are also members of the consortium).

Perhaps a consortium of the academic community and commercial companies built on similar foundations – called e.g. **“Physiomodelica Open Source Consortium”** could ensure further development of an integrated model of physiology in the future.

Acknowledgements

The authors appreciate the partial funding of this work by PRVOUK P/24/LF1 and FR Cesnet 551/2014.

References

- Amosov, N. M., B. L. Palec, B. T. Agapov, I. I. Jermakova, E. G. Ljabach, S. A. Packina, and V. P. Solovjev. 1977. *Theoretical Research of Physiological Systems: Mathematical Modeling* (in Russian). Naukova Dumka.
- Atkins, Gordon Leslie. 1969. *Multicompartment Models for Biological Systems*. Methuen London.
- Bassingthwaight, J. B. 2000. "Strategies for the Physiome Project." *Annals of Biomedical Engineering* 28 (8). Springer: 1043–58.
- Chen, Jian, Keqin Wu, William A. Pruetz, and Robert L. Hester. 2013. "HumMod Browser: An Exploratory Visualization Tool for Model Validation of Whole-Body Physiology Simulation." In *Eurographics Conference on Visualization (EuroVis)* (short Paper). researchgate.net. https://www.researchgate.net/profile/Keqin_Wu2/publication/303290077_HumMod_Browser_An_Exploratory_Visualization_Tool_for_Model_Validation_of_Whole-Body_Physiology_Simulation/links/573f6ab108ae298602e8f3cf.pdf.
- Coleman, T. G., and R. L. Summers. 1997. "Using Mathematical Models to Better Understand Integrative Physiology." *Journal of Physiology and Biochemistry* 53: 45–46.
- Fontcave-Jallon, J., and S. R. Thomas. 2015. "Implementation of a Model of Bodily Fluids Regulation." *Acta Biotheoretica* 63 (3): 269–82.
- Grodins, F. S., J. Buell, and A. J. Bart. 1967. "Mathematical Analysis and Digital Simulation of the Respiratory Control System." *Journal of Applied Physiology* 22 (2). DTIC Document: 260–76.
- Guyton, A. C. 1981. "The Relationship of Cardiac Output and Arterial Pressure Control." *Circulation* 64 (6): 1079–88.
- Guyton, A. C., T. G. Coleman, and H. J. Granger. 1972. "Circulation: Overall Regulation." *Annual Review of Physiology* 34. annualreviews.org: 13–46.
- Guyton, A. C., H. J. Granger, and T. G. Coleman. 1971. "Autoregulation of the Total Systemic Circulation and Its Relation to Control of Cardiac Output and Arterial Pressure." *Circulation Research* 28 (January): Suppl 1:93–97.
- Guyton, A. C., J. E. Hall, and J. P. Montani. 1988. "Kidney Function and Hypertension." *Acta Physiologica Scandinavica*. Supplementum 571: 163–73.
- Guyton, A. C., R. D. Manning Jr, R. A. Norman Jr, J. P. Montani, T. E. Lohmeier, and J. E. Hall. 1986. "Current Concepts and Perspectives of Renal Volume Regulation in Relationship to Hypertension." *Journal of Hypertension*. Supplement: Official Journal of the International Society of Hypertension 4 (4): S49–56.
- Guyton, A. C., and John E. Hall. 2015. *Guyton and Hall Textbook of Medical Physiology*. Elsevier Health Sciences.
- Hester, R., A. Brown, L. Husband, and R. Iliescu. 2011. "HumMod: A Modeling Environment for the Simulation of Integrative Human Physiology." *Frontiers in*. journal.frontiersin.org. <http://journal.frontiersin.org/article/10.3389/fphys.2011.00012>.
- Hester, R. L., T. Coleman, and R. Summers. 2008. "A Multi-level Open Source Integrative Model of Human Physiology." *The FASEB Journal* 22 (1 Supplement): 756.8–756.8.
- Hester, R. L., R. Iliescu, R. Summers, and T. G. Coleman. 2011. "Systems Biology and Integrative Physiological Modelling." *The Journal of Physiology* 589 (Pt 5). Wiley Online Library: 1053–60.
- Hodgkin, A. L., and A. F. Huxley. 1952. "A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve." *The Journal of Physiology* 117 (4). ncbi.nlm.nih.gov: 500–544.
- Hunter, P. 2016. "The Virtual Physiological Human: The Physiome Project Aims to Develop Reproducible, Multiscale Models for Clinical Practice." *IEEE Pulse* 7 (4). ieeexplore.ieee.org: 36–42.
- Hunter, P., J. Edmund, J. Crampin, and Poul M. F. Nielsen. 2008. "Bioinformatics, Multiscale Modeling and the IUPS Physiome Project." *Briefings in Bioinformatics* 9 (4). Oxford Univ Press: 333–43.
- Hunter, P., P. Robbins, and D. Noble. 2002. "The IUPS Human Physiome Project." *Pflügers Archiv: European Journal of Physiology* 445 (1). Springer: 1–9.
- Hunter, P. J., W. W. Li, A. D. McCulloch, and D. Noble. 2006. "Multiscale Modeling: Physiome Project Standards, Tools, and Databases." *Computer* 39 (11). ieeexplore.ieee.org: 48–54.
- Ikeda, N., F. M., M. Shirataka, and T. Sato. 1979. "A Model of Overall Regulation of Body Fluids." *Annals of Biomedical Engineering* 7 (2): 135–66.
- Kofranek, J., M. Matejak, and P. Privitzer. 2011. "Hummod-Large Scale Physiological Models in Modelica." In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical University; Dresden; Germany*, 713–24. Linköping University Electronic Press.
- Kofránek, J., M. Mateják, P. Privitzer, M. Tribula, T. Kulhánek, J. Šilar, and R. Pecinovský. 2013. "HumMod-Golem Edition: Large Scale Model of Integrative Physiology for Virtual Patient Simulators." In *Proceedings of the International Conference on Modeling, Simulation and Visualization Methods (MSV)*, 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).
- Kofranek, J., and J. Rusz. 2010. "Restoration of Guyton's Diagram for Regulation of the Circulation as a Basis for Quantitative Physiological Model Development." *Physiological Research / Academia Scientiarum Bohemoslovaca* 59 (6). Institute of Physiology: 897.
- Kofránek, J., L. D. Anh Vu, H. Snaselova, R. Kerekes, and T. Velan. 2001. "GOLEM-Multimedia Simulator for Medical Education." *Studies in Health Technology and Informatics*, no. 2. IOS Press; 1999: 1042–46.
- Kofránek, J., M. Mateják, and P. Privitzer. 2010. "Web Simulator Creation Technology." *MEFANET Report* 3: 32–97.
- Kohl, P., E. J. Crampin, T. A. Quinn, and D. Noble. 2010. "Systems Biology: An Approach." *Clinical Pharmacology and Therapeutics* 88 (1): 25–33.
- Kohl, P., and D. Noble. 2009. "Systems Biology and the Virtual Physiological Human." *Molecular Systems Biology* 5 (July): 292.
- Kulhánek, T., J. Kofránek, and M. Mateják. 2014. "Modeling of Short-Term Mechanism of Arterial Pressure Control in the Cardiovascular System: Object-Oriented and Acausal Approach." *Computers in Biology and Medicine* 54 (November): 137–44.

- Lerant, A. A., R. L. Hester, T. G. Coleman, W. J. Phillips, J. D. Orledge, and W. B. Murray. 2015. "Preventing and Treating Hypoxia: Using a Physiology Simulator to Demonstrate the Value of Pre-Oxygenation and the Futility of Hyperventilation." *International Journal of Medical Sciences* 12 (8). ncbi.nlm.nih.gov: 625–32.
- Mangourova, V., J. Ringwood, and B. Van Vliet. 2011. "Graphical Simulation Environments for Modelling and Simulation of Integrative Physiology." *Computer Methods and Programs in Biomedicine* 102 (3). Elsevier: 295–304.
- Matejak, M.. 2014. "Physiology in Modelica." *MEFANET Journal* 2 (1). Facta Medica: 10–14.
- Matejak, M. 2015. "Formalization of Integrative Physiology. Charles University in Prague." Edited by Jiří Kofránek. Ph.D., Charles University. <https://github.com/MarekMatejak/dissertation/blob/master/thesis.pdf>.
- Matejak, M., F. Ježek, M. Tribula, and J. Kofránek. 2015. "Physiolibrary 2.3-An Intuitive Tool for Integrative Physiology." *IFAC-PapersOnLine* 48 (1). Elsevier: 699–700.
- Matejak, M., T. Kulhanek, J. Šilar, P. Privitzer, F. Ježek, and J. Kofránek. 2014. "Physiolibrary-Modelica Library for Physiology." In *Proceedings of the 10 Th International Modelica Conference*; March 10-12; 2014; Lund; Sweden, 499–505. Linköping University Electronic Press.
- Matejak, M., M. Tribula, F. Ježek, and J. Kofranek. 2015. "Free Modelica Library for Chemical and Electrochemical Processes." In *Proceedings of the 11th International Modelica Conference*, Versailles, France, September 21-23, 2015, 359–66. Linköping University Electronic Press.
- Matejak, M., and J. Kofránek. 2015. "Physiomodel-an Integrative Physiology in Modelica." *And Biology Society (EMBC)*, 2015 37th ieeexplore.ieee.org. <http://ieeexplore.ieee.org/abstract/document/7318646/>.
- McCulloch, Warren S., and Walter Pitts. 1943. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *The Bulletin of Mathematical Biophysics* 5 (4). Kluwer Academic Publishers: 115–33.
- Milhorn, H. T. 1966. *Application of Control Theory to Physiological Systems*. W.B. Saunders.
- Montani, J. P. and Bruce N. Van Vliet. 2009. "Understanding the Contribution of Guyton's Large Circulatory Model to Long-Term Control of Arterial Pressure." *Experimental Physiology* 94 (4). Wiley Online Library: 382–88.
- Montani, J. P., T. H. Adair, R. L. Summers, T. G. Coleman, and A. C. Guyton. 1989. "A Simulation Support System for Solving Large Physiological Models on Microcomputers." *International Journal of Bio-Medical Computing* 24 (1): 41–54.
- Montani, J. P., H. L. Mizelle, T. H. Adair, and A. C. Guyton. 1989. "Regulation of Cardiac Output during Aldosterone-Induced Hypertension." *Journal of Hypertension. Supplement: Official Journal of the International Society of Hypertension* 7 (6): S206–7.
- Moss, R., T. Grosse, I. Marchant, N. Lassau, F. Gueyffier, and S. R. Thomas. 2012. "Virtual Patients and Sensitivity Analysis of the Guyton Model of Blood Pressure Regulation: Towards Individualized Models of Whole-Body Physiology." *PLoS Computational Biology* 8 (6): e1002571.
- Omholt, S. W., and P. J. Hunter. 2016. "The Human Physiome: A Necessary Key for the Creative Destruction of Medicine." *Interface Focus* 6 (2). Royal Society: 20160003.
- Pitts, W., and W. S. McCulloch. 1947. "How We Know Universals; the Perception of Auditory and Visual Forms." *The Bulletin of Mathematical Biophysics* 9 (3). Springer: 127–47.
- Potůček, J., M. Hájek, V. Brodan, and E. Kuhn. 1977. "The Method of Estimating Biological System Parameters on Hybrid Computer." *Kybernetika* 13 (2). Institute of Information Theory and Automation AS CR: 153–64.
- Pruett, W. Andrew, John S. Clemmer, and Robert L. Hester. 2016. "Validation of an Integrative Mathematical Model of Dehydration and Rehydration in Virtual Humans." *Physiological Reports* 4 (22). doi:10.14814/phy2.13015.
- Pruett, W., L. Husband, and R. Hester. 2014. "Understanding Variation in Salt Sensitivity in HumMod, a Human Physiological Simulator (857.11)." *The FASEB Journal* 28 (1 Supplement). http://www.fasebj.org/content/28/1_Supplement/857.11.abstract.
- Sheppard, C. W. 1948. "The Theory of the Study of Transfers within a Multi-Compartment System Using Isotopic Tracers." *Journal of Applied Physics* 19 (1). AIP: 70–76.
- Shim, E. B, Ch. H. Leem, Y. Abe, and A. Noma. 2006. "A New Multi-Scale Simulation Model of the Circulation: From Cells to System." *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences* 364 (1843): 1483–1500.
- Summers, R. L., S. Platts, J. G. Myers, and T. G. Coleman. 2010. "Theoretical Analysis of the Mechanisms of a Gender Differentiation in the Propensity for Orthostatic Intolerance after Spaceflight." *Theoretical Biology & Medical Modelling* 7 (March): 8.
- Thomas, S. R., P. Baconnier, J. Fontecave, J. P. Françoise, F. Guillaud, P. Hannaert, A. Hernández, et al. 2008. "SAPHIR: A Physiome Core Model of Body Fluid Homeostasis and Blood Pressure Regulation." *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences* 366 (1878). rsta.royalsocietypublishing.org: 3175–97.
- Von Bertalanffy, L. 1973. *General Systems Theory*. George Braziller Inc., New York.
- White, R. J., and J. C. McPhee. 2007. "The Digital Astronaut: An Integrated Modeling and Database System for Space Biomedical Research and Operations." *Acta Astronautica* 60 (4–7). Elsevier: 273–80.
- Wu, K., J. Chen, W. A. Pruet, and R. L. Hester. 2013. "Hummod Browser: An Exploratory Visualization Tool for the Analysis of Whole-Body Physiology Simulation Data." In *2013 IEEE Symposium on Biological Data Visualization (BioVis)*, 97–104. ieeexplore.ieee.org.
- Xu, L., J. Lyle, Y. Wu, Z. Pan, M. Zhang, D. H. Laidlaw, R. L. Hester, and J. Chen. 2011. "HumMod Explorer: A Multi-Scale Time-Varying Human Modeling Navigator." In *SIGGRAPH Asia 2011 Posters*, 28:1–28:1. SA '11. New York, NY, USA: ACM.
- Zhang, S., W. A. Pruet, and R. Hester. 2015. "Visualization and Classification of Physiological Failure Modes in Ensemble Hemorrhage Simulation." In *SPIE/IS&T Electronic Imaging, 939700 – 939700 – 8*. International Society for Optics and Photonics.

Sound Source Extension Library for Modelica

Johann Emhofer Raimund Zitzenbacher Christoph Reichl

Center for Energy, AIT Austrian Institute of Technology, Giefinggass 2, 1210 Wien, Austria.

{johann.emhofer, raimund.zitzenbacher.fl, christoph.reichl}@ait.ac.at

Abstract

Transient thermodynamic models in Modelica are widely used for energetic simulations of machines and systems which are located nearby people. Nevertheless, so far no libraries exist which consider the noise of such machines in the simulations. The Sound Source Extension library (SSElib) proposed in this work, should close this gap. With the aid of the SSElib, acoustic characteristics can be added to existing Modelica models (e.g. to a compressor or a pump model). The acoustic characteristic added to the existing model is frequency dependent in the one-octave band and could further depend on an input parameter like the rotational speed of a compressor. With the inclusion of sound sources into energetic models, the sound behavior of machines can be considered and control strategies can be optimized to lower the noise of machines.

Keywords: *Modelica, sound, noise, acoustics, heat pump*

1 Introduction

Due to local mechanical vibrations of a component in air, local displacements of the air atoms and changes in local pressure or density are excited. The local changes propagate to the neighboring atoms. The propagation of these vibrations are better known as sound which can be recognized by the human ear. Speech, music or acoustic signals are wanted effects of sound, whereas the noise of machines are experienced as disturbing. The aim of the Sound Source Extension Library (SSElib) is to consider the unwanted noise excited from machines. With the SSElib one can extend existing Modelica components with a sound source that can depend on a variable of the component itself. This variable could be a frequency of a fan, the pressure drop over a heat exchanger or any other variable that influences the sound characteristic of the machine. In other words, an acoustic characteristic which depends on the operating conditions of a component can be added to a new or existing component.

In the SSElib we use basic acoustic calculation methods to estimate the all over loudness of a machine. Note that the main aim of the library is not an accurate prediction of the loudness but it should show how the operation point of a machine influences their noise emission. Hence, solely simple correlations for noise propagation and reduction are included in the library to give a first hint about how the acoustic characteristic of machines behave. The strength of the SSElib is the easy integration into existing

models without the need of significant extra computational power in Modelica.

Several mature methods like Finite Element Methods (FEM), Boundary Element Methods (BEM) or Computational Aero Acoustics (CAA) exist for accurate sound propagation calculations and therefore such methods should be used if a detailed sound analysis is needed. A good overview of these methods can be found in (Crocker, 2007).

The SSElib was developed in Dymola 2016 and testing was performed within the Testers and Examples package of the library. The SSElib builds on the Modelica Standard Library (MSL) and no additional libraries are needed. Most of the *Testers* were also tested in OpenModelica 1.9.6 without any problems. Besides this publication, a *UsersGuide* package was added to the library on top level, to help users with the implementation of the SSElib into their models.

SSElib is published under the Modelica License 2.

2 Methodology

2.1 General

The following assumptions were made in order to keep the equations simple:

- All sound sources are independent point sources.
- The sound fields considered in the SSElib are assumed to be diffuse and incoherent in all frequencies.
- The noise source volume has to be less than about 0.3 to 0.4 of an enclosure volume for calculations of damping (Crocker, 2007). If the noise source occupies more than a third of the enclosed volume of the sealed enclosure, the sound field is neither reverberant nor diffuse. Nevertheless, the discussed methods will be used as a first approximation of insertion losses in an enclosure even if this requirement is not fulfilled.

In this work, we concentrate on the sound pressure p and the sound power W . The sound pressure is always connected to a location and describes the local pressure amplitude at this location. Therefore, the sound pressure depends on the distance from the sound source ($\propto 1/r^2$ for a point source) and the symmetry of the emitted sound waves from the source (monopole, dipole, etc.). From

the sound pressure one can calculate the sound intensity I which takes the material properties of the sound propagating media into account. For both, a radial symmetric point source and a plane wave moving in one direction, this gives (Crocker, 2007):

$$I = \frac{p_{rms}^2}{\rho c} \quad (1)$$

for the time averaged sound intensity, where ρ is the density of the medium (air) and c is the speed of sound in the medium. In the SSElib, the density of air and the speed of sound in air are assumed as constant values with 1.204 kg/m^3 and 343 m/s , respectively. Subsequently, the sound power W of a sound source can be calculated from integrating the sound intensity over an enclosing surface around the sound source in a far field assumption:

$$W = \int_S \frac{p_{rms}^2}{\rho c} dS = \frac{1}{\rho c} \sum_j p_{rms,j}^2 S_j \quad (2)$$

where $p_{rms,j}$ is the sound pressure and S_j is a partial area of the enclosing surface where constant sound pressure $p_{rms,j}$ is assumed. From (2) one can see, that if two variables from (W, p, S) are known, the third can be calculated. For unsymmetrical sound sources, the sound pressure on the enclosing surface becomes rapidly complex, hence only two special cases are considered in the SSElib: a spherical propagation and a one dimensional propagation. For both one finds:

$$W = \frac{1}{\rho c} p_{rms}^2 S \quad (3)$$

where S represents the surface through which the sound propagates and p_{rms} is the effective sound pressure at S .

If a point source is located in the free room, S would be $4\pi r^2$ where r is the radial distance from the point source. If a rigid surface is located below the point source, the sound will only propagate to a spherical half space with a surface of $2\pi r^2$, therefore S is only half of the free room situation. For a sound source standing on a rigid surface in front of a wall and for a sound source located in a corner, S would even be a fourth or an eighth compared to the free room situation. In other words, for a given sound power, the sound pressure could be significantly higher for different installation situations compared to the free room situation, due to the fact that the kinetic energy generated by the sound source has to be transported through a smaller surface (c.f. Table 1).

Contrary to the sound pressure, the sound power is a characteristic value of the sound source which is independent of the location or the symmetry of the sound source. Hence, once the sound power level is known, the sound pressure level can be calculated for known geometries.

Usually both the sound pressure level and the sound power level of a single frequency source, are given in log-

arithmic dB units:

$$L_p(\text{dB}) = 20 \log \left(\frac{p}{p_0} \right) \quad (4)$$

$$L_W(\text{dB}) = 10 \log \left(\frac{W}{W_0} \right) \quad (5)$$

where p_0 is the absolute threshold of hearing at $2 \times 10^{-5} \text{ Pa}$ and W_0 is 10^{-12} W .

Combining (3),(4),(5) and concerning that $W_0 = p_0^2/(\rho c)$, leads to a direct link between L_p and L_W :

$$L_p = L_W - 10 \log \left(\frac{S}{1 \text{ m}^2} \right) \quad (6)$$

As already noticed in (3), a change of the enclosed surface through which the sound from the sound source propagates leads to a reduction or an enhancement of the sound pressure level. From (6) follows for two different locations A and B :

$$L_{p,B} = L_{p,A} - 10 \log \left(\frac{S_B}{S_A} \right) \quad (7)$$

where S_A and S_B are two different enclosing surfaces with constant sound pressure level $L_{p,A}$ and $L_{p,B}$, respectively. If the enclosed surface increases e.g. if the distance between sound source and observer increases, the sound pressure level decreases. If the enclosed surfaces decreases e.g. at the inlet into a duct, the sound pressure level increases.

For a radial symmetric point source (6) leads to an equation where S can be described with the radial distance r directly:

$$L_p = L_W - 20 \log \left(\frac{r}{1 \text{ m}} \right) - 10 \log \left(\frac{4\pi p_0^2}{\rho c W_0} \right) \quad (8)$$

The last term is dominated by $10 \log(4\pi)$ and is usually approximated with 11 dB in various textbooks.




The difference of two sound pressure levels of a constant radial symmetrical point source changes with the distance. From (8) follows for two different locations A and B :

$$L_{p,B} = L_{p,A} - 20 \log \left(\frac{r_B}{r_A} \right) \quad (9)$$

where r_A and r_B are the different distances to the point source. Hence, doubling the distance leads to a reduction of 6 dB. Note that doubling the distance in (9) is equivalent to quadruple the surface of constant sound pressure.

As already discussed, p_{rms}^2 will be doubled if the sound source stands on a rigid floor and the same sound power level of the source is assumed. Hence, the sound pressure level will be enhanced by 3 dB. Similar considerations lead to an enhancement of 6 dB and 9 dB for a sound source standing on a rigid surface in front of a wall and a sound source in a corner, respectively. Therefore, simple rules as summarized in Table 1 hold for these scenarios.

Table 1. Noise enhancement for different installation situations

Situation		ΔL (dB)	ΔS
floor		+3	$S_{free}/2$
wall and floor		+6	$S_{free}/4$
corner		+9	$S_{free}/8$

2.2 Frequency analysis

Up to this point, only single frequency sources were considered. But both, sound pressure and power can be broken down into frequency bands as shown by Fourier over 200 years ago. It is common in acoustics to divide the frequency spectrum into frequency bands like the one-octave or the one-third-octave band. For an octave band the lower and upper cutoff frequencies (f_l and f_u) are defined as:

$$f_l = f_c / \sqrt{2}; f_u = \sqrt{2} f_c \quad (10)$$

where f_c is the center frequency of the band. From (10) follows the center frequency:

$$f_c = \sqrt{f_l f_u} \quad (11)$$

Furthermore, from (10) follows that the upper cut off frequency is always twice the lower cut off frequency $f_u = 2f_l$ and that the bandwidth is $\Delta f = \sqrt{2} f_c$. For the i -th frequency band, the center frequency follows:

$$f_{c,i} = 2^{(i-1)} f_{c,1} \quad (12)$$

where $f_{c,1}$ is the first center frequency (15.625 Hz in the one-octave band).

Considering frequency analysis, the frequency dependent sound pressure level $\mathbf{L_p}$ and the sound power $\mathbf{L_W}$ are generally described as vectors in the SSELlib:

$$\mathbf{L_p} = \begin{pmatrix} L_{p,1} \\ \vdots \\ L_{p,i} \\ \vdots \\ L_{p,n} \end{pmatrix}, \quad \mathbf{L_W} = \begin{pmatrix} L_{W,1} \\ \vdots \\ L_{W,i} \\ \vdots \\ L_{W,n} \end{pmatrix}$$

where each row corresponds to a frequency band with a center frequency $\mathbf{f_c} = (f_{c,1} \dots f_{c,i} \dots f_{c,n})^T$. The values of $\mathbf{f_c}$, as well as the number of rows n depend on the chosen frequency band. E.g. if the octave band is chosen, the center frequencies are $\mathbf{f_c} = (16, 31.5, 63, 125, 250, 500, 1e3, 2e3, 4e3, 8e3, 16e3)^T$ Hz and therefore $n=11$ (c.f. Table 4). The logarithmic sum of the sound pressure levels at different center frequencies gives then the total sound pressure level and the total sound power level, respectively:

$$L_{p,total}(\text{dB}) = 10 \log \left(\sum_{i=1}^n 10^{L_{p,i}/10} \right) \quad (13)$$

$$L_{W,total}(\text{dB}) = 10 \log \left(\sum_{i=1}^n 10^{L_{W,i}/10} \right) \quad (14)$$

The same addition rules (13) and (14) are valid if the sound pressure or sound power levels of independent sound sources have to be added to a total sound pressure or sound power level, respectively.

The relative loudness of sound that can be perceived by the human ear is usually calculated by weighting the instrument-measured sound levels with a frequency dependent curve or table. The most common used curve is the A-weighting curve which is defined in several national and international standards, like the (IEC 61672-1, 2003). This curve dampens the sound at low and high frequencies whereas the intermediate frequencies stay unfiltered or are slightly enhanced. Table 4 in the Appendix shows the weighting coefficients Δ_i used for A-weighting at different center frequencies in the one-octave band which are used in the SSELlib. If the one-octave band is chosen, the weighting coefficient vector is $\mathbf{\Delta_A} = (-56.7, -39.4, -26.2, -16.1, -8.6, -3.2, 0, 1.2, 1.0, -1.1, -6.6)^T$ dB.

The A-weighted total sound pressure level can then be calculated with:

$$\tilde{L}_{p,total}(\text{dBA}) = 10 \log \left(\sum_{i=1}^n 10^{(L_{p,i} + \Delta_{A,i})/10} \right) \quad (15)$$

$$\tilde{L}_{W,total}(\text{dBA}) = 10 \log \left(\sum_{i=1}^n 10^{(L_{W,i} + \Delta_{A,i})/10} \right) \quad (16)$$

It is common, that A-weighted sound levels are described with the unit dBA or dB(A).

One has to be aware if data is given in the unit dB or dB(A). However, with (15) one can always convert the units to each other.

2.3 Noise reduction

Noise reduction can be achieved with several active and passive methods. An active method could be to operate a machine in a silent operating point. Such points can be found by extending simulation models with the SSELlib. A passive method is the integration of sound absorbing materials or silencers. Currently, two main noise reduction methods are implemented in the SSELlib. The first one describes the reduction with frequency dependent differences on the sound pressure or power level. Such a description is often used for silencers in ducts. Manufacturers usually provide these data to their customers. Table 5 in the Appendix shows typical values for the differences in sound pressure level ΔL_s in a 60×60 cm rectangular silencer with a length of 50 cm. Using the vector notation one can easily describe the noise reduction with:

$$\mathbf{L_{p,out}} = \mathbf{L_{p,in}} + \mathbf{\Delta L_s} \quad (17)$$

where $\mathbf{L}_{p,out}$ is the frequency dependent sound pressure level at the outlet of the silencer, \mathbf{L}_{in} is the frequency dependent sound pressure level at the inlet of the silencer and $\Delta\mathbf{L}_s$ has to be taken from manufacturers data or text books. Note that according to (6) the reduction of \mathbf{L}_p in (17) goes hand in hand with a reduction of the sound power level \mathbf{L}_W as sound power is converted to heat in the absorber material.

The other noise reduction method implemented in the SSElib uses absorbing material mounted in the enclosure of a machine or on the enclosing walls of a room. As the assumed sound fields are diffuse and reverberant the noise in the enclosure can be described with an average sound pressure level inside the enclosure. For this, we have to introduce frequency dependent sound absorption coefficients α for given surfaces and center frequencies. The absorbed sound power is therefore given with:

$$W_{abs} = \frac{\bar{p}^2}{4\rho c} \sum_j \alpha_j S_j \quad (18)$$

where \bar{p} is the diffuse sound pressure in the enclosure, $\bar{p}^2/(4\rho c)$ represents the equivalent power per m^2 and S_j and α_j are partial surfaces and their corresponding absorption coefficients, respectively. The term $\sum_j \alpha_j S_j$ is often referred to as the equivalent area of an open window in an enclosure with no absorption.

In the steady-state, the absorbed sound power has to be equal to the sound power propagating from the sound source inside the enclosure. Hence, \mathbf{W}_{abs} is equal to \mathbf{W} . From (18) it is obvious that the sound pressure \bar{p} becomes smaller for higher absorption coefficients or higher surface areas. Using logarithmic notations one finds (see also (Möser, 2012)):

$$\mathbf{L}_{\bar{p}} = \mathbf{L}_W - 10 \log \left(\frac{\sum_j \alpha_j S_j}{1 m^2} \right) + 6 \text{ dB} \quad (19)$$

Dependent on the material and the geometry of the enclosure, a part of the absorbed sound power is converted to thermal energy whereas the other part remains sound and will be radiated from the outer surface of the enclosure. Similar to α , the transmission can be described by frequency dependent transmission coefficients τ :

$$\mathbf{W}_\tau = \mathbf{W} \circ \frac{\sum_j \tau_j S_j}{\sum_j S_j} \quad (20)$$

where \mathbf{W}_τ is the sound power propagating from the outer surface of the enclosure.

3 Implementation

3.1 Connectors













The acoustic port (●) in the SSElib consists of two variables namely, the frequency dependent sound power \mathbf{W} (flow variable) and the enclosing surface S at the ports location.

Although logarithmic units are common in sound calculations, decimal units were used to describe the sound power to circumvent problems if the flow direction is unknown. Since also negative logarithmic sound levels have still a positive sound power, the descriptions with flow variables was not possible without introducing a further variable. Therefore, and for the sake of simplicity decimal units are used in the SSElib to describe the sound power in the connectors. Currently, only the one-octave band is implemented in the SSElib, hence the sound power \mathbf{W} and the center frequencies \mathbf{f}_c are described as vectors with 11 rows according to the one-octave band (c.f. Table 4). The center frequency vector \mathbf{f}_c is located in the *Constants* package.

3.2 Components

Several components were realized in the SSElib. Table 2 gives an overview of the components and links them to the equations and tables used in this work.





Table 2. Components of the SSElib

Name		Reference
AcousticSource		-
AcousticSink		-
Add		(13),(14)
Converter_dBA		(15),(16)
InstallationSituationFloor		Table 1
InstallationSituationWall		Table 1
InstallationSituationCorner		Table 1
RadialDistance		(8),(9)
OpenWindow		(6)
Silencer		(17)
Enclosure		(18),(19)
EnclosureWithTransmission		(18),(19),(20)


3.3 Sensors

In order to observe the sound pressure and sound power levels at different locations inside the models, four sensors are provided in the library which can be connected in between an acoustic connection. For both, the sound power and the sound pressure sensors, an unfiltered version and an A-weighted version exist. Table 3 shows the four sensors. Please note that we have used dB instead of dB(A) or dBA as unit for the A-weighted variables. The reason is that dB(A) or dBA aren't valid units in Modelica.

Table 3. Sensors of the SSElib

Name		Reference
SoundPressureSensor		(4),(6),(13)
SoundPressureSensor(dBA)		(4),(6),(15)
SoundPowerSensor		(5),(6),(14)
SoundPowerSensor(dBA)		(5),(6),(16)

3.4 Sound Source Extension

The main task of the SSElib is to provide a model to extend existing models with acoustic characteristics. This model is the *AcousticExtensionOneOctave* model  located in the *SoundSourceExtension* package. It should be used in the following way:

- Create a model which represents your new acoustic component
- Extend your new model with the sound source extension from the SSElib
- Extend your new model with your non-acoustic component e.g. a fluid pump
- Use your new model in the simulations instead of the old model and modify the acoustic parameters to match your components sound characteristic. Don't forget to connect your new model to at least one acoustic sink.

A new model for an extended fluid pump was implemented in the following way using the *PrescribedPump* model from the MSL (*Modelica.Fluid.Machines.PrescribedPump*):

```

model Example_AcousticPump
extends SSElib.Extension.\
  AcousticExtensionOneOctave;
extends \
  Modelica.Fluid.Machines.PrescribedPump;
end Example_AcousticPump;

```

The extended pump can be found in the *TestersAndExamples* package. Figure 1 shows the graphical representation of the pump from the MSL extended with a sound source.

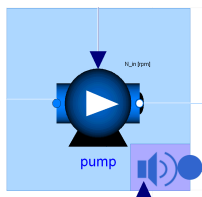


Figure 1. Fluid pump from the MSL extended with a sound source.

Contrary to the connections of the original pump model, the new model has now an additional real input and a sound source connector as depicted in the lower right corner. The additional input is simply called "soundInput" and can be used to influence the sound source levels. In the case of the pump, the rotational speed of the pump would be a reasonable input variable, due to the fact that the acoustic characteristic of the pump is expected to change significantly with the pump speed.

Figure 2 shows artificial sound data for a pump. We assume that sound power level at the rotational speeds $n = 30, 50, 70$ and 90 Hz have been measured and should be used for the simulations. They are depicted as bar plots in Fig. 2. In order to estimate the correct sound pressure levels at different rotational speeds, one could use the measured data directly in Modelica as table data and calculate the corresponding sound power levels with the aid of interpolation functions. As this method would lead to significant loss of time, we propose to use polynomial functions to describe the sound power data. The surface plot in Fig. 2 shows a polynomial function which was fitted to the data using polynomial functions from the Numerical Python library NumPy (Oliphant, 2006).

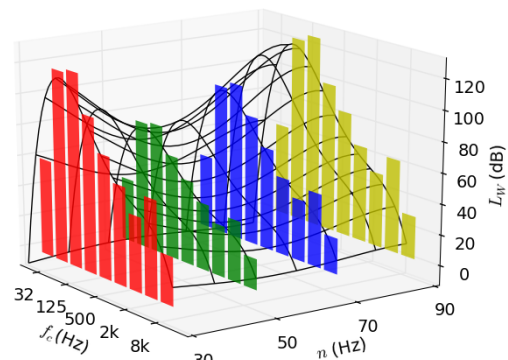


Figure 2. Sound power L_W dependency on the rotational speed of the pump n and the frequency bands around the center frequency f_c . The surface around the bar plots represents a polynomial fit.

To use polynomial functions, polynomial coefficients have to be passed to the extended model as a parameter. In general these coefficients c should have the form:

$$L_W(i, n) = \sum_{k,l} c_{k,l} i^k n^l \quad (21)$$

where i is the frequency band of the octave band (c.f. Table 4) and n is the rotational speed of the pump. Note that we used the integer of the frequency band instead the center frequency on purpose, as it is easier to fit the polynomial functions into data with linear distributed nodes. Currently, the SSElib only supports polynomials with $k=l=3$ or $k=l=4$.

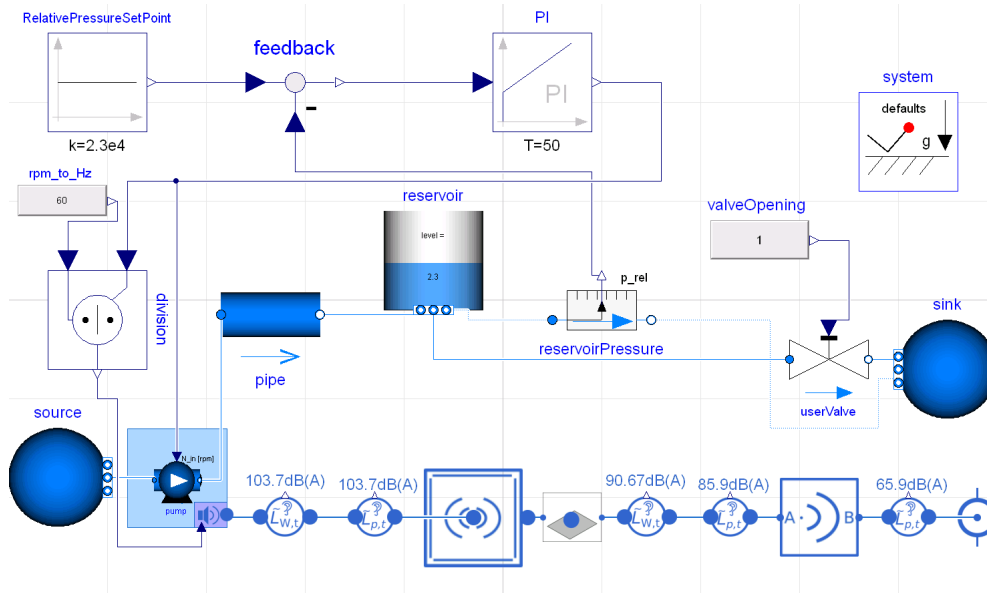


Figure 3. System sketch of the SSELlib example *SSELlib.TestersAndExamples.Example*.

3.5 Testers and Examples

Each component of the library has a testing model located in the *Testers and Examples* package. Furthermore, there is one extended component (*Example_AcousticPump*, Fig. 1) and one example (*Example*, Fig. 3) located inside this package.

The example was derived from the *PumpingSystem* example of the MSL, originally written by Francesco Casella.

Contrary to the original example the current pump can operate at different pump speeds and is controlled by a not very well designed PI-controller. Furthermore, the pump was extended with an acoustic sound source (c.f. section 3.4). A water pump which is sound-optimized for an operation around 50 Hz or 3000 rpm was assumed and polynomial coefficients were used to represent this behavior (c.f. Fig. 2).

The A-weighted sound pressure level close to the machine (1m distance) can be observed with *SensorMachine.L_p_total_dBA* and the speed of the pump can be observed with *pumps.N_in* and *pumps.soundInput* in rpm and Hz, respectively (Fig. 4). Starting from the sound-optimized speed of 3000 rpm or 50 Hz, the pump speed decreases to around 2260 rpm or 38 Hz in the steady state after around 1200 s. Simultaneously, the sound pressure level increases from 88 dB(A) to 103 dB(A) (red line in Fig. 4).

From an engineers point of view several options exists to lower the noise of the pump. One option could be, that the sound pressure measured at the machine or simulated on a computer is used to optimize the control strategy of the system. Another option could be to install sound absorption measures. The effect of the latter measure can be estimated by adding components from the SSELlib.

In this example the second option is real-

ized. The pump was covered with an enclosure (*EnclosureWithTransmission*) which stands on a floor without walls nearby (*InstallationSituationFloor*). An observer located 10 m away from the pump (*RadialDistance*) finally hears the pump with 66 dB(A) in the steady state (c.f. *SensorObserver.L_p_total_dBA* in Fig. 4).

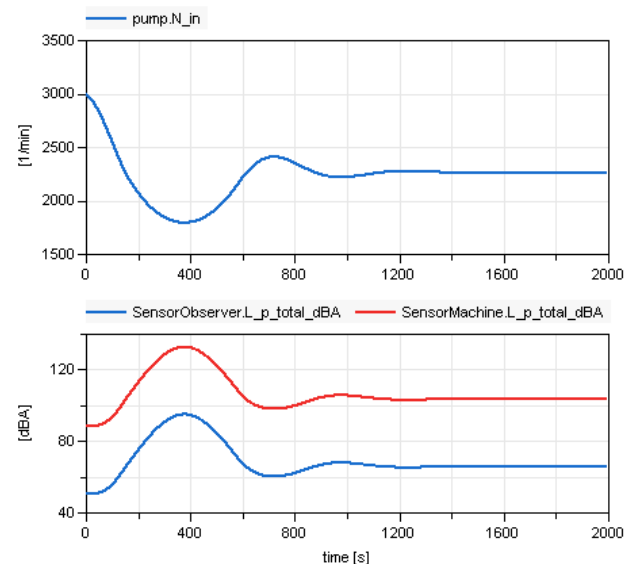


Figure 4. Time dependent pump speed (top) and time dependent total sound pressure level $L_{p,total}$ in dBA at different locations.

3.6 Heat pump example

Figure 5 shows a further example, where an air source heat pump, realized with the aid of the TIL library (TLK-Thermo GmbH) and self-written components, was extended with sound sources. Please note, that this example is not included in the SSELlib.

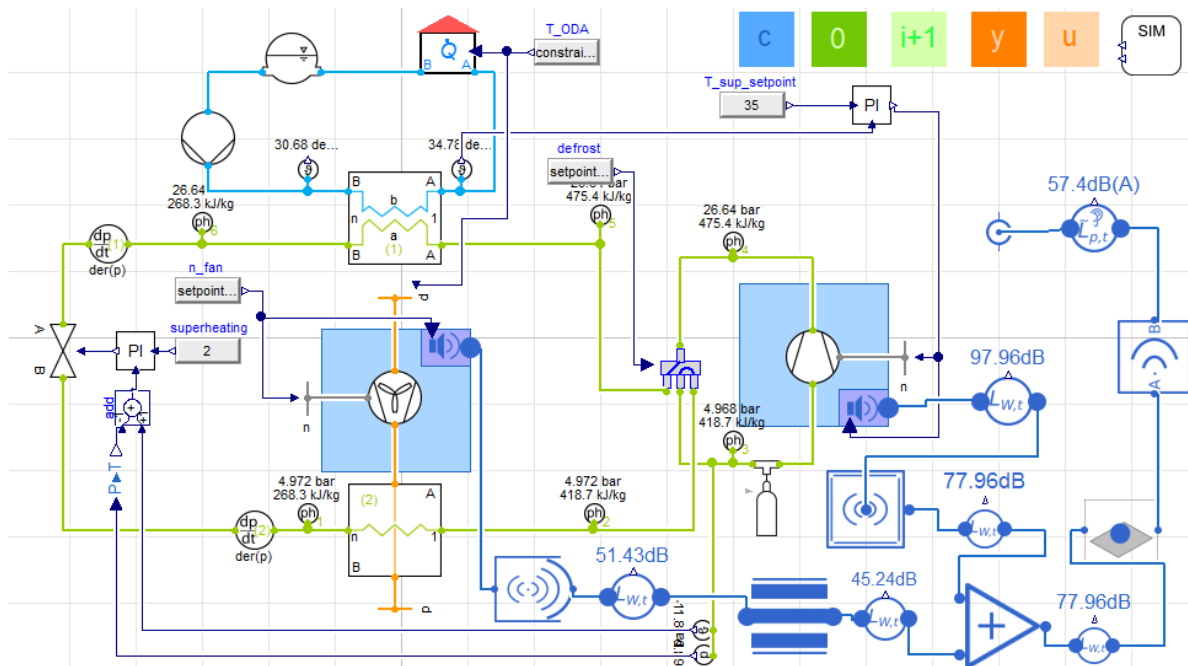


Figure 5. Air source heat pump model with an additional characteristic for the compressor and the fan. The components of the SSELlib are mainly located at the lower right corner of the figure.

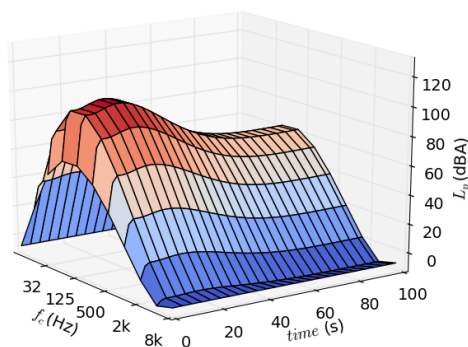


Figure 6. Transient behavior of the heat pump in the one-octave band.

The fan is housed in a casing with a $0.6\text{ m} \times 0.6\text{ m}$ opening, which is represented by the *OpenWindow* component. At the outlet of the casing a silencer (*Silencer*) is connected. The compressor is inside an casing without any free opening (*EnclosureWithTransmission*). The transmission coefficient to the outside for all frequencies is assumed with $\tau=0.01$. Both sound sources are located on a floor (*InstallationFloor*) and an observer (*SoundPressureSensor_dBA*) is 10 m away (*RadialDistance*). The total sound pressure level that the observer hears is about 57.4 dB(A) at the operating point.

Figure 6 shows the transient behavior of the sound pressure level at the observer location. After starting the heat

pump at $time = 0$, the controller needs about 100 s to reach a steady-state condition. During this time, the sound power level reaches a significant maximum around 20 s. In a sound optimized heat pump such peaks in the sound pressure level could be easily avoided if the controller considers this peaks at the start of the machine.

4 Outlook

The presented work shows the first version of a library which should continuously grow in the years to come. The following features should be implemented soon:

- Frequency resolutions in the one-third octave band
- Additional description methods for the acoustic behaviour besides the description with polynomials and constants.
- Validation of the models with measurement on an air source heat pump in AITs acoustic lab.

Acknowledgment

The Austrian Research Promotion Agency (FFG) is gratefully acknowledged for funding this work within the SilentAirHP project under Grant No. 848891. Furthermore, we thank TLK-Thermo for technical advice.

References

- M. J. Crocker, editor. *Handbook of Noise and Vibration Control*. Wiley, New Jersey, 2007.
- IEC 61672-1. *Electroacoustics - Sound level meters - Part 1: Specifications (International Electrotechnical Commission Standard No. 61672-1)*. Online on:

<https://webstore.iec.ch/publication/5708>, last visited: 2017-03-07, 2003.

M. Möser. *Technische Akustik*. Springer-Verlag, Berlin Heidelberg, 9th edition, 2012. doi:10.1007/978-3-642-30933-5.

Travis E. Oliphant. *Guide to NumPy*. Provo, UT, March 2006. URL <http://www.tramy.us/>.

TLK-Thermo GmbH. TIL suite and TIL media: Commercial library for steady-state and transient simulation of thermodynamic systems such as heat pump, refrigeration, a/c, cooling and Rankine systems. Online on: <https://www.tlk-thermo.com/>, last visited: 2017-01-22.

Appendix

Table 4. Frequency band index i , center frequencies f_c and A-weighting coefficients $\Delta_{A,i}$ for the one-octave band. The coefficients were taken from (IEC 61672-1, 2003)

i	f_c (Hz)	$\Delta_{A,i}$ (dB)
1	16	-56.7
2	31.5	-39.4
3	63	-26.2
4	125	-16.1
5	250	-8.6
6	500	-3.2
7	1 000	0
8	2 000	1.2
9	4 000	1
10	8 000	-1.1
11	16 000	-6.6

Table 5. Typical values for the differences in sound pressure level ΔL_s in a 60×60 cm rectangular silencer with a length of 50 cm taken from the data sheet of a commercial available silencer.

center frequency f_c (Hz)	ΔL_s (dB)
63	-2
125	-4
250	-10
500	-18
1000	-25
2000	-24
4000	-17
8000	-11

Table 6. Typical values for the sound absorption coefficient α of 5 cm thick melamine foam taken from the data sheet of a commercial available noise insulation material

center frequency f_c (Hz)	α (-)
125	0.13
250	0.41
500	0.6
1000	0.9
2000	0.85
4000	0.93

Towards Medical Cyber-Physical Systems: Modelica and FMI based Online Parameter Identification of the Cardiovascular System

Jonas Gesenhues¹ Marc Hein² Maike Ketelhut¹ Thivaharan Albin¹ Dirk Abel¹

¹Institute of Automatic Control, RWTH Aachen University, Germany j.gesenhues@irt.rwth-aachen.de

²Department of Anesthesiology, RWTH Aachen University Hospital, Germany

Abstract

This paper presents a concept for online parameter identification intended to be used within cardiovascular research labs and hospitals of the future featuring a data network of medical sensors. It is based on iterative non-linear optimization using a moving horizon scheme and object-oriented *Modelica* models. Special FMUs have been developed to interface the optimization module and the sensor hardware. The concept is demonstrated on an exemplary application of identifying the parameters of a model for the systemic circulation. Unlike classical online parameter identification methods, this concept allows for quickly implementing changes of the underlying model. **Keywords:** *Online Parameter Identification, Moving Horizon, FMI, ModeliChart, JModelica.org, CasADi, Cardiovascular, Medical*

1 Introduction

Throughout many countries around the globe, public health care systems are being faced by the ongoing trend of increased demand for health care services. On the one hand, this is due to the consequences of demographic changes towards an aging population. On the other hand, scientific progress allows for increased treatment possibilities (European Commission, 2016). At the same time, public hospitals, a major pillar within the health care systems, are faced by a lack of qualified health care personal. Supporting health care personal in public hospitals by smart technology might provide an essential component to meet those challenges. In this regard, ongoing trends such as digitalization of information, large scale data agglomeration (*'Big Data'*), interconnection of devices (*'IoT'*) and smart algorithms that allow for e.g. automated monitoring of a patient's status and early recognizing and possibly automatically resolving critical conditions can be expected to find their way into hospitals in the future and have the potential to improve the outcome of patients.

Within this context, the research focus of our interdisciplinary group consisting of engineers and physicians is on improving the therapy of terminal heart failure, the most prevalent cause of death in the western world (Nichols et al., 2012). Specifically, we are working on control

strategies for technological heart assist devices, such as blood pumps that are connected to the body to assist the heart (Ventricular Assist Devices). Here, mathematical models of the cardiovascular system are applied in many different ways, ranging from computer *'model in the loop'* simulations of new control strategies (e.g. Habigt et al. (2016); Ketelhut et al. (2017)) over driving test benches for *'hardware and software in the loop'* hardware tests (e.g. Misgeld et al. (2015)) to state estimation (e.g. Rüschen et al. (2016)) and model based control (Gesenhues et al., 2016).

Although much literature exists describing the observed behavior of the healthy body, few is known about the underlying mechanisms and how they are affected by diseases, drugs or the interaction with technical devices. Consequently, the adaption, refinement and creation of new models is an integral part within this field. Here, over the years the object-oriented modeling paradigm using *Modelica* has turned out invaluable for its flexibility for modifications and the concept of acausal formulation of components (Gesenhues et al., 2017) and has motivated the creation of libraries such as the *Physiolibrary* (Matejak et al., 2014) or our in-house developed library *HumanLib* (Brunberg et al., 2009).

Besides model structure, the identification of the contained model parameters is important. When it comes to biomedical dynamical systems such as the cardiovascular system, there is generally a great extent of variation considering parameters. First of all, parameters vary with countless individual characteristics of patients such as age, height, weight, gender, lifestyle etc. Second, the presence and extend of diseases directly affects the parameters. Finally, even in a specific single patient at a specific state, the parameters vary because the body possesses numerous physiological control mechanisms to adapt to external conditions such as temperature, exercise or even the current posture (standing upright or lying). Thus, the model parameters need to be considered time varying and can change within seconds. All in all, parameters identified from measured data represent a snapshot of an individual patient at a specific time.

For all of those reasons and having smart algorithms and features of hospitals in the future in mind, an au-

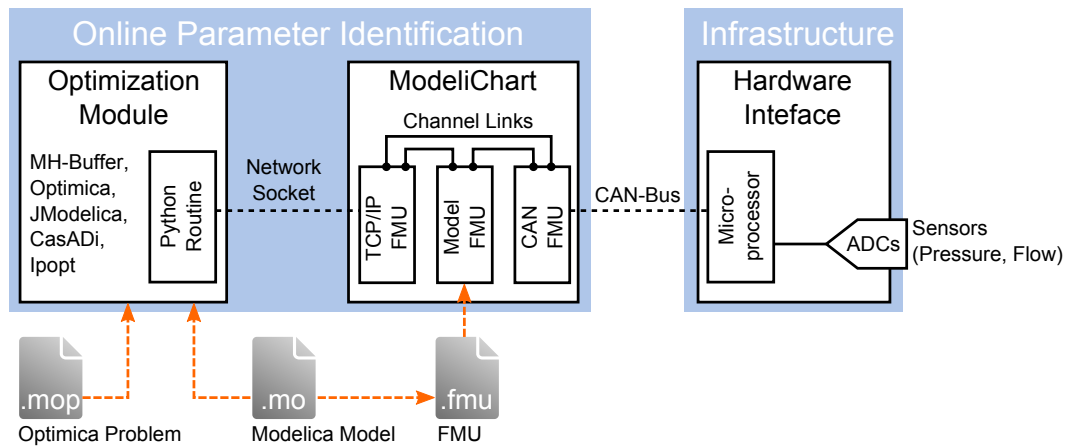


Figure 1. Architecture and components of the online identification concept. MH: Moving Horizon, CAN: Control Area Network, ADC: Analog-Digital Converter.

tomated online model parameter identification procedure that is capable to be included in a hospital's data network and which continuously identifies model parameters based on live patient data provides many benefits including diagnostic assistance to doctors (model parameters can be used to assess a patient's status), smart alarms that are raised when selected parameters exceed a certain threshold up to model based control of medical devices, for which an adequately parametrized model is an essential prerequisite. In-vivo animal trials are an integral element of our research. The infrastructure that we have developed to conduct such trials features a large number of sensors which are connected to a data network (currently we are using the Control Area Network (CAN) bus). This infrastructure bears resemblance to the possible infrastructure of future hospitals. Thus, our trials and infrastructure provide a test bed for the implementation of medical cyber-physical systems.

The current state of technology includes many classical online and offline parameter identification methods, which have been adapted and applied to virtually any physical domain. Specifically for the cardiovascular system, those include attempts using reformulation of model equations to allow for (recursive) least square techniques (Clark et al., 1980; Hann et al., 2006; Kosaka et al., 2002) and Bayesian approaches like the (extended) Kalman filter (Yu et al., 1998). Although it has been shown that the results yielded from classical approaches are valid and reliable, a major limitation consists in the fact that there is an enormous effort to reformulate the model into the specific form needed for the identification method. This generally includes manually rearranging equations and to transform the system by introducing new state variables and parameters (e.g. to resolve non-linearities). Shortly, classical methods might be satisfactory when the underlying model meets the requirements of the identification method and can be considered 'frozen' with the start of development as revisions to the model at a later time can be laborious and even impossible to implement.

As mentioned above, cardiovascular system models are subject to frequent changes. Thus, the applicability of classical methods is limited in this regard. Those limitations motivate new online identification procedures which do not require excessive reformulation efforts of the underlying model. Recently, we have started to consider non-linear optimization based identification using our *Modelica* models in combination with *Optimica*, *JModelica.org* and *CasADi* as a possible solution. A recent study focusing on the offline identification of patient specific parameters using those tools comes to the conclusion that patient specific parameter identification has the potential to be a promising component for patient assessment in the clinic (Moza et al., 2017).

The contribution of this paper is a concept that allows for the automated online parameter identification based on those ideas and tools which does not exhibit the described limitations of the previous state of the art and can be used within our animal trial infrastructure. The general idea is to repeatedly (re-)identify the current parameter values by solving a non linear optimization problem over a short time interval. The paper is organized as follows: first, the next section provides a general overview over the concept and its components and the typical setup work flow involved. Section 3 details the iteratively carried out optimization procedure. Afterwards, the concept is demonstrated by the exemplary application of identifying the parameters involved in a simple model of the systemic circulation (Section 4). Finally, the results are presented and a discussion on current limitations and further enhancements is given (Section 5).

2 Concept Overview and Work Flow

The components involved within the presented concept are summarized in Figure 1. The concept consists of our FMU-master *ModeliChart* (see Section 2.2 below) which serves as the central hub and graphical user interface and of the optimization module, which constantly calculates

current parameter values. It is further described in Section 3. The optimization module is implemented as a Python routine. Data exchange between those two components is realized through a TCP socket stream using a simple custom protocol. This design allows for either running the optimization module on the same machine that is running *ModeliChart* as well as running the optimization module on a different machine (possibly outside of the lab) connected to the local area network (LAN).

2.1 Interface FMUs

To realize the architecture depicted in Figure 1 two FMUs complying with the FMI 2.0 co-simulation standard have been developed to allow for data exchange between the individual components of the concept. Both interface FMUs have in common that they are fully configurable through the `modelDescription.xml` and additional configuration files provided as resources. So far, both FMUs only support the `Real` data type. An arbitrary number of input and output channels that appear as scalar FMU variables can be set. Input channels are intended for receiving data, output channels, which are marked with the attributes `causality="parameter"` and `variability="tunable"` are intended to send data. For each of both FMUs a convenient software tool has been developed to automatically generate the according `modelDescription.xml`, additional configuration files and the packed FMU.

The first FMU constitutes a TCP/IP based network socket interface used for the connection between the optimization module and *ModeliChart*. During the initialization of the FMU, a TCP server accepting connections on a configurable port is started waiting for a client (here the optimization module) to connect. At this point, only a single client is supported. On every execution of the `doStep(...)` method, the values of the output channel scalar variables are sent to the connected client using a simple custom protocol. Similarly, the `getReal(...)` function returns the latest value of the specified input channel scalar variable. The client is allowed to send values at any given time.

The second FMU allows for the interaction with the CAN bus of our infrastructure which distributes the sensor signals. This FMU uses the API provided by the manufacturer of the CAN interface hardware (PEAK-System Technik GmbH, Darmstadt, Germany). The CAN FMU listens to CAN messages of preconfigured message identifiers and returns the last received value whenever the corresponding `getReal(...)` function is called. Although not required in the here presented application, the CAN FMU also supports sending values to the CAN bus.

2.2 ModeliChart

ModeliChart is our self-developed freely available FMU host. The original motivation has been to provide a free and intuitive opportunity to assess and play with simulation models to physicians. However, the ease of use and

the hardware interaction capabilities through the interface FMUs described above have turned *ModeliChart* into a 'Swiss army knife' for all steps during rapid control prototyping cycles. Based on the .NET framework (Microsoft, Redmond, WA, US), it provides a simple intuitive graphical user interface. *ModeliChart* supports FMUs complying with the FMI 2.0 co-simulation standard. The main intended use case is real time operation by periodically calling the `doStep(...)` method of all FMUs after a configurable time interval. So called 'channel links' allow individual FMUs to be connected: Internally, for each channel link the `SetReal(...)` method of the receiving FMU is called at each time step. More details on *ModeliChart* can be found in (Gesenhues et al., 2017).

Within the here presented application, three FMUs are used. The CAN FMU is used to fetch the measurements of the sensors of interest from the CAN bus. Through channel links, the measurement data is handed to the TCP/IP FMU which in turn sends the measurement data to the optimization module. In this regard *ModeliChart* serves as a CAN to TCP/IP bridge. After new identified parameters are available from the optimization module, they are sent to *ModeliChart* through the TCP/IP FMU. The third FMU contains the model under investigation. Through channel links the current parameter values are set to the model FMU. In combination with the measurement data from the CAN FMU used as input into the model FMU, it is possible to compare chosen simulated signals with corresponding measured signals. Each channel can be plotted allowing to observe trends of parameter values and to visually assess the validity of the results by comparing the simulated and measured signals.

2.3 Typical Setup Work Flow

Typically, the setup starts with a *Modelica* model that contains the parameters of interest. The model should contain variables that correspond to measured signals. Obviously it is required that the parameters are adequately related to the variables representing the measured signals (i.e. observability, correlation etc.). Next, an *Optimica* file is created containing the optimization problem. The `optimization` class should extend from the *Modelica* model. In simple cases, it can be sufficient to just mark the parameters to be identified as 'free' when a quadratic penalty cost function is to be used. Nevertheless it is also possible to define custom cost functions. For consistency, certain variable expressions for optimization parameters such as final time or limits on allowed parameter values can be used which are later overwritten when the *Optimica* file is loaded in the optimization module. An exemplary *Optimica* file can be seen in Listing 2. Both files are placed in a folder accessible for the optimization module. Next, some adaption of the *Python* routine is necessary to match the measurement data to the corresponding model variables and to set optimization parameter settings (see Section 3). Besides, an FMU of the *Modelica* model to be used within *ModeliChart* is created and if necessary the

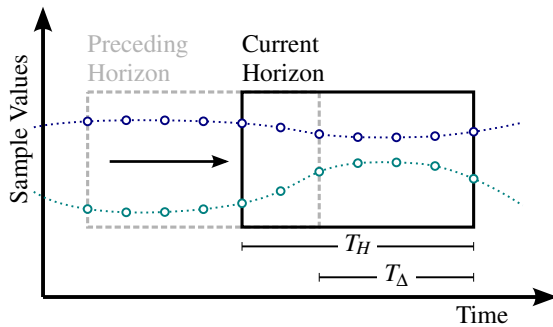


Figure 2. Illustration of the moving horizon scheme.

configuration of the interface FMUs is adapted. Finally, a *ModeliChart* setup (loading the FMUs and creating the channel links) is created and optionally saved for later use. As soon as the setup has been loaded in *ModeliChart*, the optimization module can be started.

3 Optimization Module

The online parameter identification is realized through continuously iteratively solving the optimization problem defined within the *Optimica* problem using the measurement data samples within a most recent finite time frame (horizon). Using a moving horizon scheme (Figure 2), as soon as enough samples for a new horizon spanning the time T_H and enough time since the preceding horizon T_Δ has passed, a new optimization job on the current horizon measurement data is dispatched.

The optimization module has been implemented as a Python routine. It uses the modules provided by *JModelica.org* (version 1.17) and in particular the integrated *CasADi* based optimization tool chain (Åkesson et al., 2010; Andersson et al., 2011). *CasADi* is a nonlinear optimization framework that is capable to automatically discretize the optimization problem using a collocation scheme and to calculate the necessary derivatives through algorithmic differentiation. The tool chain automatically transforms the formulated *Optimica* problem to be solved by a non linear optimization solver. Here, *Ipopt* has been used (Wächter and Biegler, 2005).

Figure 3 provides an overview of the routine. After the optimization module has been started, the *Modelica* and *Optimica* files are loaded. The variable expressions in the *Optimica* file (see Listing 2) are replaced by the configured values within the routine, i.e. `%FINAL_TIME%` is set to the value T_H . Using the according modules, the *Modelica* model is simulated with artificial input signals over the time frame T_H to provide initial trajectories of all variables. Afterwards, the optimization problem is compiled and discretized using the `prepare_optimization(...)` function of the `transfer_optimization_problem` module. The so prepared discretized problem will be used for each of the following optimizations. Since this compilation process takes significantly longer than the actual solution of the

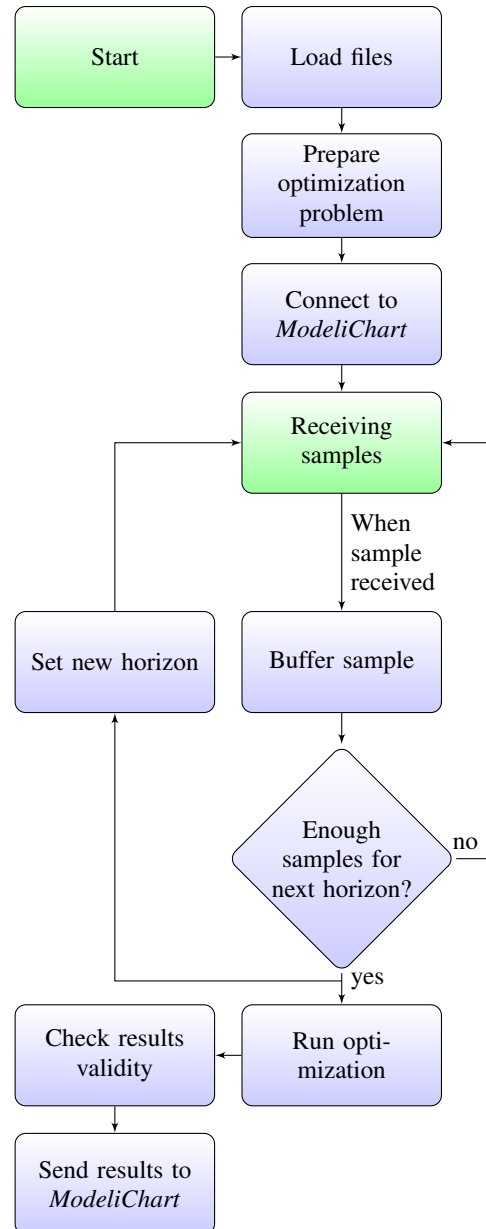


Figure 3. Flow diagram of the optimization module routine.

optimization problem, reusing the prepared discretization saves a lot of computation time. Afterwards, as soon the connection to *ModeliChart* is established, incoming samples are awaited and buffered. As soon as a new horizon is collected according to the described moving horizon scheme, the samples within the horizon are set as the new external data. Furthermore, the initial trajectories for the solution of the optimization problem are set to the solution trajectories of the preceding optimization. Afterwards the solver is started.

For a number of reasons depending on the application (some examples will be shown in Section 4), the solution obtained from the solver might be invalid or the solver might even fail to find a solution within a reasonable time. At this point, it is just checked whether the found parameters are within defined limits to decide on the validity of

Listing 1. *Modelica* model for the identification of the TEW parameters. The associated *Modelica* library 'HumanLib' can be found as online supplement.

```

model Ident3ElemWK
import HumanLib.Basics.*
import HumanLib.Vessels.*
parameter Real param_Z=0.1;
parameter Real param_R=1;
parameter Real param_C=1;
Resistance Z(R=param_Z);
Compliance C(V_0=0, C=param_C,
  V(start=100, fixed=false));
Resistance R(R=param_R);
Sources.PressureSource_Variable P_Ao;
Sources.PressureSource_Variable P_CV;
input Real AoP;
input Real CVP;
Sensors.FlowSensor Q_Ao;
equation
connect(Z.cnStreamOut, R.cnStreamIn);
connect(C.cnBloodStream, R.cnStreamIn);
connect(R.cnStreamOut, P_CVP.cnBloodStream
);
connect(P_Ao.P, AoP);
connect(P_CV.P, CVP);
connect(Z.cnStreamIn, Q_Ao.cnStreamOut);
connect(P_Ao.cnBloodStream,
  Q_Ao.cnStreamIn);
end Ident3ElemWK;

```

Listing 2. *Optimica* optimization problem for the identification of the TEW parameters.

```

optimization OptimizeWKParams(startTime=0,
  finalTime=%FINAL_TIME%)
extends Ident3ElemWK(
  param_Z(free=true,min=%MIN_Z,max=%MAX_Z%),
  param_R(free=true,min=%MIN_R,max=%MAX_R%),
  param_C(free=true,min=%MIN_C,max=%MAX_C%)
);
end OptimizeWKParams;

```

the results but in the future it might make sense to evaluate other criteria like residuals, solution time etc. Therefore, an additional parameter is reported back indicating whether the result is considered valid. This provides feedback for the user on which it can be decided to just ignore 'sporadically' invalid results or to investigate the reason.

4 Exemplary Application: Identification of the Systemic Circulation

The concept is demonstrated on the simple but relevant in practice use case of identifying the parameters involved in modeling the flow dynamics of the systemic circulation. The systemic circulation refers to all blood vessels (arteries, capillaries, veins) between the outlet of the left (side of the) heart, which pumps the blood into the systemic circulation and the right heart, which pumps blood into the pulmonary (lung) circulation (Figure 4). The vessels

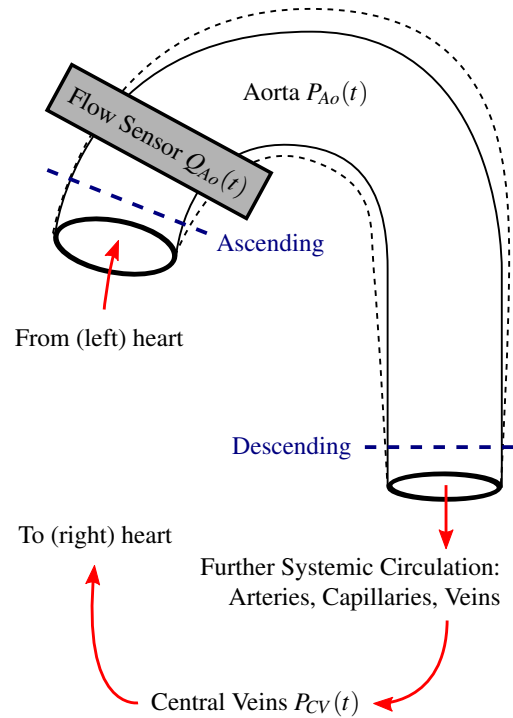


Figure 4. Schematic overview of the systemic circulation.

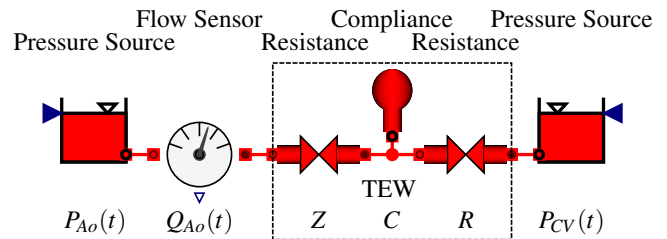


Figure 5. Graphical representation of Listing 1: the three-element-windkessel (TEW) in model combination with additional components for the parameter identification process.

within the systemic circulation start with the arteries and branch up more and more into smaller vessels (ultimately into so called capillaries) running through all parts of the body (muscles, organs) except the lungs. The capillaries end up in the veins, which in turn ultimately end in the big central veins and finally in the right heart.

The large arteries, most importantly the aorta are elastic and have the ability to distend with raising blood pressure and recoil with falling blood pressure (indicated through the dashed line in Figure 4). This leads to a damping of the amplitude of the pulsating blood pressure wave coming from the heart, an effect that is commonly referred to as the physiological windkessel effect. A very simple model for the systemic circulation is the three-element-windkessel (TEW) model (Westerhof et al., 2009). It consists of two hydraulic resistances ($\Delta P = R \cdot Q$, with ΔP being the blood pressure difference across the element, Q the blood flow and R the resistance parameter) and a compliance element ($V = C \cdot P$ with V being the current blood volume inside the element, P the current blood pressure inside the element and C being the compliance parameter).

ter). For convenience, the elements and parameters will be referred to as Z, R (resistances) and C (compliance). A graphical representation of the model can be seen in Figure 5.

During a typical in-vivo animal trial on an anesthetized pig, the chest is opened and the heart and the aorta uncovered. This allows for the placement of various sensors. For this application two invasive pressure sensors are used that are placed inside the Aorta ($P_{Ao}(t)$) and inside one of the central veins ($P_{CV}(t)$). Furthermore, an ultrasonic flow sensor is placed at the beginning of the Aorta (Q_{Ao} , see Figure 4). Listing 1 contains the complete *Modelica* model based on components from our library '*HumanLib*' (Brunberg et al., 2009). The blood connector used in Listing 1 consists of the potential variable blood pressure (traditionally denoted in mmHg where $1 \text{ mmHg} = 133.3 \text{ Pa}$, referred to atmospheric pressure) and the flow variable blood flow in ml/sec.

Listing 2 contains the according optimization problem. In this example the parameters to be identified are marked free. The optimization problem consists in finding those parameter values that minimize the quadratic difference between the measured signals and the 'simulated' signals. For this, the `ExternalData` class of the `pyjmi.optimization.casadi_collocation` module has been used. It allows to optimize for all three sensor signals at the same time without the need for deciding which signals are considered as input or output signals. The extend of individual signal differences can be weighted against each other. Here, the parameters given in Table 1 have been used to roughly normalize the signals.

To influence the parameters all sorts of experiments can be performed during an animal trial. Here, we consider the constriction of the aorta using a surgical band at two different positions: at the ascending part right at the beginning of the aorta and at the descending part of the aorta as indicated by the dashed lines in Figure 4. It is important to note that the pressure sensor measuring P_{Ao} measures the aortic pressure right behind the constriction of the ascending position but way before the descending position. Each of those constriction positions should have a different impact on the model parameters.

5 Results and Discussion

For the results presented in this section, already available raw data recorded during animal experiments conducted on anesthetized pigs (approved by local animal care authorities) has been used. There were no animal trials conducted for this study. Accordingly, the infrastructure has been emulated to generate the results presented in this section. The raw data contained the sampled values of the sensors at a sample rate of 1 kHz. The settings that have been used are summarized in Table 1.

Two different experiments are investigated. The first experiment is a short constriction of about 20 seconds

Table 1. Settings that have been used to obtain the presented results. All other settings have been left at their default values.

Name	Value
General:	
Horizon length $T_H, \%FINAL_TIME\%$	1.5 sec
Time between horizons T_Δ	0.8 sec
Validity criteria:	
Maximal value $\%MAX_R\%, \%MAX_C\%$	3.5
Maximal value $\%MAX_Z\%$	1
Minimal value $\%MIN_R\%, \%MIN_C\%$	0.1
Minimal value $\%MIN_Z\%$	0.001
Quadratic penalty weight factors:	
For P_{Ao}	10
For P_{CV}	20
For Q_{Ao}	1
Optimization settings:	
Number of collocation elements	23
Max. <i>Ipopt</i> iterations	300

at the ascending position (Figure 6). The P_{CV} signal is not shown in the plots since it remains almost constant at around 15 mmHg during the experiments. The pressure P_{Ao} is measured behind the ascending occlusion position. Hence, constricting the aorta at the ascending position limits the blood flow but hardly affects the properties of the systemic circulation. It can be seen that the parameters change only slightly during the constriction and return to their initial values some time after releasing the constriction. The immediate parameter value changes (most notably the sudden decrease of Z) can be explained by nonlinearities of the real system which become apparent when the blood flow is significantly reduced. The slow changes of parameters after the constriction are due to reactions of the bodies regulation mechanisms; trough muscle cells within the wall of some of the arteries the cross section area of the vessel and thus the resistance of the vessel can be controlled by the body. The increase in Z can be explained by the aim of the body to increase the pressure in the aorta (the so called baroreceptor reflex). Similar, the reduced supply of oxygen (hypoxia) leads to a widening of the blood vessels to allow for increased blood flow and results in a reduction of R . After the release, the regulatory mechanisms slowly revert the parameters back to the original values.

For the second experiment, a constriction at the descending position for several minutes is performed (Figure 7, release not shown in the figure). When the aorta is constricted at the descending position, the overall resistance of the systemic circulation is drastically increased which is reflected in a significant increase of R . However, due to impaired draining a significant expansion of the aorta results in a reduction of Z due to the increase of the cross section area of the aorta. For the same non-linearity reasons as in the first experiment, the expanded aorta exhibits a reduce elastance. Hence, the value of the compliance

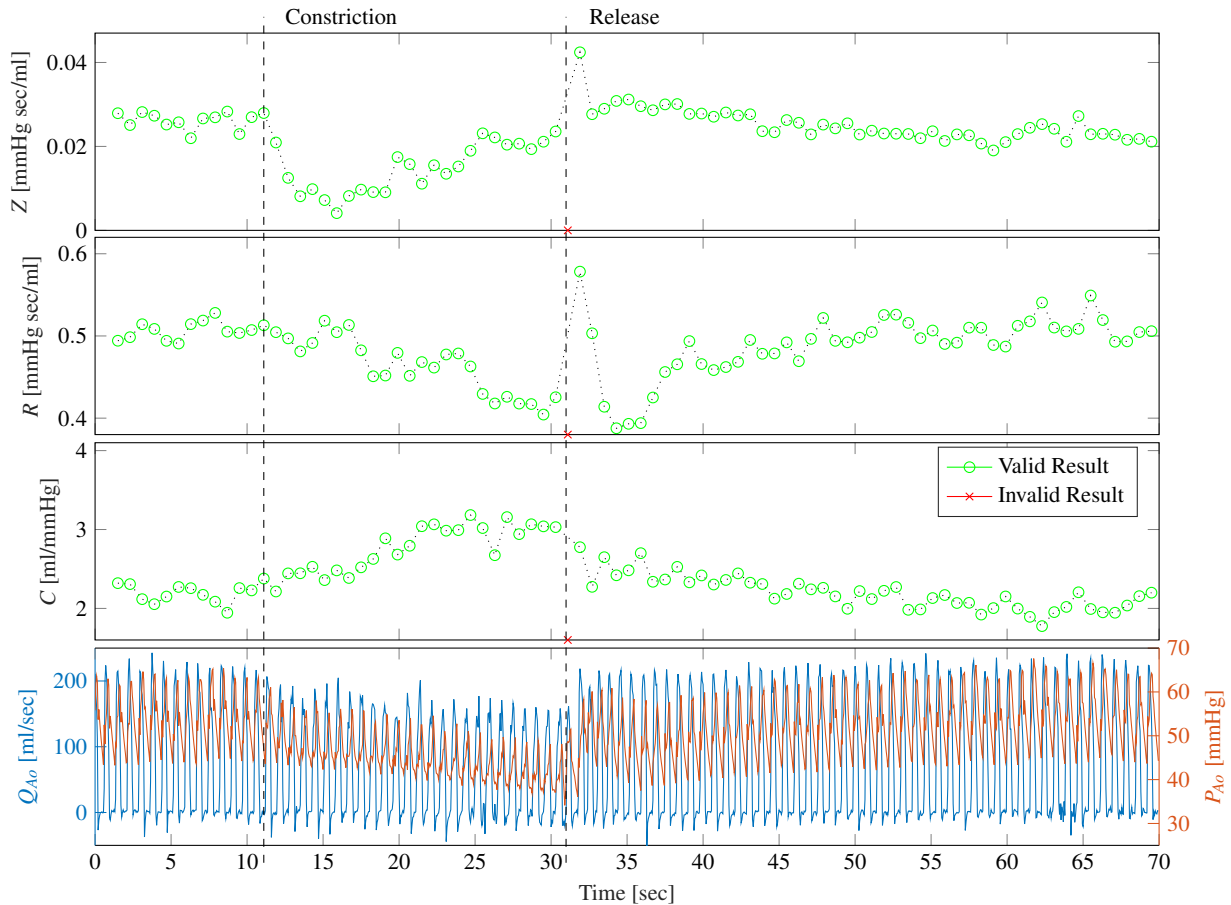


Figure 6. Short constriction of the aorta at the ascending position with subsequent release after about 20 seconds.

C decreases. Again, although small compared to the effects of the constriction, slight changes of the parameters can be observed after the constriction due to the regulatory mechanisms described above.

As seen in both experiments, the assumed proportionality between pressure and volume to model the compliance C is only valid within a limited range that has been exceeded during the experiments. Based on this findings it can be considered to revise the model accordingly. Here, the major advantage of the here presented concept consists in the fact that such model adaption can be implemented quickly.

The limitations considering what can be modeled are mostly determined by the features yet supported by *JModelica.org*, *CasADi* and *Ipop*. A major limitation is the requirement of the model equations to be twice continuously differentiable (Wächter and Biegler, 2005) for each optimization variable. This prohibits the use of switching components that commonly include `if...else` statements. In our applications, this affects models containing heart valves. To work around this limitation continuous approximations have been used (Gesenhues et al., 2016, 2017).

On a standard personal computer, the average computation time for a valid result was 0.45 sec for the first experiment and 0.36 sec for the second experiment. Since the

solution of the preceding optimization are used as the initial trajectories for the solver, the solution converges faster if there is less change of the parameters between horizons. Currently, a limitation of the current design of the routine of the optimization module is the requirement of the preceding optimization being finished before the next optimization can be started. Consequently, the time between two horizons T_Δ needs to be chosen sufficiently high to avoid additional delays from waiting for the preceding optimization to finish. Settings that affect the solution time include the number of collocation elements and the number of collocation points within the discretization of the optimization problem.

For verification purposes and to investigate the impact of the length of the horizon T_H , a test case has been constructed based on data obtained by a simulation of the model. Here, the exact parameter values that should result out of the identification are known. All three parameters have been varied during the simulation to evaluate the dynamical effects of the identification procedure. Using a sinus signal as input for $P_{Ao}(t)$ and a constant signal for $P_{CV}(t)$, the resulting $Q_{Ao}(t)$ was obtained. The so artificially created signals were used to emulate the sensor signals. The results of this test case are contained in Figure 8. As it can be seen, a smaller value for the horizon length T_H results in a faster response to changing parameters. On

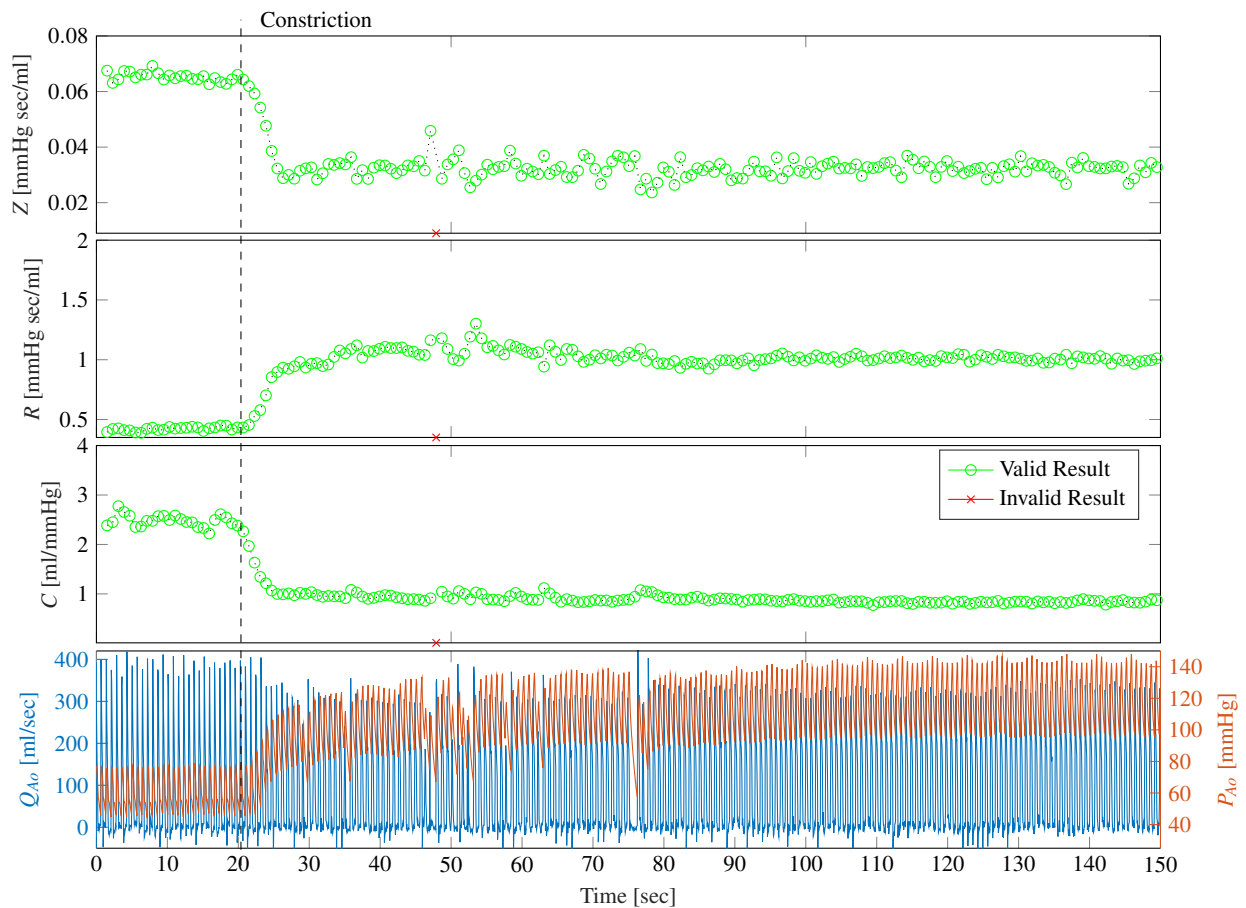


Figure 7. Long constriction at the descending position of the aorta.

the other hand, fluctuations are more damped for bigger values of T_H . Especially in the presence of noise, choosing a bigger value for T_H might be preferable. However, significant parameter changes within the horizon lead to invalid results (the outliers for $T_H = 5$ sec in Figure 8).

In the future, it is planned to further evaluate the concept considering other aspects of the cardiovascular system. Besides, further improvements to the optimization module will be made to improve the robustness and the performance. This includes canceling optimizations that do not converge within a given time. Similarly, it will be considered to adapt the time between horizons T_Δ depending on the necessary solution time. Another interesting idea that has come up is the synchronization of the horizons to heartbeats. Besides, improvements will be made to the setup work flow aiming at eliminating the need to adapt the Python code for new setups. Furthermore, we will be looking into changing the settings of within the optimization module through the *ModeliChart* interface without the need to stop and restart the optimization module.

Concluding, the concept that has been integrated into our lab infrastructure presents a valuable addition for our research on the cardiovascular system and has the potential to be used as a clinical tool in the future.

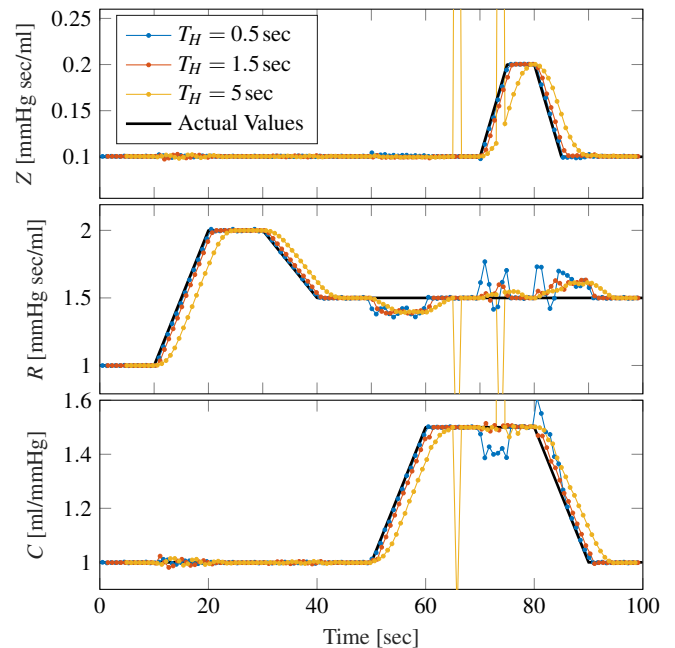


Figure 8. Verification trough identification of known parameter values for different horizon lengths T_H . The outliers in the $T_H = 5$ sec test are invalid results.

Acknowledgments

This work was supported by the German Research Foundation (DFG) within the Smart Life Support 2.0 PathoMod project (PAK 183-2).

References

- J. Åkesson, K.-E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit. Modeling and optimization with Optima and JModelica.org—languages and tools for solving large-scale dynamic optimization problems. *Computers & Chemical Engineering*, 34(11):1737–1749, November 2010. doi:10.1016/j.compchemeng.2009.11.011.
- Joel Andersson, Johan Åkessona, Francesco Casellad, and Moritz Diehl. Integration of CasADi and JModelica.org. In *Proceedings from the 8th International Modelica Conference, Technical University, Dresden, Germany*, pages 218–231. Linköping University Electronic Press, June 2011. doi:10.3384/ecp11063218.
- A. Brunberg, J. Maschuw, R. Autschbach, and D. Abel. Object-oriented model library of the cardiovascular system including physiological control loops. In *IFMBE Proceedings*, pages 166–169. Springer Nature, 2009. doi:10.1007/978-3-642-03895-2_48.
- John W. Clark, Robert Y. S. Ling, R. Srinivasan, J. S. Cole, and Roderick C. Pruett. A two-stage identification scheme for the determination of the parameters of a model of left heart and systemic circulation. *IEEE Transactions on Biomedical Engineering*, BME-27(1):20–29, January 1980. doi:10.1109/tbme.1980.326687.
- European Commission. Joint report on health care and long-term care systems & fiscal sustainability. *European Economy Institutional Papers*, (037), October 2016. doi:10.2765/680422.
- Jonas Gesenhues, Marc Hein, Moritz Habigt, Mare Mechelinck, Thivaharan Albin, and Dirk Abel. Nonlinear object-oriented modeling based optimal control of the heart: Performing precise preload manipulation maneuvers using a ventricular assist device. In *2016 European Control Conference (ECC)*. Institute of Electrical and Electronics Engineers (IEEE), June 2016. doi:10.1109/ecc.2016.7810603.
- Jonas Gesenhues, Marc Hein, Maïke Ketelhut, Moritz Habigt, Daniel Rüschén, Mare Mechelinck, Thivaharan Albin, Steffen Leonhardt, Thomas Schmitz-Rode, Rolf Rossaint, Rüdiger Autschbach, and Dirk Abel. Benefits of object-oriented models & ModeliChart: Modern tools and methods for the interdisciplinary research on smart biomedical technology. *Biomedical Engineering / Biomedizinische Technik*, 2017. doi:10.1515/bmt-2016-0074.
- Moritz Habigt, Maïke Ketelhut, Jonas Gesenhues, Frank Schrödel, Marc Hein, Mare Mechelinck, Thomas Schmitz-Rode, Dirk Abel, and Rolf Rossaint. Comparison of novel physiological load-adaptive control strategies for ventricular assist devices. *Biomedical Engineering / Biomedizinische Technik*, 2016. doi:10.1515/bmt-2016-0073.
- C.E. Hann, J.G. Chase, and G.M. Shaw. Integral-based identification of patient specific parameters for a minimal cardiac model. *Computer Methods and Programs in Biomedicine*, 81(2):181–192, February 2006. doi:10.1016/j.cmpb.2005.11.004.
- Maïke Ketelhut, Frank Schrödel, Sebastian Stemmler, Jesse Roseveare, Marc Hein, Jonas Gesenhues, Thivaharan Albin, and Dirk Abel. Iterative learning control of a left ventricular assist device. In *19th IFAC World Congress, Accepted for publication*. Toulouse, France, 2017.
- Ryo Kosaka, Yoshiyuki Sankai, Tomoaki Jikuya, Takashi Yamane, and Tatsuo Tsutsui. Online parameter identification of second-order systemic circulation model using the delta operator. *Artificial Organs*, 26(11):967–970, November 2002. doi:10.1046/j.1525-1594.2002.07112.x.
- Marek Mateják, Tomáš Kulhánek, Jan Šilar, Pavol Privitzer, Filip Ježek, and Jiří Kofránek. Physiobrary - modelica library for physiology. In *Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden*. Linköping University Electronic Press, March 2014. doi:10.3384/ecp14096499.
- Berno J.E. Misgeld, Daniel Rüschén, Sebastian Schwandner, Stefanie Heinke, Marian Walter, and Steffen Leonhardt. Robust decentralised control of a hydrodynamic human circulatory system simulator. *Biomedical Signal Processing and Control*, 20:35–44, July 2015. doi:10.1016/j.bspc.2015.04.004.
- Ajay Moza, Jonas Gesenhues, Dirk Abel, Rolf Rossaint, Thomas Schmitz-Rode, and Andreas Goetzenich. Patient specific parameter estimation for cardiovascular system models based on clinical measurements. *Biomedical Engineering / Biomedizinische Technik*, 2017. doi:10.1515/bmt-2016-0078.
- M Nichols, N Townsend, R Luengo-Fernandez, J Leal, A Gray, P Scarborough, and M Rayner. European cardiovascular disease statistics 2012. european heart network, brussels, european society of cardiology, sophia antipolis. 2012. *Cerebrovasc. Dis*, 25:457–507, 2012.
- Daniel Rüschén, Miriam Rimke, Jonas Gesenhues, Steffen Leonhardt, and Marian Walter. Online cardiac output estimation during transvalvular left ventricular assistance. *Computer Methods and Programs in Biomedicine*, August 2016. doi:10.1016/j.cmpb.2016.08.020.
- Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, April 2005. doi:10.1007/s10107-004-0559-y.
- Nico Westerhof, Jan-Willem Lankhaar, and BerendE. Westerhof. The arterial windkessel. *Medical & Biological Engineering & Computing*, 47(2):131–141, 2009. ISSN 0140-0118. doi:10.1007/s11517-008-0359-2.
- Yih-Choung Yu, J.R. Boston, M.A. Simaan, and J.F. Antaki. Estimation of systemic vascular bed parameters for artificial heart control. *IEEE Transactions on Automatic Control*, 43(6):765–778, June 1998. doi:10.1109/9.679017.

The DLR RailwayDynamics Library: the Crosswind Stability Problem

Andreas Heckmann¹ Gustav Grether²

^{1,2}Institute of System Dynamics and Control, German Aerospace Center (DLR), Germany,
andreas.heckmann@dlr.de

Abstract

High crosswinds affect the stability of railway vehicles, in particular if they run on very high speed to reduce traveling time, if they are configured as double-deck cars to increase the number of passenger seats and if they use light-weight design in order to reduce life-cycle costs. This is why crosswind stability is an active field of research within the project Next Generation Train. However, this field relies on the cooperation of two different domains, namely aerodynamics and vehicle dynamics. With this background a crosswind stability tool was implemented in Modelica as a part of the DLR RailwayDynamics Library. This tool gathers data from scaled wind tunnel measurements and multibody data on the railway vehicle in order to rapidly analyze and assess the risk of overturning due to high crosswinds. To a large extent the tool is oriented towards the associated homologation rules and standards. However, the tool is as well supposed to support future advancements of these standards by providing capabilities for the stochastic analysis of the crosswind stability problem.

vehicle dynamics, aerodynamics, railway vehicles, crosswind stability, aerodynamic admittance, stochastic analysis

1 Introduction

1.1 Motivation

Crosswind stability addresses the risk, that vehicles running on high speed are prone for overturning, if high crosswinds occur. At DLR, it is a particular subject of research, since the long-term internal project Next Generation Train (NGT) copes with three key features that aggravate the problem: it is a very high-speed train in double deck configuration and light-weight design. Nevertheless, this train concept has been proposed since it facilitates objectives such as low energy consumption and low life-cycle costs per passenger even for reduced traveling times.

From the technical point of view, crosswind stability is a multidisciplinary issue, since aspects of aero- and vehicle dynamics have to be taken into account. This is why the homologation rules, specified by regulation of the European Commission (TSI HS RST 2008) and the associated standard (EN 14067-6: 2010) address both: scaled wind tunnel experiments in order to characterize the aero-

dynamic properties and multibody simulations to study the mechanical behavior of the railway vehicle under consideration.

With this background, a subpackage of the DLR RailwayDynamics Library (Heckmann et al., 2014a), (Schwarz et al., 2015) has been implemented in Modelica which aims at the assessment of railway vehicles with respect to crosswind stability. To a large extent, the tool is oriented towards (EN 14067-6: 2010) and the simulation procedures defined there including a non-normative but promising stochastic approach in Appendix J. But beyond that, the implementation is intended to support various research activities in aerodynamics and vehicle dynamics control within the NGT project, see e.g. (Fey et al., 2014), (Heckmann et al., 2014b).

1.2 Overview on Vehicle Assessment

The basic assessment scenario is defined in (EN 14067-6: 2010) as follows: At given train speed, wind velocity and wind direction, aerodynamic forces and torques are applied to a multibody train model. These applied forces and torques lead to an unloading of the wheels at the windward side of the train. This unloading is interpreted to indicate the risk of overturning.

In detail, a major assessment issue is the so-called critical wind speed that is defined as the wind velocity, at which 90 % wheel unloading compared to the static load occurs. In general, the evaluation procedure requires to iteratively vary the wind velocity as an input parameter till the multibody simulation results show that the remaining wheel load is 10 % of the static load.

The critical wind speed is evaluated for several train velocities, these sample points are then connected to construct a curve called the Characteristic Wind Curve (CWC). In order to meet the homologation criterion the CWC of the considered vehicle must completely run above the Characteristic Reference Wind Curve (CRWC) defined by (TSI HS RST 2008). As illustrative examples, Fig. 1 presents the CWC of the NGT train head which meets the homologation criterion, while the initial NGT coach design does not.

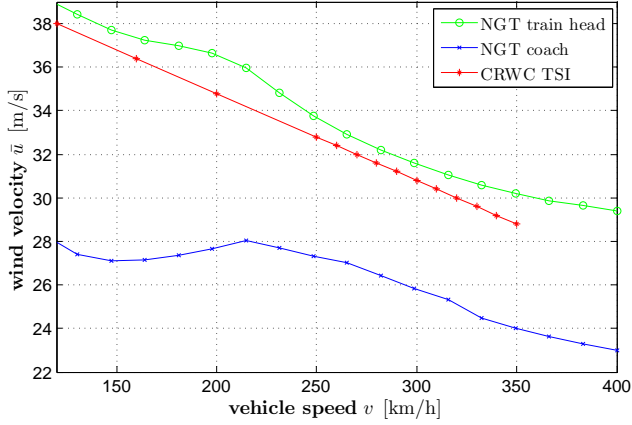


Figure 1. Characteristic Wind Curve (CWC) of the NGT train head and the NGT coach compared to the reference curve CRWC from (TSI HS RST 2008), cf. (Heckmann et al., 2014b).

2 Aerodynamic Loads

2.1 Fundamentals

The above described assessment scenario requires to specify loads, i.e. the aerodynamic forces and torques that are to be applied to the mechanical train model. These loads are commonly expressed by means of aerodynamic coefficients $c_i = c_i(\beta)$ and $c_j = c_j(\beta)$ as function of the yaw angle β as follows (Baker et al., 2009):

$$\bar{f}_i = \frac{1}{2} \rho A c_i(\beta) \bar{V}^2 \quad (1)$$

$$\bar{m}_j = \frac{1}{2} \rho A h c_j(\beta) \bar{V}^2. \quad (2)$$

In (1) and (2), \bar{f}_i and \bar{m}_j represent the vector components of force and torque, ρ is the density of the air, A and h the reference area and height, respectively. \bar{V} denotes the wind speed relative to the vehicle, which follows from vectorial decomposition considering the vehicle speed v , the wind velocity \bar{u} and the angle β_w between track and wind, see Fig. 2.

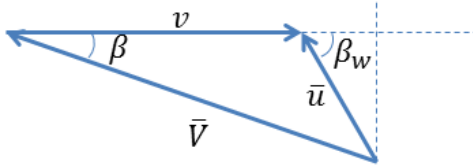


Figure 2. Vector decomposition to evaluate the wind speed relative to the vehicle \bar{V} .

It is state-of-the art to identify aerodynamic coefficients in scaled wind tunnel measurements as visualized in Fig. 3 and normalize the results with $A = 10 \text{ m}^2$ and $h = 3 \text{ m}$ according to (TSI HS RST 2008). Note that the (EN 14067-6: 2010) also approves numerical CFD simulation under certain conditions.

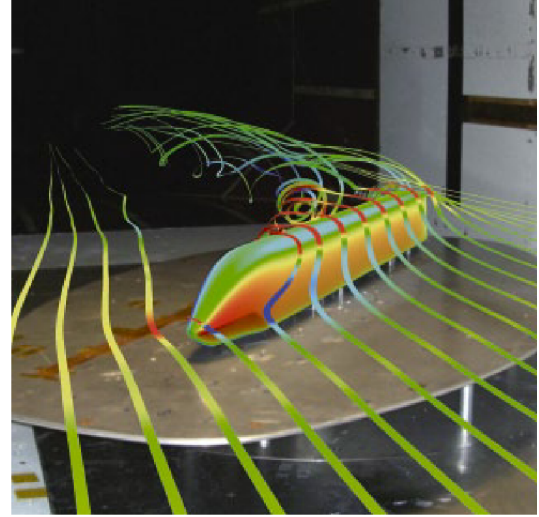


Figure 3. Calculated streamlines and pressure distribution on the NGT model scale 1:25 in the Cologne Cryogenic Wind Tunnel under cross wind conditions, see (Heckmann et al., 2014b).

Eq. (1) and (2) define the aerodynamic loads as a function of \bar{V} which in turn depends on the wind velocity \bar{u} according to Fig. 2 or more general, on the underlying wind gust model. Three different approaches to represent the wind gust are implemented in the Modelica RailwayDynamics Library:

1. Steady approach: $\bar{u} = \text{const}$
2. Quasi-steady approach: Eq. (1) and (2) are assumed to be valid even if the wind velocity changes in time, i.e. $\bar{u} = u(t)$. In detail, (EN 14067-6: 2010) defines the so-called Chinese Hat gust model in Fig. 4 which specifies a wind field fixed along the track at which the vehicle runs at constant speed v . For (1) and (2), \bar{u} then follows from $\bar{u} = \bar{u}(s) = \bar{u}(v \cdot t)$,
3. Unsteady approach: The mean wind forces \bar{f}_i and torques \bar{m}_j are superimposed with fluctuating parts f'_i and torques m'_j (Baker, 1991), i.e.

$$f_i = \bar{f}_i + f'_i, \quad m_j = \bar{m}_j + m'_j, \quad (3)$$

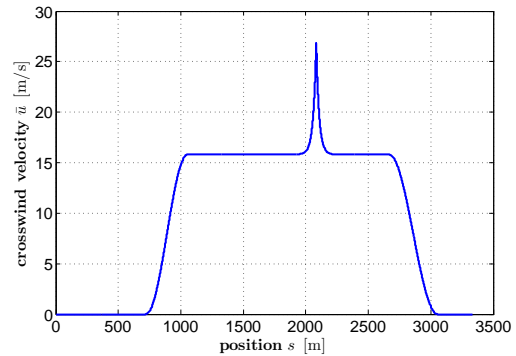


Figure 4. Crosswind velocity definition "Chinese Hat" according to (EN 14067-6: 2010).

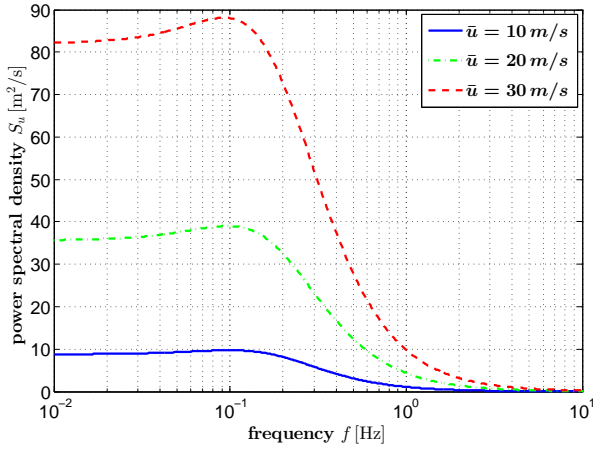


Figure 5. Power spectral density example of the turbulent wind according to (Cooper, 1984) for three different wind velocities \bar{u} , train speed $v = 400$ km/h and wind angle $\beta_w = 90^\circ$.

which are specified in the frequency domain as described in the next section.

2.2 Unsteady Aerodynamics

Unsteady wind may be approximated as a Gaussian stochastic process and therefore may be characterized by the power spectral density (PSD). Cooper derived the following wind spectrum relative to the moving vehicle S_u as function of the frequency f (Cooper, 1984):

$$S_u = \frac{4\tilde{f}}{f} \frac{\sigma_u^2}{[1 + 70.8\tilde{f}^2]^{\frac{5}{6}}} \left[c_u + (1 - c_u) \frac{0.5 + 94.4\tilde{f}^2}{1 + 70.8\tilde{f}^2} \right] \quad (4)$$

that include the following definitions and parameter values:

- the coefficient c_u : $c_u = \left(\frac{u}{\bar{v}} \cos \beta_w + \frac{\bar{u}}{\bar{v}} \right)^2$.
- the root mean square of the wind velocity fluctuations σ_u : $\sigma_u = 0.245 \cdot \bar{u}$, here inserted according to (EN 14067-6: 2010).
- the turbulence length scale in wind direction xL_u : ${}^xL_u = 96.0395$ m, see (EN 14067-6: 2010).
- the compound length scale L_u : $L_u = {}^xL_u \sqrt{c_u + 0.706(1 - c_u)}$
- the normalized frequency \tilde{f} : $\tilde{f} = f \frac{L_u}{\bar{v}}$

Fig. 5 shows exemplary spectra in order to illustrate (4).

In order to generate a representation of S_u in time domain, the fluctuation wind velocity u' may be evaluated by superimposing n discretized frequencies f_i with amplitude $A_i = A_i(f_i)$ and random phase ϕ_i :

$$u' = \sum_{i=1}^n A_i \sin(2\pi f_i \cdot t + \phi_i), \quad A_i = \sqrt{2S_{u,i}(f_{i+1} - f_i)}. \quad (5)$$

In reality, it has been observed that force fluctuations don't follow wind velocity fluctuations without attenuation or

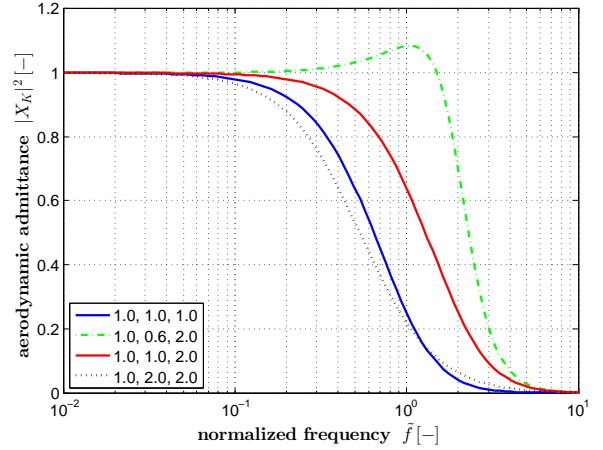


Figure 6. Aerodynamic admittance function for various parameter triples \hat{k} , $\hat{\xi}$, \hat{f} .

lag, so that an additional dynamics transfer behavior has to be considered by the aerodynamic admittance function $|X_K(f)|^2$ for each of the six force or torque components K and their associated spectra S_K :

$$|X_K(f)|^2 := \frac{1}{(\rho A c_k \bar{u})^2} \frac{S_K}{S_u}, \quad k = i, j, \quad K = f'_i, m'_j. \quad (6)$$

The measurement of the aerodynamic admittance for high-speed trains is a field of active research, which the RailwayDynamics Library is supposed to support. As an initial approach, it is referred to the following model, which uses three free, dimensionless parameters \hat{k} , \hat{f} and $\hat{\xi}$, which are fitted to approximate wind tunnel measurements as good as possible (Sterling et al., 2009):

$$|X_K(f)|^2 := \frac{1}{\hat{k} \left(\left[1 - \left(\frac{\tilde{f}}{\hat{f}} \right)^2 \right]^2 + \left[2\hat{\xi} \frac{\tilde{f}}{\hat{f}} \right]^2 \right)^2} \quad (7)$$

Fig. 6 gives an impression on parameter values from literature specifying the aerodynamic admittance.

3 Vehicle Dynamics

The (EN 14067-6: 2010) refers to three vehicle models with increasing complexity:

- The 2D three-mass model, which is not implemented in the RailwayDynamics Lib.
- The five-mass model without wheel-rail contact which is supposed to be used in a steady-state scenario.
- The multibody model with wheel-rail contact which is intended to be utilized for transient simulation tasks, in which quasi-steady or unsteady aerodynamic loads are applied.

3.1 The Simplified Five-Mass Model

Fig. 7 presents the structure of the five-mass model, as it is defined in (EN 14067-6: 2010, Appendix H). It considers two unsprung bodies which are the wheelsets of the railway vehicle, two primary suspended bodies representing the bogie frames including all attachments and the car body, which is connected to the bogies via secondary suspensions.

The four wheel-rail forces $Q_{ij}, i, j = 1, 2$, in Fig. 7 are interpreted as supporting or reaction forces due to the condition that the wheels do not lift off. Note, that at most 90 % wheel unloading is permitted which in turn defines that lift-off or loss of contact at the wheel-rail interface is not admissible and motivates to disregard the wheel-rail contact for the sake of simplicity.

The model takes all together 11 degrees of freedom into account. These are the lateral, vertical and roll motion of bogies and the car body, which may in addition rotate around its pitch and yaw axis. The vertical and lateral spring elements of the primary and secondary suspensions are equipped with bump stops specified by a non-linear stiffness characteristic as well proposed in (EN 14067-6: 2010, Appendix H). The rotational stiffness of an anti-roll bar as part of the secondary suspensions is supposed to reduce the tilting of the carbody due to wind torques acting around the longitudinal or x-axis.

Since this model is used in a steady-state scenario, transient behavior or modeling of damping devices actually is irrelevant but nevertheless is introduced in parallel to all spring elements. Due to the bump stops, the model is non-linear and the analysis is organized as a time simulation that is intended to converge against its final and steady state as a result of the applied constant wind loads.

In summary, the five-mass model is supposed to facilitate a simple analysis to be feasible in early engineering phases yielding conservative results with comparable low critical wind speeds. To this aim, the suspension elements are not modeled considering design details but are represented by a set of generalized stiffness parameters and

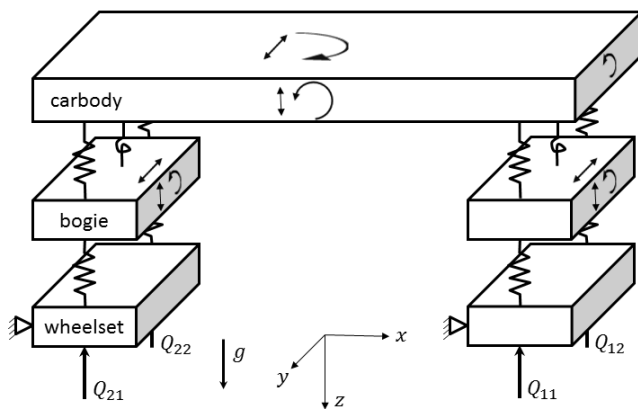


Figure 7. Structure of the simplified five-mass model according to (EN 14067-6: 2010).

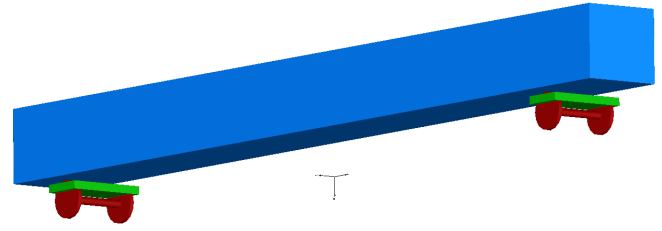


Figure 8. Animation of the five-mass-model.

very essential geometric information. Fig. 8 shows an animation of the five-mass model in Modelica.

3.2 The Multibody Model

The multibody model to be described here is tailored to the lightweight intermediate car of NGT project, but may easily be adapted to conventional high-speed railway vehicles. The most prominent up-grade to the five-mass model concerns the non-linear wheel-rail contact that is considered on basis of the geometry of the standardized UIC 60 rail and the WS 1002 profile geometry.

To this aim, the Modelica RailwayDynamics Library (Heckmann et al., 2014a) employs the distance $\Delta = \Delta(s, y, \varphi, \psi)$ as a function of the wheel profile coordinate s , $\underline{s} \leq s \leq \bar{s}$, and the lateral displacement, roll and yaw angle between wheel and rail, see Fig. 9. The contact position s^* defined as a weighted mean value of s using the regularization parameter α is a continuous function of y, φ and ψ and thus constitutes a smooth contact formulation, see (Arnold and Netter, 1997):

$$s^* := \frac{\int_{\underline{s}}^{\bar{s}} s e^{\frac{\Delta}{\alpha}} ds}{\int_{\underline{s}}^{\bar{s}} e^{\frac{\Delta}{\alpha}} ds}. \quad (8)$$

However for transient analysis, the (EN 14067-6: 2010) allows to monitor the wheel-rail forces utilizing a 2 Hz low-pass filter to evaluate the wheel unloading criterion. Therefore, it cannot be completely ruled out that loss of contact occurs and is admissible for very short time periods. In addition, overloading and in turn wheel lift-off

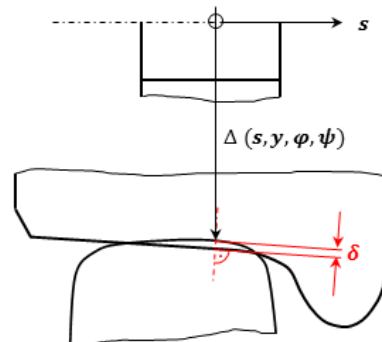


Figure 9. Sketch to illustrate the wheel-rail contact quantities (exaggerated presentation of the penetration δ).

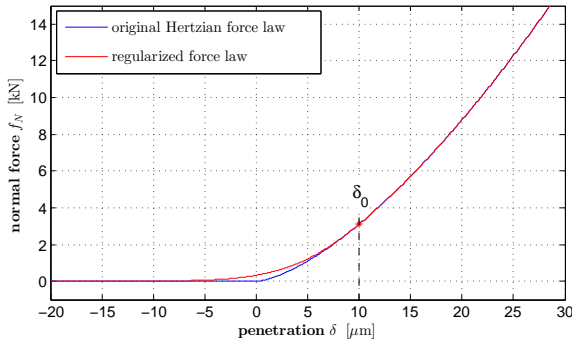


Figure 10. Illustration of the regularized contact force law.

may occur during the iteration process to determine the critical wind speed.

In order to take this into account, the wheel-rail contact algorithm of the Modelica RailwayDynamics Library had to be adapted. In particular the kinematic constraint (Heckmann et al., 2014a, (6)) is replaced by a regularized penalty contact formulation, in which the penetration δ of the wheel and the rail body in Fig. 9 features a non-linear spring element to evaluate the normal contact force f_N according to the Hertzian theory (Hertz, 1882):

$$f_N = \begin{cases} \delta < \delta_0: & a \cdot e^{-b(c+\delta)^2}, \\ \delta \geq \delta_0: & c_H \cdot \delta^{\frac{3}{2}}, \end{cases} \quad (9)$$

where $\delta_0 > 0$, $a := c_H e^{\frac{3}{4} \delta_0^{\frac{3}{2}}}$, $b := \frac{3}{4 \delta_0^2}$, $c := -2\delta_0$.

The coefficient c_H is a function of the material properties and the local curvatures of the contact partners at the point of contact, while a, b and c are defined in such a way that f_N is two times continuously differentiable for all values of δ and in particular for $\delta = \delta_0$, see Fig. 10. The proposed regularization prevents chattering in the vicinity of wheel lift-off situations and therefore improves the numerical robustness. In addition, this elastic contact, to be distinguished to the quasi-elastic formulation in (Heckmann et al., 2014a), requires damping to be numerical feasible.

It is state of the art in multibody analysis of railway vehicles to take the elastic compliance of the track into account that is excited by large wheel-rail forces. This is in particular important if these forces are a significant result of the analysis. Therefore, the track superstructure shown in Fig. 11 is presented as one body with three sprung and damped degrees of freedom, which is assumed to follow each wheel pair as a so-called moving track model, e.g. see (Iwnicki, 2006, Ch. 12)

The NGT running gears use so-called independently rotating wheel sets (IRW set) with two wheels attached to a wheel carrier. Active guidance control (Kurzeck et al., 2014) provides running stability and low wear properties of this configuration which is a main objective of the NGT project.

In order to be consistent, these model upgrades implicate a more complex motion composition compared to

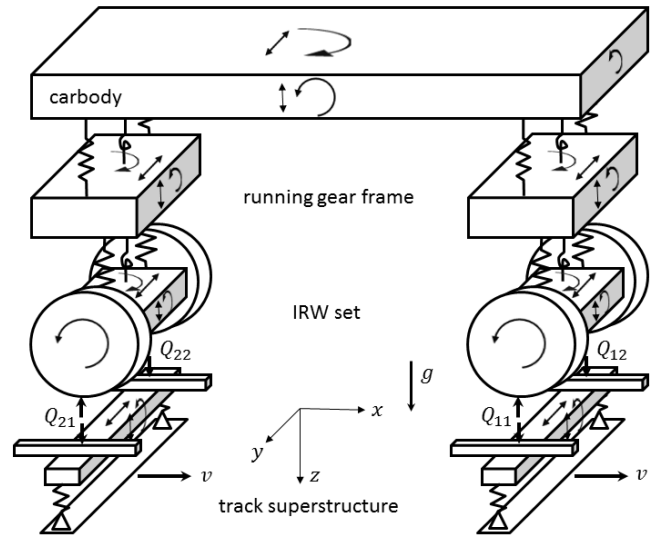


Figure 11. Structure of the multibody model of the NGT intermediate car (all springs include dampers connected in parallel).

Sec. 3.1 that requires the consideration of all together 31 degrees of freedom which are indicated in Fig. 11. Apart from that, the multibody model, whose animation is shown in Fig. 12, sticks to the concept of the five-mass model to present all suspension elements by a set of generalized stiffness and damping parameters and very essential geometric information. This modeling idea is well suited to be used in early engineering phases, when detailed information on the design of the suspensions are not yet available and therefore fits to the scope of the RailwayDynamics Library.

3.3 Negotiating Curves

For comfort reason, the maximum lateral acceleration that passengers are supposed to experience while the vehicle is negotiating curves is restricted to $1m/s^2$. Therefore, the lay-out of railway tracks also includes superelevations and the maximum speed at which the vehicles runs through the curve is limited as a function of curve superelevation and radius in order to meet this requirement.

This property is exploited by the (EN 14067-6: 2010) to

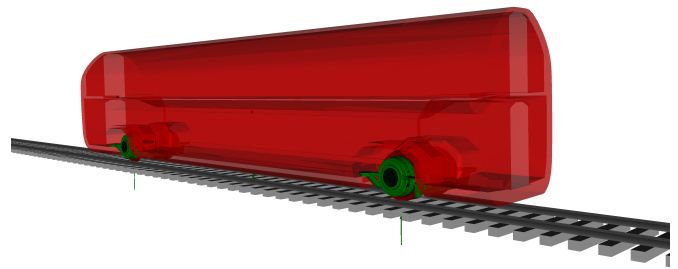


Figure 12. Animation of the NGT intermediate car multibody model.

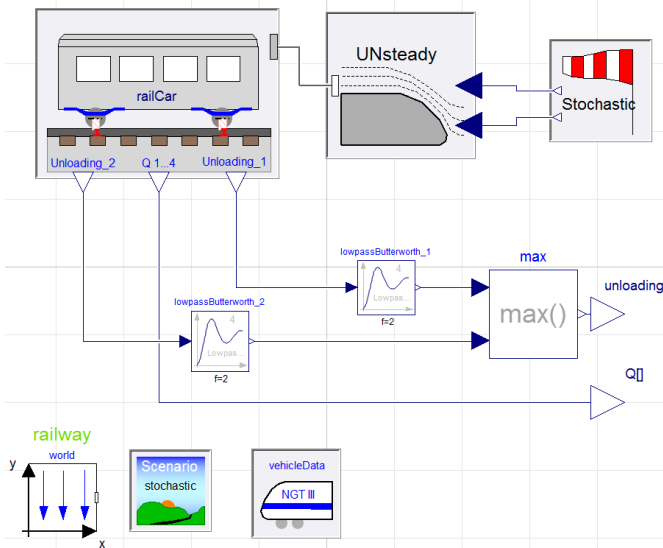


Figure 13. Diagram layer of a crosswind stability model of the Modelica RailwayDynamics Library.

address the crosswind stability while negotiating curves. The evaluation of the CWCs are additionally parametrized with the so-called unbalanced lateral acceleration a_q , which can take values between -1 and 1 $[m/s^2]$, i.e. $-1m/s^2 \leq a_q \leq 1m/s^2$. The introduction of this parameter into the Modelica crosswind scenario is straight forward by specifying the direction and value of the gravity vector accordingly.

4 Implementation

Fig. 13 gives an overview on the structure of a crosswind stability model of the Modelica RailwayDynamics Library. Vehicle data are separated from the model instances and organized by data records, which in turn are substructured in aerodynamical and mechanical information. Another record called *Scenario* organises information to perform the specific simulation task, see Fig. 14.

Wind generation, aerodynamical load evaluation and vehicle running dynamics are separated in three model components, so that it is easy to exchanged e.g. the stochastic wind with a chinese hat wind gust instance or substitute steady for unsteady aerodynamic approach.

In order to facilitate robust initialization, the application of the aerodynamic loads f_i and m_j from (3) is delayed in time using a first order low pass filter with time constant t_0 , i.e.:

$$\check{f}_i = (1 - e^{-\frac{t}{t_0}}) f_i, \quad \check{m}_j = (1 - e^{-\frac{t}{t_0}}) m_j. \quad (10)$$

The *railCar* instance in Fig. 13 provides the wheel-unloading of each running gear as a function of time, which then is low-pass filtered according to (EN 14067-6: 2010). The output of the *max()* block evaluates the final simulation result or the crosswind stability criterion, respectively.

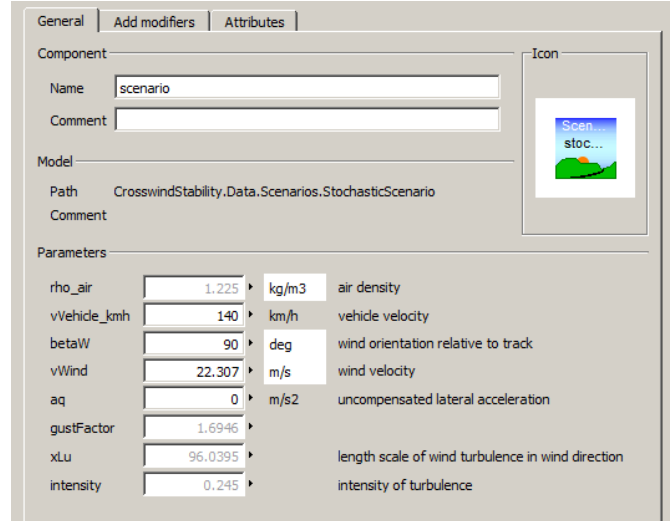


Figure 14. Parameter menu of the scenario data record.

In order to determine the critical wind speed, a function *find_Vcwc* is defined, which iteratively simulates the crosswind stability model, while the wind speed is varied systematically. The function terminates and returns the critical wind speed, if the model simulation results show 90% wheel unloading:

```
function find_Vcwc
  input Real vVehicle_kmh (unit="km/h")=80 "
    vehicle speed";
  input SI.Angle betaW=
    Modelica.SIunits.Conversions.from_deg
    (90) "wind angle";
  input SI.Acceleration aq=1 "unbalanced
    lateral acceleration";
  input SI.Velocity vW[2]={20,30} "range to
    look for v_cwc";
  input String modelName;
  input Real tolerance=0.1 "tolerated
    deviation of target unloading=0.9";
  input Real t_stop=110 "end time"
  output SI.Velocity v_cwc;
```

A second function *plot_CWC* not only evaluates one critical wind speed, but provides a plot of the critical wind curve as shown in Fig. 1.

```
function plot_CWC
  "iteration process to evaluate v_cwc=
    v_cwc(vVehicle, betaW, aq)"
  input Real vVehicle_kmh[2] (unit="km/h")
    ={120,400} "considered speed range";
  input SI.Angle betaW=
    Modelica.SIunits.Conversions.from_deg
    (90) "wind angle";
  input SI.Acceleration aq=1 "unbalanced
    lateral acceleration";
  input SI.Velocity vW[2]={20,30} "range to
    look for v_cwc";
  input String modelName;
  input Integer numberOfVehicles= 15 "
    number of speed sampling points";
  input Real tolerance=0.1 "tolerated
    deviation of target unloading=0.9";
```

```

input Real t_stop=110 "end time"
output SI.Velocity v_cwc[
  numberOfVehicles];

```

5 Exemplary Results

5.1 The Five-Mass Model

The (EN 14067-6: 2010) offers the opportunity to verify the implementation of the steady scenario with the five-mass model, since its Appendix H contains the input data and the results of two example vehicles. Fig. 15 presents a comparison of Modelica RailwayDynamics Library results with the CWC from the EN standard for Vehicle 2 and two unbalanced accelerations. The large correspondence of results can be stated, a slight derivation only occurs for $v = 80 \text{ km/h}$ vehicle speed, which is to be further investigated.

The CWC of the NGT in Fig. 1, which has been introduced in order to give an overview on the vehicle assessment methodology in Sec. 1.2, has also been generated using the five-mass model, see also (Heckmann et al., 2014b) for a more detailed discussion.

5.2 The Multibody Model of the NGT Coach

With the exception of Fig. 19, all results to be given in this section have been obtained using the scenario shown in Fig. 14, which will turn out to lead to 90% wheel unloading. The associated transient wind velocities are plotted in Fig. 16.

The parameters to be used in (6) are intended to be gained in wind tunnel measurements, which are not yet available. Therefore, hypothetical admittance parameters have been introduced in order to evaluate preliminary and exemplary results. Following a proposal of (Baker, 2010), the values below have been chosen for the most important force and torque components:

$$\begin{aligned}
 \text{side force:} \quad & \hat{k} = 1, \quad \hat{\xi} = 1, \quad \hat{f} = 2.0 \cdot \sin(\beta), \\
 \text{lift force:} \quad & \hat{k} = 1, \quad \hat{\xi} = 1, \quad \hat{f} = 2.5 \cdot \sin(\beta), \\
 \text{roll moment:} \quad & \hat{k} = 1, \quad \hat{\xi} = 1, \quad \hat{f} = 2.0 \cdot \sin(\beta).
 \end{aligned}
 \tag{11}$$

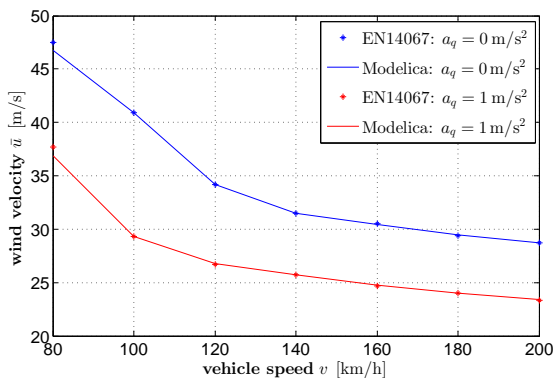


Figure 15. Comparison of the CWC evaluated in Modelica with (EN 14067-6: 2010), Vehicle 2 in Appendix H.

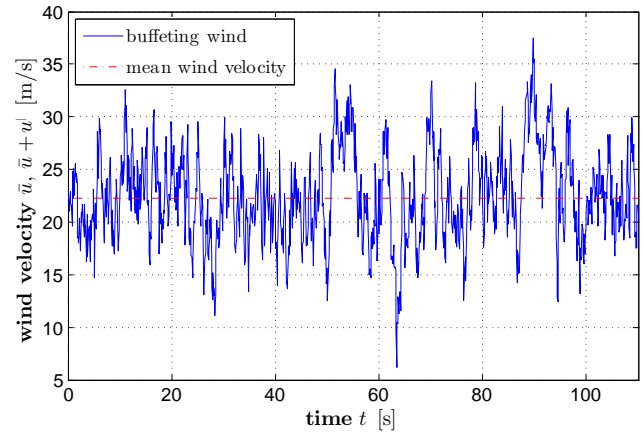


Figure 16. Transient velocities of buffeting wind ($\bar{u} = 22.307 \text{ m/s}$, $\sigma_u = 0.245 \cdot \bar{u}$, $\beta_w = 90^\circ$, $v = 140 \text{ km/h}$).

The admittance of all other wind load components have been set to $|X_K(f)|^2 := 1$ as recommended in (EN 14067-6: 2010, Appendix J).

Fig. 17 presents the transient wheel forces, which all start from the static wheel load $f_w(t=0) = 57727 \text{ N}$, since the wind loads are applied according to (10) with time constant $t_0 = 2 \text{ s}$.

The lowest frequency that has been considered in (5) to transfer the PSD in Fig. 5 into the time domain is $f_1 = 0.01 \text{ Hz}$. In the (EN 14067-6: 2010, Appendix J), it is proposed to choose the simulation time in such a way, that one full period of the lowest frequency is covered, i.e. 100s here. The additional 10s have been appended in order to account for the low passed filtered load application at the beginning of the simulation. Note, the (EN 14067-6: 2010) requests to repeat this simulation with different random phases ϕ_i in (5) and to statistically determine the mean value of the critical wind velocity and its confidence interval.

The fifth curve in Fig. 17 displays the low pass filtered wheel unloading that reaches the crosswind stability criterion at $t = 90.3 \text{ s}$.

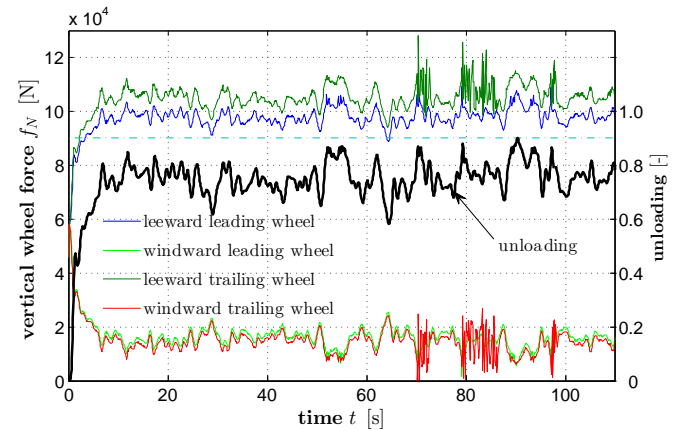


Figure 17. Transient vertical wheel force and unloading results based on hypothetical admittance parameters.

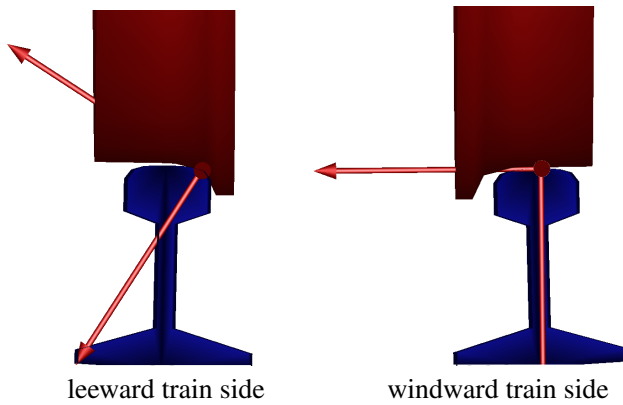


Figure 18. Wheel-rail contact configuration with flange contact at the leeward train side

Fig. 18 illustrates the corresponding contact configuration between wheel and rail. Due to the lateral wind load the complete car is displaced in lateral direction until the wheel flange is touched and counteracts the load force.

Fig. 19 shows the CWC of the NGT coach, which summarizes the critical wind velocities for 15 different vehicle speeds ($a_q = 0 \text{ m/s}^2$). Note, that all curves in Sec. 5.2 are preliminary results. Measured admittance functions for the NGT are not yet available, so that the associated parameters only have been chosen on a trial basis. The introduced root mean square of the wind fluctuations $\sigma_u = 0.245 \cdot \bar{u}$ that lead to rather large peaks of the wind velocities in Fig. 16 is another parameter to be substantiated in the future.

182cpu-s on a lap-top with Core-i7 processor and 2.9 GHz clock rate were required to get the above given results associated to 110s simulation time. It took less than an hour to evaluate the CWC in Fig. 19, which relies on an iterative process to obtain the critical wind velocity for all 15 vehicle speeds.

6 Conclusions and Outlook

In the course of the DLR project Next Generation Train the subpackage CrosswindStability of the DLR Railway-

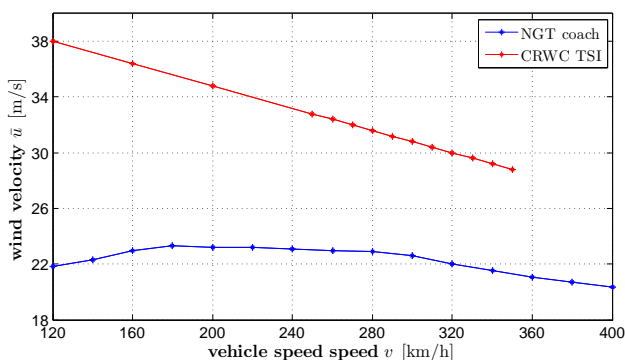


Figure 19. Preliminary critical wind curve of the NGT coach based on hypothetical admittance parameters for stochastic analysis.

Dynamics Library has been implemented. The tool offers the capability of use and combine several vehicle models and aerodynamic approaches in order to assess the crosswind stability of railway vehicles.

The consideration of unsteady aerodynamics within this task is a active field of research at DLR. The current focus is the measurement of the aerodynamic admittance function in a reproduceable and reliable manner. The RailwayDynamics Library now affords to rapidly analyze the vehicle dynamics once a aerodynamic admittance is available and that way provides a quick insight on further implications with respect to the risk of overturning.

Another future field of application of the presented capabilities concerns the dimensioning of suspension parameters of a railway vehicle in early engineering phases. Although the multibody model considers all relevant degrees of freedom and suspension components only moderate computational resources are required. The employment of the vehicle dynamics model in optimization tasks seems to be feasible, which may include multiple design objectives such as passenger comfort or running behavior besides crosswind stability.

Acknowledgment

An initial version of the Modelica crosswind stability tool has been implemented by Dr. Antonio Carrarini during his period of employment at DLR.

Wind tunnel measurements on the steady aerodynamics of the NGT have been provided by Sigfried Loose and his colleagues from the DLR Institute of Aerodynamics and Flow Technology, see Fig. 3 and cf. (Heckmann et al., 2014b).

References

- M. Arnold and H. Netter. Wear profiles and the dynamical simulation of wheel-rail systems. *Progress in Industrial Mathematics at ECMI*, 96:77–84, 1997.
- Chris Baker, Federico Cheli, Alexander Orellano, Nicolas Paradot, Carsten Proppe, and Daniele Rocchi. Cross-wind effects on road and rail vehicles. *Vehicle System Dynamics*, 47(8):983–1022, 2009. doi:10.1080/00423110903078794.
- C.J. Baker. Ground vehicles in high cross winds part II: unsteady aerodynamic forces. *Journal of fluids and structures*, 5(1): 91–111, 1991.
- C.J. Baker. The simulation of unsteady aerodynamic cross wind forces on trains. *Journal of Wind Engineering and Industrial Aerodynamics*, 98(2):88–99, 2010.
- R.K. Cooper. Atmospheric turbulence with respect to moving ground vehicles. *Journal of wind engineering and industrial aerodynamics*, 17(2):215–238, 1984.
- EN 14067-6: 2010. Railway Applications -Aerodynamics- Requirements and test procedures for crosswind assessment., 2010.

- Uwe Fey, Johannes Haff, Mattias Jönsson, Sigfried Loose, and Claus Wagner. Experimental investigation of topological changes in the flow field around high-speed trains with respect to reynolds number scaling effects. In J. Pombo, editor, *The Second International Conference on Railway Technology: Research, Development and Maintenance*, number P32 in Civil Comp Proceedings, pages 1–20. Civil-Comp Press, Stirlingshire, UK, 2014. doi: 10.4203/ccp.104.32.
- A. Heckmann, A. Keck, I. Kaiser, and B. Kurzeck. The Foundation of the DLR RailwayDynamics Library: the Wheel-Rail-Contact. In *10th International Modelica Conference*, 2014a.
- Andreas Heckmann, Bernhard Kurzeck, Tilman Bunte, and Sigfried Loose. Considerations on active control of cross-wind stability of railway vehicles. *Vehicle System Dynamics*, 52(6):759–775, 2014b.
- Heinrich Hertz. Über die Berührung fester elastischer Körper. *Journal für die reine u. angewandte Mathematik*, 92, 1882.
- S. Iwnicki. *Handbook of railway vehicle dynamics*. CRC Press, 2006.
- B. Kurzeck, A. Heckmann, C. Wesseler, and M. Rapp. Mechatronic track guidance on disturbed track: the trade-off between actuator performance and wheel wear. *Vehicle System Dynamics*, 52(sup1):109–124, 2014.
- Christoph Schwarz, Andreas Heckmann, and Alexander Keck. Different models of a scaled experimental running gear for the DLR RailwayDynamics Library. In *11th International Modelica Conference, 21.-23. Sep. 2015*, 2015.
- Mark Sterling, Chris Baker, Abdessalem Bouferrouk, Hugh ONeil, Stephen Wood, and Ewan Crosbie. An investigation of the aerodynamic admittances and aerodynamic weighting functions of trains. *Journal of Wind Engineering and Industrial Aerodynamics*, 97(11):512–522, 2009.
- TSI HS RST 2008. 2008/232/EC: Commission Decision of 21 February 2008 concerning a technical specification for interoperability relating to the rolling stock sub-system of the trans-european high-speed rail system, 2008.

The OneWind[®] Modelica Library for Floating Offshore Wind Turbine Simulations with Flexible Structures

Mareike Leimeister Philipp Thomas

Fraunhofer Institute for Wind Energy and Energy System Technology IWES Northwest, Germany,
{mareike.leimeister, philipp.thomas}@iwes.fraunhofer.de

Abstract

Floating offshore wind turbines are getting more and more into the focus of interest, as industries aim for larger turbines and deeper water areas. Fully coupled analyses of those highly complex systems are challenging. In this paper, the hierarchical programming structure in Modelica is used to model a fully flexible floating wind turbine system. The single components, as well as special difficulties that have to be dealt with during modeling, are addressed. On basis of a reference floating offshore wind turbine, the implemented fully flexible model is compared with its rigid equivalent, as well as results from code-to-code comparisons of free-decay simulations. The findings are satisfactory and confirm the theoretical assumptions. In addition, further applications of the created model are shown.

Keywords: *offshore wind energy, floating platform, fully coupled aero-hydro-servo-elastic simulation, Euler-Bernoulli beam, OneWind Modelica Library, MultiBody*

1 Introduction and Outline

Many promising offshore sites for wind energy utilization are in deep water. For water depths larger than 50 m, commonly used bottom-fixed foundations, as for example monopiles, jackets, or tripods, are no longer suitable. However, floating platforms, such as spar-buoys, semi-submersibles, or TLPs (tension leg platforms), could be a potential solution for deep water operations. Easier and faster installation due to onshore assembly, as well as reduced noise during erection are some advantages that floating support structures have over bottom-fixed designs. On the other hand, floating wind turbines are very complex systems. Motion-coupling, wave excitation, and additional components like mooring lines are inter alia new challenges for accurately modeling and simulating those systems, and allowing fully coupled load analyses.

Extensive research on floaters is conducted and several prototypes are designed^{1,2,3}. Even in the IEA Wind

Tasks⁴, floating wind turbine systems are included. In order to contribute to code-to-code comparison analyses, a fully flexible model for floating wind turbine systems is developed in the OneWind[®] Modelica Library.

In this paper, first, the different components of a floating offshore wind turbine system and their implementation in Modelica, based on the Modelica MultiBody Library, are explained in Chapter 2. Chapter 3 outlines the limitations of the implemented floating wind turbine model. The OCx offshore wind turbine designs, elaborated in the IEA Wind Tasks, are used in Chapter 4 as basis for comparison of reference load case simulation results, as well as for demonstrating the high flexibility for adaptations and ease of model modifications. Finally, Chapter 5 summarizes the developed approach and gives recommendations for further work on fully flexible floating offshore wind turbine systems in Modelica.

2 Components and Implementation in Modelica

Object-oriented programming in Modelica enables a hierarchical structure of the complex wind turbine system. The implemented floating wind turbine model contains six main components (rotor, nacelle, operating control, support structure, wind, and waves), which are possibly using further subcomponents, as presented in the following Modelica code and in Figure 1.

```
model OffshoreWindTurbine
extends OneWind.WindTurbine.OffshoreWT
(
  //=== rotor ===
  ,redeclare model Rotor = OneWind.Rotor
  (
    redeclare record RotorData
    ,redeclare model Hub
    ,redeclare model Blade
  )
  //=== nacelle ===
  ,redeclare model Nacelle =
    OneWind.Nacelle
)
```

¹<https://www.statoil.com/en/news/hywindscotland.html> (Accessed: 02 March 2017)

²<http://principlepowerinc.com/en> (Accessed: 02 March 2017)

³<http://ideol-offshore.com/en> (Accessed: 02 March 2017)

⁴<http://www.ieawind.org/taskWebSites.html> (Accessed: 23 September 2016)

```

(
  redeclare record NcData
  , redeclare model Drivetrain
  , redeclare model Generator
  , redeclare model YawController
)
//=== operating control ===
, redeclare model OperatingControl =
  OneWind.OperatingControl
(
  redeclare record OperatingControlData
  , redeclare model MainControl
  (
    redeclare model PitchControl
    , redeclare model
      GeneratorTorqueControl
  )
  , redeclare model GeneratorSpeedFilter
)
//=== support structure ===
, redeclare model SupportStructure =
  OneWind.FlexibleFloater
//=== wind ===
, redeclare model Wind = OneWind.Wind
(
  redeclare record WindData
)
//=== waves ===
, redeclare model Waves = OneWind.Wave
(
  redeclare record WaveData
)
);
end OffshoreWindTurbine;

```

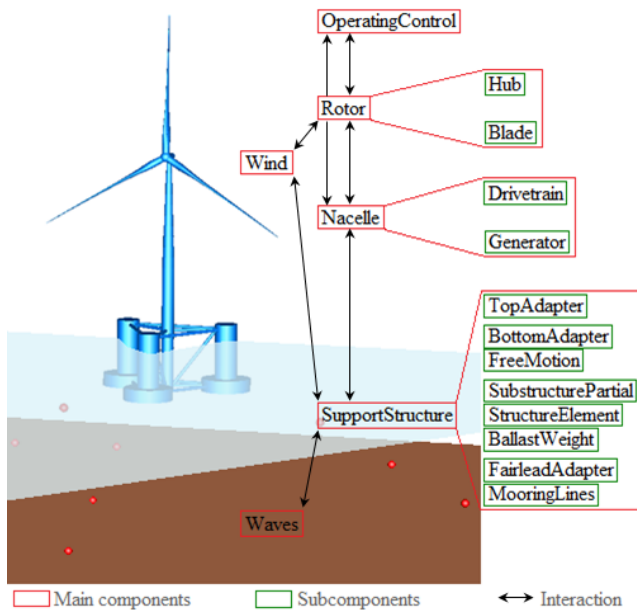


Figure 1. Components and interactions of a floating wind turbine, using the example of a semi-submersible platform.

2.1 Rotor

The rotor model extends the basic model for a hub with one blade to a three-bladed rotor. The blades

are implemented either as rigid bodies or as flexible structures, which could be based on modal reduction techniques or finite-elements (Thomas et al., 2014). The structure model is connected to the aerodynamic model, which uses unsteady blade element momentum theory for load calculation, and takes aero-structure-coupling into account.

2.2 Nacelle

The model of the nacelle contains two subcomponents: the drivetrain and the generator. Furthermore, the yaw controller is included. The nacelle is basically represented as rigid link with mass and inertia, while drivetrain and generator provide also stiffness and damping (Strobel et al., 2011).

2.3 Operating Control

The operating control covers algorithms and parameters for pitch and generator torque control, using either built-in PID-algorithms (Jonkman et al., 2009) or an external control DLL. The latter one is obtained from a simulation tool, Bladed (GL Garrad Hassan, 2010) or Hawc2 (Larsen and Hansen, 2014), and accessed via a generic DLL interface. A bus system forms the link between rotor, nacelle, and operating control (Otter, 2009). There is no direct link to the support structure, as the control parameters are initially adjusted based on the floating system design. Furthermore, different operating phases, such as startup, shutdown, or idling, can be realized.

2.4 Support Structure

The support structure model defines everything related to the floating device, including the tower from the RNA (Rotor Nacelle Assembly) down to the substructure, the floater itself, station-keeping system, and all loads acting on the entire support structure. Furthermore, it contains the *FreeMotion*, relevant for modeling the motions of the floating body. An overview of the structure of the model *SupportStructure* is given in the following:

```

model SupportStructure
  //— substructurePartial —
  extends OneWind.SubstructurePartial;
  //— structureElement —
  TopologyData = OneWind.FloaterTopologyData;
  StructureElement = OneWind.BernoulliBeam3D;
  //— adapters —
  OneWind.AdapterFemFrameToFrame_free
    topAdapter;
  OneWind.AdapterFemFrameToFrame_fixed
    bottomAdapter;
  OneWind.AdapterFemFrameToFrame_free
    fairleadAdapter[3];
  //— additional weights —
  OneWind.AdditionalWeightLoadElement
    ballastWeight[noElementsBallast];

```

```

OneWind.AdditionalWeightLoadElement
  capsWeight[ noElementsCaps ];
//— station-keeping system —
OneWind.DynamicMooringLines mooringLines
  [3];
MultiBody.Parts.FixedTranslation
  fTrFairleads[3];
MultiBody.Parts.FixedTranslation fTrAnchors
  [3];
//— freeMotion —
FreeMotion = MultiBody.Joints.FreeMotion;
end SupportStructure;

```

2.4.1 SubstructurePartial

The basis of the support structure model is formed by the partial model *SubstructurePartial*. This covers all main loads, as well as the visualization of the environment, represented by a squared *FixedShape* for the seabed and a moving surface for animating the wave motion, and contains the model *World*. The structure of the partial model *SubstructurePartial* is presented in the following:

```

partial model SubstructurePartial
outer MultiBody.World world;
//— visualization —
MultiBody.Visualizers.FixedShape ground;
MultiBody.Visualizers.Advanced.Surface
  surface;
//— wave loads —
OneWind.MorisonLoadElement waveLoads[
  noElementsUnderWater ];
OneWind.MorisonLoadHeavePlate
  morisonLoadHeavePlate[ noHeavePlates ] if
  heavePlates;
//— wind loads —
OneWind.TowerLoadElement windLoads[
  noElementsOverWater ];
//— buoyancy loads —
OneWind.BuoyancyLoadElement buoyancyLoads[
  noStructureElements ];
end SubstructurePartial;

```

The determination of the loads due to waves, wind, and buoyancy is covered in the following in more detail. The gravity force is not elaborated explicitly, as its

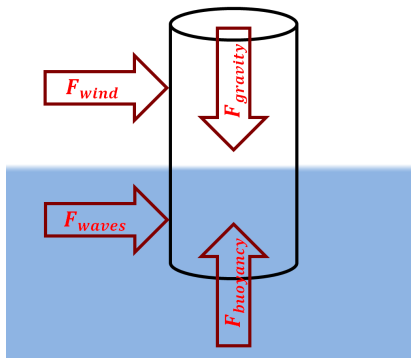


Figure 2. Schematic representation of the main loads acting on the support structure.

computation is directly included in the setup of the subcomponent *StructureElement* (Subsection 2.4.2). A schematic overview of these main loads is presented in Figure 2.

Wave Loads The hydrodynamic load calculation uses Morison's equation, as given in Equation 1, and is performed for each structure element that is initially below the water surface, based on its diameter D , length ∂z , hydrodynamic drag coefficient C_D , and added mass coefficient C_a , as well as velocities (structure velocity \dot{q} , water particle velocity v_{water} , relative velocity $v_{\text{water}} - \dot{q}$), accelerations (structure acceleration \ddot{q} , water particle acceleration \dot{v}_{water}), and water density ρ_{water} .

$$\begin{aligned}
 F_{\text{waves}} = & \frac{1}{2} \rho_{\text{water}} C_D D (v_{\text{water}} - \dot{q}) |v_{\text{water}} - \dot{q}| \partial z \\
 & + \rho_{\text{water}} (1 + C_a) \frac{\pi D^2}{4} \dot{v}_{\text{water}} \partial z \\
 & - \rho_{\text{water}} C_a \frac{\pi D^2}{4} \ddot{q} \partial z
 \end{aligned} \quad (1)$$

As offshore wind turbines often have to deal with large dimensioned support components, a separate parameter is introduced to select whether a fixed value for the added mass coefficient should be used, which is only valid for slender structures, or the added mass coefficient is calculated depending on the wave number, known as MacCamy-Fuchs approach for large diameters (Yu, 2015). Furthermore, if the floater is equipped with heave plates, acting as motion suppression device, as it is the case for semi-submersible platforms, an additional hydrodynamic heave force due to these heave plates is included.

Wind Loads In the aerodynamic load calculation, the drag forces at each emerged support structure element are computed by means of Equation 2, based on the density of air ρ_{air} , the aerodynamic drag coefficient C_d of the cylindrical element, its diameter D and length ∂z , as well as the local relative velocity, resulting from the local wind speed v_{wind} and the velocity of the structure element \dot{q} .

$$F_{\text{wind}} = \frac{1}{2} \rho_{\text{air}} C_d D (v_{\text{air}} - \dot{q}) |v_{\text{air}} - \dot{q}| \partial z \quad (2)$$

Buoyancy Loads Because a floating wind turbine system is considered, buoyancy force and center of buoyancy will vary with the motion of the floater. Therefore, these two variables have to be computed for each structure element at each time step, depending on the actual position. The coordinate system, used in this calculation, as well as the degrees of freedom (DoFs) of a floating wind turbine are presented in Figure 3.

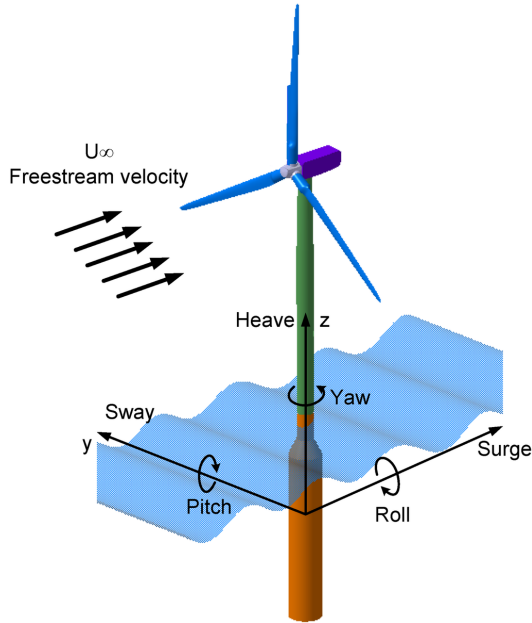


Figure 3. Coordinate system of a floating wind turbine, using the example of a spar-buoy platform, adapted by the author from (Tran et al., 2014).

In an extensive computation, first, the sequential rotation, defined by roll, pitch, and yaw angles, is transformed into a combined rotation, expressed in terms of a combined rotation angle α_{combined} and the corresponding axis of the combined rotation. Instead of having the complex floater geometry rotated, the following approach is used: it is assumed that the floater remains in its initial position and the water plane area is rotated with the combined rotation angle around the axis of combined rotation, however, in opposite direction, as schematically shown in Figure 4.

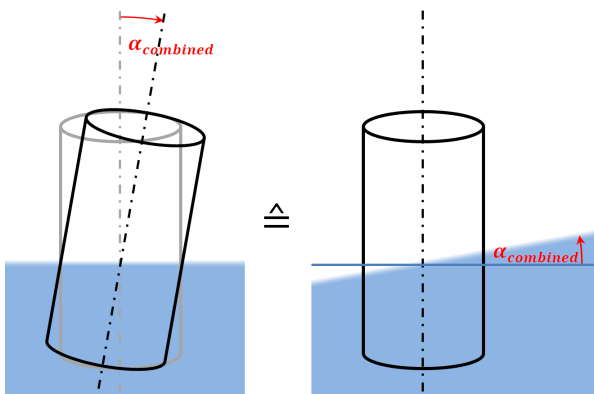


Figure 4. Schematic representation of the used buoyancy calculation approach.

Including the translational motion of the floater, given by surge, sway, and heave values of the platform, and used as the distance from the initial origin to the moved water plane, the equation for the rotated and translated water plane can be set up. With the node positions of the con-

sidered beam, defined in the subcomponent *StructureElement*, as further explained in Subsection 2.4.2, a straight equation can be defined for the considered structure element. The intersection of this straight with the moved water plane is analyzed according to the following case discrimination:

- An infinite number of cross points corresponds to the structure element lying exactly in the water plane, leading to a buoyant volume of half of the element volume.
- The solution of having no cross points corresponds to the structure element being parallel to the water plane. Depending on the node positions in relation to the translational motion, the element is either fully submerged or not submerged at all.
- Finally, when having one cross point of straight and plane, the buoyant volume can be computed as fraction of the element volume, if the cross point lies within the actual length of the structure element. If the straight would intersect the water plane at an extension of the structure element, the buoyant volume is either equal to the element volume or zero, depending on the relative position of the element nodes to the translated water plane.

From the determined buoyant volume V_B , the buoyancy load of each structure element at each time step is obtained by multiplication with the water density ρ_{water} and gravitational acceleration g , as given in Equation 3.

$$F_{\text{buoyancy}} = V_B \rho_{\text{water}} g \quad (3)$$

The buoyancy force is then connected to the *frame_c* of the element (introduced in Subsection 2.4.2), which is located in the middle of the element axis, including deformation. As, however, the point of attack of the buoyancy force varies with the motion of the floater, the distance from the actual point of attack to the central point (*frame_c*) is computed according to the different element positions elaborated in the above case discrimination. The resulting moment due to the shifted center of buoyancy is finally added as torque load to the *frame_c*, so that correct loads due to buoyancy are represented.

2.4.2 StructureElement

In the subcomponent *StructureElement*, all members of the support structure (floater and tower up to the RNA) are defined, based on a record for the topology data. This record contains number and coordinates of the nodes, as well as number and definition of the members, specified by the two end nodes and the structural properties of the element. The tubular beam properties are defined by an isotropic material (with elastic modulus and shear modulus, density, and Rayleigh damping parameters),

as well as start and end diameters and wall thicknesses. This record can either be written manually, or generated by means of a MATLAB code. The latter method makes it easy to change subdivisions of the beam elements and is thus useful for a more comprehensive structural analysis. The *TopologyData* record is used for setting up the structure elements, using an extended 3D-Bernoulli beam element model, which is also applicable to branched geometries and has an external load connector *frame_c*.

Avoiding Closed Loops Offshore substructures might be branched structures, like semi-submersibles, TLPs, or also bottom-fixed support structures, such as tripods and jackets. This will lead to closed loops in multibody applications that use the floating frame of reference, what makes it impossible to calculate the unique orientation of each frame, especially where branches are connected. This problem is addressed here by excluding orientation from the node connectors that are used to build up the substructure by defining the position of each member and connecting the members. In case internal forces need to be resolved between local beam and world frame, a local beam orientation is constructed by means of *absoluteRotation()* and *axesRotations()*. This beam orientation only depends on the initial node positions and is independent of the body motion. Therefore it is combined with a reference orientation from the *bottomAdapter* to calculate an approximation of the local beam orientation, assuming small flexible body motion, which is sufficient for rigid body motion. Since the multiplicity of external floating frame connectors rely on correct orientation, each frame orientation is exactly calculated from a combination of reference orientation and local elastic rotation.

Adapters Since the structure is built by 3D-Bernoulli beams, which have FEM-nodes with node position, cut force, cut torque, elastic displacement, and elastic rotation as variables, adapters between the *FemFrame* connectors and the common Modelica *Frame* connectors, not having the elastic deformation variables but the frame orientation in addition, are needed. Two different adapters are used: *AdapterFemToFrame_fixed* and *AdapterFemToFrame_free*. The “fixed” adapter (*bottomAdapter*), where the boundary conditions are set, is needed for the structure node that will be connected to the *FreeMotion*, while the “free” adapter is for connecting any other components, such as mooring lines at the fairleads (*fairleadAdapter*), or RNA on the tower top (*topAdapter*).

2.4.3 Additional Weights

Besides the main loads due to waves, wind, and buoyancy, which are already included in the partial model *SubstructurePartial*, covered in Subsection 2.4.1, additional weight due to column caps, not considered as beam elements,

and ballast have to be integrated into the model. The subcomponent *AdditionalWeightLoadElement*, similar to the *BuoyancyLoadElement*, however, just using the simple time- and position-independent weight calculation, has the weight as output, which is implemented in the vertical component of a force equation.

2.4.4 Station-Keeping System

The station-keeping system contains three different components: fairleads, mooring lines, and anchors, as schematically shown in Figure 5.

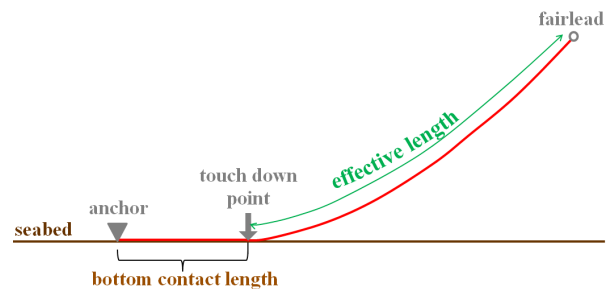


Figure 5. Schematic representation of a catenary mooring line.

Fairlead and anchor positions are defined by fixed translations, while the mooring lines are implemented by means of a separate model. This (1) models the mooring lines, divided into several elements, as mass-spring-damping systems, (2) considers velocity-dependent (Morison) and inner damping, (3) computes weight and buoyancy of the lifted parts of the catenary lines, and (4) includes bottom contact reaction forces. Thus, the shape of the mooring lines, as well as the effective lengths are internally determined at each time step, based on the common catenary equation, given mooring line parameters, and the actual fairlead positions. (Feja, 2013)

2.5 Wind

Several wind models, either deterministic, based on gust models, or stochastic, using binary or ASCII data, are available. Different gust profiles, such as 1-cosine gust, extreme coherent gust, extreme direction change, or extreme operating gust, can be selected; wind shear can be included by means of an exponential or logarithmic profile; and the tower effect can be considered either for upwind or downwind turbines. Two different guidelines can be chosen: IEC-61400-1 edition 3 (International Standard, 2005) or GL guideline for certification of wind turbines (GL Rules and Guidelines, 2010). Corresponding to this, the wind turbine class (I, II, III), depending on the reference wind speed average over 10 minutes, the turbulence characteristic (A, B, C), for high, medium, or low turbulence, as well as the turbulence model (e.g. normal or extreme) have to be specified. For the simulation of the wind, ramped, steady, turbulent, as well as upwind or downwind steady or turbulent wind types

for turbine wake simulation using two or more turbines, can be chosen. Finally, the basic parameters, such as hub wind speed, density and dynamic viscosity of air, wind direction, and flow inclination, are defined in the *WindData* record. (Thomas et al., 2014)

2.6 Waves

Two basic wave models are implemented in Modelica: one model for regular waves and one for irregular random waves. Water parameters, like water depth and density, as well as the option to use Wheeler stretching or linear extrapolation method, are common for both wave models. The regular waves are further specified by wave period, wave height, and phase angle. The irregular waves, on the other hand, are defined by a Pierson Moskowitz or JONSWAP wave spectrum, significant wave height, spectral period, random phase angles, and number of frequencies, because irregular waves are obtained as superimposition of several regular waves of different frequencies.

3 Limitations

Holistic modeling of a flexible floating offshore wind turbine system is, because of non-linear physics and fully coupled aero-hydro-servo-elastic simulation, very complex and extensive. Therefore, some simplifications have to be made in the first step of implementation, which are depicted in the following:

- Additional weights due to caps and ballast are computed for each element and connected to their *frame_c*, which is located at the midpoint along the central axis of each element. However, this does not correspond to the correct center of gravity in case of the caps and the uppermost element containing ballast, if this element is only partially filled with ballast. This inaccuracy can be removed by adding a torque load to the *frame_c* resulting from the different center of gravity, similar to the method applied in the buoyancy load calculation, described in Subsection 2.4.1.
- In case of a branched structure, like the semi-submersible floater, there is an overlap of elements. For example, the pontoons are connected to the nodes at the central axis of the columns, however, the pontoon structure itself should just start from the column surface instead of the column center. This leads to some additional incorrect weight, which has to be removed, for example by using massless elements for connecting branched elements to the surface of another element. However, the type and characteristics of those massless elements have to be chosen such that they would not affect the real structural performance.

- Wave and wind loads are actually only calculated based on the elements above and below the still water level in the initial undisplaced position, not taking into account that elements could emerge or submerge during simulation due to the motion of the floater. In addition, the wave load calculation only accounts for relative velocities but not for relative accelerations, as otherwise the initialization of the time-domain simulation in Dymola does not finish. Any loads on the submerged structure due to currents are not included. Furthermore, for correct simulation of floating offshore wind turbines in different sea states, the actual wave height has to be included in the buoyancy calculation.

Those simplifications are rather minor and do not affect the system performance in the free-decay simulations, except for the neglect of the relative acceleration in the wave load calculation, as shown in Section 4.1. However, for an offshore floating wind turbine system, which should represent accurate system performances and valid results for any simulation and load case, the above mentioned points have to be included in the model.

4 Results and Applications

The practical use of the offshore wind turbine model in Modelica is presented by analyzing simulation results based on the implemented code (Section 4.1) and pointing out the feasibility of model adaption (Section 4.2).

4.1 Simulation Results

In order to examine the developed code for a floating offshore wind turbine system, the NREL offshore 5-MW reference wind turbine (Jonkman et al., 2009) on top of a floating spar-buoy, defined in OC3⁵ Phase IV (Jonkman, 2010), is implemented in Modelica, based on the created floating wind turbine model presented in Chapter 2, as shown in Figure 6. In order to point out

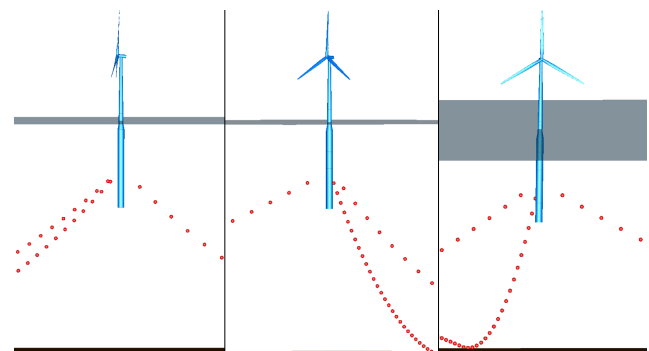


Figure 6. Visualization of the OC3-spar floating wind turbine system (top grey area: water surface, dotted red lines: mooring lines, bottom brown area: seabed).

⁵Offshore Code Comparison Collaboration

the complexity of the implemented model, some main statistics are presented in Table 1. The simulation settings and performance, as well as the hardware properties are listed in Table 2.

Table 1. Dymola statistics of translated OC3-spar wind turbine model.

Statistical Parameter	Value
Continuous time states (scalars)	866
Time-varying variables (scalars)	34,341
Sizes after manipulation of linear systems	{436, 3, 2}
DAE scalar equations	121,139

Table 2. System properties, simulation settings, and performance.

Parameter	Value
Clock frequency	3.10 GHz
Operating system	64-bit
Simulation interval	600 s
Output interval length	0.05 s
Solver	Esdirk45a
Tolerance	1.0E-4
CPU time for integration	38,041.8 s
CPU time for initialization	83.3 s

With this model, free-decay simulations, as specified in OC3 Phase IV (Jonkman et al., 2010), are carried out in Dymola⁶. OC3 mainly focuses on “(1) discussing modeling strategies, (2) developing a suite of benchmark models and simulations, (3) running the simulations and processing the simulation results, and (4) comparing and discussing the results” (Jonkman et al., 2010, pp. 1-2). The OC3 participants, together with their simulation tools, are listed in Figure 7, which represents the legend to Figure 8(a).

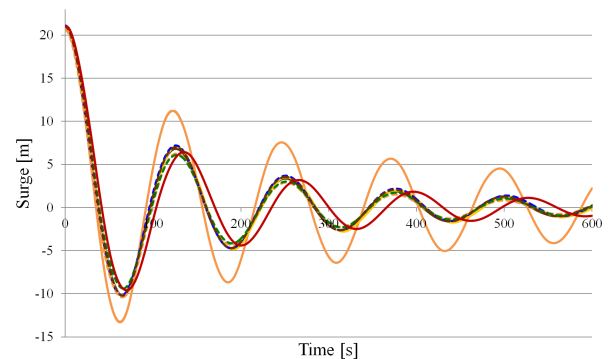


Figure 7. Participants and used simulation tools within the OC3 code-to-code comparison.

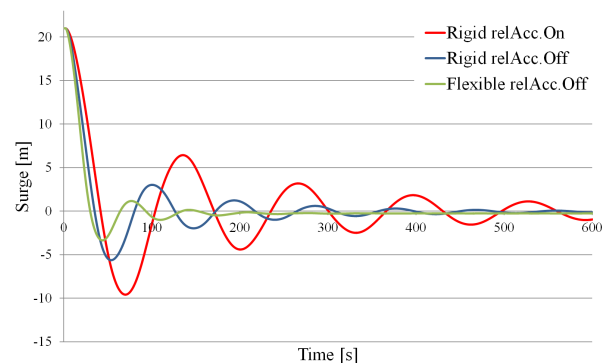
The free-decay tests are performed with the fully flexible support structure, while the turbine is modeled as rigid structure. Furthermore, aerodynamic damping is deactivated, so that the hydrodynamic damping can be elaborated in detail. The obtained motion response

time series are then compared with the results from the code-to-code comparisons (Jonkman et al., 2010). As, however, the relative acceleration is not included in the wave load calculation, the same simulations are performed with the fully rigid equivalent of the floating wind turbine model, once considering, once neglecting the relative acceleration.

Figure 8 presents the time series of the free-decay simulations exemplarily for the surge DoF. The rigid wind turbine model, taking the relative acceleration into account, yields similar results as obtained by the code-to-code comparison (Jonkman et al., 2010), shown in Figure 8(a). The effect of neglecting the relative acceleration is shown on the rigid model and compared to the results from the fully flexible model, not yet capable of taking this parameter into account. From Figure 8(b) it can be seen that the shorter eigenperiod and stronger damping, obtained by the time series of the fully flexible model, are mainly due to the disregarded relative acceleration in the wave load calculation.



(a) Rigid model and code-to-code comparison results, legend given in Figure 7



(b) Consideration and neglect of relative acceleration

Figure 8. Free-decay time series in surge.

The system response by the end of the decay process turns out to depend on the chosen solver. This, however, is expected to be caused by the damping parameters set in the *TopologyData* record of the *StructureElement*. At this stage, the Rayleigh damping parameters are computed manually, based on the system eigenfrequencies, and

⁶<http://www.3ds.com/products-services/catia/products/dymola> (Accessed: 22 August 2016)

used for all beam elements. However, for the sake of accuracy and in order to obtain more realistic estimates for the structural damping parameters, it is recommended to compute those Rayleigh damping parameters for each beam element individually and internally within Modelica.

4.2 Model Adaption

Due to the hierarchical programming in Modelica and the multibody approach, single components can easily be adapted or exchanged. This way, other floating wind turbine designs, bottom-fixed offshore or even onshore wind turbine systems can be modeled, using the basic structure of the implemented fully flexible model for a floating offshore wind turbine system. Thus, the presented model can be used as a simple tool for elaborating new research topics and different or innovative wind turbine system designs.

This flexibility of model adaption is demonstrated on the example of the OC4 semi-submersible platform (Robertson et al., 2014), the OC3 tripod (Nichols et al., 2009), and the OC4 jacket (Jonkman et al., 2012), shown in Figures 9(a), 9(b), and 9(c), respectively. Furthermore, Table 3 compares the complexity of those models, using the same statistics from Dymola as presented in

Table 1 for the spar-buoy floating wind turbine model. This underlines the enlarged calculation effort due to the increased number of system parameters, which comes with more complex and highly branched structures. But nevertheless, it is feasible to model and simulate very sophisticated wind turbine system designs.

5 Conclusion and Outlook

This paper presents the modeling of a fully flexible floating offshore wind turbine system in Modelica. Based on the Modelica MultiBody Library and the hierarchical programming structure in Modelica, the complex system is implemented via six main components and several subcomponents. Floating systems bring new challenges, such as buoyancy, free motion, station-keeping system, as well as relative velocities and accelerations. Furthermore, closed loops have to be avoided, when handling branched structures in multibody applications, and certain adapters, as well as a special load frame, are needed to connect external components to the flexible Bernoulli beams.

Due to the complexity of fully flexible modeling of a floating offshore wind turbine system, some simplifications are made. Most of them have minor impact on the behavior of the system and the simulation results. Never-

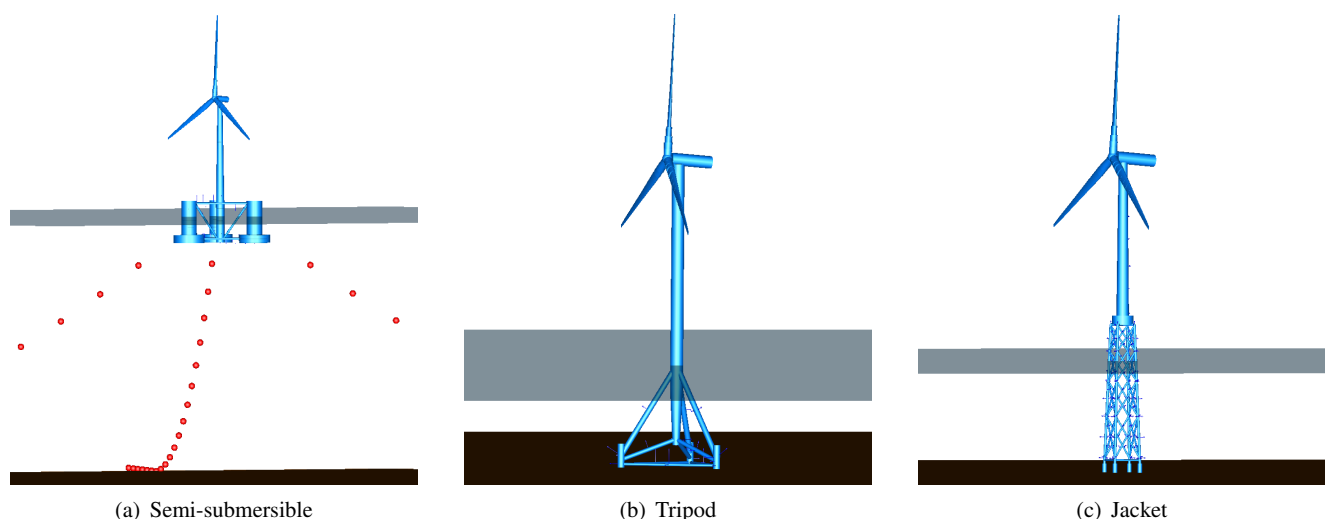


Figure 9. Other wind turbine systems, implementation based on the basic model.

Table 3. Dymola statistics of translated adapted wind turbine models.

<i>Statistical Parameter</i>		<i>Semi-submersible</i>	<i>Tripod</i>	<i>Jacket</i>
Continuous time states (scalars)		1,658	796	2,056
Time-varying variables (scalars)		64,139	35,079	75,425
Sizes after manipulation of linear systems	Vector length	78	94	366
	Maximum value	983	499	1,681
DAE scalar equations		237,246	148,435	374,173

theless, the proposed methods for accurate modeling have to be implemented in the next stage. More challenging and more relevant for correct simulation results, however, is the inclusion of the relative acceleration in the wave load calculation. In order to account for this, further work on an alternative way to connect external loads to the structure elements is in progress. In this approach, each structure element is made up of two Bernoulli beams, while the mid-node is added separately and connected to a “free” adapter. To avoid too small beam elements, the *TopologyData* record is adapted and the number of nodes and members is reduced. A more realistic estimation of the Rayleigh damping parameters, which could be directly included as internal computation within Modelica, is as well of relevance for obtaining correct system responses. Further fine tuning and detailed examination of the environmental models should be carried out in order to get even closer to the reference results. Finally, with regard to the computational effort and simulation performance, additional work on speeding up the initialization process is recommended.

Thus, the presented model should be seen rather as a work in progress than as a fully established Modelica code, as further work on small but important details is still needed for proper representation of fully flexible floating wind turbine systems. However, the implemented model is a very good basis for simulation of fully flexible floating offshore wind turbines and already reproduces the dynamics of such a complex system quite well. Furthermore, using the Modelica MultiBody Library and the object-oriented programming in Modelica comes with the great advantage to quickly adapt the implemented basic model. This makes it a simple tool, which can be used in other research projects and for modeling of novel wind turbine designs.

Acknowledgements

This work is financially supported by the German Federal Ministry of Economics and Technology, funding code 0325841A.

References

- P. Feja. *Dynamische Modellierung von Verankerungsleinen für schwimmende Offshore-Windenergieanlagen mit Modelica*. Bachelor's Thesis, RWTH Aachen University, Fraunhofer Institute for Wind Energy and Energy System Technology (IWES), Germany, 2013.
- GL Garrad Hassan. *V4 Bladed Multibody dynamics*. Garrad Hassan & Partners Ltd., Bristol, UK, Bladed User Manual, Version 4.0, 2010.
- GL Rules and Guidelines. *IV: Industrial Services, Part 1: Guideline for the Certification of Wind Turbines*. Germanischer Lloyd, Hamburg, Germany, 2010.
- International Standard. *Wind turbines - Part1: Design requirements*. International Electrotechnical Commission, Geneva, Switzerland, IEC 61400-1, 3rd edition, 2005.
- J. Jonkman. *Definition of the Floating System for Phase IV of OC3*. Technical Report NREL/TP-500-47535, National Renewable Energy Laboratory (NREL), Golden, Colorado, USA, 2010.
- J. Jonkman, S. Butterfield, W. Musial, and G. Scott. *Definition of a 5-MW Reference Wind Turbine for Offshore System Development*. Technical Report NREL/TP-500-38060, National Renewable Energy Laboratory (NREL), Golden, Colorado, USA, 2009.
- J. Jonkman, T. Larsen, A. Hansen, T. Nygaard, K. Maus, M. Karimirad, Z. Gao, T. Moan, I. Fylling, J. Nichols, M. Kohlmeier, J. Pascual Vergara, D. Merino, W. Shi, and H. Park. 'Offshore Code Comparison Collaboration within IEA Wind Task 23: Phase IV Results Regarding Floating Wind Turbine Modeling'. In *2010 European Wind Energy Conference and Exhibition (EWECE)*, Warsaw, Poland, April 2010. Conference Paper NREL/CP-500-47534. doi: 10.13140/2.1.3576.5768.
- J. Jonkman, A. Robertson, W. Popko, F. Vorpahl, A. Zuga, M. Kohlmeier, T.J. Larsen, A. Yde, K. Saeterstro, K.M. Okstad, J. Nichols, T.A. Nygaard, Z. Gao, D. Manolas, K. Kim, Q. Yu, W. Shi, H. Park, A. Vasquez-Rojas, J. Dubois, D. Kaufer, P. Thomassen, M.J. de Ruiter, J.M. Peeringa, H. Zhiwen, and H. von Waaden. 'Offshore Code Comparison Collaboration Continuation (OC4), Phase I - Results of Coupled Simulations of an Offshore Wind Turbine with Jacket Support Structure'. In *Proceedings of the 22nd International Society of Offshore and Polar Engineers Conference, 17-22 June 2012, Rhodes, Greece*, pages 337–346, June 2012. Conference Paper NREL/CP-5000-54124.
- T.J. Larsen and A.M. Hansen. *How 2 HAWC2, the user's manual*. Risø-R-Report Risø-R-1597(ver. 4-5), Risø National Laboratory, 2014.
- J. Nichols, T. Camp, J. Jonkman, S. Butterfield, T. Larsen, A. Hansen, J. Azcona, A. Martinez, X. Munduate, F. Vorpahl, S. Kleinhansl, M. Kohlmeier, T. Kossel, C. Böker, and D. Kaufer. 'Offshore Code Comparison Collaboration within IEA Wind Annex XXIII: Phase III Results Regarding Tripod Support Structure Modeling'. In *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition, 5-8 January 2009, Orlando, Florida, USA*, January 2009. Conference Paper NREL/CP-500-44810.
- M. Otter. *Modeling, Simulation and Control with Modelica 3.0 and Dymola 7*. Technical Report, Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR) - Institut für Robotik und Mechatronik, Wessling, Germany, 2009.
- A. Robertson, J.M. Jonkman, F. Vorpahl, W. Popko, J. Qvist, L. Frøyd, X. Chen, J. Azcona, E. Uzunoglu, C. Guedes Soares, C. Luan, H. Yutong, F. Pengcheng, A. Yde, T.J. Larsen, J. Nichols, R. Buils, L. Lei, T.A. Nygaard, D. Manolas, A. Heege, S. Ringdalen Vatne, T. Duarte, C. Godreau, H.F. Hansen, A.W. Nielsen, H. Riber, C. Le Cunff, F. Beyer, A. Yamaguchi, K.J. Jung, H. Shin, W. Shi, H. Park, and M. Alves. 'Offshore Code Comparison Collaboration

- Continuation Within IEA Wind Task 30: Phase II Results Regarding a Floating Semisubmersible Wind System'. In *Proceedings of the ASME 2014 33rd International Conference on Ocean, Offshore and Arctic Engineering, OMAE 2014, San Francisco, California, USA*, volume 9B, OMAE2014-24040, page V09BT09A012. American Society of Mechanical Engineers, June 2014. doi:10.1115/OMAE2014-24040.
- M. Strobel, F. Vorpahl, C. Hillmann, X. Gu, A. Zuga, and U. Wihlfahrt. 'The OnWind Modelica Library for Offshore Wind Turbines - Implementation and first results'. In *Proceedings of the 8th International Modelica Conference 2011, Dresden, Germany*, pages 603–609. Modelica Association, March 2011.
- P. Thomas, X. Gu, R. Samlaus, C. Hillmann, and U. Wihlfahrt. 'The OneWind[®] Modelica Library for Wind Turbine Simulation with Flexible Structure - Modal Reduction Method in Modelica'. In *Proceedings of the 10th International Modelica Conference 2014, Lund, Sweden*, pages 940–948. Modelica Association, March 2014. doi:10.3384/ECP14096939.
- T. Tran, D. Kim, and J. Song. 'Computational Fluid Dynamic Analysis of a Floating Offshore Wind Turbine Experiencing Platform Pitching Motion'. *Energies*, 7(8):5011–5026, 2014. doi:10.3390/en7085011.
- W. Yu. *Dynamic Modeling of a Floating Wind Turbine*. Technical Report, Fraunhofer Institute for Wind Energy and Energy System Technology (IWES), Germany, 2015.

Modelica Based Naval Architecture Library for Small Autonomous Boat Design

Thom Trentelman¹ Joshua Sutherland² Kazuya Oizumi² Kazuhiro Aoyama²

¹Maritime Engineering, Delft University of Technology, The Netherlands,
t.r.trentelman@student.tudelft.nl

²Systems Innovation, University of Tokyo, Japan, {joshua, oizumi, aoyama}@m.sys.t.u-tokyo.ac.jp

Abstract

This paper describes a method for early stage boat design by creating and utilizing a library of naval architecture based boat components in Modelica. The method involves the construction of stand-alone boat components which can be assembled into a simulation model. Structuring the model into multiple system levels provides a clear overview. Utilizing the partial-complete methodology ensures that all system levels are replaceable within the simulation. This allows the user to construct many different boat models and experiment with unconventional or innovative designs. By comparing the performance and behaviour of different assemblies of components the most ideal design for a given purpose can be found and used as a starting point for the in-depth design process. By organizing the components in a library they can be re-used in future projects as well. It is noted that when additional libraries are utilized the effectiveness of this design method increases significantly. As the availability of component models increases, users can save time on the physical design and modelling of the individual components and instead focus on assembling working simulation models right from the beginning. To illustrate this, the construction of a few simple boat components is described in this paper. These components are then combined to simulate multiple concept designs.

Keywords: early stage ship design, model based design, object-orientated, innovative naval architecture

1 Introduction

Shipping nowadays is one of the world's most important means of transport. Not only for product or passenger transport, but also for leisure such as pleasure yachts and sport boats. The volume of world seaborne trade exceeded 10 billion tons in the year 2015. Together this volume was accountable for more than 80% of the total worldwide merchandise trade, growing with 2.1% over the year 2014 (UNCTAD/RMT, 2016). A downside to this key global trade enabler is the overall environmental pollution that comes from the industry. In 2012 ship emissions were responsible for 3.4% of the worldwide

emissions of greenhouse gasses (GHG), ranking 2nd as transport related polluter after road transport (24.3%) and just above aviation (3.1%). Although road transport is the major source of transport related emissions, shipping is the only sector where the GHG emission rates are still rising (Maragkogianni, Papaefthimiou, & Zopounidis, 2016). A publication of the International Maritime Organisation (Buhaug et al., 2009) described a scenario where without political involvement the growth of ship's GHG emissions would grow by 150~250% between 2007 and 2050.

1.1 The Importance of Innovative Naval Architecture Design

The design process of ships from scratch to seaborne is costly and requires large amounts of detailed analysis, is subject to extensive safety and environmental legislation and to remain competitive, the time-span for this design process is ever decreasing. There is little room for uncertainty or assumptions. And whilst most drastic and influential decisions are made in the early stages of the design process, most design software tends to emphasise more on the detailed phases in later stages of the design process (Abt, Bade, Birk, & Harries, 2001; Bole & Forrest, 2005). For these reasons ship design often loses its innovative character and mostly evolves around the slight improvement of existing designs.

However, due to international political involvement the imposed legislation on the emission of GHG has increased in the recent years. Stricter emission quotas and the introduction of eco-zones at major ports have forced the industry to innovate (Rue & Anderson, 2015). Moreover, not only stricter legislation has moved the industry; in recent years the direct influence of global warming have exposed itself more clearly where conditions on sea have become more challenging for existing vessels due to the changing climate. Reports of an increase in storms and higher rising waves have set high performance demands for the vessels in order to maintain safety for their passengers and crew on board (Bitner-Gregersen, Eide, Hørte, & Skjong, 2013).

1.2 A New Method for Innovative Early Stage Concept Design

Using Model Based Design (MBD) in the early design stage makes it possible to experiment with innovative techniques. For example, 3-Dimensional simulation gives direct performance output of the created concept. However, the level of detail of the simulation output will vary greatly based on the complexity of the simulation model. This results in the necessity for further investigation. This in-depth design stage follows a spiral path. Such a design path has an iterative character; the concept design should run down all design stages multiple times. At the end of every iteration the concept design should be adjusted to ultimately result in a suitable design (Papanikolaou, 2014). Even though eventually this high detail design path is inevitable for every concept design, this method only states whether the chosen concept is suitable for its goal or not after one or more iterations. When the concept turns out to be unsuitable for its goal, the process must start over from the very beginning. Although this method allows innovative decisions to be made regarding concept designs, it does not solve the time issue.

In this paper, we present a complimentary method utilizing the Modelica modelling language in the early design stages, with the simulation model subdivided into multiple system levels. The advantage is that these system levels have replaceable properties, with every system level consisting of one or more components. The components can be anything the user wants to attach to his simulation model e.g. thrusters, motors, solar panels, gearboxes and batteries. Connecting these components all the way through to the top level creates a complete simulation model arranged in a hierarchy to aid the management of complexity. By swapping components within these levels multiple concepts are established on the same structure. Still the level of detail of the simulation output is heavily reliant on the complexity of these individual components. However, at this design phase the suitability of the designs can be determined by comparing the performance of various concepts. Only the most promising concept designs will be subjected to the advanced design stage. Meanwhile all created components can be exchanged or stored in a library for use in later projects.

2 Background to Model Based Design and its Application in Naval Architecture

2.1 Model Based Design

Model Based Design (MBD) is a design method which enables the rapid creation of design concepts in software by leveraging advances in computing technology.

While various modelling languages exist their choice to be used on a specific project and lifecycle stage within

that project must be made based on what value they provide. Conceptual modelling languages such as SysML provide a structure to describe a system and its purpose, but fail to provide feedback on the performance of the system being designed. Conversely 3D CAD, can give clear and direct feedback on the geometry of a system but requires the designer to commit to a great deal of detail which often occurs later in the design lifecycle.

1 Dimensional Computer Aided Engineering (1D CAE) is a broad term used to cover methodologies and tools which aid the early stages of engineering lifecycles by the utilization of computers (Sawada, 2012). By deliberately neglecting 3D geometry the engineer can quickly prototype designs and crucially; predict the performance by simulating the models created rather than creating a physical prototype. Likely leading to cost and time savings for a given project and making it easier to complete difficult trade-off decisions which must be made to select a combination of components to form a system which serves its purpose best.

Modelica is an example of a numerical modelling language popular in the 1D CAE paradigm. Equations capture the behaviour of individual components which are then connected together to develop subsystems which ultimately form the system being modelled. Its object orientated features make it possible to quickly create and modify models including the sharing of interfaces and inheritance of common attributes. Features of which are vital for the handling highly complex systems being developed by large teams.

This is particularly useful when attempting to create highly innovative designs where experimental or extraordinary components can be realized as models by their particular domain experts and integrated into a system by an engineer who is not a specialist in those particular domains.

2.2 Current Applications of Modelica to Naval Architecture

A review of the current literature shows the use of Modelica in the naval architecture has been focusing on the design of highly detailed specific systems aboard ships. For example (Dong, Wu, Zhang, & Peng, 2011) have constructed a simulation model of a hydraulic rudder and used it for the analysis of shock resistant effects while (Marty, Corrigan, Gondet, Chenouard, & Hétet, 2012) focussed on simulating energy flows and fuel consumption on board of a large cruise vessel.

The high level of detail of these simulation models is typical for the advanced design stage of a ship or boat and as such these stand-alone simulations are not meant to be integrated into a larger holistic simulation model of the entire ship or boat. The amount of detail in these individual component models would be excessive for the construction of an early design stage model.

Through the course of a literature review no prior attempt at using Modelica to model an entire ship or boat was found. As such we conclude the navel architecture preliminary design process is likely still reliant on other software languages other than Modelica and so is not experiencing the benefits described in Section 2.1 of this paper.

3 Contributions of this Paper

Earlier literature from authors including (Sutherland, Oizumi, Aoyama, Eguchi, & Takahashi, 2016; Sutherland, Oizumi, Aoyama, Takahashi, & Eguchi, 2016; Sutherland, Salado, Oizumi, & Aoyama, 2017) has focused on the practical use of a Modelica based naval architecture library. It describes the process of designing a race winning model boat for an annual student contest. The boat model needed to be rather innovative as it was assigned to sail autonomously on solar energy. This paper will describe the details and architecture of the navel architecture library used.

The focus will be on the replaceable components, system level characteristics and the hierarchal structure. Successively the architecture of several components will be described to aid reader comprehension.

Through this paper the authors aim to inspire engineers from the naval community to take benefit from advanced design methods and tools. The effectiveness of the design method will grow exponentially as more naval component model libraries become available.

4 Methodology for Creating Replaceable System Levels

As shown in Figure 1 the simulation model is subdivided into four systems levels. All system levels are linked to one another and ultimately make up the Solar Boat Model. All system levels are replaceable since they are all constructed using the partial-complete methodology. With one click a component, subsystem or system of interest can be swapped for an alternative.

4.1 System Levels

The hierarchical structure provided by the four system levels offers a clear overview of the simulation model as demonstrated in Figure 1. In Modelica these system levels are introduced as separate *packages*. Underneath these, every system level has their own sub-*packages* containing relevant models.

At the bottom level (Level 4) all individual components are arranged in a library. This library provides fundamental elements to the simulation model, where other system levels are introduced for enabling quick and drastic concept changes. The number of possible components is infinite and the level of complexity per component is variable. Components can often be broken down into separate sub-components. These sub-components can then be reconnected at the subsystem level. However, it is important to acknowledge that this method is used in the early (lower detail) design stage.

The subsystem level (Level 3) contains assemblies of Level 4 components which interact one-on-one to function properly in a system. Every unique

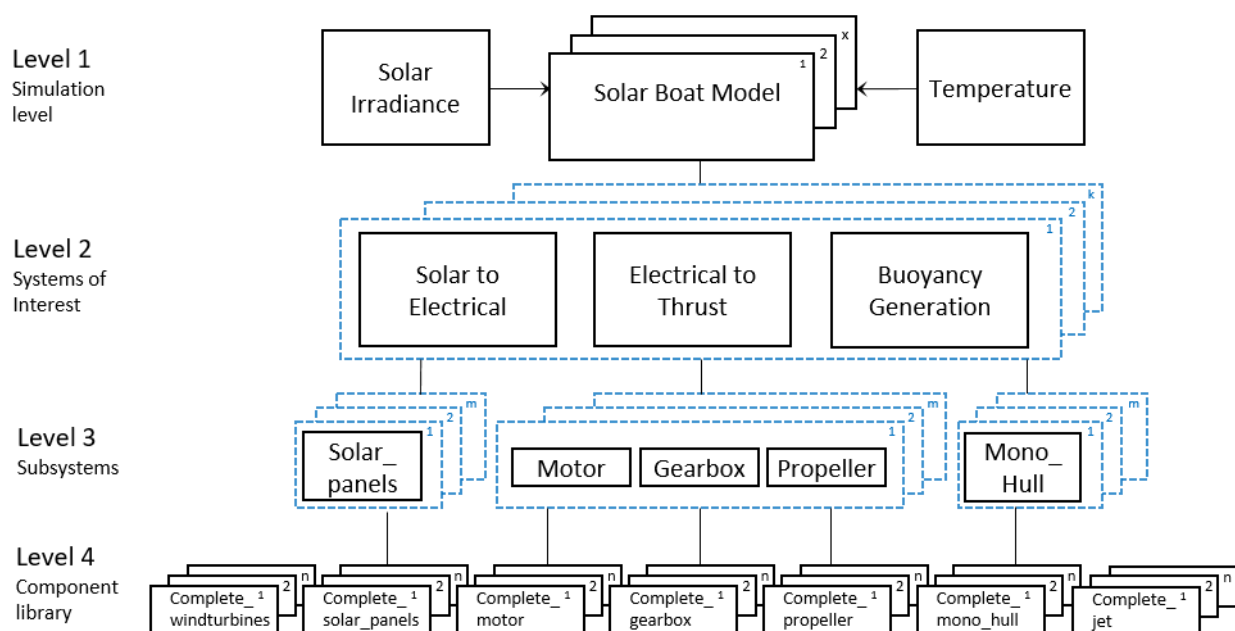


Figure 1. Schematic view of system level hierarchy; enabling the creation of several ready to simulate Solar Boat models by alternating various replaceable complete components from the library (Level 4). System Levels 2 and 3 are also replaceable as shown in Figure 4 and Figure 5.

combination of components within a subsystem requires a different subsystem name at this level.

The systems of interest level (Level 2) combines all subsystems which will be used in the concept. It is at this level that changes will radically influence the appearance of the concept design.

At the top level (Level 1) the final concept design is assigned and connected with external factors. These external factors could be weather conditions, but also payloads or obstacles in the simulated space the vehicle operates in. At this level the model is complete and ready to simulate.

4.2 Partial-Complete Methodology

The partial-complete methodology, as the name suggests, makes use of a partial model to create multiple complete models of the same type. This object-oriented inheritance scheme uses a partial model to state the characteristic parameters and equations for every component, subsystem or system of interest. Each complete model then extends their corresponding partial model and assigns real values to the parameters. The partial-complete methodology allows the models to be replaceable and enables the rapid creation of multiple complete components using the same architecture. Figure 2 shows a flow-chart which demonstrates the methodology for a simplified (Level 4) component.

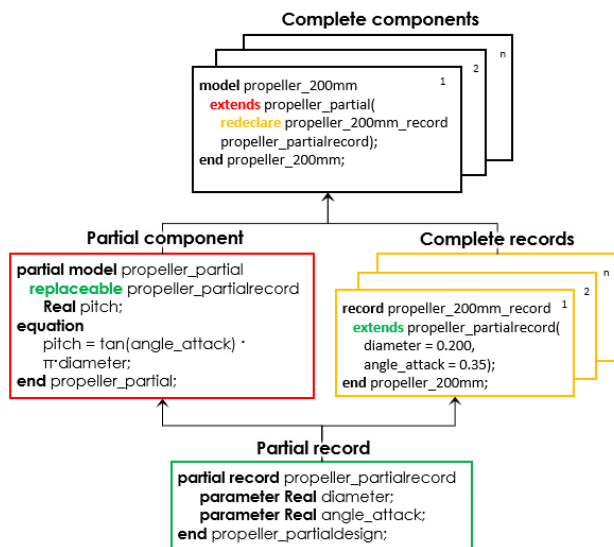


Figure 2. Flow chart demonstrating the partial-complete methodology for creating a replaceable propeller component (Level 4). The component parameters and equations have been heavily simplified.

4.2.1 Component Modelling (Level 4)

Not only do all components have to be replaceable, for experimenting it is also desirable to make the components easily adjustable and scalable. The use of records ensures this. A record formulates all constant parameters which are used in the physical description of the component. The very base of a component is a partial record.

A partial record states all the parameters without assigning a value to them. For example, these parameters could be dimensions, mass, cost, conductivity, material characteristics and cost. A complete record extends the partial record and ultimately assigns values to the parameters. By creating various complete records with alternating values, every time extending the same partial record, many different components of the same type can be created at the one instant. Using the partial-complete methodology on a record level ensures that components of the same type are using identical parameters.

The partial component is used to describe the outlines of the component and can include both interfaces and behaviour. When used for describing the outline of a component, it must declare replaceable the partial record and therefore gain access to the parameters. These parameters must then be complemented with additional variables which will solely be used in the characteristic set of equations of the component. Note that every time a component desires a new set of equations, a new partial component has to be created.

In order to make the components connectable to other components all output forces expressed by the different components must be attached to the same frame. To enable this a connector from the Modelica Mechanics.Multibody library is used in the *Diagram* view of the partial specifically: `Interfaces.Frame_a`. By using this library the Modelica compiler can automatically combine the forces and torques excreted by the components and compute the resulting net acceleration on the entire boat assembly. In some cases (e.g. components affected by hydrodynamics) it is useful to attach `Mechanics.Multibody.Sensors` to the frame. The data from this sensors could be used in the equations of the same partial model (e.g. compute drag from velocity). Lastly, for a clearer understanding of the 3D simulation of the model `Mechanics.Multibody.Visualizers` can also be attached to the frame. The template for the *Diagram* view is shown in Figure 3.

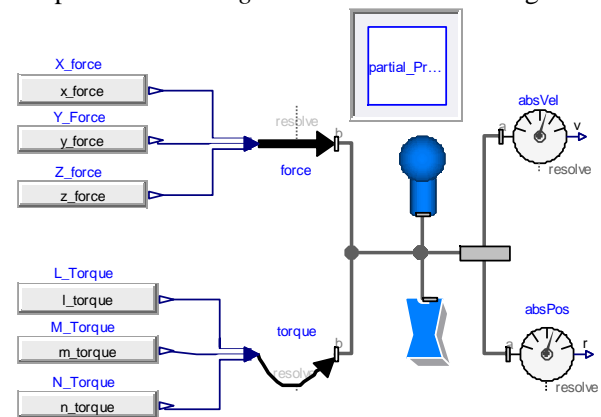


Figure 3. Diagram template used for creating general boat components (Level 4).

The last step for creating a Level 4 component is to create a complete component. The complete model extends its corresponding partial component and inherits the values for the parameters by re-declaring the partial record with the chosen complete record. Every complete record should be re-declared in a complete model in order to create the eventual replaceable components.

4.2.2 Subsystem Modelling (Level 3)

Subsystems (Level 3) are models that state one or more component types which the designer decides should be grouped together. In the partial subsystem the partial components are assigned. Constraining the replaceable partial components to their own types (complete components) makes a subsystems distinctive. This is illustrated in the following Modelica code for a partial Electrical to Thrust partial subsystem which is also shown as a diagram in the central layer of Figure 4:

```
model ElectricalToThrust
  replaceable motors.motor_partial
    constrainedby motor;
  replaceable gearbox.gearbox_partial
    constrainedby gearbox;
  replaceable propellers.propeller_partial
    constrainedby propeller;
end ElectricalToThrust;
```

Clearly, within a subsystem, components should be connected in using the appropriate connectors (i.e. mechanical, electrical, etc.). And to enable interaction with other subsystems external connectors must be added.

The complete subsystem is constructed in the same way as at the component level. First it extends the partial subsystem, then re-declares the partial components with their corresponding complete components. Figure 4 demonstrates the construction of two different complete subsystem architectures by alternating the thruster component (propeller to jet).

4.2.3 System of Interest Modelling (Level 2)

The Systems of Interest level (Level 2) is the level the full architecture of the simulation model is brought together into a complete concept design. The method for coding is no different than on the subsystem level. Again the subsystems should be connected at this level as required. Figure 5 demonstrates the construction of two different complete systems of interest by alternating the energy source subsystems.

4.2.4 Simulation Level Modelling (Level 1)

The simulation level assigns the concept design from the System of Interest level. On this level external factors

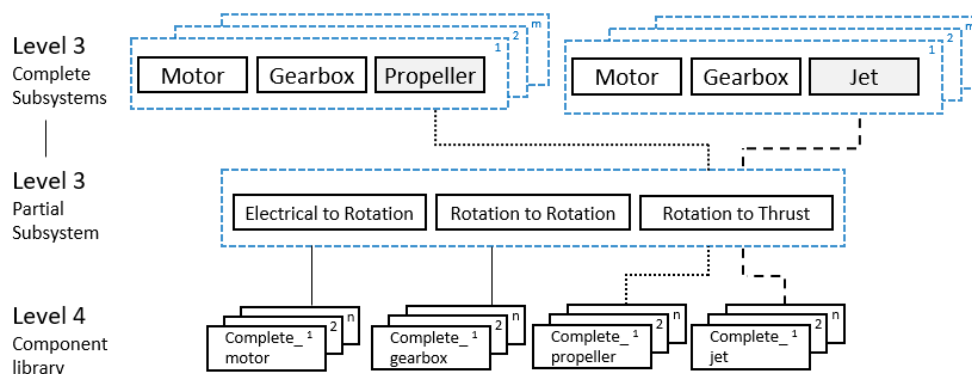


Figure 4. Schematic view of the partial-complete methodology as used for the creation of replaceable subsystems (Level 3). In this specific example both replaceable complete subsystems will use a motor and a gearbox. The difference comes from the variation of propeller (dotted line) and jet (dashed line) as main thruster.

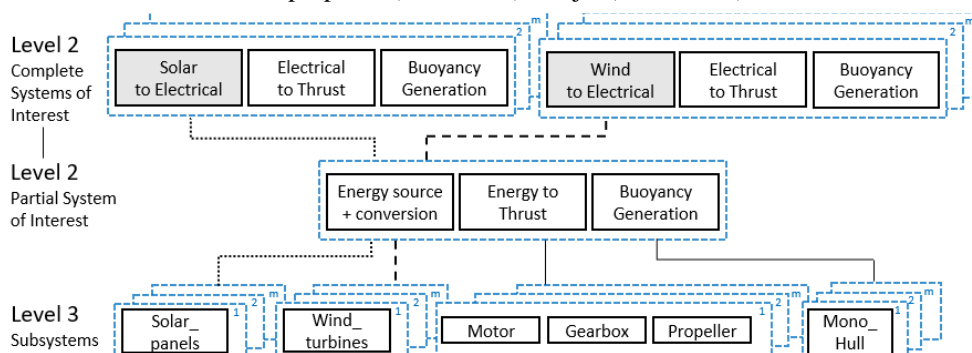


Figure 5. Schematic view of the partial-complete methodology used for the creation of replaceable systems of interest (Level 2). In this specific example both replaceable complete systems of interest will use a motor, gearbox and propeller propulsion system, as well as a mono-hull buoyancy generation. The difference comes from the variation of solar panels (dotted line) and windmills (dashed line) as main energy source.

such as weather can be linked to the concept design, making it ready to simulate.

5 The Modelica Naval Architecture Library

This chapter demonstrates the practical application of the design method as described in Chapter 4 in order to construct a complete simulation model. The simulation model will be a solar powered model boat. Although this design method has the potential to model the simulation model with six degrees of freedom (see Figure 3), for reasons of simplification this paper will model in only two degrees of freedom. This means that the eventual simulation output will only contain a boat travelling in a straight line and moving up and down along its vertical axis.

5.1 The Library: Component Modelling (Level 4)

This section describes the individual components which are used in the simulation model. Even without precise knowledge about the physics of every type of component, it is still possible to construct functioning components by yourself. This method allows using existing literature to formulate component characteristics and equations. This makes it possible to construct components even without knowledge or thorough research. In this paper the solar panel components are based on existing literature. In some occasions components, such as motors and gearboxes, could be implemented from the *Modelica Standard Library*. As a result, very few parameters and equations have to be formulated in the partial models, keeping the components simple. In other occasions the components will be based on the authors' expertise, resulting in more complex, yet more adjustable components (as shown with the propeller and buoyancy components later in the paper).

5.1.1 Solar Panel Components

The solar panel components are implemented as modelled by (Esrām, 2010). Most parameters are pre-defined and do not need to be changed. However, for simulation purposes the solar panel components needs additional parameters such as surface area, mass and cost. These additional parameters should be formulated in the partial record along with the other pre-defined parameters and later assigned by a complete record in a complete component.

Also it is possible to vary the distribution of the solar panels over the length of the boat model. Every unique distribution requiring a different partial component, later extended by corresponding complete components.

For a solar panel to generate electricity it needs solar irradiation. This solar radiation needs to be modelled on the simulation level (Level 1). In order for this external

value to reach the solar panel component it needs to be implemented as an *input* in the *Modelica Diagram* view of the partial solar panel component.

5.1.2 DC Electrical Motor and Gearbox Components

The electrical motor component and the gearbox component are selected from the *Modelica Standard Library*. Without any adjustment these component will respond to the appropriate inputs and generate an output. These inputs and outputs need to be connected within a subsystem, and on the system of interest level.

Both the electrical motor and the gearbox need some additional adjustment before they can be successfully implemented in the simulation model. Just like the solar panel component, the motor and gearbox need to have parameters for size, mass and cost assigned. Also, it is important to determine a gearbox ratio. Which of course is also adjustable.

Every time an adjustment in one of the parameter values is desired, a new complete component has to be created, extending the partial component.

5.1.3 Propeller Component

The propeller is a propulsion device often mounted at the stern of the boat. Although there are many different types of propulsion devices, the propeller is the most commonly used. A boat can be designed having more than one propeller. The direction of the propeller can be fixed or with the ability to rotate around its vertical axis in order to adjust the direction of the boat, like the propellers on typical outboard motors. In this simulation, a fixed single propeller is used as the thruster of the boat.

The propeller generates the thrust which will ultimately displace the boat in the simulation. The thrust is calculated using the basic equation:

$$Thrust = k_t \cdot \rho \cdot n^2 \cdot D^4 \quad (1)$$

At the same time a propeller generates a torque. This torque is described by using a similar basic equation:

$$Torque = k_q \cdot \rho \cdot n^2 \cdot D^5 \quad (2)$$

Where (ρ) is the density of the water, n is the revolutions per second of propeller, D the diameter of the propeller and ($k_t(J)$) and ($k_q(J)$) are the parametrised thrust and torque coefficients, respectively. Since ρ is determined by the environment and n is as a result of interaction with torque from either the gearbox or the motor, only the diameter and the coefficients have to be formulated in the partial record of the propeller.

The coefficients are found by a 4th-order polynomial in the form of $a \cdot J + b \cdot J^2 + c \cdot J^3 + d \cdot J^4$, where J is the advance ratio of the propeller. The advance propeller is determined using the equation:

$$J = \frac{V_a}{n \cdot D} \quad (3)$$

Where V_a stands for the advance velocity (the velocity of the water as it reaches the propeller, which in practice is influenced by the shape of the hull). For simplification, the advance velocity in this simulation is equalled to the absolute velocity of the simulation, the value for which is extracted from the implemented velocity sensor.

The a, b, c and d *polyfits* within the 4th order polynomial are determined for a particular propeller design by iterative calculation using a calculation executed in Matlab as described by the blade element theory presented by (Auld & Srinivas, 2016). But of course experimentally derived functions could also be used.

5.1.4 Buoyancy Component

The buoyancy component describes the hull used in the simulation and the interaction with its surrounding. The hull of a boat can come in various shapes and sizes. In addition there is possibility to design a mono-hull, double-hull or even a triple-hull. The shape and weight of the hull mainly affects the resistance from the water and the stability of the boat. However its hydrodynamic outline and the interaction with the water are extremely difficult to accurately model without resorting to calculations within the field of advanced 3D fluid dynamics. But in the preliminary design stage it is not necessary to calculate so accurately and some basic equations are sufficient as described in the following subsections.

5.1.4.1 Resistance force

To reduce complexity the resistance of the buoyancy component calculation is primarily calculated by the friction force (Equation 4) using the ITTC-57 equation (Equation 5).

$$R_{total} = R_{friction} \quad (4)$$

$$R_{friction} = \frac{1}{2} C_f \cdot \rho \cdot S \cdot V^2 \quad (5)$$

Where C_f is the friction coefficient, ρ is the density of the water, S is the wetted surface area of the hull and V is the absolute velocity of the model.

The friction coefficient itself is a function of the number of Reynolds Number, hence the following equations:

$$C_f = \frac{0.075}{(\log_{10}(Re) - 2)^2} \quad (6)$$

$$Re = \frac{V \cdot L_{wl}}{\nu} \quad (7)$$

Reynolds Number as shown in equation 7 is calculated using the length of the waterline (L_{wl}) of the hull rather than the overall length but for simplification this could be equalled to the overall length. It is multiplied with the absolute the velocity and then divided by the kinematic viscosity of the water (ν). This parameter is determined as a function of temperature and water density, but could also be implemented as a constant. For salt water with a density of 1025kg/m³ and a temperature of 15°C, $\nu = 1.188 \cdot 10^{-6}$ is used.

The wetted surface (S) is for simplification equalled to the surface of the under half of an ellipsoid and assumed to be constant:

$$S = \frac{1}{2} \cdot 4\pi \frac{((ab)^{1.6} + (bc)^{1.6} + (ac)^{1.6})^{\frac{1}{1.6}}}{3} \quad (8)$$

Where a is the overall length of the hull, b the greatest width and c the depth, vertically measured from the bottom to top of the ellipsoid.

The frictional resistance always works in the opposite direction of the traveling direction of the boat. More resistance components such as air resistance and wave-making resistance could be added in order to increase the accuracy of the resistance simulation but they are neglected at this time.

5.1.4.2 Buoyancy force

The buoyancy force keeps the vessel floating on the water and maintains its stability. It has an upward force component along the z-axis and restoring moments around the x-, y- and z-axes of the construction. In the simulation the restoring moments have been neglected for simplification. The equation for the buoyancy force along the z-axis then becomes:

$$F_{buoyancy} = \nabla \cdot \rho \cdot g \quad (9)$$

The water displacement (∇) would normally be calculated using the block coefficient, length of the water line, breadth and draft of the hull. The equation is as follows:

$$\nabla = C_b \cdot L_{wl} \cdot B \cdot T \quad (10)$$

However, to remain consistent with previous calculations, the displacement will again be calculated using an ellipsoid equation:

$$\nabla = \frac{1}{2} \cdot \frac{3}{4} \pi abc \quad (11)$$

Again only the bottom half of the ellipsoid will be submersed. The coefficients abc are used the same as in equation 8.

5.1.5 Overhead Components

The overhead components make up for weight or cost values of components which have not been modelled or used in the simulation model. Moreover they could be used as the safety margin of the design. The overhead components have no further dimensions or forces applied other than weight.

5.2 The Library: Subsystems (Level 3)

The subsystems which are used in the boat model are Solar to Electrical, Electrical to Thrust, Buoyancy Generation and Overhead Structures (as shown in Figure 1, overhead structures excluded).

5.2.1 Solar to Electrical

The Solar to Electrical subsystem consists of solely the solar panels. The distribution of solar panels is determined at this level. The output from the solar panels is an electrical connection which has to be connected to an electrical motor. In the subsystem an `Electrical.Analog.Interfaces.PositivePin` and a `.NegativePin` are utilized. In the systems of interest level these pins can be used to connect the electrical current to the electrical to thrust subsystem (see Figure 9). Figure 6 show a possible composition for the Diagram view of the Solar to Electrical subsystem.

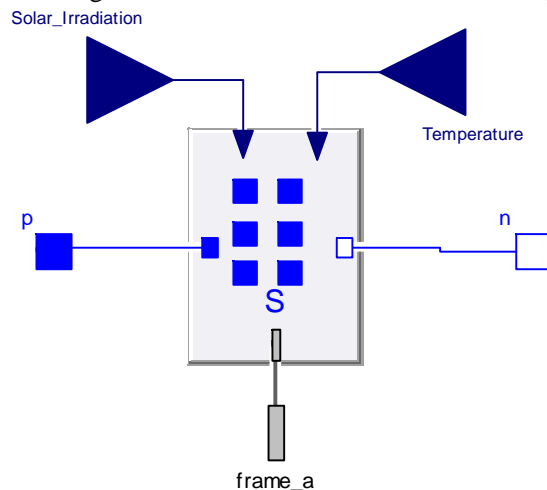


Figure 6. Diagram view of Solar to Electrical subsystem.

5.2.2 Electrical to Thrust

The electrical to thrust subsystem is responsible for the transformation of the electrical current generated by the solar panels into a thrusting force. It accesses electrical current via the connection of the pins at the system of interest level (Level 2).

Several configurations could be made to convert the electrical signal and the resulting thrusting force with every configuration should be constructed in a separate partial subsystem.

Figure 7 shows a Diagram view of an electrical to thrust subsystem containing a motor, gearbox and propeller. These components are mechanically linked to each other. The motor component has two wires with pins attached, which will be used to link this components in the system of interest level to the energy source (see Figure 9).

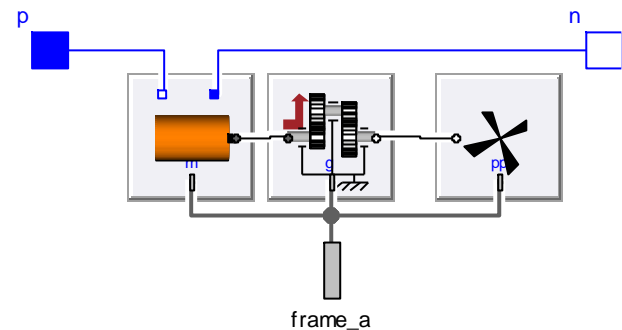


Figure 7. Diagram view of Electrical to Thrust subsystem.

5.2.3 Buoyancy Generation

The buoyancy generation subsystem is used to connect the hull component with external factors such as water velocity or wind speed. A “single hull” subsystem will contain only one hull component.

If the concept design should contain multiple hull components, as seen on catamaran or trimaran boats, a separate partial subsystem must be created. Two or more replaceable components can be implemented and attached to the frame.

Figure 8 show a Buoyancy Generation subsystem with a single hull component, connected to a water velocity input.

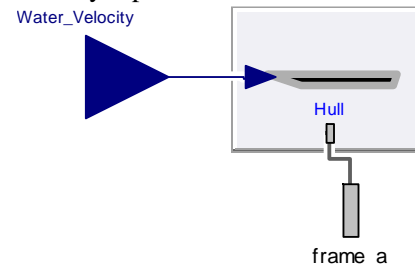


Figure 8. Diagram view of Buoyancy Generation subsystem.

5.2.4 Overhead Structures

The overhead structures subsystem is a stand-alone subsystem which assesses the overhead structure component of choice and make it possible to fit this into the simulation in the systems of interest level.

5.3 The Library: Systems of Interest (Level 2)

The systems of interest model combines various subsystems to form a model of the system for assessment (i.e. a boat). At this system-level (Level 2) it is possible to change entire subsystems with the use of the partial-complete methodology described previously.

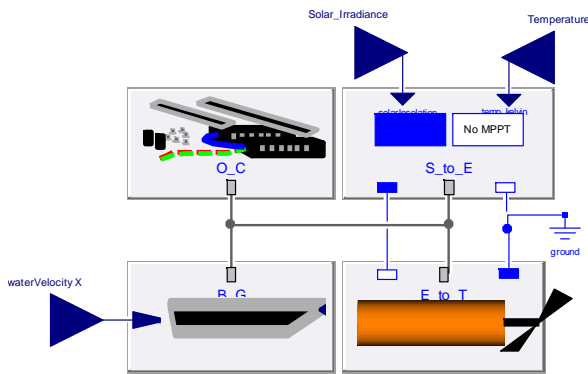


Figure 9. Diagram view of Systems of Interest (Level 2).

5.4 The Library: Simulation Level (Level 1)

The simulation level contains the chosen concept design and connects it to all the external parameters such as environmental elements or payloads. It is at this level that the simulation is ready to run. Chapter 6 gives an example of simulating multiple concept designs.

6 Simulation

In this section we present the results of simulating boat concept designs constructed in Modelica. With the use of 3D animation (by means of `Mechanics.Multibody` library) it is possible to visualize the behaviour of a particular concept (see Figure 10). While detailed performance can be analysed using charts. When multiple different concept designs are simulated, the data from the charts can be compared to determine the most suitable design for its purpose.



Figure 10. Screenshot of the 3-Dimensional simulation.

6.1 Simulating the Concepts

To demonstrate, we compare the results of simulating two competing boat designs. Both designs are based on the basic set of components as described in Chapter 5. However, they vary with Concept 1 being powered by mid-range efficiency solar panels (12.5%, 3.24kg). Demanding a higher maximum boat velocity, the second simulation will be powered by more efficient, however slightly heavier solar panels (21.5%, 5.40kg). It should be noted that the maximum allowable solar panel surface may not exceed 2 m². Six mid-range efficiency solar panels almost perfectly fill up all the available space, where six of the high efficiency solar panels only fill up 89% of all the available space. There is no room

left for adding a seventh solar panel. By simulating both boats subject to the same environmental conditions of average solar irradiance of a day in August in Japan (610 W/m²) it is possible to construct Figure 11 with time series of velocity and Table 2 containing data selected from the point when the simulation reached a steady state (simulation time = 25s). As mentioned previously no steering component is attached to the boat and hence the boat only moves in the X and Z axis.

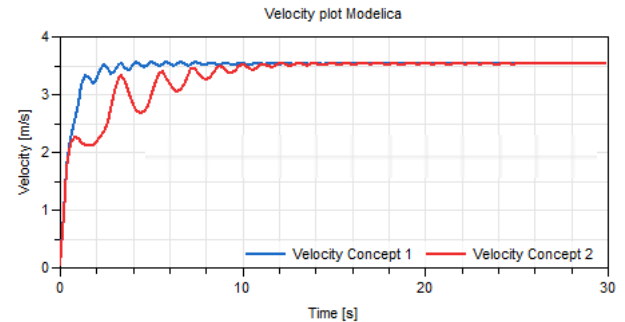


Figure 11. Simulation results of two concept designs.

Table 2. Simulation data of two concept designs.

Concept	1	2
Efficiency of solar panels [%]	12.5	21.5
Area per solar panel [m ²]	0.333	0.297
Area of six solar panels [m ²]	1.99	1.78
Mass of six solar panels [kg]	3.24	5.40
Spin speed motor [rpm]	12900	13300
Thrust generated [N]	111	118
Draft boat [m]	0.166	0.185
Maximum velocity [m/s]	3.56	3.53

6.2 Analysing the Simulation Results

The expectation would be that the design with the most efficient solar panels, generating more power, will result in a faster concept. However, by looking at the output results from both simulations we find both concepts reaching about the same maximum velocity. We therefore conclude that the more efficient solar panels, used in Concept 2, are not appropriate to increase the velocity of the boat.

In order to increase the velocity of the boat we could run more simulations with different hull designs in order to decrease the draft and ultimately the drag, swap motors, gearboxes and propellers or simply try to find lighter solar panels with a sufficient efficiency.

7 Discussion

7.1 Benefits

By using the design strategy utilizing Modelica described in this paper for preliminary ship design engineers are able to rapidly construct working assemblies and cut valuable time on their early design research. Even without detailed domain knowledge of naval architecture a user is able to assemble and compare different models.

By using Modelica's object oriented features the methodology enables radical or minor changes to a simulation model with one mouse-click. With the possibility to directly run a 3D animation of the simulation model the reaction to these changes can be reviewed immediately.

There is basically no limit of components to add to the simulation and it is also possible to break up existing components into several separated components. For example, a rudder can be simply modelled by the deflection of alternating a single value for the angle, or if desired be split up into different components, where the mechanism to initiate this deflection is implemented as individual components.

7.2 Limitations

As for every design method, the simulation output will never be an exact representation of reality. The methods used for describing the boat components are by necessity based on approximations. Given the stated goals of the IDCAE method can only be used for preliminary ship design, the engineer must still run detailed design iterations on the eventual chosen model in order to create a system which is producible and successful.

8 Conclusions and Future Work

8.1 Conclusion

This paper set out to describe a new method for the early stage design of ships which are highly innovative. We achieved this by introducing the use of Modelica for complete ship design. Combined, they form a working simulation of a boat that travels in a straight line. While the accuracy of the simulation may not be very high, at this stage it is already possible to determine the difference between the performance and behaviour of different assemblies. Only once the demands for the eventual design rise, must the simulation increase in accuracy.

8.2 Future Work

Primarily the content of the library must be expanded in the future. Further, the availability of many different types of components will make it possible to assemble innovative simulation models. Therefore, the authors hope to see more marine engineers finding their way to Modelica.

In addition, given the complexity level can vary per component and every ship design has its own purpose and requires different accuracies from the simulation a balance must be made on how complex to make each component. However, if for example waves are to be simulated all forces in the six degrees of freedom must be formulated in every component. Which in turn will make the library more useful.

References

- Abt, C., Bade, S., Birk, L., & Harries, S. (2001). Parametric hull form design-a step towards one week ship design. In *8th international symposium on practical design of ships and other floating structures* (pp. 67–74).
- Auld, & Srinivas. (2016). Blade Element Propeller Theory. Retrieved November 10, 2016, from <http://s6.aeromech.usyd.edu.au/aerodynamics/index.php/sample-page/propulsion/blade-element-propeller-theory/>
- Bitner-Gregersen, E. M., Eide, L. I., Hørte, T., & Skjong, R. (2013). *Ship and offshore structure design in climate change perspective*. Springer.
- Bole, M., & Forrest, C. (2005). Early stage integrated parametric ship design. In *Proc. ICCAS* (pp. 447–460).
- Buhaug, Ø., Corbett, J. J., Endresen, Ø., Eyring, V., Faber, J., Hanayama, S., ... others. (2009). Second IMO GHG Study 2009. London UK: International Maritime Organization.
- Dong, R., Wu, C., Zhang, J., & Peng, W. (2011). Modeling and simulation for ship hydraulic rudder system based on Modelica/MWorks [J]. *Ship Science and Technology*, 11, 20.
- Esrām, T. (2010). Modeling and Control of an Alternating-Current Photovoltaic Module. University of Illinois at Urbana-Champaign.
- Maragkogianni, A., Papaefthimiou, S., & Zopounidis, C. (2016). Shipping Industry and Induced Air Pollution. In *Mitigating Shipping Emissions in European Ports* (pp. 1–9). Springer International Publishing.
- Marty, P., Corrigan, P., Gondet, A., Chenouard, R., & Hétet, J.-F. (2012). Modelling of energy flows and fuel consumption on board ships: application to a large modern cruise vessel and comparison with sea monitoring data. In *Proceedings of the 11th International Marine Design Conference, Glasgow, UK* (pp. 11–14).
- Papanikolaou, A. (2014). *Ship Design: Methodologies of Preliminary Design*. Athens Greece: Springer.
- Rue, C. D. L., & Anderson, C. B. (2015). *Shipping and the Environment*. New York USA: Routledge.
- Sawada, H. (2012). Upstream design and 1D-CAE. *Journal of System Design and Dynamics*, 6(3), 351–358.
- Sutherland, J., Oizumi, K., Aoyama, K., Eguchi, T., & Takahashi, N. (2016). System-Level Design Tools Utilizing OPM and Modelica. In *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference IDETC2016*. Charlotte, North Carolina.
- Sutherland, J., Oizumi, K., Aoyama, K., Takahashi, N., & Eguchi, T. (2016). System-Level Design Trade Studies by Multi Objective Decision Analysis (MODA) utilizing Modelica. In *The First Japanese Modelica Conference, May 23-24, Tokyo, Japan* (pp. 61–69). Tokyo Japan: Linköping University Electronic Press.
- Sutherland, J., Salado, A., Oizumi, K., & Aoyama, K. (2017). Implementing Value-Driven Design in Modelica for a racing solar boat. In *15th Annual Conference on Systems Engineering Research*. Los Angeles, California (accepted for presentation).
- UNCTAD/RMT. (2016). *Review of Maritime Transport 2016*. Geneva Switzerland: United Nations Publication.

FMI Go! A simulation runtime environment with a client server architecture over multiple protocols

Claude Lacoursière, `clacours@hpc2n.umu.se`

Tomas Härdin, `tomas.hardin@umu.se`
HPC2N/UMIT, Umeå University
SE-901 87, Umeå, Sweden

Abstract

We present a distributed software infrastructure to perform distributed simulations with Functional Mockup Interface (FMI) compatible components. The current implementation supports both TCP/IP and MPI. This is a client-server design where the client is the global simulation stepper and the servers are the simulation modules. Features on the master time stepping algorithm currently include several time stepping algorithms including one which can handle algebraic constraints, root finding for cases involving loops, and support for asynchronous data exchange with “monitors” and “observers” which only consume data. The servers provide support for numerical directional derivatives, filtering, and interpolation. Support is provided for the System Specification and Parameterization (SSP), an emerging standard aimed at supporting the FMI.

The software is open source with a permissive license and designed to be used inside simulation environments and platforms with user interfaces. The focus being on the mathematical and runtime aspect of FMI based simulations.

1 Introduction

No one simulation tool can satisfy everyone’s needs and yet, full system simulation is the order of the day. Models created using different tools must be made compatible with each other for data transfer at least, and by force of reality, a lowest common denominator must be found for numerical time integration of modular, heterogeneous systems. In this model, subsystems are black boxes connected with simple elements representing boundary conditions. The (FMI)(MODELISAR, 2014) standard specifies an API which answers the first question of data formats as well as fundamental functionality to initialize and terminate modules, and defines semantics to handle events etc. However, this standard does not specify the requirements on the runtime environment or the master stepper.

We consider both these issues with the aim of providing a minimal runtime infrastructure which is fully standards compliant as well as open. We also intend to develop a number of numerical methods for time integration. This should allow academics to test their new numerical meth-

ods on nontrivial examples. The hub based design should also allow people to write their own interfaces to connect with the data analysis and visualization tools they prefer, and can serve as a foundation for commercial integration tools with sophisticated user interfaces.

In what follows we describe the nature of the problem we are trying to resolve in Sec. 2, then cover some previous work in Sec. 3. We then describe some details of our architecture in Sec. 4. Force based model coupling and is described in Sec. 5 and a kinematic coupling as well as a differential algebraic stepper is found in Sec. 6. Experiments and discussions are in Sec. 7,8,9 and in Sec. 10.

2 Problem statement and objectives

Software tool interoperability requires a standardized interface to be implemented by vendors, as well as a standard format to describe and exchange source or binary code implementing a Functional Mockup Unit (FMU). Also needed mathematical model which corresponds to the interface, a configuration format and editor for producing configuration files. Then comes a runtime environment which can read these, load the FMUs and perform time integration. One also needs data collection from the runtime environment, data formats and communication protocols. Of course, one also needs one needs numerical methods for time integration. When all this is in place, one can create simulations, run them, gather data, and analyze it with the tools of their choice.

The FMI specifies only the first three items: interface, exchange formats, and high level mathematical formulation. The emerging standard System Specification and Parameterization (SSP) (Köler et al., 2016) aims at defining the structure of a simulation – which FMU connects to which and on what port – as well as parameterization, including unit conversion etc. This is in the process to be adopted by the FMI committee. Editors for SSP are under development by vendors. There is also a Software Development toolKit (SDK)(QTronic, 2017) which is a reference implementation of the FMI API and can serve as a foundation for writing runtime environments.

We decided to develop components of the runtime environment including SSP, protocols and formats for data communication and handling, as well as stepping methods. We believe that these are the components missing to

achieve a genuinely modular solution which avoids vendor lock-in, as well as a sufficiently complete environment for researchers to test their numerical methods and simulation master algorithms. We are also considering IP and secrecy issues which can be supported with our client server design. Of course, we believe that performance is a fundamental aspect.

Loading many shared libraries as is suggested by the FMI documentation always leads to problems. This is why we chose a client server architecture which we implemented over TCP/IP for LAN and WAN configurations, and MPI for standalone of cluster ones.

User interfaces we leave for others.

The objective is to provide a robust runtime environment with good numerical methods which goes from SSP to data files, based on standards and protocols so that visualization and analysis software of choice can be plugged in easily.

3 Previous work

Nearly twenty FMI *import* tools are listed on the FMI website (MODELISAR, 2014) and these fall unevenly into two categories. First come the well established simulation packages which support import functionality in order to connect to third party tools. Second come *integration* tools only designed to couple simulation and analysis tools, which is close to our own work. These divide further into commercial and open source ones. Of the latter, DACCOSIM (Galtier et al., 2015) is the closest to our effort.

Yet all integration tools we know of aim at providing a full environment and it appears that the issue of the quality of time integration is secondary at best, yet the numerical methods are locked down which isn't good if one has a particularly recalcitrant and difficult model. We don't see the need to keep numerical methods secrets. And this prevents experimentation on real-life problems by academics. One of our motivations.

Distributed and modular simulations isn't new and predecessors include the High Level Architecture (HLA) (IEEE, 2010) for instance, which has even been adapted to FMI (Awais et al., 2013). Focusing on FMI compatible efforts, Ptolemy (Ptolemaeus, 2014) is an object oriented peer-to-peer agent based simulation environment and has now FMI (Broman et al., 2013; Cremona et al., 2016) capabilities with an eye on meeting the requirements for discrete-continuous simulations (Zeigler, Praehofer, and Kim, 2000), and DACCOSIM (Galtier et al., 2015) which is most similar to ours. There are others yet but too numerous to list here.

Why a new effort? First because there is a need for a test environment for new time integration methods which is not possible with commercial tools. The open source projects did not seem to have this as a focus.

Then comes a more contentious issue: software license. The commercial dimension here weights heavily so we

chose the permissive MIT (MIT, nodate) license to avoid any problem.

It is clear from the literature about time integration for cosimulations methods (Fiedler and Arnold, 2014; Martin, Christoph, and Tom, 2013; Schierz, Arnold, and Clauss, 2012; Schierz and Arnold, 2012; Arnold, 2010; Bernhard Schweizer, Li, and Daixing Lu, 2015; Bernhard Schweizer, Daixing Lu, and Li, 2015; B. Schweizer and D. Lu, 2015) that it is hardly possible to control errors or reach stability without having access to directional derivatives or at least the ability to rollback which isn't available in too many cases. However, both of these features can be provided by the runtime environment with numerical differentiation and various brute force techniques. This is clearly not addressed in any of the tools we looked at. One can provide these features when wrapping a ME FMU into a CS FMU, since one, which means that it might be advantageous to export as ME when possible and let a wrapper take care of more advanced features.

The DAE stepper presented in Sec. 6 is different from that of Schweizer (Bernhard Schweizer, Daixing Lu, and Li, 2015) in that we are using a previously published relaxation and regularization technique (Lacoursière, 2007) which is provably linearly stable, unlike the variants Schweizer analyzed (Ascher and Petzold, 1993). Experience has proved that our method does not require the solution of nonlinear systems of equations as the linearized approximation is sufficiently stable and produces no systematic drift.

Therefore, we believe that our work has much orthogonality with what already exists, enough to add yet one more FMI runtime environment to the list.

4 Software design

We opted for a client-server architecture in which each server process hosts an individual FMU. The global stepper is then a client, consuming results produced by the FMUs, and serves also as a data hub. It is also a server to *monitors* which are read-only applications for interactive, online visualization and data analysis, as well as data storage. See Fig. 2. We decided against peer-to-peer communication

We used Protobuf (*Protocol Buffers* 2017) to map the twenty or so functions in the FMI API to messages which can be passed via ZeroMQ (iMatrix, 2017). The servers dynamically load an FMU and using the QTronix SDK (QTronic, 2017). The same was repeated to use the Message Passing Interface (MPI) (MPI, 2017) which has the benefit of not needing to pack and unpack data. We chose to use (MPICH, 2017) because it is better than other implementations at handling oversubscription, i.e., when there are more processes than cores available. On Windows, we use the native library (Microsoft, 2017).

The numerical Jacobians were implemented in the servers using simple first order finite differences. These computations are done in the servers which can exploit

parallelism to perform this task. This requires the ability to rollback a simulation one or several times, and if possible, to clone it for parallelism. In the best case, FMUs provide rollback functionality. The second best case is a functional serialization and deserialization. We are investigating other methods, as well as the trade off between doing much more work per step vs step size and accuracy.

Support for Model Exchange (ME) FMUs is in development on two fronts. One is a global stepper capable of handling ME FMUs or combinations of ME and CS FMUs, i.e., integrating discrete and continuous systems (Zeigler, Praehofer, and Kim, 2000). The other is to include a local ME stepper inside the servers so that ME FMUs can be transformed to be CS ones. The advantage here is that even though CS export tools make a choice of numerical time integration which cannot be changed, yet is critical for stability and application dependent. Therefore, delaying the decision until runtime is appealing. In addition, ME FMUs are required to be able to cancel a step as long as no event is crossed which makes it easier to compute numerical derivatives and rollback. Access to the ME FMU also allows the support of extrapolation and interpolation methods (Bernhard Schweizer, Li, and Daixing Lu, 2015) after the model has been exported, and to introduce other types of filters on the inputs (M. Benedikt et al., 2013; Drenth, 2016). We have already automated the latter. Briefly, assuming a module has continuous states x , inputs u and outputs $y = g(x, u)$, the dynamics is augmented so that

$$\begin{aligned}\dot{x} &= f(x, u, t) \\ \dot{z} &= x,\end{aligned}\tag{1}$$

and then the reported output is

$$y = \langle g(\langle\langle x \rangle\rangle, \langle\langle u \rangle\rangle) \rangle\tag{2}$$

instead of $g(x, u)$ at the end of the communication step. The averages can be, e.g.,

$$y = g\left(\frac{1}{2H}(z_0 + z_1), \langle\langle u \rangle\rangle\right),\tag{3}$$

where H is the communication step, and z_0, z_1 are the values of z at the beginning and end of the communication step (Drenth, 2016). The advantage of such filters is to offset the noise produced by the discontinuous inputs at each communication point. Such functionality is clearly not possible when using a CS FMU, and we believe that it is good to leave the choice open. We have automated the augmentation of the equations of motion. Other types of filters are straight forward to implement.

The overall design is shown in Fig. 1. Here, one goes from a FMU via the Qtronic library to the FMI API. At this point, depending on whether the FMU is ME or CS, support libraries are used to deliver additional functionality to the global stepper. This funnels through the FMI/X communication library towards the global stepper, and said returns results to be processed by the FMU.

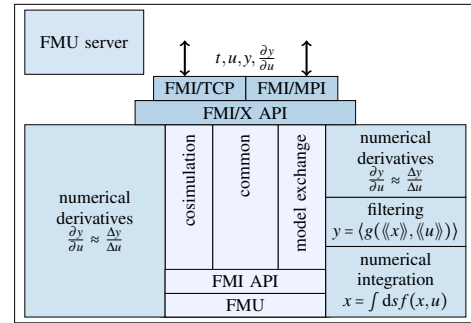


Figure 1. Server architecture

The global stepper program has a barebone, command line interface to describe the connections, as well as support for SSP files which is much more convenient.

The global stepper can also resolve loops at initialization using Newton-Raphson's method, and we intend to use this feature for the ME stepper so that it can process DAEs.

As the kinematic stepper requires the solution of linear problems, we currently use UMFPACK (Davis, 2004).

The overall design of the system appears diagrammatically in Fig. 2.

For TCP/IP, there are a number of issues related to the GRID computing concept of "network weather service", which is about resource discovery and allocation. This is not implemented yet but there are simple tools for this.

Hardware in the Loop (HIL) functionality has not been developed at this time though the architecture is compatible with this.

The software runs on Linux, Mac OS X and Windows.

To emphasize, this type of design is not entirely novel as mentioned in Sec. 3. What is different however is the restriction we imposed ourselves to the runtime environment and not the user environment. In addition, the existing functionality and what is in planning will hopefully provide building blocks for the development of new numerical time integrators, since typical restrictions of FMUs – absence of directional derivatives or rollback functionality – will be compensated for by the support modules, as described above, i.e., wrappers for ME FMUs to transform them into CS FMUs.

We follow the UNIX philosophy here: "Do one thing and do it well". For our case, the pipe model is "initial conditions in, data out". Clearly, there is more than one thing going on here, but we are aiming at being atomic and modular: all that's needed to perform time integration of systems made of FMUs, but only that.

To confirm that this is a position statement, we believe that numerical algorithms have little if anything to do with trade secrets, yet should be tested extensively in real situations. When an engineer runs a simulation, the same wants to have confidence that the results make sense. For that reason, the numerical time integration software should open. If successful, the best methods will be available for all to use.

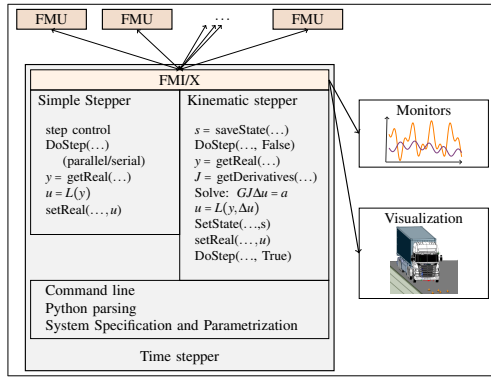


Figure 2. Overall architecture

5 Force subsystem couplings

Splitting a system leads to having one variable, x , say, which appear in two subsystems. For instance, the output shaft of an engine is the very same as the input shaft of a clutch so the angle of said should be the same in each module.

As the systems are integrated independently, the duplicates, $x^{(1)}, x^{(2)}$ cannot remain in sync. One strategy for physical systems at least is to introduce a generally stiff spring-damper in one or both subsystems. There are several choices as described previously (Bernhard Schweizer, Li, and Daixing Lu, 2015) namely, force-displacement or force-velocity in which one of the units contains a spring-damper but not the other. These we call “holonomic” and “non-holonomic”, respectively. Then there is displacement-displacement coupling in which case there are spring and dampers on both sides. Finally, there is the “spring free” case described in Sec. 6 which requires a global solver to compute the force required so that $x^{(1)} = x^{(2)}$ at each communication step, an algebraic condition. The latter requires both rollback and directional derivatives which is not often supported. Rollback is also required for “iterative” methods which are essentially fixed point iterations (Bernhard Schweizer, Li, and Daixing Lu, 2015) and have good stability properties. As mentioned, we aim at making our software capable of applying these methods for any FMU, whether it has these features natively or not.

We chose force-velocity in our examples.

Whenever there is a spring-damper at the input of a system, system 1, say, there is an input signal

$$u^{(1)} = x^{(2)}, \text{ or } u^{(1)} = v^{(2)} \text{ or both (with abuse of notation).} \quad (4)$$

But this signal is not available continuously, only at the beginning once per communication step. This values are denoted as $\bar{x}^{(2)}$ and $\bar{v}^{(2)}$. The coupling force, given spring and damping constants $k^{(c)}, \gamma^{(c)}$, respectively, is then

$$f^{(c)} = -k^{(c)}(x^{(1)} - \bar{x}^{(2)}) - \gamma^{(c)}(x^{(1)} - \bar{v}^{(2)}). \quad (5)$$

The reaction force $-\bar{f}^{(c)}$ at the end of the reported to the

coupled element. When $\bar{x}^{(2)}$ is not reported, the approximation

$$x(t) - \bar{x} \approx - \int_0^t ds(v - \bar{v}) \quad (6)$$

is used and $x(t) - \bar{x}$ is reset to 0 at the beginning of each communication step.

A variety of methods can be used to improve on the Zero Order Hold (ZOH) including extrapolation, or combination of extrapolation and interpolations, often called iterative methods (Bernhard Schweizer, Li, and Daixing Lu, 2015, and references therein).

One thing remains though, the spring-dampers $k^{(c)}, \gamma^{(c)}$ are not in the original model and introduce artificial dynamics. One needs to keep the frequencies due to the couplings much higher than the *design* frequencies, i.e., time scales involved by the couplings should be much smaller than those of interest in order to not interfere with the results as we show in Sec. 9, and this leads to communication steps which are orders of magnitude smaller than the time scales of interest, and introduces stiffness in the individual modules as they fight against the spring force in Eqn. (5). As we show below in Sec. 9, coupling frequencies need to be at least one order of magnitude above the design frequencies, meaning that coupling springs must be two order of magnitude above the stiffness of the internal force derivatives.

There are alternatives to this which do not involve a global solver as the one in Sec. 6, such as bilateral delay lines (TLM) (Dag Fritzson, 2007; Krus, 1995). This is still a spring-damper coupling but motivated by the fact that a force takes finite time to traverse any form of physical coupling, interactions are interpolated between the two previous steps. The main issue here is that this only works with intermediate steps within the `DoStep` calls. Something which can only be addressed with ME FMI using state machines without continuous states, one of our next steps. An effort similar to ours but based on TLM is in the process of being released to the public (Sjölund et al., 2010).

As far as trying to damp the high frequencies due to coupling and avoid oversampling the system, an anti-aliasing technique as been presented recently (Drenth, 2016) which is promising. We have introduced it into our software though we are not including this in our results as explained below in Sec. 11.

6 A differential algebraic stepper

As mentioned in Sec. 5, a split model involves algebraic conditions, and these can be taken care of directly by a DAE method (Bernhard Schweizer, Daixing Lu, and Li, 2015; Bernhard Schweizer, Li, Daixing Lu, and Meyer, 2015; Bernhard Schweizer and Li, 2015; B. Schweizer and D. Lu, 2015), though that requires rollback and directional derivatives. There are no spring-dampers in this model and therefore, no parasitic dynamics. Also, the problems related to choosing suitable spring and damping constants for the couplings is now entirely avoided, so are

artifacts introduced by the numerical integration methods because of stiffness, or unnaturally small time steps.

We reuse ideas from multibody dynamics (Lacoursière, 2007) to design a stepper which computes the interaction forces required to maintain the constraints at least linearly, and uses damping to stabilize the symmetric average of the algebraic conditions.

Consider two systems with output variables $y^{(1)}, y^{(2)}$ as well as time derivatives $\dot{y}^{(1)}, \dot{y}^{(2)}$. These variables are constrained either holonomically or nonholonomically, respectively, meaning that in discrete time we should have

$$g(y_k^{(1)}, y_k^{(2)}) = 0 \text{ or } q(y_k^{(1)}, y_k^{(2)}) = G^{(1)}\dot{y}_k^{(1)} + G^{(2)}\dot{y}_k^{(2)} = 0, \quad (7)$$

respectively, where k is the discrete time index. We also write $G = \partial g / \partial y$ so that $G\dot{y} = 0$ holds for both cases, abusing notation. First we make the assumption that

$$y_{k+1}^{(j)} \approx y_k^{(j)} + h\dot{y}_{k+1}^{(j)}, \quad (8)$$

where H is the communication step is a reasonable approximation. This is the case for the SHAKE (Hairer, Lubich, and Wanner, 2001) stepper and a variant of ours (Lacoursière, 2007). Note that $k+1$ is used on the time derivatives. Next we write a time translation operators as

$$y_{k+1}^{(j)} = \Phi_k^{(j)}(u_k^{(j)}) \text{ and } \dot{y}_{k+1}^{(j)} = \Psi_k^{(j)}(u_k^{(j)}). \quad (9)$$

The aim now is to compute $u_k^{(j)}$ in such a way that Eqn. (7) is satisfied at $k+1$. Assuming that all modules can rollback, we start with a guess $\bar{u}_k^{(j)}$ and from this we expand the constraint equations in Eqn. (7) to compute $u_k^{(j)} = \bar{u}_k^{(j)} + \delta u_k^{(j)}$ such that the constraints are satisfied. This requires the directional derivatives

$$\frac{\partial \dot{y}_k^{(j)}}{\partial u_k^{(j)}} = \frac{\partial \Psi_k^{(j)}}{\partial u_k^{(j)}}, \quad (10)$$

which are mobilities in the case of multibody dynamics, or admittance for electrical circuits. Dropping superscripts on all variables and writing G for the agglomerated Jacobian of the constraint equations and g_k for the value of the constraint equation at discrete time k , δu should satisfy

$$\begin{aligned} g_{k+1} &\approx g_k + hG\dot{y}_{k+1} \\ &= g_k + hG\Psi_k(\bar{u}_k + \delta u) \\ &\approx g_k + hG\dot{y}_k + h\left[G\frac{\partial \Psi_k}{\partial u_k}\right]\delta u. \end{aligned} \quad (11)$$

This linear approximation can be stabilized as shown in our previous work (Lacoursière, 2007), which is unconditionally stable, unlike more popular methods (Ascher and Petzold, 1993) variants of which have been studied also in the context of cosimulation (Bernhard Schweizer, Daixing Lu, and Li, 2015). However, methods mentioned above all based on spring-damper ideas and introduce second order

dynamics on the algebraic condition. Our method is of first order only and this is what provides stability. We also use a symmetric form of the constraint so that in fact, we are enforcing

$$\frac{1}{4}(g_{k+1} + 2g_k + g_{k-1}) + \frac{\tau}{h}G_k v_{k+1} + \frac{\varepsilon}{h}u_{k+1} = 0, \quad (12)$$

where τ is a relaxation time and serves as stabilization. This is clearly of first order in and of itself, but of course, the averaging does introduce oscillations, damped by the τ term. Such symmetric projections have been shown to have good energy preservation properties (Hairer, 2000) when $\tau = 0$ and the nonlinear equations are solved exactly. The parameter ε has the same unit of the inverse of a spring constant if we assume that u has units of force, and is there only to prevent against constraint degeneracy but can be shown to introduce physical compliance when τ is sufficiently small. However, τ serves as a low pass filter and when $\tau = 2h$, the oscillations are just below critical damping. This is needed to stabilize on the constraint manifold. This analysis is found in part in our own work cited above. With this parameterization, τ/h is the rate of exponential decay of the constraint violation. There is no such guarantee of stability with the standard scheme (Ascher and Petzold, 1993). A more thorough stability analysis is in preparation. We linearize Eqn. (12) to avoid having to solve the nonlinear system of equations. Introducing the parameter

$$\gamma = \frac{1}{1 + 4\tau/h} \quad (13)$$

we need to solve the following linear system of equations for δu

$$\left[G\frac{\partial \Psi}{\partial u} + \frac{4\gamma\varepsilon}{h}\right]\delta u = -\frac{4\gamma}{h}g_k + \gamma G\dot{y}^{(k)} - G\dot{y}^{(k+1)} \quad (14)$$

In practice, one performs a step with some guess for inputs \bar{u}_k to obtain a preliminary estimate on the velocities \dot{y}_{k+1} , rollback, compute δu , and then step forward again with $u_k = \bar{u}_k + \delta u$. This has been shown to work very well even for nonsmooth, event driven systems (Lacoursière and Sjöström, 2014) dozens of units simulated in parallel.

Note here that if there are n observables $y \in \mathbb{R}^n$ and m control inputs $u \in \mathbb{R}^m$, the system is underactuated if $m < n$, overactuated if $m > n$ and fully actuated when $m = n$. The matrix $G\partial\Psi/\partial u$ has full row rank when $m \leq n$ but is degenerate otherwise, and this is where $\varepsilon \geq 0$ comes in to regularize the system.

The control flow is described in the following. In this notation, each statement is understood to be applied to all FMUs in parallel and all superscripts are removed.

```
s = GetState(...)
SetXXX(...),  $\bar{u}$ 
DoStep(..., t+h, False)
GetReal(...),  $\bar{y}$ 
GetDirectionalDerivative(...,  $\partial\Psi/\partial u$ )
```

```

Assemble system matrix  $G\partial\Psi/\partial u$ 
Solve for inputs  $\delta u$  from Eqn. (14)
SetState(...,s)
SetRealXXX(..., $\bar{u}+\delta u$ )
DoStep(...,t, h, True)
    
```

7 Experimental methodology

In evaluating the performance of FMIGo! we focused on the latency introduced by the FMI/X communication layer. We looked at scalability here and worked strictly with the best case scenario using loopback ports. Overhead due to TCP/IP routers or switches varies greatly, but are in the millisecond range.

For the rest, we compare both the accuracy of cosimulated systems with respect to reference or analytic solutions, and pay attention to the small time scales introduced by the spring-damper couplings.

Two examples are considered, chains of spring damper systems with uniform masses, and a simple truck model often used for elementary analysis (Eriksen and Nielsen, 2014). The latter example contains large mass ratios.

In all cases, we choose our time steps using dimensional analysis and compare them the periods of oscillations in the systems, the rationale being that accurate solutions should require around twenty steps per period of oscillation – the smallest in the system –, as is the case for most good numerical time integration methods as easily verified. For instance, an embedded Runge Kutta method of order 4/5 requires 15 steps per period for the simple harmonic oscillator to reach local tolerance of 10^{-4} , involving 90 function evaluations, though forward Euler requires at least 50 function evaluations (steps) to produce a reasonable solution, though more than 200 to deliver any form of accuracy. And because ZOH techniques are very akin to forward Euler, this is the best one can expect.

8 Timing

Measurements on 4 cores i7 2.8GHz, sufficient memory and a vanilla Linux installation. We had 4GB available, but the footprint of the program was small enough as to be irrelevant for common hardware. Results will vary for different systems but this should give a good idea of the overhead involved.

Using `/usr/bin/time` utility we extracted `user`, `system` and `wall` time. The `user` time is spent in computation and signal routing and communication packaging and a part of the time spent in moving data via sockets.

For TCP/IP version, `system` includes time for polling, waiting and going through the TCP/IP stack. For the MPI version, `system` includes interprocess communication which depending on the MPI library, might also go via sockets and TCP/IP. So, this time has to be considered in the present case as it is used by the application. There are other unrelated processes counted in `system` but that was made negligible by stopping all irrelevant applications.

The `wall` time is larger than the sum of `user` and

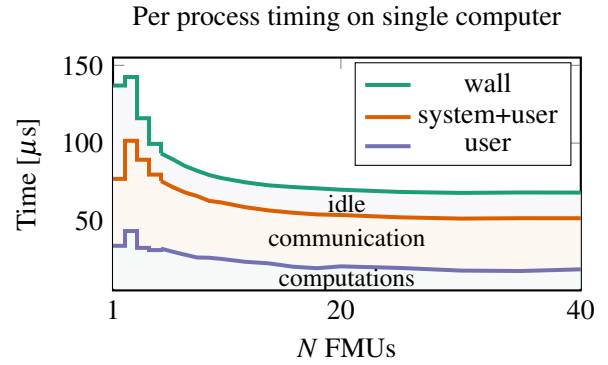


Figure 3. Timing measurements in loopback configuration

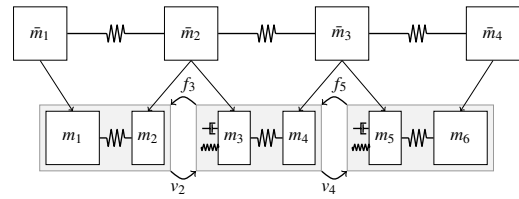


Figure 4. A chain of mass spring-dampers split into subunits

system because the CPU is not fully utilized and spends cycles waiting for packets and communication.

Note that this timing is sensitive to the choice of MPI library for the case of oversubscribing, i.e., when there are more processes than cores. OpenMPI performs badly in this case and seems to have quadratic complexity. MPICH however is well behaved and delivers linear performance as shown in Fig. 3. The Microsoft MPI library did perform well also.

Our experiment consisted of minimal FMUs which contained a point mass and a spring-damper. Computations were minimal and represent a lower bound on any practical simulation. What is therefore included here is all the time needed to perform time integration on said physical model, communication to the master stepper, routing of signals and communication back to the individual FMUs.

The conclusion is that a distributed design, at least when using MPI, is negligibly slower than one based purely on dynamic loading. The benefits of MPI however are immense as one can simulate on clusters, and as for the TCP/IP version, enables IP protection by hosting FMUs on secured computers.

9 Chains of mass-spring-dampers

The purpose of this experiment is to see at which point the time scales of the models and those of the couplings are sufficiently far apart that the dynamics of interest is negligibly disturbed and from there, made an estimate of the kind of time step required, in proportion to that one would use for the individual systems.

We consider a chain of N elements with ideal springs as

seen in Fig. 4

$$\bar{m}^{(j)} \ddot{\bar{x}}^{(j)} = -k^{(j-1)}(\bar{x}^{(j)} - \bar{x}^{(j-1)}) - k^{(j)}(\bar{x}^{(j)} - \bar{x}^{(j+1)}), \quad (15)$$

with $j = 2, 3, \dots, N-1$, and

$$\begin{aligned} \bar{m}^{(1)} \ddot{\bar{x}}^{(1)} &= -k^{(1)}(\bar{x}^{(1)} - \bar{x}^{(2)}) \text{ and} \\ \bar{m}^{(N)} \ddot{\bar{x}}^{(N)} &= -k^{(N-1)}(\bar{x}^{(N)} - \bar{x}^{(N-1)}). \end{aligned} \quad (16)$$

We call these spring constants the *design* variables as they are the ones included in the original model. The undamped case is the only irrelevant one for this analysis as it is the worst case scenario for testing the schemes. This considerably reduces the dimension of the parameter space. Then we split each mass except the first and last so that for $j = 2, 3, \dots, N-2$

$$\begin{aligned} m^{(i)} &= \bar{m}^{(1)}, m^{(2N-2)} = \bar{m}^{(N)}, \text{ and} \\ m^{(2j)} + m^{(2j+1)} &= \bar{m}^{(j+1)}. \end{aligned} \quad (17)$$

The interaction between $m^{(2j-1)}$ and $m^{(2j)}$ is the same as that between $m^{(j)}$ and $m^{(j+1)}$, but we now introduce spring-damper coupling $k^{(c)}, \gamma^{(c)}$ between $m^{(2j)}$ and $m^{(2j+1)}$ so that

$$\begin{aligned} m^{(2j)} \ddot{x}^{(2j)} &= -k^{(j)}(x^{(2j)} - x^{(2j-1)}) + f^{(2j,2j+1)}, \text{ and} \\ m^{(2j+1)} \ddot{x}^{(2j+1)} &= -k^{(j)}(x^{(2j+1)} - x^{(2j+2)}) - f^{(2j,2j+1)}, \\ m^{(1)} \ddot{x}^{(1)} &= -k^{(1)}(x^{(1)} - x^{(2)}), \\ m^{(N)} \ddot{x}^{(N)} &= -k^{(N)}(x^{(N)} - x^{(N-1)}), \text{ and} \\ f^{(2j,2j+1)} &= -k^{(c)}(x^{(2j)} - x^{(2j+1)}) \\ &\quad - \gamma^{(c)}(\dot{x}^{(2j)} - \dot{x}^{(2j+1)}), j = 1, 2, \dots, N-1. \end{aligned} \quad (18)$$

and therefore, we should have

$$x^{(2j)} \xrightarrow[k^{(c)} \rightarrow \infty]{} x^{(2j+1)}. \quad (19)$$

if the damping is correctly adjusted. We chose the non-dimensional damping parameter such that

$$\zeta = \frac{\gamma^{(d)}}{2\sqrt{\mu k^{(c)}}} = 0.7 \text{ where } \mu = \frac{m^{(2j)}m^{(2j+1)}}{m^{(2j)} + m^{(2j+1)}}. \quad (20)$$

This means that $\gamma^{(c)} \rightarrow \infty$ as $k^{(c)} \rightarrow \infty$ as required for convergence. The effect of this is also that in the stability analysis, we are at the same location in the complex plane as long as

$$h_1 \omega_1^{(c)} = h_2 \omega_2^{(c)}, \text{ where } \omega_i^{(c)} = \sqrt{\frac{k_i^{(c)}}{\mu}}. \quad (21)$$

Note that it is not generally possible to pick the damping constant γ optimally since internal inertiae and frequencies of any given FMU cannot be assumed to be

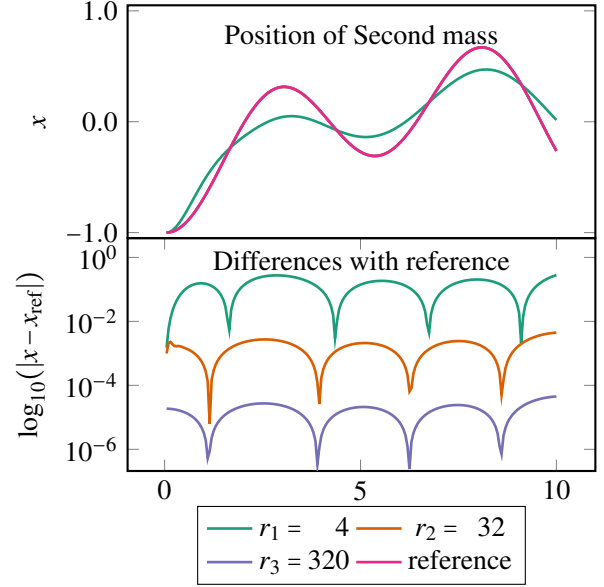


Figure 5. Influence of coupling springs on natural dynamics

known. We expect the time step to decrease linearly with the smallest period of the system, which should be $O(k^{(c)})$ unless an implicit integration strategy can be implemented.

We now look at ratio ρ between the coupling frequencies $\omega^{(c)}$ and those of the original system $\omega^{(d)}$, and estimate how large ρ must be to minimize interference. In our experiments, we set $x^{(1)}(0) = 1$ so as to inject energy in the natural modes. As expected for a linear system, the modes are separated and do not interact very much so the overall dynamics is similar. What is worrisome however that it takes a ratio of coupling of more than 100 before the errors go below 10^{-3} .

As seen in Fig. 4, one needs frequency ratios of around 30 to start recovering the correct solution and from Fig. 5 there is quadratic convergence towards the original solution. But given that this is a forward Euler technique, we get less. We found that we needed at least 50 times more step per coupling period than the minimum required by a good numerical integrator to have stability and some stability. This means $30 \cdot 50 = 1,500$ more steps per unit time than for the isolated models. That's three orders of magnitude more work. The DAE stepper of Sec. 6 produced very good solutions with a step commensurate with the design frequencies. We used a holonomic coupling here and included the positions in the model.

10 Experiments with a truck model

Here we investigate a simple truck model with an engine modeled with a point mass – the flywheel –, a PI control which aims at reaching a given speed, a clutch, a gearbox and a shaft, each represented with a pair of masses coupled with spring-dampers – piecewise linear for the clutch as in Fig. 7–, and a trailer modeled as a point mass but interacting with a road with variable slope following a sine

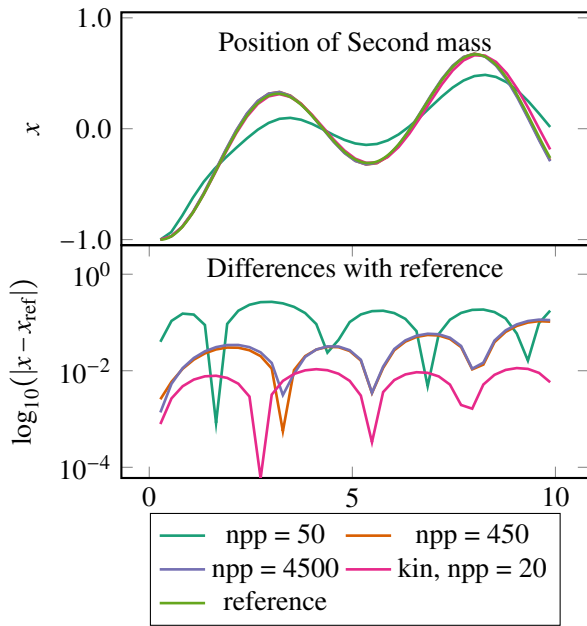


Figure 6. Simulation results for chains

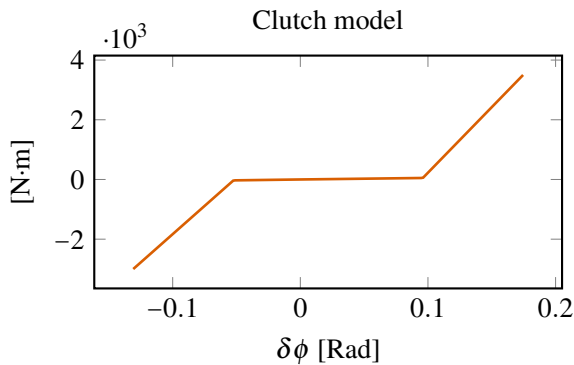


Figure 7. A piecewise linear clutch model

wave, and subject to gravity, rolling and dry friction, as well as air resistance. This is a textbook model (Eriksen and Nielsen, 2014).

The engine delivers 1,350 Nm max torque, the target speed is 100 km/h, and the trailer has a mass of 10,000 kg. The slope of the road was made sinusoidal. The interesting aspects here are the mass ratio and the large torques involved. The kinematically coupled model simply constraints velocities and coordinates between the components, i.e., the flywheel angle should match that of the in plate of the clutch, etc.

Each FMU has its own time integration and we chose the GSL for that. We used the fourth order Runge Kutta method `rk45` for our experiments with 10^{-6} tolerance. For the kinematic stepper, we computed the effective mobility by integrating and rollback and then using finite differences.

Here we compare our kinematic stepper of Sec. 6 with a step of $1/20$ and $1/120$ of the smallest period in the design. In this case, this is in the clutch and gearbox dynam-

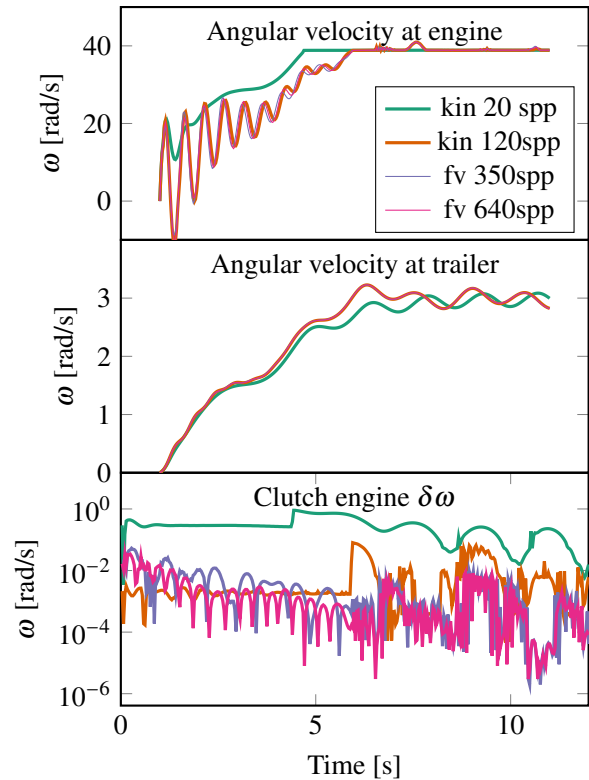


Figure 8. Cosimulation of a truck model. The “spp” key stands for steps per period.

ics. The case of 120 steps per period offers good results, though as low as 10 steps per period as stable. However, for the force-velocity coupling using either sequential or parallel simulation required more than 350 steps per period before stability. Good results come after 640 steps per period. Things were worse yet when we used holonomic coupling, i.e., including spring dampers for positions as well as velocities. We needed more than 10 times as many steps for the force-velocity versions, though with kinematic coupling, we had high accuracy at 20 steps per period already (results not shown). To be considered here is that the mass of the trailer is so much larger than the driveline that very stiff springs would be needed to reach the correct result. Considering the previous experiment in Sec. 9 we used coupling springs 30 times larger those in the design. This leads to $30 \cdot 350 / 20 \approx 500$ more steps than necessary just for stability. The DAE stepper does perform more work per step: solving a small system of linear equations, computing directional derivatives, and performing two sub-steps per step. But even that’s not a fair comparison since the FMUs in the kinematic coupling setup do not contain stiff coupling springs and therefore, they also perform an order of magnitude less work. In this case, the best result from the DAE stepper used four times fewer integration steps overall, and that’s despite the fact that we used numerical directional derivatives, which takes four times as many steps.

11 Discussion

The main lesson here is that when it comes to force-velocity couplings at least, but this applies to the other cases of spring-damper type couplings, one needs at least 30:1 ratio of frequencies and as far as ZOH techniques goes, this introduces orders of magnitude more work than an all-at-once method, or a DAE based one. Though the frequency ratio appears inevitable, there is recent work (Drenth, 2016) indicating that the communication step size can be kept modest by filtering the high oscillations. We have not been able to use this technique or reproduce the results in our experiments, something which the author relates to admittance or mobility ratio between the FMUs.

Kinematic coupling offers very good solutions at relatively large steps and were able to use holonomic couplings as well at moderate steps. Of course, this requires functionality that is not often seen in CS FMUs, namely, rollback and directional derivatives.

12 Conclusion

The software we introduce should be of interest for being minimalistic and offering a good foundation to build integration environments for cosimulation. It offers very good performance for a standalone computer yet can be distributed over WAN. We hope that the functionality we are developing will open the door to more advanced time integration methods. We will soon start collaborations with the OpenModelica and OpenCPS groups, which will provide user interfaces.

Our kinematic stepper can produce very good results in keeping time steps commensurate with the model frequencies and offers parallelism, at least on simple models. Further investigation is needed clearly, but the benefit in comparison to force-velocity coupling is significant and we believe that, despite the difficulties associated with roll-back and computation of directional derivatives, is worth much attention.

As part of future work, we intend to support TLM as it is popular, provide a fully functional ME simulation master, and improve the numerical directional derivatives functionality to be fully parallel. Other features such as extrapolation and interpolation in the ME FMU wrapper, as well as iterative and implicit time integration methods, or various types of filtering are under consideration.

The software is available on a request basis at this time at *git clone* at https://mimmi.math.umu.se/users/sign_in. Anonymous access is forthcoming.

Acknowledgments

This work is a part of the project "Virtual Truck and Bus" supported by the Swedish Energy Authority, and is a collaboration between Scania CV AB, Algoryx Simulation AB, Modelon AB, Umeå University and Volvo Car Corporation. Previous contributions to the software development were made by Stefan Hedman and Adeel Ashgar.

Bibliographic References

References

- Arnold, Martin (2010). "Stability of Sequential Modular Time Integration Methods for Coupled Multibody System Models". In: *Journal of Computational and Non-linear Dynamics* 5.3, pp. 031003–031003.
- Ascher, Uri M. and Linda R. Petzold (1993). "Stability of Computational Methods for Constrained Dynamics Systems". In: *SIAM J. Sci. Computing* 14.1, pp. 95–120.
- Awais, M. U. et al. (2013). "Distributed hybrid simulation using the HLA and the Functional Mock-up Interface". In: *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, pp. 7564–7569.
- Broman, D. et al. (2013). "Determinate composition of FMUs for co-simulation". In: *2013 Proceedings of the International Conference on Embedded Software EM-SOFT*, pp. 1–12.
- Cremona, F. et al. (2016). "Step revision in hybrid Co-simulation with FMI". In: *2016 ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, pp. 173–183.
- Dag Fritzson Johas Ståhl, Iakov Nakimovski (2007). "Transmission line co-simulation of rolling bearing applications". In: *The 48th Scandinavian Conference on Simulation and Modeling*. Ed. by Claus Führer Peter Bonus Dag Fritzson, pp. 24–39.
- Davis, Timothy A. (2004). "Algorithm 832: UMFPACK — an Unsymmetric-Pattern Multifrontal Method". In: *ACM Transactions on Mathematical Software* 30.2, pp. 196–199.
- Drenth, Edo (2016). "Robust Co-Simulation Methodology of Physical Systems". In: *9th Graz Symposium Virtual Vehicle*.
- Eriksen, Lars and Lars Nielsen (2014). *Modeling and control of engines and drivelines*. John Wiley & Sons.
- Fiedler, Robert and Martin Arnold (2014). "Coupled differential algebraic equations in the simulation of flexible multibody systems with hydrodynamic force elements". In: *PAMM* 14.1, pp. 523–524.
- Galtier, Virginie et al. (2015). "FMI-based Distributed Multi-simulation with DACCOSIM". In: *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*. DEVS '15. Alexandria, Virginia: Society for Computer Simulation International, pp. 39–46.
- Protocol Buffers (2017). <https://github.com/google/protobuf>.
- Hairer, E. (2000). "Symmetric Projection Methods for Differential Equations on Manifolds". In: *BIT Numerical Mathematics* 40 (4), pp. 726–734.
- Hairer, E., C. Lubich, and G. Wanner (2001). *Geometric Numerical Integration*. Vol. 31. Springer Series in Computational Mathematics. Berlin: Springer-Verlag.

- IEEE (2010). “IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Framework and Rules”. In: *IEEE Std 1516-2010*, pp. 1–38.
- iMatrix (2017). *ZeroMQ*. <http://zeromq.org/>.
- Köler, Jochen et al. (2016). “Modelica-Association-Project “System Structure and Parametrization” – Early Insights”. In: *Proceedings of the 1st Japanese Modelica Conference*. Modelica Association. Linköping University Electronic Press, pp. 35–42.
- Krus, Petter (1995). “Modelling of Mechanical Systems using Rigid Bodies and Transmission Line Joints”. In: *ASME J. Dyn. Sys., Meas., Control* 121.4, pp. 606–611.
- Lacoursière, Claude (2007). “Ghosts and Machines: Regularized Variational Methods for Interactive Simulations of Multibodies with Dry Frictional Contacts”. PhD thesis. Dept. of Computing Science, Umeå University.
- Lacoursière, Claude and Sjöström (2014). *A non-smooth event-driven, accurate, adaptive time stepper for simulating switching electronic circuits*. Tech. rep. UMINF 16.15. Dept. of Computing Science, Umeå University.
- M. Benedikt et al. (2013). “NEPCE - A nearly energy-preserving coupling element for weak-coupled problems and co-simulations”. In: *V International Conference on Computational Methods for Coupled Problems in Science and Engineering*. Ed. by S. Idelsohn, M. Papadrakakis, and B. Schrefler, pp. 1021–1032.
- Martin, Arnold, Clauss Christoph, and Schierz Tom (2013). “Error Analysis and Error Estimates for Co-Simulation in FMI for Model Exchange and Co-Simulation V2.0”. In: *Archives of Mechanical Engineering* 60. 1, pp. 75–94.
- Microsoft (2017). *Microsoft MPI*. <http://tinyurl.com/mpilib-microsoftv85>.
- MIT (n.d.). *MIT license*.
- MODELISAR (2014). *FMI website*. last retrieved 2017-01-22. URL: <https://www.fmi-standard.org>.
- MPI (2017). *A Message Passing Interface Standard*. <http://mpi-forum.org/>.
- MPICH (2017). *High performance, widely portable implementation of the Message Passing Interface*. <http://mpi-forum.org/>.
- Ptolemaeus, Claudius, ed. (2014). *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org. URL: <http://ptolemy.org/books/Systems>.
- QTronic (2017). *QTronic FMI SDK*. <http://www.qtronic.de/en/fmusdk.html>.
- Schierz, Tom and Martin Arnold (2012). “Stabilized overlapping modular time integration of coupled differential-algebraic equations”. In: *Applied Numerical Mathematics* 62.10. Selected Papers from NUMDIFF-12, pp. 1491–1502.
- Schierz, Tom, Martin Arnold, and Cristoph Clauss (2012). “Co-simulation with communication step size control in an FMI compatible master algorithm”. In: *Proceedings of the 9th International MODELICA Conference*.
- Schweizer, B. and D. Lu (2015). “Predictor/corrector co-simulation approaches for solver coupling with algebraic constraints”. In: *ZAMM* 95 (9), pp. 911–938.
- Schweizer, Bernhard and Pu Li (2015). “Solving Differential-Algebraic Equation Systems: Alternative Index-2 and Index-1 Approaches for Constrained Mechanical Systems”. In: *Journal of Computational and Nonlinear Dynamics* 11.4, pp. 044501–044501.
- Schweizer, Bernhard, Pu Li, and Daixing Lu (2015). “Explicit and Implicit Cosimulation Methods: Stability and Convergence Analysis for Different Solver Coupling Approaches”. In: *Journal of Computational and Nonlinear Dynamics* 10.5, pp. 051007–051007.
- Schweizer, Bernhard, Pu Li, Daixing Lu, and Tobias Meyer (2015). “Stabilized Implicit Cosimulation Method: Solver Coupling With Algebraic Constraints for Multibody Systems”. In: *Journal of Computational and Nonlinear Dynamics* 11.2, pp. 021002–021002.
- Schweizer, Bernhard, Daixing Lu, and Pu Li (2015). “Co-simulation method for solver coupling with algebraic constraints incorporating relaxation techniques”. English. In: *Multibody System Dynamics*, pp. 1–36.
- Sjölund, Martin et al. (2010). “Towards Efficient Distributed Simulation in Modelica using Transmission Line Modeling”. In: *Proceedings of the 3rd International Workshop on Equation-Based Object-Oriented Languages and tools*. Ed. by Peter Fritzson et al., pp. 71–80.
- Zeigler, Bernard P., Herbert Praehofer, and Tag G. Kim (2000). *Theory of Modeling and Simulation*. 2nd ed. Academic Press.

Experimenting with Matryoshka Co-Simulation: Building Parallel and Hierarchical FMUs

Virginie Galtier¹ Michel Ianotto¹ Mathieu Caujolle² Rémi Corniglion² Jean-Philippe Tavella²
José Évora Gómez³ José Juan Hernández Cabrera³ Vincent Reinbold⁴ Enrique Kremers⁵

¹CentraleSupélec, France, {first.last}@centralesupelec.fr

²EDF R&D, France, first.last@edf.fr

³SIANI, Spain, jose.evora@siani.es, josejuanhernandez@siani.es

⁴University of Leuven, Belgium, vincent.reinbold@kuleuven.be

⁵EIFER, Germany, enrique.kremers@eifer.uni-karlsruhe.de

Abstract

The development of complex multi-domain and multi-physic systems, such as Smart Electric Grids, have given rise to new challenges in the simulation domain. These challenges concern the capability to couple multiple domain-specific simulators, and the FMI standard is an answer to this. But they also concern the scalability and the accuracy of the simulation within an heterogenous system. We propose and implement here the concept of a Matryoshka FMU, i.e. a first of its kind FMU compliant with the version 2.0 of the FMI standard. It encapsulates DACCOSIM – our distributed and parallel master architecture – and the FMUs it controls. The Matryoshka automatically adapts its internal time steps to ensure the required accuracy while it is controlled by an external FMU-compliant simulator. We present the JavaFMI tools and the DACCOSIM middleware used in the automatic building process of such Matryoshka FMUs. This approach is then applied on a real-life Distributed Energy System scenario. Regarding the Modelica system simulated in Dymola, improvements up to 250% in terms of computational performance are achieved while preserving the simulation accuracy and enhancing its integration capability.

Keywords: co-simulation tool, multi-threaded execution, master algorithm, FMU, FMI standard

1 Introduction

Complex systems can be characterized by a great number of heterogeneous entities in interaction. The Smart Grids provide a typical example: over a large territory a multitude of devices produce, transport, store and consume electricity, while some are being monitored and controlled in order to best adjust the dynamic configuration of the electric network to the current and forecasted weather conditions and client needs. Co-simulation is essential to design and study such complex systems.

In this context, the FMI (Functional Mockup Interface) standard (Blochwitz and Otter, 2011) allows users to share and combine their models across simulation tools by wrapping them with a native solver in a package, called

an *FMU for Co-Simulation*, that is composed of an XML model description and a compiled C code. But the orchestration of the execution of the multiple FMUs forming the co-simulation of a complex system is up to the user. DACCOSIM, as an FMU-based co-simulation platform able to define and simulate complex calculation graphs, proposes an answer to this matter.

Furthermore, solvers usually used to simulate multi-physics systems are single-threaded. They may thus encounter scaling problems when simulating larger systems. This is the same for those included in FMUs. DACCOSIM provides a master code orchestrating the execution of FMUs in parallel, synchronizing their data exchanges and adjusting the internal step size to ensure accuracy.

Our objective is to get the best of both worlds by wrapping a DACCOSIM co-simulation in an FMU. We refer to this englobing FMU as a "Matryoshka" FMU.

This article is organized as follow: Section 2 provides a quick overview of DACCOSIM features and inner architecture. Section 3 lists the benefits of encapsulating DACCOSIM within an FMU. Section 4 presents JavaFMI, a tool which greatly facilitates the construction of FMUs from Java code. Section 5 explains how a Matryoshka FMU is built with and by DACCOSIM. A real-life Distributed Energy System is then considered and the results obtained in terms of accuracy and computation efficiency are presented in Section 6. Finally Section 7 points out a few directions we would like to explore in the future.

2 DACCOSIM, a Powerful FMI for Co-Simulation Platform

DACCOSIM (Galtier et al., 2015) is a Java co-simulation middleware able to define and simulate complex calculation graphs consisting of multiple FMUs compliant with the FMI 2.0 standard for Co-Simulation. It relies on JavaFMI (see Section 4) and is available¹ under AGPL for both Windows and Linux operating systems, whether 32-bit or 64-bit.

¹<https://daccosim.foundry.supelec.fr>

It consists of two complementary parts:

- A **user-friendly Graphical User Interface (GUI)** that facilitates the definition of multi-domain studies (Figure 1). It enables to easily design the calculation graph of the simulation case, i.e. the FMUs involved and the variables exchanged in-between. It also allows the user to set the resources used for the simulation case (local multi-threaded machine or HPC cluster) as well as its setup (simulation duration, co-initialization method, time step control strategy, tolerance allowed to internal solver and variables...). Results can be displayed *a posteriori* or in real-time during the simulation. In addition, a Domain Specific Language allows the user to write scripts to define, configure and run parametric studies on large co-simulation cases involving hundreds of FMUs and thousands of variable exchanges.

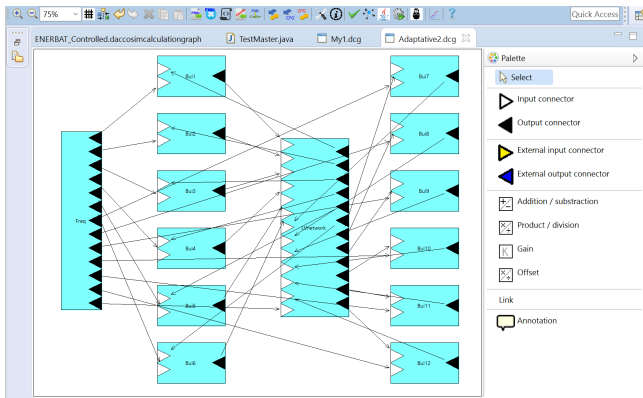


Figure 1. Screenshot of DACCOSIM GUI for a system of 14 FMUs with 110 variables exchanged

- A **parallel and distributed execution architecture** which manages the initialization and the execution of the involved FMUs. To maximize performance and scalability, DACCOSIM runs the FMUs involved in the co-simulation in parallel, using multiple threads on a node, and using multiple nodes when a cluster is available. Each FMU is executed by a wrapper directly connected to other wrappers to import and export variable values at each communication step. To provide the best trade-off between precision and computational speed, DACCOSIM integrates fixed and adaptive time step control strategies to dynamically adjust the simulation step size of all FMUs to the estimated error. In order to perform this co-ordinated step-size adjustment, DACCOSIM relies on a hierarchy of "masters", one on each computation node, controlling the set of FMU wrappers executing on this node. This architecture (Figure 2) is used during both co-initialization and co-simulation stages.

The transition from the calculation graph designed with DACCOSIM GUI to its execution with DACCOSIM calculation engine relies on Acceleo²: the graph is translated

²<https://www.eclipse.org/acceleo/>

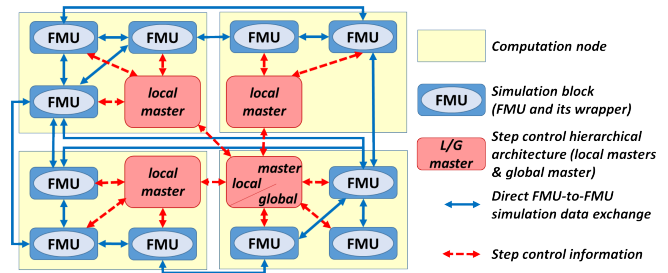


Figure 2. DACCOSIM distributed architecture

into one or more DACCOSIM masters depending on the resources considered. These masters launch the simulation and run concurrently till the end of the simulation case.

If it is above all a robust and scalable co-simulation middleware able to simulate large and complex use cases, **DACCOSIM is also an experimental playground for the FMI standard** where innovative features are tested, such as the ahead implementation of proposed FMI primitives (Tavella et al., 2016), or the Matryoshka FMU approach presented in this paper.

3 The Benefits of Encapsulating DACCOSIM within an FMU

DACCOSIM itself is a powerful FMI for co-simulation middleware able to perform fully parallel and distributed co-initialisation and co-simulation tasks. But as a standalone tool, its scope remains limited:

- Only FMUs compliant with the FMI 2.0 for CS standard are supported. Consequently simulators such as NS-2 (a communication networks simulator), or HLA federates with no FMI interface cannot be included into its co-simulation graph.
- It cannot be integrated within domain-specific tools able to import FMUs, tools which become more and more widespread nowadays.

Designing a specific control API for DACCOSIM would help to meet these needs, but encapsulating it all into a Matryoshka FMU fulfills even more of them:

- Such an FMU can be imported into any FMI compliant simulation tool or platform such as Dymola or MECSYCO (Vaubourg et al., 2015). This opens new perspectives since some of these tools might as well handle non-FMI components with which DACCOSIM is not able to directly interact.
- Taking advantage of DACCOSIM efficient, multi-threaded, step-size control solution helps simulating faster larger models within traditional mono-threaded simulation tools. It makes particular sense for domains where few parallel solvers are available.
- Initialization of complex graphs is taken care of within the Matryoshka thanks to DACCOSIM generalized co-initialization algorithm.
- A complex simulation graph can be reused directly

without having to re-write anything and with no risk of disclosing industrial and intellectual property.

- The co-simulation process can be finely tuned: when a solver typically uses only one accuracy objective for the whole model, DACCOSIM allows the user to define different tolerance values for every output and internal variable of each FMU.

4 JavaFMI Tools to Generate and Execute FMUs

JavaFMI is a software project devoted to provide a toolbox that allows to import and export Functional Mock-up Units (FMU) to/from Java in conformance with the FMI-CS 1.0 and 2.0 standards. This project is developed by SIANI³ university institute and its license is LGPL. Main contributors of this project are EDF Lab, EIFER, and CentraleSupélec. This project is composed of two main tools: a wrapper and a builder.

4.1 FMI Wrapper

The **FMI wrapper** allows to import FMUs into a Java application supporting the creation of Master Algorithms (Evora et al.). It provides **two types of interface**: simulation (simplified interface) and access (full interface).

The **simulation class** (*FmiSimulation*) provides a very simplified access to the FMU. This way, the user of the wrapper can load FMUs without having a deep knowledge of the FMI standard. Its methods are `init`, `doStep`, `terminate`, `read` and `write variable`, `getSimulationTime`, `isTerminated` and `reset`.

The **access class** allows invoking all available methods of the standard depending on the version that is being used. This way, the simulation class can be wrapped by the access class allowing for an advanced usage. Methods like `get`, `set` and `free state` can be invoked among others. Basically, all the primitives specified in the FMI-CS standard can be found as methods in this class.

4.2 FMI Builder

The **FMI builder** allows to **create an FMU based on a Java application** or any program that can be controlled by a simple Java code. That is, any Java simulation can be exported to an FMU. This tool provides an automated solution to create an FMU covering the development of the dynamic libraries, the generation of a model description file and the packaging of the needed resources.

The builder provides a framework to convert a Java simulation into an FMU. It is required to extend the *FmiSimulation* class where, at least, the following methods should be implemented:

- **define**. It returns a model that contains the information to be rendered in the `modelDescription.xml`
- **init**. It is called in the instantiation process of the FMU. It should register all input and output variables

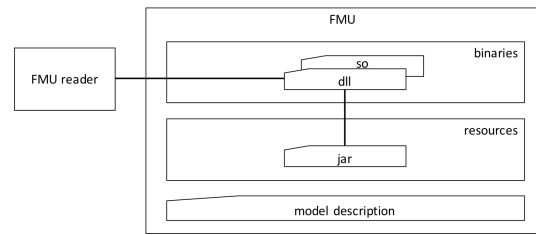


Figure 3. Communication between the JavaFMI wrapper and the FMU JAR through the libraries (dll, so)

with their corresponding getter and/or setter methods so that the framework can later get and set the FMU variables during the initialization and simulation stages.

- **doStep**. It advances the simulation according to the given step size.
- **reset**. It resets the simulation to its initial state.
- **terminate**. If needed, it should be filled with a termination code.

Once these methods are implemented, the *FmiSimulation* class is packaged into a JAR (Java ARchive) file and processed by the builder so that an FMU is created. The builder creates an FMU file containing:

- Dynamic libraries (dll and so) in the binaries folder.
- Model description.
- JAR file tuned to the model in the resources folder.
- Additional FMU resources in the resources folder if any are defined by the user.

The resulting FMU is compliant with the version 2.0 of the FMI standard which makes it applicable within any FMI compliant tool. Basic primitives like `init`, `doStep`, `terminate`, etc. are available as well as advanced ones like `get`, `set` and `free state`. For these advanced methods, the FMI builder has a default implementation that can be overridden in case a custom implementation is needed.

At runtime, the FMU dynamic libraries are programmed so that an instance of a Java Virtual Machine (JVM) is created in order to load the FMU JAR file. Once this happens, all functions invoked by the user of the library are directly bridged to the Java application by using pipes. Associated data flows are explained in Figure 3.

When using JavaFMI wrapper to load the FMU, this data flow can be shortened: if the JavaFMI wrapper detects that the FMU has been built with the JavaFMI builder, it takes the JAR located in the resources folder, loads it in the JVM in which the wrapper is, and communicates directly with the FMU methods through Java (Figure 4). This yields a significant improvement in the communication speed.

The JavaFMI project also contributes to make the FMI standard evolve. New co-simulation concepts (Tavella et al., 2016) are being trialed and validated by implementing newly defined primitives linking compliant FMU-generating tools to master algorithms.

³<http://www.siani.es>

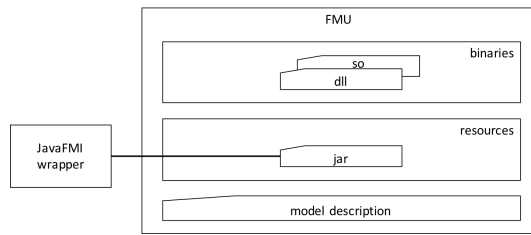


Figure 4. Direct communication between the JavaFMI wrapper and the FMU JAR created by the builder

5 Matryoshka FMU Building Process

In this section we first describe the steps taken by a DACCOSIM user to build a Matryoshka FMU for his co-simulation test case. Next we expose the behind the scene mechanisms, i.e. how DACCOSIM 2017 was augmented to support the construction of a Matryoshka FMU and which operations it performs during the building process. Last we present the result of the building process.

5.1 The User's Perspective: How to Build a Matryoshka FMU from DACCOSIM

Exporting a Matryoshka FMU is quite simple for DACCOSIM users. Only a few additional steps are required after having designed the co-simulation graph.

During this initial stage, the user sets the simulation configuration as he would do for any co-simulation test case. These settings determine the internal behavior of the Matryoshka, with in particular:

- the **co-initialization mode** (none, sequential output propagation, Newton or a mix of both),
- the **step size control method** (constant step size or the adaptive Euler, Richardson or Adams-Bashforth methods) and its step size characteristics (initial, minimum and maximum step size),
- the **event detection method** (bisectional approach (Camus et al., 2016) or minimum step-size).

The user's only task is then to define the inputs and outputs of the Matryoshka FMU and link them to the variables of its internal FMUs:

1. The user uses a specific interface (Figure 5) to **create the external variables** of the graph and set for each its **name**, **causality** (input or output), **type** (real, integer, boolean, string, enumeration), **variability** (constant, discrete or continuous) and **initialization mode** (exact, approximated or calculated). Adding a **description** of the variables is also possible.
2. He **defines default initial values** for each external input variable.
3. He **adds external connectors**, connects them to the FMUs own connectors in the graph and associates their variables as depicted in Figure 6.
4. Finally he **generates the Matryoshka FMU** by clicking the toolbar export button of the GUI.

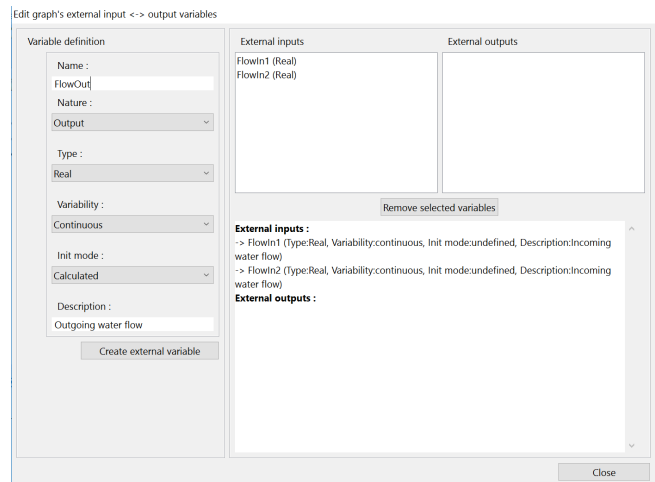


Figure 5. DACCOSIM interface enabling external IO definition

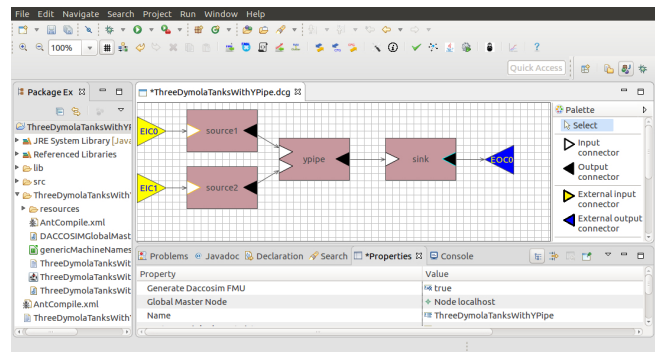


Figure 6. DACCOSIM graph with external connectors

5.2 Behind the Scene: the Steps Towards the Matryoshka FMU

We describe in the following subsections the sequence of actions that are automatically performed by DACCOSIM and result in the Matryoshka FMU generation when clicking the "Generate DACCOSIM Matryoshka FMU" button of DACCOSIM GUI toolbar.

5.2.1 Creating DACCOSIM master external API

DACCOSIM 2017 was augmented to be controlled from the outside when executed on a local machine. The result is a **DACCOSIMGlobalMaster** class that is tailored to a particular co-simulation configuration, and can be instantiated from another Java program. The obtained master class retains its internal mechanism specificities (multi-threaded architecture, adaptive step size control...) while adapting to the constraints imposed by the control program (external step size, input values...): if the internal step size leads to exceed the external step bound, the internal step size is truncated to meet this limit. It is afterward restored to its non-truncated value at the beginning of the following external step. Only DACCOSIM cluster features, i.e. its distributed architecture, are for now not exploited in the context of the Matryoshka FMU.

The master class is generated with Acceleo. It inte-

grates a set of basic functions enabling external interactions with the master, among which:

- **instantiating** DACCOSIM global master;
- **setting the start and stop times** of the simulation;
- **setting and getting the value of the external variables** of the Matryoshka. Inner variables are not accessible for now;
- **changing the state of the master**, i.e. initialization or simulation mode;
- **performing the co-initialization of the internal graph** considering imposed input variable values;
- **performing a co-simulation step** whose size is imposed by the control program;
- **terminating** the co-initialization and/or co-simulation process.

5.2.2 Specifying Matryoshka interface to JavaFMI

All these functions are called by a **Java interface code extending the *FMISimulation* class** defined in the JavaFMI tools. This interface is used to perform the mapping between the primitives defined by the FMI standard and DACCOSIM master's own interaction functions. It is automatically generated with Acceleo. The *modelDescription.xml* file of the Matryoshka FMU is later generated based on the information specified in this interface class, and especially the list and characteristics of the external input and output variables of the DACCOSIM co-simulation graph.

One characteristic of the Matryoshka that has to be calculated prior to the interface generation is the *dependency of the external output variables regarding its external input variables*. This information is important when performing the co-initialization of a co-simulation scheme involving the Matryoshka to ensure that the variables are initialized in the correct order.

The calculation of the Matryoshka output dependencies is automated by DACCOSIM. It first computes the *oriented acyclical causality graph* of the Matryoshka co-simulation scheme (Figure 7). The graph is then reversibly parsed from the external outputs (blue dots) until it reaches the *graph seeds* that include the external inputs (large yellow dots). Optionally, this process can be disabled to let the user define the dependencies manually.

5.2.3 JavaFMI interface compilation with Ant

The two Java files (Master and Interface) are put into Java packages and compiled into a JAR using Ant. The Ant command file is tuned to each use-case and generated with Acceleo. The resulting JAR is an essential input component for the FMU builder.

5.2.4 Building the FMU using JavaFMI builder

The Matryoshka FMU is finally created by using JavaFMI builder (see Section 4). The following components are assembled as the super FMU resources:

- the **previously constituted Jar file**;
- the **resources required by DACCOSIM master**, i.e. the inner FMUs, the csv files defining the variables

to log and the variables exchanged, a *modelDescription* file generated by DACCOSIM (different from Matryoshka's own *modelDescription* file generated by the builder, even though they're quite similar);

- the **library files required by DACCOSIM calculation engine**. If most are platform independent, a few such as OMQ require OS specific components. This results in an OS specific Matryoshka that can be used either on Windows 64 bit or Linux 64 bit systems.

A simple call to the FMU builder command line pointing to these resources is then sufficient to automatically create the Matryoshka FMU.

5.3 What is a Matryoshka FMU like

The result is an FMU embedding DACCOSIM with the following capabilities:

- Can manage variable simulation time step.
- Can be instantiated several times.
- Cannot get and set FMU state, serialize its state or provide directional derivatives.

Generated Matryoshka FMUs have been successfully tested with the FMU checker, as well as imported and run in FMI 2.0 compliant tools (Dymola, DACCOSIM...).

6 Application to an Industrial Simulation Use Case

6.1 Presentation of the District Energy System Use Case

A District Energy System (DES) consists of components that enable the delivery of energy services in a district. This includes all possible carriers, most frequently electricity, heating, cooling and gas networks. Research interests mainly focus on the modelling of electrical and heat grids on a neighbourhood scale to optimize the topology and sizing of the electrical network, as well as to design the energy management system (Baggi et al., 2014; Zucker et al., 2016; Wetter et al., 2015).

6.1.1 Problematics

One of the main issues of such models is their lack of scalability, i.e. the inability to study a growing number of buildings connected to real size distribution networks in an appropriate amount of time: long time-scale simulation of a DES can thus easily reach limits in terms of memory and simulation time when using one generic solver since most of them are mono-threaded. As a result, the simulation of large and complex DES usually leads to simplifications either on the building side or on the network side. The alternative is to distribute the simulation by decomposing the problem into smaller interconnected sub-problems. DACCOSIM is then a suitable candidate tool.

6.1.2 Model Description

We consider a fixed district model written in Modelica (Baetens et al., 2015) involving 12 grid-connected Smart Buildings in a heterogeneous district (Figure 8).

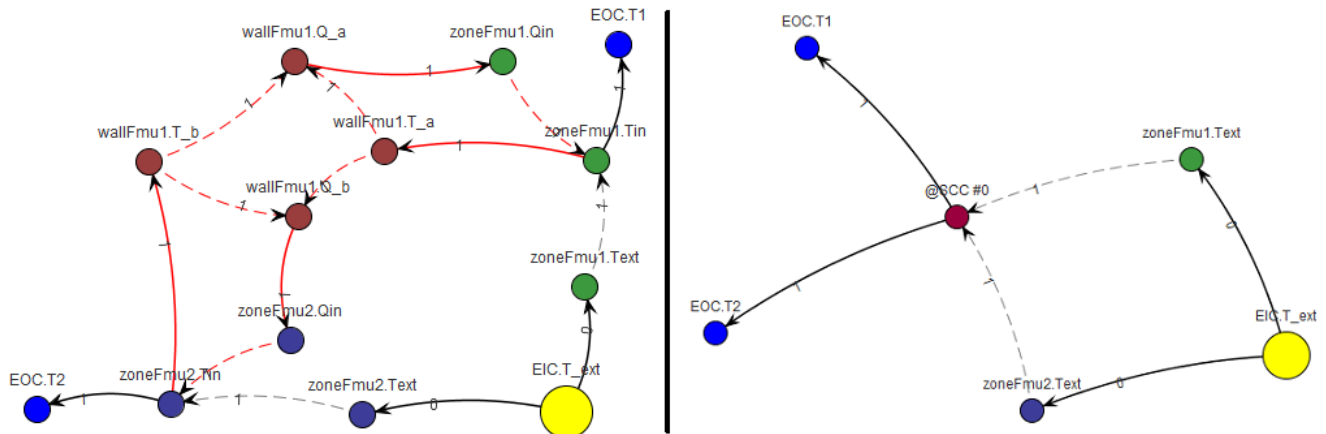


Figure 7. Complete causality graph (on the left) and its acyclic view (on the right) of a simple co-simulation graph.

For each building, we consider 3 thermal zones and one heating-pump (HP) connected to a 3-phases linear feeder. Thermal, electrical, ventilation, hot-water demand and occupancy profiles are heterogeneous and derive from a stochastic model (Baetens and Saelens, 2015). We employ complex quasi-stationary equations of the grid in order to study the influence of the load demand on the maximal/minimal tension of the grid (Protopapadaki et al., 2015). The impact of the MV network is also considered: it is modeled by a voltage source following real unbalanced LV busbar measurements. No hot water network is considered here.

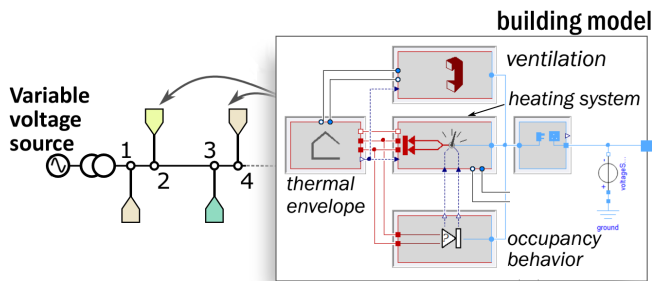


Figure 8. Illustration of the District Energy System use case

This basic scenario is already complex enough to exhibit scalability issues when using a standard solver. For illustration, it takes about 1 full day for 2 weeks of simulated time on our standard PC with Dymola 2016.

To distribute this use-case, the global model must be divided into multiple FMUs. The use of component-oriented modeling languages like Modelica usually makes the cutting decisions easy. Moreover, DES usually offer a lot of similarities, thus, one could consider creating communications between clusters of buildings, buildings or, even deeper, between heating systems, thermal envelope, and the network. In this section, we consider each building as one FMU, and the electrical network as another one, in order to simplify the understanding of the results. A smart handling of occupancy profiles has been implemented in the building models: the identity key, noted

$idOcc \in \{1, \dots, 12\}$, is related to resources profiles, *i.e.* electrical energy and hot water demands, occupancy and reference temperatures. This allows to keep the FMU general so that only a single profile needs to be loaded. The frequency of the network propagated to each of its components is also represented as an FMU, as is the LV voltage information imposed on the busbar.

This results in a system composed of 15 FMUs. All of them except for the LV voltage FMU are included within the Matryoshka FMU (Figure 9). This split allows to make the Matryoshka sensitive to its electrotechnical environment, *i.e.* to the MV network behavior. This would also allow to easily connect several DES Matryoshka FMUs to a MV network and see how they interact.

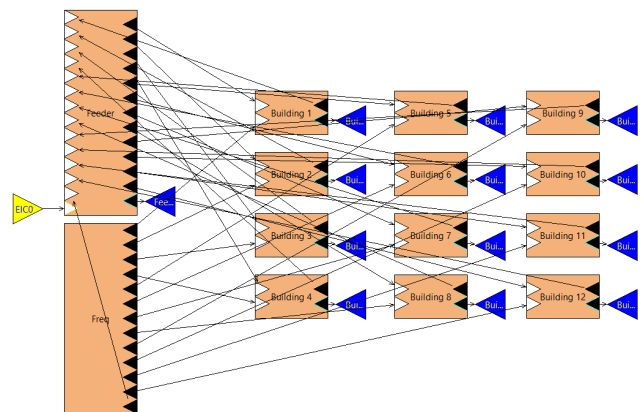


Figure 9. Screenshot of DACCOSIM showing the Matryoshka of the 12 Smart Buildings as a system of 14 FMUs with its external inputs and outputs

6.2 Description of the Experiments

The pure Modelica model simulated under Dymola is used as the reference for all the simulations performed in this article. This allows us to assess the performance of the other solutions in both their accuracy and computational time. The Matryoshka FMU is compared to this reference. This FMU is built by DACCOSIM integrating the electrical

network and its 12 Smart Buildings as FMUs. Each inner FMU is generated using Dymola 2016.

Similar tests are performed in DACCOSIM and Dymola environments: considering a constant step size, a voltage is imposed to the DES system FMUs. This input comes from a Modelica block in Dymola and from its FMU counterpart in DACCOSIM.

The comparisons are carried out for a simulated time from one to five days. Realistic demand and occupancy data as well as weather data are used, therefore creating a variability in the calculations between each simulated day. The accuracy of several variables is relevant regarding the validation of the design of the electric grid with Smart Buildings, among them:

- the inner temperature of the buildings;
- the norm of the voltage and the current;
- the correct capture of the extrema of these quantities.

The Dymola model was simulated with tolerances of 10^{-4} , 10^{-5} and 10^{-6} . Output points are saved every 60 s. The Matryoshka is set up with a relative tolerance on each FMU internal solver of 10^{-4} , as well as a relative tolerance on the outputs of the FMUs of 10^{-3} for temperature and voltage, and 10^{-2} for currents. Euler algorithm is used inside the Matryoshka to manage the step size, with a minimum step size of 1 s and a maximum of 40 s. The Matryoshka is then simulated along the voltage data FMU with a constant step size of 60 s. The results obtained for these four configurations for a one and five days of continuous simulation are shown in Table 1. It is clear that the results of the Matryoshka co-simulations are closer to the ones of the Dymola model with a tolerance of 10^{-6} than to the other Dymola setups. Thus in the following, the performance of the Matryoshka will be compared with the Dymola model with a tolerance of 10^{-6} on longer simulations.

All the simulations are performed on a laptop computer with 4 physical cores and 8 logical threads with a maximum speed of 2.50 GHz (Intel i7-4710MQ) and 8 Gb of RAM running under Windows 8.1 64 bit.

6.3 Results

6.3.1 Matryoshka Accuracy

Providing sufficient accuracy is a key-aspect of a co-simulation. Splitting a model such a DES into smaller subparts exported and then interconnected as FMUs creates propagation delays: they depend on the largest number of linked FMUs separating the start from the end of the co-simulation graph and the sum of the varying time step sizes observed for each propagation sequence. It is thus important to use time step adaptive strategies to shorten the time steps when model dynamics are important and enlarge them when they stabilize.

Using Euler adaptive approach in the Matryoshka, we obtained the cumulated distribution displayed in Figure 10. It illustrates the repartition of the absolute error of the Matryoshka model to the pure Modelica model chosen

as reference, respectively for voltage, current and temperature norms for the five days simulation case. 95.0 % and 99.8 % of the measurement points have an absolute error lower than 10^{-1} for respectively current and voltage. This is to be compared with the current Smart Meter capabilities: on voltage an accuracy of 0.5 % of the nominal voltage, i.e. about 1.15 V, is expected.

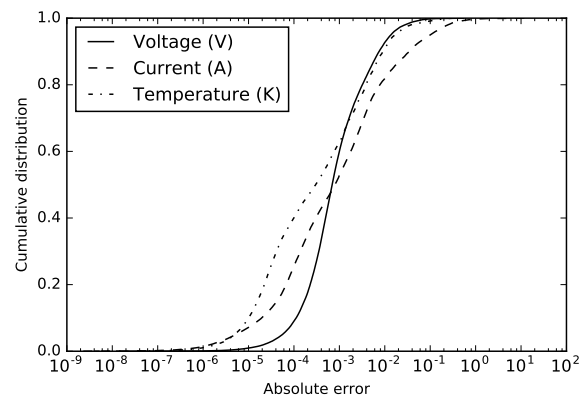


Figure 10. Cumulated distribution of the absolute error on voltage, current and temperature in one building zone: Matryoshka compared to pure Modelica simulation in the 5 days case

It is especially important to correctly capture the minimum and maximum values of both grid voltages and currents in order to properly design the grid. The extrema of the Matryoshka simulation should be close to the ones computed with the pure Modelica model so that we can use DACCOSIM results with as much trust as Dymola ones. Table 2 shows the minimum, mean and maximum error over the 12 buildings on the maximum values of current and voltage. The error is kept low with maximum errors of 5 mA, 0.7 mV and 1.6 mV.

With such error levels, using a Matryoshka in a co-simulation is relevant for distribution grid design. The representation of the use case dynamics as well as its accuracy are sufficient for a correct simulation of the network and its usages.

6.3.2 Matryoshka Computational Performance

The computation time of the pure Modelica model simulated under Dymola should not be lower than the one of the FMU co-simulation under DACCOSIM to make it relevant to use Matryoshka FMUs.

The results of the execution time measurement can be seen on Figure 11. The speed up starts around 1.5 for one day, grows and stabilizes itself around 3.5. This performance is quite interesting when doing simulation on long time scales. The changes of the speed up might be due to the variable calculation load induced by the different occupancy and weather profiles considered for every day.

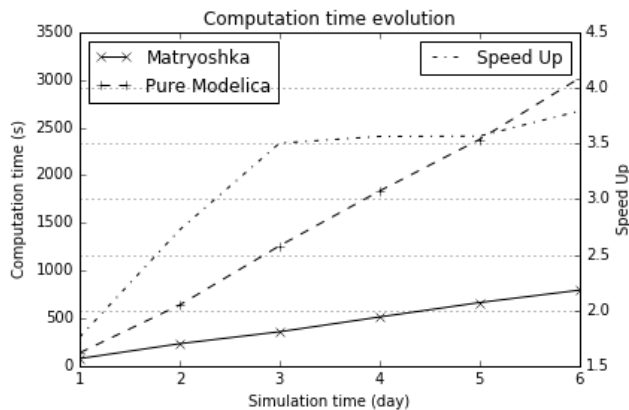
Using a Matryoshka co-simulation also enables to tune the tolerance on the relevant variables when doing simulations for design purposes. The user can thus have the accuracy he needs in a shorter time.

Table 1. Performances of the different configurations for one and five days simulations

Model	Mean RMSE over the 12 buildings					
	Computation time (s)		On current (A)		On voltage (V)	
	1 Day	5 Days	1 Day	5 Days	1 Day	5 Days
Dymola (10^{-4} tolerance)	191	2633	1.34	8.69×10^{-1}	1.17×10^{-1}	7.10×10^{-2}
Dymola (10^{-5} tolerance)	169	2329	3.25×10^{-1}	2.52×10^{-1}	3.13×10^{-2}	2.04×10^{-2}
Dymola (10^{-6} tolerance)	139	2375	reference	reference	reference	reference
Matryoshka	79	666	7.99×10^{-2}	1.39×10^{-1}	7.08×10^{-3}	1.18×10^{-2}

Table 2. Errors on extrema aggregated on the 12 buildings

Error type	Absolute Error
Max. error on max. voltage	6.97×10^{-1} mV
Max. error on min. voltage	1.62 mV
Max. error on max. current	5.04 mA

**Figure 11.** Computation time of the pure Modelica and FMU simulations with speed up

7 Conclusions and Future Work

The *Matryoshka FMU* we have presented in this paper and successfully implemented on a real-life test case is a *first of its kind* that is compliant with the latest version of the FMI 2.0 standard and built with an open-source solution DACCOSIM. The FMIBench commercial tool can also build hierarchical FMUs but supports fully only the version 1.0 of the FMI standard and we have no knowledge about the co-initialization and co-simulation features implemented within the embedded master.

By taking advantage of the FMI standard capabilities, a Matryoshka FMU can be easily *integrated within any FMI-CS compliant simulator* on any Windows or Linux 64 bits system. Such FMU could even be easily deployed on a node of a HPC-cluster environment. The use of DACCOSIM parallel master architecture allows to *achieve both computational efficiency and accuracy* thanks to its internal adaptive time step mechanisms and its capability to finely tune the tolerance on its variables. The JavaFMI

builder makes its *generation automatic*: once the simulated use-case is set, a single click in the DACCOSIM user interface generates such an FMU.

With DACCOSIM Matryoshka FMUs, complex real-life systems can thus be easily simulated, finely tuned, and improved in their computation efficiency while allowing an easy implementation within any FMI-CS compliant simulation environment.

Work is currently being carried out to further improve their capabilities. Some can be performed with the current FMI standard, while others require new attributes :

- When the user chooses the target platform and architecture (Linux, Windows or both) for the Matryoshka FMU we will check that the choice is conform with the platform(s) targeted by the inner FMUs.
- We are working to allow the Matryoshka FMU to save and restore its state (if all its inner FMUs have this capability). So, the Matryoshka could be included into any co-simulation which might require FMUs to rollback.
- We are investigating a way to build a Matryoshka which distributes its simulation on multiple cluster nodes. We also wish to include new information about the number of inner FMUs in its modelDescription.xml file. This would provide useful information about the number of created threads in order to automate its placement on HPC cluster nodes.

8 Acknowledgment

Authors thank Region Grand Est and RISEGrid institute for their support to this research. The modeling of the electrical network and the smart buildings was conducted within the EFRO-SALK project, which receives the support of the European Union, the European Regional Development Fund, Flanders Innovation & Entrepreneurship and the Province of Limburg.

References

- R. Baetens and D. Saelens. Modelling uncertainty in district energy simulations by stochastic residential occupant behaviour. *Journal of Building Performance Simulation*, (September), 2015.

- R. Baetens, R. De Coninck, F. Jorissen, D. Picard, L. Helsen, and D. Saelens. OPENIDEAS - An Open Framework for Integrated District Energy Simulations. In *Proceedings of Building Simulation 2015*, 2015.
- S. Baggi, D. Rivola, V. Medici, G. Corbellini, D. Strepparava, and R. Rudel. Modeling and Simimulation of a residential Neighborhood with Photovoltaic System Coupled to Energy Storage Systems. In *29th European Photovoltaic Solar Energy Conference and Exhibition*, 2014.
- T. Blochwitz and M. Otter. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. *8th International Modelica Conference*, 2011.
- B. Camus, V. Galtier, and M. Caujolle. Hybrid Co-simulation of FMUs using DEV and DESS in MECSYCO. In *Symposium on Theory of Modeling and Simulation*, 2016.
- J. Evora, J. J. Hernandez, and O. Roncal. JavaFmi. URL <https://bitbucket.org/siani/javafmi/>.
- V. Galtier, S. Vialle, C. Dad, J-P. Tavella, J-P. Lam-Yee-Mui, and G. Plessis. FMI-Based Distributed Multi-Simulation with DACCOSIM. In *Symposium on Theory of Modeling and Simulation - TMS'15*, 2015.
- C. Protopapadaki, R. Baetens, and D. Saelens. Exploring the impact of heat pump-based dwelling design on the low-voltage distribution grid. In *14th Conference of International Building Performance Simulation Association*, 2015.
- J-Ph. Tavella, M. Caujolle, S. Vialle, C. Dad, C. Tan, G. Plessis, M. Schumann, A. Cuccuru, and S. Revol. Toward an Accurate and Fast Hybrid Multi-Simulation with the FMI-CS Standard. *IEEE ETFA Track 9 - Information and Communication Technology in Energy Systems*, 2016.
- J. Vaubourg, Y. Presse, B. Camus, C. Bourjot, L. Ciarletta, V. Chevrier, J-P. Tavella, and H. Morais. Multi-agent Multi-Model Simulation of Smart Grids in the MS4SG Project. In *PAAMS'15*, 2015.
- M. Wetter, M. Bonvini, and T. Nouidui. Equation-based languages - A new paradigm for building energy modeling, simulation and optimization. *Energy and Buildings*, 2015.
- G. Zucker, F. Judex, M. Blöchle, M. Köstl, E. Widl, S. Hauer, A. Bres, and J. Zeilinger. A new method for optimizing operation of large neighborhoods of buildings using thermal simulation. *Energy and Buildings*, 2016.

Scaling FMI-CS Based Multi-Simulation Beyond Thousand FMUs on Infiniband Cluster

Stephane Vialle^{1,2} Jean-Philippe Tavella³ Cherifa Dad¹ Remi Corniglion³ Mathieu Caujolle³
Vincent Reinbold⁴

¹UMI 2958 - GT-CNRS, CentraleSupélec, Université Paris-Saclay, 57070 Metz, France

²LRI - UMR 8623, 91190 Gif-sur-Yvette, France

³EDF Lab Saclay, 91120 Palaiseau, France

⁴University of Leuven, Department of Civil Engineering, 3001 Leuven, Belgium

Abstract

In recent years, co-simulation has become an increasingly industrial tool to simulate Cyber Physical Systems including multi-physics and control, like *smart electric grids*, since it allows to involve different modeling tools within the same temporal simulation. The challenge now is to integrate in a single calculation scheme very numerous and intensely inter-connected models, and to do it without any loss in model accuracy. This will avoid neglecting fine phenomena or moving away from the basic principle of equation-based modeling.

Offering both a large number of computing cores and a large amount of distributed memory, multi-core PC clusters can address this key issue in order to achieve huge multi-simulations in acceptable time. This paper introduces all our efforts to parallelize and distribute our co-simulation environment based on the FMI for Co-Simulation standard (FMI-CS). At the end of 2016 we succeeded to scale beyond 1000 FMUs and 1000 computing cores on different PC-clusters, including the most recent HPC Infiniband-cluster available at EDF.

Keywords: Multi-Simulation, FMI, Scaling, Multi-core, PC Cluster

1 Introduction and Objectives

A multi-simulation based on the FMI for Co-Simulation standard is a graph of communicating components (named FMUs) achieving a time stepped integration, under supervision of a global control unit (named *Master Algorithm* in the standard), as illustrated in Fig. 1. During a time step, all FMU inputs remain constant and all FMUs can be run concurrently. When all FMU computations of a time step are finished, the FMU outputs are routed to connected FMU inputs, and all FMUs communicate with the Master Algorithm. When using adaptive time steps or managing events inside the time steps, the Master Algorithm has a complex role to decide on the next time step to execute.

In order to run wide multi-simulations requiring large memory and heavy CPU resource consumption, we first need to distribute and process a co-simulation graph on several PCs. Second, to achieve *scaling* we need (1) to speedup a co-simulation using more computing resources (cores and PCs), and (2) to strive to maintain the same execution time when running larger co-simulations on more

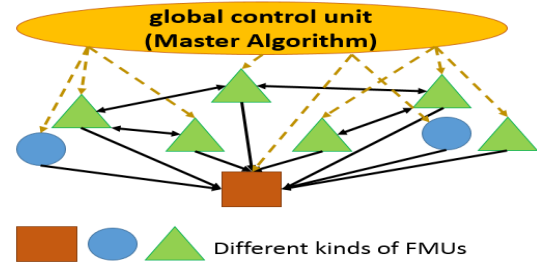


Figure 1. Generic FMU graph implementing a multi-simulation

PCs. Of course to achieve this, we attempt to minimize the global execution time as the sum of all the parallel FMU computation substeps, the FMU communication substeps, and the graph control substeps.

In 2014 we designed a distributed and parallel FMI based multi-simulation environment, named DACCOSIM¹ (Galtier et al., 2015), integrating a hierarchical and distributed Master Algorithm. Available under AGPL for both Windows and Linux operating systems, DACCOSIM² achieves a multi-threaded execution of local FMUs on each node with concurrent run of different FMUs on cluster nodes, and frequent data exchange between nodes (see section 2). Then, we decided to optimize and facilitate the execution of our environment on multi-core PC clusters, which are our typical computing platforms. Unfortunately, FMUs are kind of opaque computing tasks frequently exchanging small messages. They are very different from optimized High Performance Computing tasks, and we faced different difficulties. In 2016 we succeeded to exhibit scaling on a co-simulation of heat transfers in a set of n three-floor buildings, and we efficiently run up to 81 FMUs on a 12-node PC cluster with a 10 Gbit/s Ethernet interconnect and 6 cores per node (Dad et al., 2016). However, we needed to identify the optimal distribution of one building on a minimum set of cluster nodes after running several benchmarks, and we scaled our co-simulation replicating our initial building and its optimal distribution. This approach has proven it is possible to achieve scaling on distributed FMI based co-simulations, despite the unusual features of an FMU task graph, from a classic parallel computing point of view. We have carried on with this work, in order to automate the ef-

¹<https://daccosim.foundry.supelec.fr>

²Partially supported by Region Lorraine (France)

efficient distribution of wide co-simulations on large multi-core PC clusters, aiming to distributed thousands of FMUs on thousands of computing cores.

The paper is organized as follow. Next section describes the features of an FMU graph execution, the principles of its execution, and the choices done when designing the DACCOSIM architecture. Section 3 lists all sources of parallelism and also performance losses in an FMU graph running on a multi-core PC cluster, in order to design an efficient software architecture of multi-simulation. Section 4 investigates the distribution of the FMU graph on a multi-core PC cluster, with poor or rich meta-data on the FMU graph, in order to maximize performance. Then, section 5 introduces our new large scale benchmark detailing reached numerical results, performance and scalability. Finally, section 6 lists our current results and remaining challenges.

2 FMI-CS based Multi-Simulations

2.1 FMI-CS Strengths and Limitations

Modern electric systems are made of numerous interacting subsystems: power grid, automated meter management, centralized and decentralized production, demand side management (including smart charging for electric vehicle), storage, ICT resources. . . Beyond a consensus on the language to use, modeling wide and complex systems in one universal modeling tool implies to make some simplifications that may lead to minimize important phenomena. As historic and domain-specific tools validated their business libraries since a long time, the most rational approach to simulate wide Cyber Physical Systems (CPSs) consists in recycling specialized simulation tools in a co-simulation approach.

The Functional Mock-up Interface for Co-Simulation (FMI-CS) specification can now be considered as a well-established standard for co-simulation thanks to numerous developments done by industrial parties (Blochwitz et al., 2011). A growing number of business tools - like EMTP-RV³ for electromagnetic transient modeling - have adopted the standard and added FMI connectors to their products. FMI-CS allows to obtain a fairly realistic representation of the whole system behavior since all the subsystems are equally taken into account without the pre-eminence of a domain (e.g. ICT) on another (e.g. physics). It allows the building of stand-alone active components (FMUs) that can be executed independently of each other. FMUs exchange data (with other FMUs or with external components) only at some discrete communication points. In the time interval between two communication points each FMU model is simulated by its own numerical solver, and a *Master Algorithm* controls the FMU graph at each communication point (see Fig. 1).

An FMU for Co-Simulation consists of a ZIP file containing an XML-based model description and a dynamic

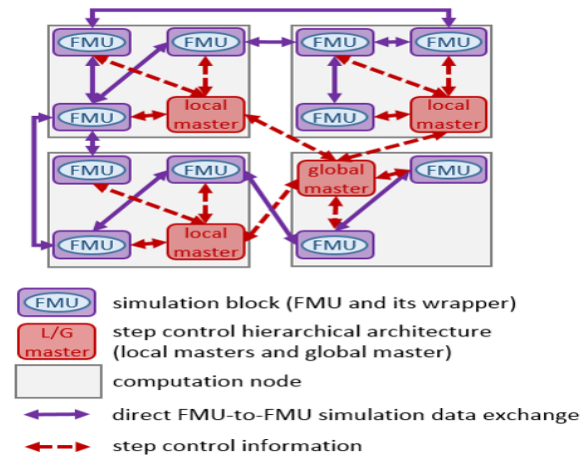


Figure 2. DACCOSIM software architecture

library being either a self-contained executable component or a call to a third-party tool at run-time (tool coupling). FMI-CS is focused on the slave side (FMU) and remains very discreet on the master side (Master Algorithm). The way a simulation tool utilizes the functionality provided by FMUs is strongly related to the simulation paradigm on which the tool relies:

- Some co-simulation middleware use the Agent & Artifact paradigm (Ricci et al., 2007) to describe an heterogeneous multi-model, and they rely on the Discrete Event System Specification (DEVS) formalism (Zeigler et al., 2000) to conceive a decentralized execution algorithm respecting causality constraints. But conservative algorithms, such as Chandy-Misra-Bryant (Chandy and Misra, 1979) used in some A&A tools like MECASYCO⁴, do not integrate the concept of rollback. They must be adapted to restore FMUs to a previous state and adjust the step size in case of events or fast system dynamics (Camus et al., 2016).
- Another class of tools is based on the synchronization of the communication points of all the FMUs involved in a calculation graph. Unfortunately these tools often stick to the master pseudo codes given as examples in the FMI standard with a centralized algorithm acting as a bottleneck for all the communication (data exchanges and control information).

2.2 DACCOSIM Architecture Choices

In our attempts to design, distribute and co-simulate on cluster nodes very wide systems composed of thousands of FMUs, we need both to synchronize all the communication points (for accuracy purpose) and decentralize the usual control function of the Master Algorithm (for performance purpose). First versions of these features were available within DACCOSIM in 2014.

DACCOSIM 2017 emphasizes a complete and user-friendly Graphical User Interface (GUI) to configure and perform local or distributed co-simulations with potentially many heterogeneous FMUs compliant with the co-

³<http://emtp-software.com/>

⁴<http://mecsyco.com/>

simulation part of the FMI 2.0 standard (FMI-CS 2.0). A DACCOSIM calculation graph consists of blocks (mainly FMUs) that are connected by data-flow links and potentially distributed on different computation nodes. The graph is then translated into Java master codes in conformance with the features described in the FMI-CS 2.0 standard. More precisely, DACCOSIM notably offers for the co-initialization of its calculation graph:

- Automatic construction of the global causal dependency graph, built both from the FMUs internal dependencies and the calculation graph external dependencies. An acyclic view of the graph is generated by aggregating each cycle as a super-node composed of Strongly Connected Components (SCCs);
- Generalized distributed co-initialization algorithm, mixing a sequential propagation method applied to the acyclic dependency graph, and a Newton-Raphson method solving its SCCs.

And for co-simulation, it offers among others:

- Implementation of each FMU wrapper as two threads allowing to concurrently run computations and send messages (FMU & control) while receiving incoming messages;
- Overlapped (optimistic) or ordered (pessimistic) data synchronization inside distributed masters (see section 3.3), that can operate with constant time steps or with adaptive time steps controlled by one-step methods (Euler or Richardson) or a multi-step method (Adams-Bashforth);
- Approximate event detection while waiting for a new version of the FMI standard able to correctly handle hybrid co-simulations (Tavella et al., 2016).

DACCOSIM generated master codes follow a centralized hierarchical approach (see Fig. 2). A unique global master located on one cluster node is in charge of handling the control data coming from several local masters located on other cluster nodes and taking step by step decisions based on this information. Every master, whether global or local, aggregates these control data that are coming from each FMU wrapper present on its cluster node. This is done before communicating synthesized information to the global master. The control data exchanged between masters and between FMUs and masters are called vertical data. Of course when the co-simulation is run on a single machine, only one master code is generated.

An original feature of the DACCOSIM architecture lies in the fact that the FMU variable values to be exchanged at each communication step are directly transmitted from the senders to receivers without passing by a master. The masters, the wrapped blocks (mainly FMUs with wrappers) as well as the communication channels between them are automatically generated by DACCOSIM by translating the calculation graph defined by the user via its GUI. All communications are implemented using ZeroMQ (or ZMQ)

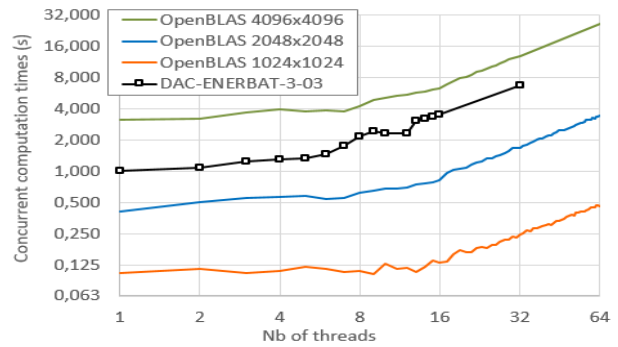


Figure 3. Concurrent run times on a 2x4-core node

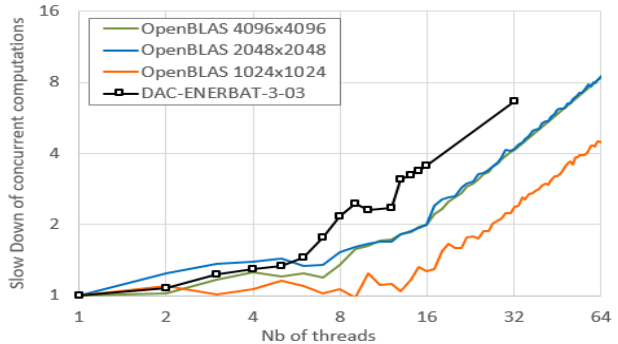


Figure 4. Slow down of concurrent runs on a 2x4-core node

middleware, allowing direct communications between different threads located on the same or on different PC cluster nodes. For intra-node communications, a mechanism of shared message queue is also available.

3 Parallelism Sources and Limitations

3.1 FMU Computations

Each computing substep is a high source of parallelism, as all FMUs can concurrently achieve their computations. However, running n_f FMUs on n_c cores of the same computing node can lead to imperfect concurrency: (1) when there are less cores than FMUs ($n_c < n_f$), and (2) when the FMU computations access the node memory and saturate the memory bandwidth. Taking into account this FMU concurrency imperfection, we will deduce the optimal number of FMUs to run on each computing node, and so the total number of nodes to use (see section 4).

3.1.1 FMU Concurrency Experiment

We concurrently run HPC computing kernels of *dense matrix product* ($C = A \times B$, a reference HPC benchmark) on one of our cluster computing node. We used an *OpenBLAS* *dgemm* kernel, and a NUMA dual-haswell node with 2×4 physical cores at 3.5 GHz, and 2×15 MB of cache memory. Fig. 3 shows the execution times measured on different problem sizes, with optimized *OpenBLAS* kernels blocking data in cache to minimize the memory bandwidth consumption. For each problem size, we concurrently run from 1 up to 64 threads, each thread executing one complete call to the kernel on locally allocated data structures. We considered (1) two large matrix sizes: 2048×2048 matrices of 32 MB and 4096×4096 matrices of 128 MB,

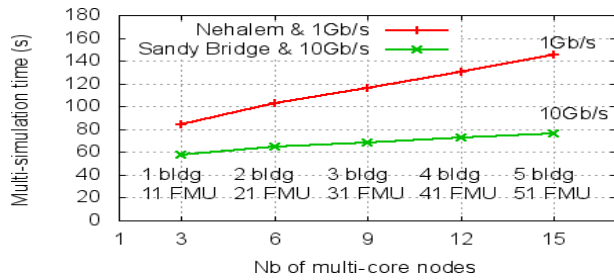


Figure 5. Size up experiments on 1 Gb/s and 10 Gb/s clusters

each matrix being larger than the entire node cache, and (2) a smaller problem with 1024×1024 matrices of 8 MB, allowing to store the three matrices of the problem into one node cache. Each curve illustrates the global execution time evolution when running more concurrent computations.

Fig. 4 shows the slow down observed when increasing the number of concurrent computations: $SD(n) = t(n)/t(1)$. Our optimized OpenBLAS kernel exhibits good concurrent performance, with a very limited slow down up to the 8 physical cores, followed by a first slow down increase up to the 16 logical cores (exploiting hyper-threading), and a linear increase when running more concurrent tasks serially processed by the node cores. Then, we concurrently run several threads executing the same FMU⁵, modeling heat transfers and achieving significant computations with the *cvode* solver⁶. We can observe in Fig. 4 that these concurrent FMU executions (1) exhibit a limited slow down, up to $(n_c - 2)$ FMU computing threads, similar to the behavior of concurrent OpenBLAS kernels, and (2) then quickly increase their slow down beyond $(n_c - 2)$ FMU computing threads per node, going away from OpenBLAS kernel curves.

In fact some extra tasks are running in parallel of the FMU computation threads in DACCOSIM, and it is not surprising the slow down starts to increase a little bit before deploying one FMU per physical core. But the slow down increase appears stronger than with OpenBLAS kernels, and is militantly in favour of running only $(n_c - 2)$ FMUs per computing node and using additive nodes. Of course, this experimental study will need to be conducted on different cluster nodes with different FMUs in the near future to confirm the definition of the ideal number of FMUs to deploy and run on a multi-core cluster node.

3.1.2 Unsuccessful Performance Improvement

When the number of available computing nodes is limited, it leads to run many FMUs on a same node, many more than $(n_c - 2)$. Then, we attempted to reduce the computation time limiting the number of FMUs simultaneously run in parallel on a same computing node. We implemented a semaphore-based synchronization-mechanism, authorizing only n_{max} FMUs to concurrently run their computa-

tions (a new FMU could enter its computation substep only when a previous one finished its substep).

But performance measured when limiting the concurrency of many FMUs on a same node were disappointing: the total computation time still increased. We have not succeeded to improve the execution of many concurrent FMUs per node with a basic scheduling mechanism.

3.2 FMU Communications

3.2.1 Main Features of Inter-FMU Communications

There is no order in the inter-FMU communications of a time step, they can all be routed in parallel, fully exploiting the cluster interconnect bandwidth. Moreover, FMU communications inside a computing node can be achieved faster (no crossing of network connections no network software layer). But data exchanged between two FMUs are usually small (like one or a few floating point values). Each FMU communication is sensitive to the network and applicative latency: time to transfer a byte from one JVM (running FMUs) on one node to another JVM on another node, in current DACCOSIM implementation. Moreover, an FMU graph has many connections generating communications at the end of each time step.

So, communication features of multi-simulations are different from classic HPC application ones, which always attempt to group data and exchange large messages not too frequently. FMU communications are small, numerous and frequent, however their implementation can be parallelized.

3.2.2 Sensitivity to Latency and Bandwidth

Respective weights of FMU computations and communications depend on the FMU graph and the multi-simulation. We have run some size up experiments on our multi-simulation of heat transfer inside buildings. We have implemented larger simulations using greater number of computing nodes, replicating buildings on new nodes. Theoretically, the execution time of the multi-simulation should have remain almost constant (FMU computation time remained constant on each node, and communications were routed in parallel). Experimentally, Fig. 5 shows the execution time increase on PC clusters with 1 Gb/s and 10 Gb/s Ethernet interconnects. This time increase is more limited on the 10 Gb/s Ethernet interconnect. These experiments of heat transfer multi-simulations have pointed out the importance of the communications and the sensitivity to the interconnect speed.

3.2.3 Difficulty to Fully Use Infiniband Interconnect

In order to reduce cost of these intensive and short communications, an interesting issue consists in using low latency and high bandwidth interconnects of standard HPC clusters, like some Infiniband networks. However, it requires to use some constrained middleware or communication libraries from HPC technologies (like MPI library), with native Infiniband interface. Using modern and comfortable middleware (like ZMQ in DACCOSIM environ-

⁵FMUs were designed at EDF Lab Les Renardières using BuildSysPro models, and generated with Dymola 2016

⁶Sundials suite of nonlinear and differential/algebraic equation solvers, of the LLNL's Center for Applied Scientific Computing

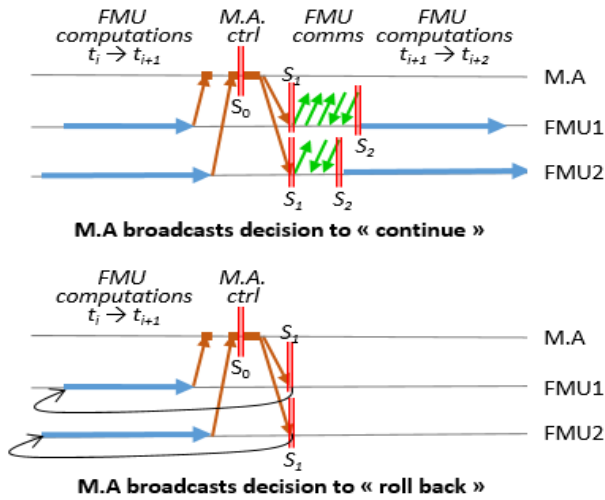


Figure 6. Relaxed synchronization of time step subparts

ment) leads to use Infiniband networks over TCP/IP adaptors and to loose their very low latency (Secco et al., 2014).

We attempted to use the MPI library to implement our multi-simulation communications, but MPI has been designed for process-to-process communications and appeared not adapted to DACCOSIM thread-to-thread communications, where each thread manages an FMU. We have currently suspended this investigation, and we use Infiniband networks over TCP/IP adaptors.

3.2.4 Minimizing Message Sizes

Current communication mechanisms of DACCOSIM send FMU output data as strings, and send input name strings instead of short input identifiers (like input indexes). Future versions of DACCOSIM will implement shorter data encoding in order to reduce message sizes and bandwidth consumption.

3.3 Time Step Subparts Orchestration

3.3.1 Ordered Orchestration with Relaxed Synchronization

Basic orchestration of a time step is illustrated on Fig. 6, and follows a *relaxed synchronization* mechanism. All FMU computations are run in parallel to progress from t_i up to $t_{i+1} = t_i + h$, and as soon as an FMU has finished its computation substep it sends its requirements to the Master Algorithm (M.A.): to roll back and rerun with a smaller time step (to increase accuracy), to continue with the same time step, or to continue with a greater time step. Then, the M.A. processes each received requirement but awaits all requirements (synchronization point S_0) before taking a global decision, and broadcasting its decision to all FMUs. All FMUs wait for the M.A. global decision, and as soon as an FMU receives the M.A. decision (sync. point S_1), it rolls back or continues its time step.

- If an FMU receives the command to continue (top of Fig. 6), it enters its communication substep, sending its output results to connected FMUs and waiting for the update of all its input values (sync. point S_2). Finally,

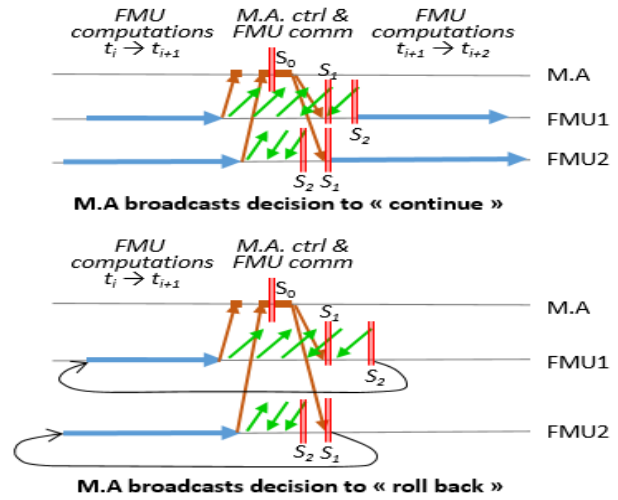


Figure 7. Overlapped orchestration mode

when it has updated all its inputs, it enters its next computation substep.

- If an FMU receives the command to roll back (bottom of Fig. 6), it restores its previous state at t_i and reruns its computation step, but progresses from t_i up to $t'_{i+1} = t_i + h'$, with $h' < h$ the new time step broadcasted by the M.A.

So, the only global synchronization barrier is the M.A. decision broadcast, that all FMUs are waiting for. Others synchronization points are *relaxed ones*, that stop only one task (the M.A. or one FMU). Then, each task going over a relaxed synchronization point carries on with its work independently of others tasks. Relaxed synchronization allows to increase performance, avoiding time consuming synchronization barriers and avoiding to synchronize all FMUs on the current slowest ones (the ones with longest computations or communications at current time step). Algorithms with relaxed synchronization schemes are usually more complex to implement and to debug, but ZMQ middleware has allowed an easy and efficient implementation of these communication and synchronization mechanisms between threads across a PC cluster.

3.3.2 Overlapping Strategy

To still reduce the communication cost, a solution consists in overlapping some of the communications with some FMU computations, and with the *Master Algorithm* decision pending. Fig. 7 illustrates these mechanisms. When an FMU has finished its computation substep, it sends its requirements to the M.A. and, not waiting for M.A. decision broadcast, enters its communication substep. So, FMUs update their input values while the M.A. collects their requirements and broadcasts its global decision.

But, depending on the pending time of the M.A. decision and on the number of inter-FMU communications, each FMU can cross its synchronization points S_1 (M.A. decision broadcast) and S_2 (all input update received) in any order (see FMU1 and FMU2 examples on Fig. 7). So, when both S_1 and S_2 synchronization points have been

crossed, each FMU considers the M.A. decision:

- If the M.A. has broadcasted a command to continue, then each FMU enters its new computation substep (see top of Fig. 7), and has saved some execution time achieving its inter-FMU communications while the M.A. decision was pending.
- If the M.A. has broadcasted a command to rollback, then each FMU waits for the end of its communications, restores its state at the beginning of the time step, and reruns its computation from t_i up to t'_{i+1} . In this case, the overlapping mechanism has a little bit increased the execution time, achieving unnecessary inter-FMU communications.

From a theoretical point of view, our overlapping mechanism reduces the execution time when there are few rollbacks, or when using constant time steps. But from a technical point of view, some threads will work to send and receive messages while some threads will achieve the end of long FMU computations (M.A. decision broadcast is no longer a synchronization barrier). The communication threads could disturb the ongoing computations and slow down the multi-simulation, especially when running more threads than available physical cores (see section 3.1). Nevertheless, our overlapped orchestration mode has appeared efficient on our multi-simulation of heat transfer inside three floor building, run on a 6-core node cluster with a 10 Gb/s Ethernet interconnect. Section 5 will show the performance achieved on our benchmark of power grid multi-simulation.

3.4 Event Handling Impact

To increase their genericity, it seems necessary for CPSs to handle more signal kinds especially *continuous & piecewise differentiable* signals, *piecewise continuous & differentiable* signals and *piecewise constant* signals, which are sources of *events*. The current FMI-CS 2.0 release (Blochwitz et al., 2011) can theoretically approach events thanks to for example a bisectional search using variable step size integration (Camus et al., 2016). But only events due to piecewise constant signal changes can be detected. And the solution involves bad performance as it is based on rollbacks and finally some inaccuracies appear due to the last non zero integration step size.

We proposed to add new primitives in the FMI-CS standard (Tavella et al., 2016) in order to integrate hybrid co-simulation in a pure FMI-CS environment. Our solution does not require any model adaptation and allows to couple physics model with continuous variability and controllers with discrete variability. Moreover, parallelism is not reduced by our approach, as all FMUs continue to run concurrently either when processing shorter time steps, or when executing rollbacks. So, event handling by the FMI-CS evolution we have proposed does not require to change our parallel and distribution strategy of FMU co-simulation graph.

From a computing performance point of view, this FMI-

CS standard improvement leads to execute a maximum of one rollback per FMU each time an unpredictable event appears during a time step. In the end, we do not know in advance how much the execution time will decrease but we are sure to achieve higher accuracy while improving the computation performance.

4 FMU Placement Strategies

4.1 Not a Dependence Graph Problem

A DACCOSIM program running a total of n_F FMUs is composed of n_F FMU wrapper tasks, n_F data receiver tasks, plus a local or global control task per computing node (implementing our hierarchical M.A., see section 2.2). Of course, a DACCOSIM program can be considered as a dependency task graph, and some strategies exist to distribute such a graph on a PC cluster (Sadayappan and Ercal, 1988; Kaci et al., 2016). However, a DACCOSIM task graph has some specific task dependencies. During one time step in ordered orchestration mode, all FMUs execute three substeps as illustrated on Fig. 8:

- The computation substep (Fig. 8 part *a*): all FMU wrapper tasks run concurrently and autonomously, achieving the FMU computations. There is no dependence between these tasks during this substep. The only optimizations consist in load balancing the FMU computations among the computing nodes, and to set only $n_c - 2$ FMU per nodes when there are enough available computing nodes, according to section 3.1.
- The control substep (Fig. 8 part *b*): each FMU wrapper task sends its wish to the control task for the next operation (rollback or continuation, and size of the future time step) and waits for its global decision. There is a total dependence of the control task to all the wrapper tasks, followed by a total dependence of all the wrapper tasks to the control task, close to a *synchronization barrier* for the FMU wrapper tasks (see section 3.3). There is no optimization to achieve when distributing the FMU graph, excepted to implement a local control task on each node to manage its FMU wrapper tasks.
- The communication substep (Fig. 8 part *c*): it is only achieved when no rollback is ordered by the control task. Each FMU wrapper task sends its new outputs to connected FMU inputs, while each FMU receiving task ensures the reception of the new input values of the FMU. These communication operations are not CPU consuming. So, we run in parallel up to $2 \times n_f$ tasks on each computing node hosting n_f FMUs, in order to optimize the communications (see section 3.3). All these tasks run without any synchronization nor dependence during the communication substep, and the only optimization when distributing the FMU graph consists in grouping on the same node the most strongly connected FMUs (fighting with the load balancing objective).

In fact, we can classify our DACCOSIM task graph as a *periodic task graph*. Its period is equal to one time step,

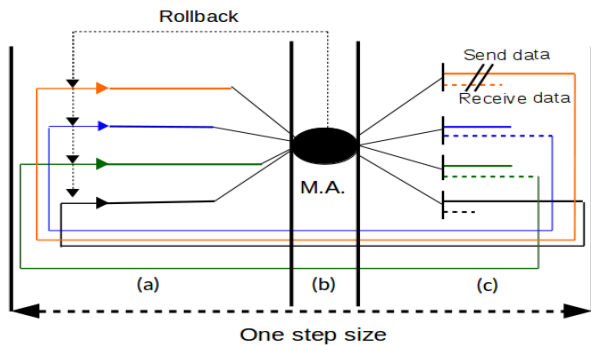


Figure 8. Multi-task synchronization overview (*ordered mode*)

and includes 2 phases (*a* and *c*) with pure concurrent task executions, and one phase (*b*) which is a kind of synchronization barrier (with only the control task working). So, we do not consider the task dependencies to distribute our FMU graph on a PC cluster. We focus on load balancing, on grouping the most connected FMUs, and on limiting the number of FMU per nodes (when there are enough available nodes).

IFP EN and INRIA succeeded to parallelize computation inside wide FMUs thanks to a fine scheduling of basic operation executions on one multi-core node (Saidi et al., 2016). The practical speed-up observed by our colleagues is achieved by imposing a constrained allocation of all the operations of a same FMU to the same core. Their approach is complementary to ours as they optimize the co-simulation of FMUs on the different processors of the same calculation node while we are optimizing the deployment of a calculation graph composed with lots FMUs on a possibly wide set of multi-core nodes.

4.2 Different Contexts and Approaches

The main trouble to establish a good distribution of the FMU graph on a PC cluster is the lack of metadata about FMU computations in the FMI-CS standard. There is no information about FMU computation time, or computation complexity. Dynamic load balancing is out of reach of our current implementation, and static load balancing of the computations on the nodes of a PC cluster remains difficult. This section introduces the different approaches we identified to distribute FMU graphs.

4.2.1 Previous Experimental Approach

In the beginning of 2016 we distributed on two PC clusters a first multi-simulations of heat transfers inside buildings. Each building was a subgraph of only 10 FMUs. We ran a small one-building problem setting only one FMU per node, so that FMUs could run the real simulation without disturbing each other (not sharing cache memory, nor memory bandwidth, nor cores...). We measured the computation time of each FMU (to *characterize* our different FMUs), and then we established the most load balanced FMU distributions on various number of nodes. Finally, some complementary experiments allowed to identify the most efficient distribution of a one-building problem on each PC cluster.

When the best distribution of a one-building problem (using n_0 nodes) was identified, we enlarged the problem with k buildings, replicating our best distribution (using $k \times n_0$ nodes). We successfully *scaled up* (Dad et al., 2016): processing larger problem on larger cluster in similar time. But this approach takes too long development times, and replicating the best elementary distribution leads to use too many nodes, some cores remaining unused. This approach cannot be a generic solution.

4.2.2 Approach function of the User Knowledge

From our point of view, distributing a totally unknown FMU graph or a fully characterized FMU graph should be infrequent DACCOSIM use cases. Users build co-simulations based on their expertise and have an initial knowledge about computation loads and communication volumes in their FMU graphs, allowing to use basic distribution mechanisms. When testing and improving their co-simulations they accumulate knowledge on their FMU graphs, and can use more complex heuristics. However, it is not obvious to design an efficient heuristic.

During the development phase of a co-simulation many FMU graphs are only run a few times and the FMU graph distribution has to be computed quickly, without long calibration steps. But when a co-simulation design is finished and successful, it can enter a long exploitation phase, requiring frequent runs. Then, it can be profitable to make detailed performance measurements and to compute a fine distribution of the FMU graph, in order to use less computing nodes and/or to decrease the co-simulation time.

Considering current and future usage of DACCOSIM at EDF, for smart grid co-simulations, we identified 3 levels of user knowledge, and we propose 3 associated generic FMU distribution approaches.

a - Identifying FMU Families: when users have only minimal technical and skill information about their co-simulations, and are able just to group the FMUs in families with close computing load.

Proposed approach: each family will form an FMU list, and the concatenated list of all FMU families will be distributed on the computing nodes according to a round-robin algorithm. This approach requires light knowledge on the FMUs used, and succeeded to load balance the FMU computations on our benchmark (see section 5).

Extreme use case: If no information is available on some external FMUs, it is possible to group these FMUs in a particular family to spread over the computing nodes. If no pertinent information is available on any FMUs, it is also possible to group all FMUs in a unique family, to achieve a random distribution and to track a statistical load balancing.

b - Cumulating Knowledge for Heuristics: when users progress in their co-simulation development they improve their knowledge about their FMUs and FMU graph. This extra-knowledge can be exploited by more or less generic heuristics to improve the FMU distribution. For example:

- running and testing different configurations of the FMU graph allows to learn some relative computing weights (ex: $t_{FMU2}^{comput} \approx 0.5 \times t_{FMU1}^{comput}$),
- analyzing the FMU graph allows to detect some regular patterns strongly connected (ex: a city area connected on one medium voltage network of a smart grid).

Proposed approach: design and use an heuristic (1) to optimize load balancing in order to reduce the global computation time, and/or (2) to group on same nodes the FMUs strongly interconnected in order to reduce the global communication time.

Warning: our experiments have shown the load balancing optimization is the most important criterion, however designing an efficient heuristic (improving performance of the previous approach) remains difficult.

c - Building Models of Computation and Communication Times: when an FMU graph enters an exploitation phase, it can be profitable to establish an execution time model of the co-simulation, to optimize its distribution and the computing resource usage.

Proposed approach: (1) run smaller but similar co-simulations, deploying only one or very few FMUs per node (to avoid mutual disruption between FMUs) and measure each FMU computation time on the nodes of the target cluster, (2) analyse the FMU graph to compute the volume of each inter-FMU communication, and measure the experimental applicative latency and bandwidth on the target cluster. Then, establish a computation and a communication time model of the co-simulation, to feed a solver looking for the best distribution of the FMU graph.

Warning: This approach requires long experimental measurements.

The IDEAS test case described in section 5, has been distributed on different PC clusters according to the *Identifying FMU families* and the *Cumulating knowledge for heuristics* approaches.

4.3 FMU Distribution on Virtual Nodes

When the FMU graph is defined, the DACCOSIM software suite distributes the FMUs and generates Java source files for a set of *virtual nodes*, and maps the *virtual nodes* on the available *physical computing nodes* at runtime. Then the Java source codes are compiled, a JVM is started for each virtual node and its Java program is executed. We defined intermediate *virtual nodes* in order to generate Java source files independent of physical node names and IP addresses, and to make the deployment more flexible on nodes with Non Uniform Memory Architecture (NUMA).

Modern computing nodes have several processors and memory banks interconnected across a small network. But memory access time becomes function of the distance between the core running the code and the memory bank storing the data (NUMA principle). Creating one process (one JVM, one virtual node) per NUMA subnode in-

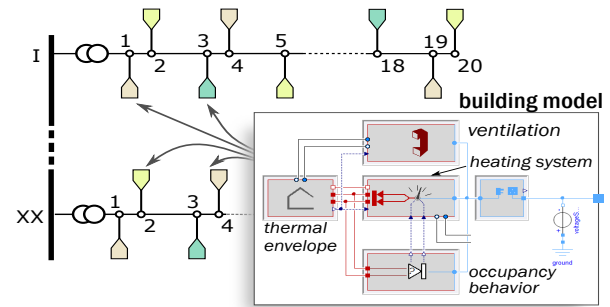


Figure 9. Topology of the large scale testbed using IDEAS lib.

stead of creating a unique JVM per node managing all the threads, can increase data locality and performance.

5 Large Scale Experiments

5.1 Experiment Objectives

The co-simulation of a large scale District Energy System was chosen as a testbed. Co-simulation methods are foreseen to handle several bottlenecks encountered during CPS simulation on one single simulation tool, such as:

- Multi-Physics integration (electrical, hydraulic, thermal, etc.),
- Multiple time-scales and dynamics,
- Implementation of controllers,
- Scalability, i.e. the capability to study a growing number of buildings and the growing size of the power grid.

The numerical experiment consists in a complex multi-physical district energy system. The main purpose of this section is thus to propose a proof of concept of co-simulation with lots of FMUs on a HPC cluster and to highlight the advantages of such an approach for large scale systems.

5.2 Testbed Description

In this section, we propose to assess DACCOSIM Master Algorithm efficiency by co-simulating an electrical distribution grid using a variable number of cluster nodes. The model has been completely implemented using Modelica and the OpenIDEAS library⁷ (Baetens et al., 2015). Neither the electrical grid, the heating systems nor the building envelopes have been simplified.

The general structure of the use case is shown on Fig. 9. It is composed of 1000 buildings connected to low voltage (LV) feeders, each of them including a thermal envelope, ventilation and heating systems and a stochastic occupancy behavior. The buildings are dispatched on 20 low voltage LV feeders, each modeled as one FMU, noted I to XX on Fig. 9. These feeders are connected to a medium voltage (MV) network that is also simulated with a single FMU. A data-reading FMU provides real medium voltage measurements that are imposed at the MV substation busbar. The electric grid frequency is provided to different FMUs (buildings and feeders) by 20 additional FMUs.

⁷EFRO-SALK project, with support of the European Union, the European Regional Development Fund, Flanders Innovation & Entrepreneurship and the Province of Limburg

This distributed frequency FMU implementation is meant to reduce inter-node communications since the frequency has to be dispatched to all the FMUs of the use case. Finally, the co-simulation holds a total of 1042 FMUs exported from Dymola 2016 FD01 in conformance with the FMI-CS 2.0 standard. A smaller use-case with less buildings and only 442 FMUs, has also been designed to evaluate the scalability of our solution.

The test case was run on two different clusters: (1) *Sarah* at CentraleSupélec Metz, composed of dual 4-cores Intel Xeon E5-2637 v3 at 3.5 GHz (Haswell) with a 10 Gb/s Ethernet communication network, and (2) *Porthos* at EDF R&D, composed of dual 14-cores Intel Xeon E5-2697 v3 at 2.60 GHz (Haswell) with Infiniband FDR communication network. These clusters are labeled "sar" and "por" on performance curves of section 5.4. On both clusters, DacRun is used to deploy and run the DACCOSIM co-simulation. DacRun is implemented in Python 2.7, is compliant with OAR and SLURM cluster management environments, and can also be used on standalone machines (for small experiments). It achieves Java source files compilation, virtual/physical nodes mapping, JVMs starting and can ensure to gather the results and logs.

5.3 Numerical Results

The runs are done for a one-day simulation with one-minute constant step size. The co-simulation gives realistic results according to expert judgment. Moreover the energy consumption of the buildings follows the same trend as the one observed on a Dymola simulation limited to one 20-building feeder. To assess the correctness of the co-simulation on cluster, we selected a single building of the test case and simulated it with Dymola by injecting sampled voltage data obtained from the cluster co-simulation. The power consumed by the building simulated with Dymola and the one co-simulated on cluster should be the same as the two selected buildings have the same environment: same input voltage, same weather data and same occupancy data. The root mean square error on the current between those two simulations is 1.16×10^{-2} A, with current mainly in the range 1 – 10A. The two currents for the one-day simulation are plotted on top of Fig. 10 with a close-up on its bottom. The dynamic of the power consumption is well reproduced thus the cluster co-simulation seems reliable.

5.4 Performance and Scaling

The FMUs were dispatched on the nodes following two different approaches introduced in section 4.2: with (1) a *Cumulated knowledge for heuristic* approach exploiting the problem topology with balanced load ("KHBL" on Fig. 11), and (2) according to an *Identifying FMU families* approach associated to a round-robin mechanism ("FFRR" on Fig. 11). Experimentations were conducted on our clusters in the ranges 32 – 1024 and 112 – 1792 cores, with *overlapped* and *ordered* orchestration modes ("over" and "order"), on both 442 and 1042 FMUs use-cases.

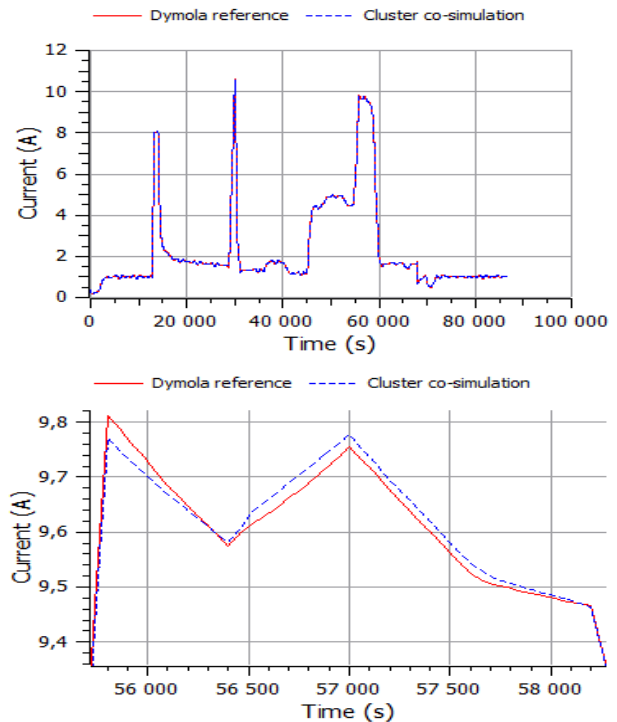


Figure 10. Current from a building of the DACCOSIM co-simulation and its Dymola counterpart

Scalability Achievement: time curves on Fig. 11 appear very linear on this full logarithmic scale graphic, slope is close to -1 on HPC Porthos cluster, and time curves of different problem sizes are parallel. So, execution time regularly decreases when using more cores, and similar performance can be achieved when running larger problems on larger number of cores (from 442 to 1042 FMU benchmark curves). Of course, when using as many cores as FMUs the execution stops to decrease (right-hand side of Porthos curves).

Interconnect and Communication Impact: time curve slope is smaller on Sarah cluster and its 10 G/s Ethernet interconnect, than on Porthos and its high performance Infiniband FDR interconnect. Communications are not negligible, and a high performance interconnect (low latency and high bandwidth) improves the scalability.

Complex Choice of the Orchestration Mode: Overlapped mode was the fastest one on a previous use case run on a cluster with smaller nodes (Dad et al., 2016). But when running IDEAS use case on Sarah cluster, the overlapped mode appears slower than the ordered one, and when run on Porthos cluster, both orchestration modes have close performances up to allocate enough nodes to get one core per FMU. Beyond this limit it remains free cores on each node to manage communications, and the execution time of the overlapped mode roughly decreases and really becomes the smaller one. So, both orchestration modes are interesting, but strategy to foresee the right one is still under investigation.

Difficulty to Design Efficient Heuristics: our heuristic based on FMU graph knowledge, aiming to group connected FMUs on the same node with respect to load bal-

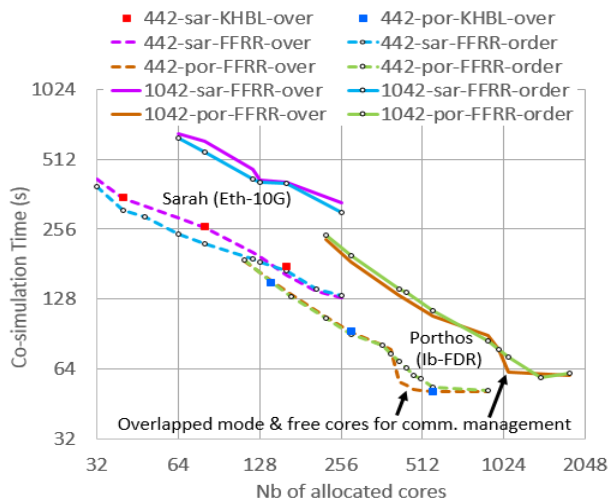


Figure 11. IDEAS benchmark with 442 FMUs run on clusters

ancing, requires in our testbed some accurate number of nodes (5, 10 or 20 on our example) and does not achieve better performance than round-robin distribution of FMU families. An efficient heuristic remains hard to design and our round-robin on FMU families algorithm appears a good solution

6 Conclusion and Perspectives

With DACCOSIM generating Java files for Linux and its Python add-in DacRun easily compiling, running and collecting the results of a DACCOSIM application on clusters, we have illustrated in this paper the capability of our FMI-CS based environment to manage very wide co-simulations. Our testbed is a realistic case study using the OpenIDEAS library and involving the detailed modeling of 1000 buildings scattered on a distribution grid. We have demonstrated the feasibility of scaling-up the multi-simulation by pushing very far the limits of the simulation and taking advantage of Porthos, the EDF cluster ranked 310th in the 48th edition of the TOP500 list published in November 2016.

Work is currently being carried out to further improve the capabilities of our co-simulation tools suite. Some can be performed with the current FMI-CS 2.0 standard (e.g. minimizing inter-FMU message sizes), while others would require an evolution of the standard (e.g. event handling of accurate hybrid co-simulation).

A collection of generic heuristics for FMU graph distribution, when knowledge on co-simulation has been accumulated, is also under development, to make easier large scale deployments of more complex co-simulations.

References

R. Baetens, R. De Coninck, F. Jorissen, D. Picard, L. Helsen, and D. Saelens. OPENIDEAS - An Open Framework for Integrated District Energy Simulations. In *Proceedings of Building Simulation Conference 2015 (BS 2015)*, Hyderabad, India, December 2015.

T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß,

H. Elmquist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J.-V. Peetz, and S. Wolf. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In *Proceedings of the 8th International Modelica Conference*, Dresden, Germany, March 2011.

B. Camus, V. Galtier, and M. Caujolle. Hybrid Co-Simulation of FMUs using DEV and DESS in MECSYCO. In *Proceedings of the 2016 Spring Simulation Multiconference, Symposium on Theory of Modeling and Simulation (TMS/DEVS'16)*, Pasadena, CA, USA, April 2016.

K. M. Chandy and J. Misra. Distributed Simulation: A Case Study in Design and Verification of Distributed Programs. *IEEE Trans. Softw. Eng.*, 5(5), September 1979.

C. Dad, S. Vialle, M. Caujolle, J.-Ph. Tavella, and M. Ianotto. Scaling of Distributed Multi-Simulations on Multi-Core Clusters. In *Proceedings of 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2016)*, Paris, France, June 2016.

V. Galtier, S. Vialle, C. Dad, J.-Ph. Tavella, J.-Ph. Lam-Yee-Mui, and G. Plessis. FMI-Based Distributed Multi-Simulation with DACCOSIM. In *Proceedings of the 2015 Spring Simulation Multiconference, Symposium on Theory of Modeling and Simulation (TMS/DEVS'15)*, USA, April 2015.

A. Kaci, H. N. Nguyen, A. Nakib, and P. Siarry. Hybrid Heuristics for Mapping Task Problem on Large Scale Heterogeneous Platforms. In *Proceedings of the 6th IEEE Workshop on Parallel Computing and Optimization (PCO 2016), IPDPS Workshop 2016*, Chicago, IL, USA, May 2016.

A. Ricci, M. Viroli, and A. Omicini. Give Agents Their Artifacts: The A&A Approach for Engineering Working Environments in MAS. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*, Honolulu, HI, USA, May 2007. ACM.

P. Sadayappan and F. Ercal. Cluster-partitioning Approaches to Mapping Parallel Programs Onto a Hypercube. In *Proceedings of the 1st International Conference on Supercomputing (ICS 1988)*, Athens, Greece, June 1988. Springer-Verlag.

S. E. Saidi, N. Pernet, Y. Sorel, and A. Ben Khaled. Acceleration of FMU Co-Simulation On Multi-core Architectures. In *Proceedings of 1st Japanese Modelica Conference*, Tokyo, Japan, May 2016.

A. Secco, I. Uddin, G. P. Pezzi, and M. Torquati. Message Passing on InfiniBand RDMA for Parallel Run-Time Supports. In *Proceedings of the 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2014)*, Turin, Italy, February 2014.

J.-Ph. Tavella, M. Caujolle, S. Vialle, C. Dad, Ch. Tan, G. Plessis, M. Schumann, A. Cuccuru, and S. Revol. Toward an Accurate and Fast Hybrid Multi-Simulation with the FMI-CS Standard. In *Proceedings of the IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA 2016)*, Berlin, Germany, September 2016.

B. P. Zeigler, T. G. Kim, and H. Praehofer. *Theory of Modeling and Simulation : Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, 2000.

Development of an open source multi-platform software tool for parameter estimation studies in FMI models

Javier Bonilla^{1,3} Jose A. Carballo^{1,3} Lidia Roca^{1,3} Manuel Berenguel^{2,3}

¹CIEMAT-PSA, Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas - Plataforma Solar de Almería, Spain, {javier.bonilla, jose.carballo, lidia.roca}@psa.es

²Department of Informatics, University of Almería, Almería, Spain, beren@ual.es

³CIESOL, Solar Energy Research Center, Joint Institute University of Almería - CIEMAT, Almería, Spain

Abstract

This paper presents the current development of an open source multi-platform software tool intended for estimating or optimizing parameters of Functional Mock-up Interface (FMI) compliant models. Parameter estimation and optimization is a powerful tool in many engineering and science fields. Nevertheless, the effort and time that must be devoted to coupling and integrating complex modeling languages and tools together with analysis and optimization methods and algorithms sometimes is high. As a consequence of that, commonly the most convenient and easy-to-use optimization mechanisms are applied. Therefore, the focus on the development of this tool is in facilitating such coupling while being customizable. The main toolkit and libraries used in the development of the tool are presented, all of them are open source. Two application examples are also presented, one of them is a parameter optimization study considering a steady state model, while the other is a parameter estimation study of a dynamic model against experimental data. Finally, current tool limitations are presented, ongoing work and ideas for future features are also commented.

Keywords: parameter estimation, parameter optimization, model calibration, Functional Mock-up Interface (FMI), open source software tool

1 Introduction

The application of optimization to complex dynamic models has become recently more usual in industry, as well as in academia. Optimization can be online, such as optimal control in the form of Model Predictive Control (MPC) or offline, such as parameter estimation, state estimation or parameter optimization.

In parameter optimization, also known as design optimization, some parameters are optimized to improve the system dynamics or response according to some criteria. Parameter estimation, model calibration or parameter identification, comprises estimating some unknown parameters in a particular model. To that end, several simulations are performed and results are compared against experimental data. The unknown parameters are therefore determined by numerical optimization algorithms. This

procedure is a powerful tool in many engineering and science fields and has its origin in the least squares method proposed by Gauss.

In order to apply optimization techniques to complex dynamic models, a suitable modeling language, that can deal with dynamic systems and the increasing complexity of research and engineering needs, is advisable. Modelica (Modelica Association, 2014b) is one of those modeling languages, easing the model development, maintenance and reuse thanks to the equation-based object-oriented paradigm and other useful features. Nevertheless, there are other commonly used modeling languages and simulation tools, being one of the most representative Matlab/Simulink (The MathWorks Inc., 2016). For this reason, the support of an independent standard devoted to Model Exchange (ME) and co-simulation, such as FMI (Modelica Association, 2014a), would be advisable when considering the model interface for an optimization software tool.

Even though there are modeling languages and tools for developing complex dynamic models, as well as a standard format to exchange those models, and advanced methods for optimization, sometimes the time required to couple all these tools is high. This task involves writing scripts for bindings and using several programming languages and tools.

With the aim of facilitating the integration of these tools and methods, an easy-to-use open source multi-platform software tool which performs parameter estimation studies in Functional Mock-up Units (FMUs) is currently being developed. This software tool performs parameter estimation studies using a global-search Multi-Objective Genetic Algorithm (MOGA). The tool also supports linear and non-linear equality and inequality constraints.

This paper is organized as follows. Section 1.1 is a brief summary of Modelica and FMI-based tools for parameter estimation. Section 2 describes the still under development software tool and its architecture. In Section 3, two examples are presented. Section 3.1 shows a steady-state parameter optimization study, whereas Section 3.2 presents a model calibration study against experimental data from a Thermal Energy Storage (TES) tank. This tank is used for research on solar thermal storage.

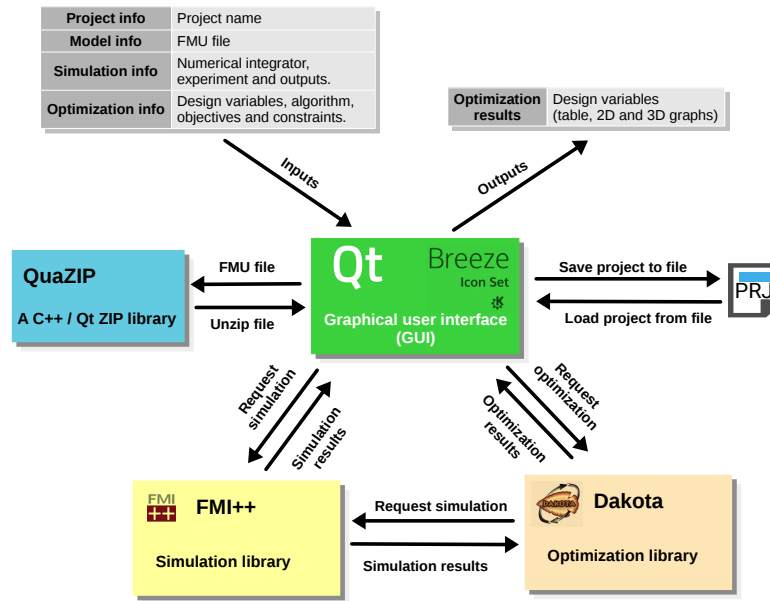


Figure 1. Optifmus information exchange

1.1 State of the art

Most commercial Modelica tools have parameter estimation and optimization libraries for Modelica models, i.e. Model Design Tools (Elmqvist et al., 2005; Pfeiffer, 2012) in Dymola (Dassault Systemes, 2016) and corresponding libraries in SimulationX (ESI ITI GmbH, 2016), MapleSim (MapleSoft, 2016) and SystemModeler (Wolfram, 2016), among others. GenOpt (Wetter, 2001) is an optimization program that can be coupled with Modelica models compiled in Dymola. BuildingsPy (Berkeley Lab, 2016) is a Python package that can run Modelica simulations using Dymola, additional Python packages for parameter estimation can be used by means of scripting, such as those from SciPy.org (SciPy developers, 2017). The OpenModelica tool (OSMC, 2016) includes the OMOptim tool (Thieriot et al., 2011) for parameter estimation of Modelica models in Windows platforms.

Modelica models can be also exported as FMUs and imported in commercial and open source numerical computational tools such as Matlab/Simulink and Scilab (Scilab Enterprises, 2015). Parameter estimation studies can be performed by means of scripting in these tools. JModelica.org (Åkesson et al., 2010) supports parameter estimation of Modelica and FMI models also by scripting. The RaPid Parameter Identification (RaPid) toolbox (Vanfretti et al., 2016) is a modular and extensible toolbox for parameter estimation of FMI models in Matlab/Simulink.

2 Optifmus Software Tool

The under development software tool is called Optifmus. Although it is at an early development stage, it is functional with respect to FMU simulations and parameter estimation studies using a MOGA. The tool is composed of the following main elements.

- **Graphical User Interface (GUI).** The GUI allows the user to provide all the required information: model information (FMU file and parameters), simulation information (numerical solver and its configuration, inputs and simulation interval), optimization information (algorithm and its configuration, parameters, objective functions and constraints). The GUI also shows the obtained results. Results are presented in tables, 2D and 3D graphs.
- **Optimization toolkit.** This toolkit collects all the information introduced in the GUI and calls the FMU simulator to perform the needed simulation runs and carry the optimization out using the selected algorithm. Optimization results are presented to the user in the GUI.
- **FMU simulator.** The simulator performs the model simulation according to the supplied data (model and simulation information) and provides the results to the GUI or to the optimization toolkit.

2.1 Software Architecture

Optifmus is being developed in C++ using open source multi-platform libraries and tools. The following list briefly describes the main libraries. Figure 1 shows the information exchange between them.

- **Qt toolkit** (The Qt Company, 2016). It is a cross-platform application framework used for developing application software. The Qt Core and Qt Widgets modules are used for the GUI. The Qt Charts module is used for 2D graphs, whereas the Qt Data Visualization module is used for 3D graphs.
- **Breeze icons** (KDE Community, 2016). The GUI icons belong to this open source library.

- **FMI++** (Widl et al., 2013). The FMI++ library is a high-level utility package for FMI-based software development. It provides high-level features, which ease the handling and manipulation of FMU models: an eXtensible Markup Language (XML) parser and numerical integration capabilities. The FMI++ library relies on the Odeint library (Ahnert and Mulansky, 2011), and optionally on Suite of Nonlinear and Differential/ALgebraic Equation Solvers (SUNDIALS) (Hindmarsh et al., 2005), for the numerical integration of FMUs.
- **QuaZIP** (Tachenov, 2016). It is a C++ wrapper for accessing ZIP files. This wrapper uses the Qt toolkit and therefore it is a multi-platform wrapper. It is used in *Optifmus* to handle the extraction of FMU files.
- **Dakota** (Adams et al., 2016). The Dakota toolkit is intended as a flexible, extensible interface between simulation codes and a variety of iterative systems analysis methods. Dakota is a powerful toolkit which provides the following functionality: optimization, uncertainty quantification, nonlinear least squares methods, and sensitivity/variance analysis. Dakota uses Sandia-developed libraries, as well as external optimization and design of experiments libraries. For further details consult Sandia Corporation (2016). Dakota can be used as a standalone application or as a C++ library.

Additionally, reading and writing operations of input and result files in Comma-separated Values (CSV) or trajectory mat format are performed by means of C functions available in the source code of the OpenModelica tool.

3 Examples

In order to show the *Optifmus* capabilities and illustrate how a parameter estimation study can be performed, the following sections introduce two examples. Section 3.1 shows a steady-state parameter optimization study, whereas Section 3.2 shows a model calibration study against experimental measurements from a real facility.

3.1 Parameter optimization

The general formulation for a optimization problem description is given by Equation 1. It can be formulated as optimize (minimize or maximize) several objective functions $f(x)$ that depend on some parameters or design variables x subject to several constraints: upper and lower bounds for design variables, x_l and x_u , equality constraints, $g(x)$ and inequality constraints, $h(x)$.

$$\begin{aligned}
 &\text{optimize} && f(x) \\
 &\text{with respect to} && x \in \mathbb{R}^j \\
 &\text{subject to} && x_l \leq x \leq x_u, \\
 & && g(x) = 0, \\
 & && h(x) \leq 0,
 \end{aligned} \tag{1}$$

where,

$$\begin{aligned}
 f(x) &= \{f_1(x) \cdots f_i(x)\}, \\
 x &= \{x_1 \cdots x_j\}, \\
 x_l &= \{x_{l,1} \cdots x_{l,j}\}, \\
 x_u &= \{x_{u,1} \cdots x_{u,j}\}, \\
 g(x) &= \{g_1(x) \cdots g_k(x)\}, \\
 h(x) &= \{h_1(x) \cdots h_n(x)\}.
 \end{aligned}$$

The example considered in this section, known as Srinivas' problem, can be found in the Dakota User's Manual (Adams et al., 2016), section *Additional examples* → *Multiobjective test problems* → *Multiobjective test problem 3*. The problem has two design variables, x_1 and x_2 with their respective upper and lower bounds,

$$\begin{aligned}
 -20 &\leq x_1 \leq 20, \\
 -20 &\leq x_2 \leq 20,
 \end{aligned}$$

two objective functions, f_1 and f_2 , which must be minimized,

$$\begin{aligned}
 f_1(x_1, x_2) &= (x_1 - 2)^2 + (x_2 - 1)^2 + 2, \\
 f_2(x_1, x_2) &= 9x_1 - (x_2 - 1)^2,
 \end{aligned}$$

and two inequality constraints h_1 and h_2 ,

$$\begin{aligned}
 h_1(x_1, x_2) &= x_1^2 + x_2^2 - 225 \leq 0, \\
 h_2(x_1, x_2) &= x_1 - 3x_2 + 10 \leq 0.
 \end{aligned}$$

The first step is to generate a FMU file of this model, most Modelica tools support exporting Modelica models to FMUs. The Modelica code of this model is as follows.

```

model mogatest3
  parameter Real x1 = 0 "Parameter x1";
  parameter Real x2 = 0 "Parameter x2";
  output Real f1 "Function f1";
  output Real f2 "Function f2";
  output Real h1 "Constraint h1";
  output Real h2 "Constraint h2";
equation
  f1 = (x1-2)^2 + (x2-1)^2 + 2;
  f2 = 9*x1 - (x2-1)^2;
  h1 = x1^2 + x2^2 - 225;
  h2 = x1 - 3*x2 + 10;
end mogatest3;

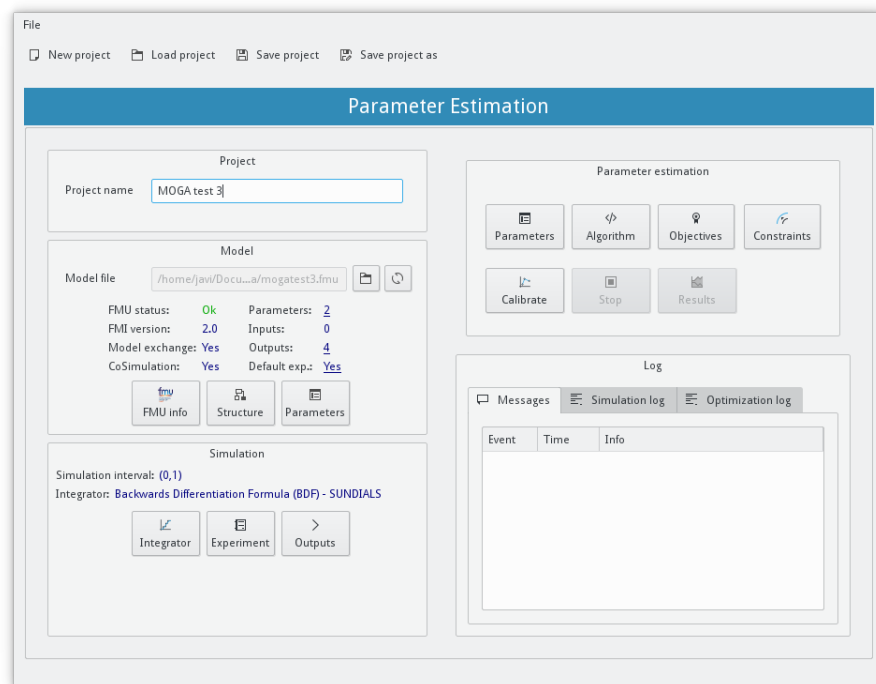
```

The next step is to create a new project in *Optifmus*. Currently, two kinds of projects can be created: simulation and parameter estimation. Projects can be saved to and loaded from files. Figure 2 shows the *Optifmus* GUI for parameter estimation. The information that must be completed is divided in groups in the GUI and it is described as follows.

1. **Project information.** A descriptive name can be given to easily identify the project.
2. **Model information.** A FMU file must be specified. Once the file is loaded, some information is

Table 1. Numerical integrators

Numerical integrator	Library	Step size	Order
Forward Euler	Odeint	Constant	1
4 th order Runge-Kutta	Odeint	Constant	4
Adams-Bashforth-Moulton	Odeint	Constant	Adjustable
5 th order Runge-Kutta-Cash-Karp	Odeint	Controlled	5
5 th order Runge-Kutta-Dormand-Prince	Odeint	Controlled	5
8 th order Runge-Kutta-Fehlberg	Odeint	Controlled	8
Bulirsch-Stoer	Odeint	Controlled	Controlled
4 th Rosenbrock	Odeint	Controlled	4
Backwards Differentiation Formula (BDF)	SUNDIALS	Controlled	Controlled
Adams-Bashforth-Moulton	SUNDIALS	Controlled	Controlled

**Figure 2.** Optifmus GUI for parameter estimation.

displayed in the GUI. There are three buttons in this group. The FMU info button shows some information about the model, see Figure 3a. The structure button shows the model structure, see Figure 3b. The parameters button allows the user to give values to the model parameters, see Figure 3c. Since in our case, both parameters x_1 and x_2 are going to be calibrated, there is no need to give them values.

3. **Simulation information.** There are also three buttons in this group: numerical integrator, experiment and outputs. The first one allows us to select the numerical integrator, Figure 4a. The step size is only needed if it is not controlled by the integrator. If it is controlled, absolute and relative tolerance are used instead. Some integrators allows the users to specify the order whereas others have a fixed constant or a controlled order. The FMI++ library can use the numerical integrators given in Table 1 from the Odeint

and SUNDIALS libraries. The numerical integrator (BDF) and tolerances (10^{-4}) are left by default in our example. The experiment window permits selecting the simulation interval, start and stop times, and matches model inputs with data from files. Since our model does not have inputs and it is a steady-state model, this step can be omitted and thus leaving the simulation interval by default, [0,1] seconds. Section 3.2 shows how to use this window.

The outputs window, see Figure 4b, shows the model outputs, furthermore allows us to specify the number of intervals, i.e. the number of points that will be sampled for the output trajectory. The number of points can be also set by a time step instead of by a fixed number. Since our model is a steady-state one, there is no need to sample more than one interval. One interval means that values at the beginning and end of the simulation are stored.

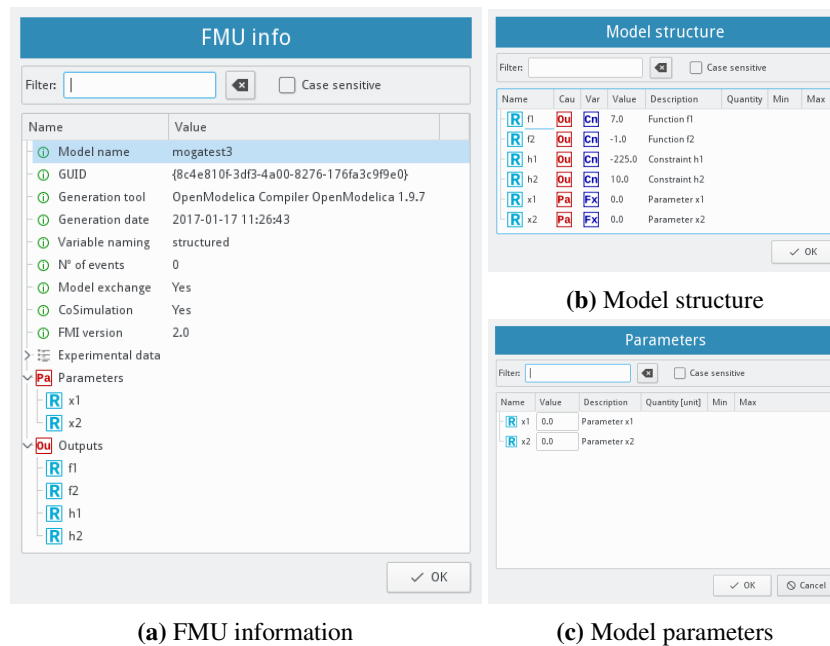


Figure 3. Model information

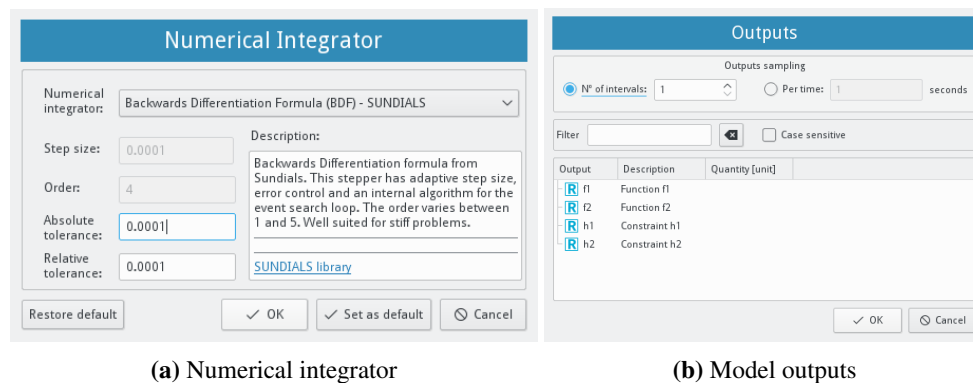


Figure 4. Simulation information

4. **Parameter estimation information.** This GUI group gathers all the information for the parameter estimation study: parameters or design variables, optimization algorithm, objective functions and constraints.

- **Parameters.** The parameters to be calibrated can be selected in the parameter window, see Figure 6. For each parameter, it can be specified its initial value, its lower and upper bounds, and if scaling is considered. If this is the case, the scaling type can be a fixed value, logarithm scale or automatic. The two first require a scale value. Consult Dakota documentation for further information about scaling of design variables (Adams et al., 2016).
- **Algorithm.** Currently, the only Dakota algorithm considered in the tool is the MOGA from the Sandia-developed JEGA library (Eddy and Lewis, 2001). This is a multi-objective algorithm which supports general constraints: bounded design variables, linear and nonlinear equality and inequality constraints.

The algorithm is highly configurable. Figure 5 shows the window to configure the algorithm. The options selected in our case are those indicated in Dakota documentation for this example. The number of model evaluations is set to 2000.

- **Objectives.** The objectives can be selected in the objectives window, see Figure 7. For each objective the following options are available: criterion (maximize or minimize), trajectory reduction, weight, scaling type and value. The trajectory reduction option reduces the whole trajectory for each objective function to a single value in each simulation run. This is required because the optimization algorithm needs a single value per objective function and simulation run. The following options are available: root mean of squares, mean of absolute values, max, min or last value. The weight value is used if the option to use weights, and therefore transform the multi-objective problem to a single-objective problem, is checked.

Parameter Estimation - Algorithm

Algorithm

Multi-objective Genetic Algorithm (MOGA)

Search method

Free derivative

Search domain

Global

Type of constraints

Nonlinear

MOGA stands for Multi-objective Genetic Algorithm, which is a global optimization method that does Pareto optimization for multiple objectives. It supports general constraints and a mixture of real and discrete variables.
[Dakota MOGA documentation](#)

Filter:

☐ Case sensitive

Property	Value
Fitness type	Domination count
Replacement type	Below limit
Below limit	6
Shrinkage fraction	0.9
Convergence type	
Metric tracker	
Percent change	0.1
Num generations	10
Postprocessor type	
Orthogonal distance	
Max iterations	100
Max function evaluations	2000
Population size	50
Log file	JEGAGlobalLog
Print each population	
Initialization type	Unique random
Crossover type	
Crossover type	Multi point parametrized binary
Crossover rate	0.8
Mutation type	
Mutation type	Offset normal
Mutation scale	0.1
Mutation rate	0.08
Seed	10983
Convergence tolerance	0.0001

Info property

Select the fitness type for JEGA methods.

Info item

Rank each member by the number of members that dominate it.

OK

Cancel

Figure 5. Optimization algorithm configuration

Parameter Estimation - Selected parameters

Filter:

☐ Case sensitive

Name	Value	Description	Quantity	Min	Max	Scaling type	Scaling value
<input checked="" type="checkbox"/> R x1	0	Parameter x1		-20	20	None	1
<input checked="" type="checkbox"/> R x2	0	Parameter x2		-20	20	None	1

OK

Cancel

Figure 6. Selected design variables

Parameter Estimation - Constraints

Linear equality constraints

Linear inequality constraints

Nonlinear equality constraints

Nonlinear inequality constraints

	Lower	s	Constraint	s	Upper	Scaling type	Scaling value	
1	0		x1		-3			

Constraint

x1 - 3*x2 ≤ -10

OK

Cancel

Figure 8. Linear inequality constraints

Parameter Estimation - Objective functions

☐ Apply weights to transform a multiobjective problem to a unique composite objective function

Filter:

☐ Case sensitive

Objective	Description	Quantity [unit]	Criterion	Reduction	Weight	Scaling type	Scaling value
<input checked="" type="checkbox"/> R f1	Function f1		Minimize	Last value	1	None	1
<input checked="" type="checkbox"/> R f2	Function f2		Minimize	Last value	1	None	1
<input checked="" type="checkbox"/> R h1	Constraint h1		Minimize	Root mean square	1	None	1
<input checked="" type="checkbox"/> R h2	Constraint h2		Minimize	Root mean square	1	None	1

OK

Cancel

Figure 7. Objective functions

Parameter Estimation - Constraints

Linear equality constraints

Linear inequality constraints

Nonlinear equality constraints

Nonlinear inequality constraints

	Lower	s	Constraint	s	Upper	Scaling type	Scaling value	
1	0		h1		0			

Constraint

h1 ≤ 0

OK

Cancel

Figure 9. Nonlinear inequality constraints

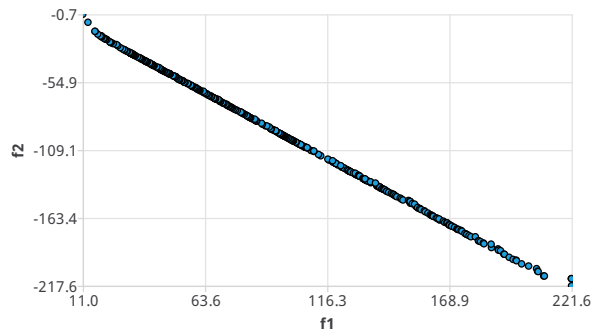


Figure 10. Srinivas' problem - 2D graph

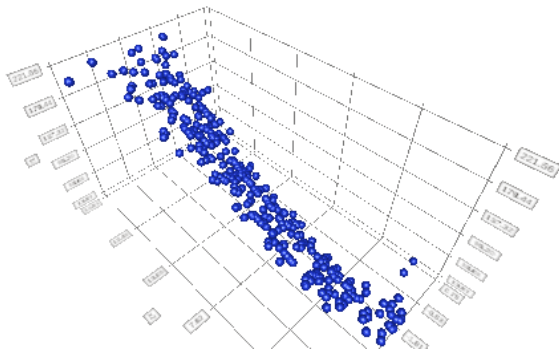


Figure 11. Srinivas' problem - 3D graph

The scaling option has the same meaning than for design variables, but there are some limitations for objective functions, consult Dakota documentation for further details. In our example, for both objective functions the criterion is minimize. Since this is a steady-state model, reductions are set to last simulation values. Scaling is not used and weight are not enabled because this is a multi-objective optimization problem.

- Constraints.** The tool supports linear and nonlinear equality and inequality constraints. Scaling options are also available. Linear constraints with respect to design variables can be directly specified in the appropriate tab in the constraint window. In our example, the linear constraint h_2 was defined in the Modelica code, but this was not necessary since it can be directly defined in the GUI, see Figure 8. On the other hand, it can be also defined in the model as in our example. Nonlinear constraint functions must be defined in the model, in our example h_1 , then both or only one limit per constraint must be set in the GUI, see Figure 9.

Once all previous information is defined, the optimization process can be performed hitting the calibration button, see Figure 2. The stop button allows us to stop the current optimization process. When the calibration process is completed, the results button will be enabled to show

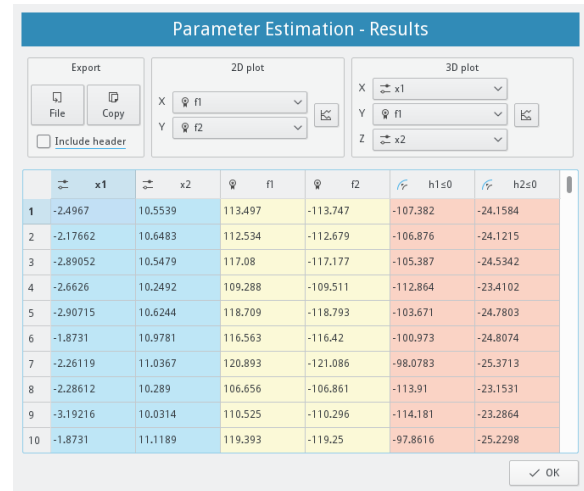


Figure 12. Srinivas' problem - optimization results

information about the optimization results. Our optimization example took less than 5 seconds for 2000 model evaluations in a conventional laptop (4 x Intel Core i5 2.60 GHz, 8 Gbytes of RAM). During the process, messages and log information are shown in the GUI.

Optimization results are shown in the results window, see Figure 12. The MOGA algorithm provides all the solutions found in or close to the Pareto front. In our case, the MOGA algorithm found 421 solutions after 2000 model evaluations. Design values, objective functions and constraints for each solution are shown in a table. In the result window, any design variable, objective function or constraint can be selected to be plotted in 2D or 3D graphs. Figure 10 shows a 2D graph of f_2 with respect to f_1 , which is the Pareto front of our problem. Figure 11 shows a 3D graph of f_1 with respect to x_1 and x_2 .

3.2 Model calibration

This section presents the calibration of a TES tank dynamic model that it is under development. This kind of tanks is used in solar thermal power plants in order to store thermal energy and dispatch it at night or under unfavorable meteorological conditions. A complete description of the model is out of the scope of this paper, but a brief summary is given in the following lines. The storage fluid is commonly molten salts, which can reach high temperatures.

The dynamic model considers two control volumes and dynamic mass and energy balances for molten salt and the inert gas in the facility, nitrogen. Tank geometry, slope and dimensions are considered in the model. The pump inside the tank is also modeled, assuming a simplified geometrical form. The position of the level meter and thermocouples in the tank are also taken into account. The model considers different kinds of heat transfer processes: convection, conduction and radiation between molten salt, gas, tank walls, roof, floor, pump, insulation and foundation. The variables of interest are tank level, together with molten salt and gas temperatures.

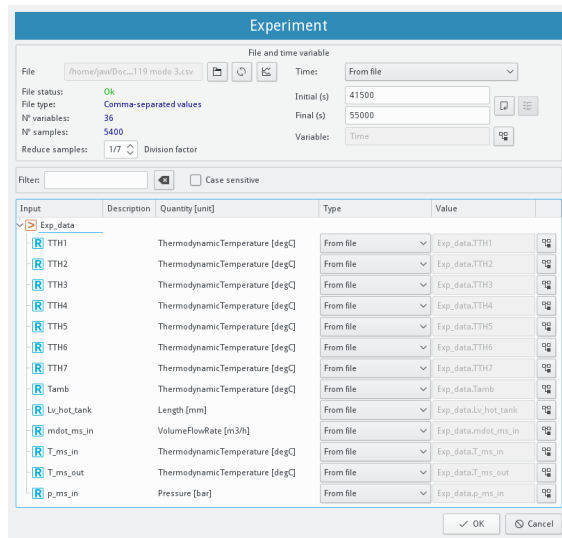


Figure 13. Tank calibration experiment

Molten salt tank foundation designs are commonly outside of standards for foundations, since these standards do not cover the temperature range where TES systems operate. The modeled tank has several foundation layers made of concrete with steel fibers and a compacted light expanded clay aggregate. The thermal insulation is usually made of several layers from different materials. The outer thermal insulation layer is frequently covered with an aluminum jacket for weather protection. For all these reasons, thermal conductivities in the insulation and foundation are difficult values to estimate. In this model, those thermal conductivities are assumed as mean constant values and have been calibrated using the *Optifmus* tool. Needed measurements are available from experimental campaigns carried out in the facility. It is located at CIEMAT - Plataforma Solar de Almería (PSA). Therefore, our model calibration problem can be formulated as minimizing the differences between molten salt and gas experimental and simulated temperatures by tuning the mean insulation and foundation thermal conductivities.

The steps to perform the calibration are similar to those in Section 3.1, for example setting the project (step 1) and model (step 2) information. The remaining steps are briefly summarized in what follows.

3. **Simulation information.** The experiment window is used to match model inputs with experimental data stored in files, see Figure 13. The values in the input file can be visualized thanks to a plotting tool included in *Optifmus*. In case the input file is over-sampled, a factor can be used to reduce the number of samples. Model inputs can be also fixed to constant values in this window if needed. Time values can be read from the loaded file or a number of time intervals between the start and stop times can be specified in the GUI. In our example, the simulation interval is set to [41500, 55000] seconds. The number of samples is reduced by 1/7 in order to reduce the

calibration time, since samples were taken each five seconds. Model inputs and time are matched to file data. The numerical integrator is left by default. The number of intervals is set to 500 in the output window in order to capture several points in the output trajectories.

4. **Parameter estimation information.** There are no constraints in our example, the remaining configuration options are described as follows.

- **Parameters.** Insulation and foundation mean thermal conductivities (k_{ins}, k_{fou}) are the design variables, guess initial values are $0.15 \text{ W}/(\text{m K})$. They are bounded in the $[0, 1]$ interval.
- **Algorithm.** The MOGA algorithm is used with its default values, besides the number of model evaluations (1000) and the seed (1) in order to obtain reproducible results.
- **Objectives.** The differences between experimental and simulated molten salt and gas temperatures are the two objective functions ($T_{ms,diff}, T_{gas,diff}$), they must be minimized. The trajectory reduction was set to root mean square values, therefore both objective functions provide the mean temperature difference in the trajectory. There is no need to use scaling since both objective functions represent temperature differences. Weights are not used because we are considering a multi-objective minimization problem.

The calibration process took 21 minutes and found 157 solutions in or close to the Pareto front for 1000 model evaluations. Figure 14 shows 10 of those solutions, where the objective functions give the mean temperature difference between experimental data and simulation results. Figure 15 shows the Pareto front.

The first solution in Figure 14, and pointed out by the arrow in Figure 15, was used to compare the model results against a different set of experimental data. All figures shown in this section were created in the *Optifmus* plotting tool. The system was exposed to several mass flow rate steps in this experiment, see Figure 16. Figure 17 shows experimental and simulated tank levels. Horizontal lines point out the position in height of the thermocouples. The gas experimental temperature corresponds to that from the highest-placed thermocouple, whereas the molten salt temperature is obtained from the highest-placed thermocouple immersed in molten salts. Figure 18 shows the experimental and simulated molten salt temperatures. Notice that there is no experimental molten salt temperature until the tank level reaches the first thermocouple position. This is why the experimental temperature is set to a constant value at the beginning of the simulation. Figure 19 shows the experimental and simulated gas temperatures. Notice that the molten salt level reaches the last thermocouple at the end of the simulation, therefore there are no available measurements for gas temperature.

	Tank.k_ins	Tank.k_fou	Tms_diff	Tgas_diff
1	0.240195	0.465693	1.55434	2.0533
2	0.240195	0.464505	1.55488	2.05292
3	0.240195	0.471824	1.55155	2.05566
4	0.240195	0.473722	1.55069	2.05635
5	0.240195	0.453473	1.56001	2.04869
6	0.240195	0.473362	1.55085	2.05633
7	0.240195	0.451157	1.5611	2.04785
8	0.240195	0.447909	1.56263	2.04661
9	0.240195	0.446876	1.56312	2.0462
10	0.240195	0.440346	1.56626	2.0438

Figure 14. Tank model calibration results

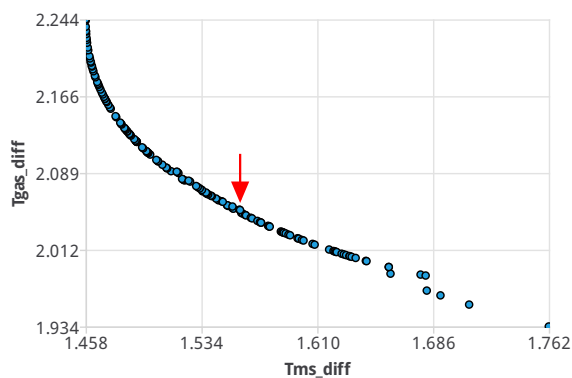


Figure 15. Tank model calibration Pareto front

4 Optimus limitations

The current main Optimus limitations are listed here.

- The software tool has been tested only in Linux. It is planned to be tested in Windows and Mac platforms.
- Only ME FMUs version 1.0 and 2.0 are supported.
- Only continuous real design parameters, objectives functions and constraints are supported.
- All the MOGA options are configurable from the GUI besides the niching type and the use of surrogate models which are not supported.

5 Ongoing work and future ideas

Ongoing work is summarized in the following list.

- Optimize the code to improve speed.
- Unit support when setting parameter values.
- Load FMI++ logs in the Optimus GUI.
- More graphic configuration options.
- Include an option to simulate the model with the parameters of the selected row in the result table.

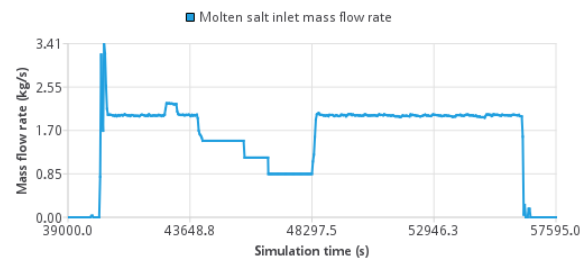


Figure 16. Tank simulation - mass flow rate

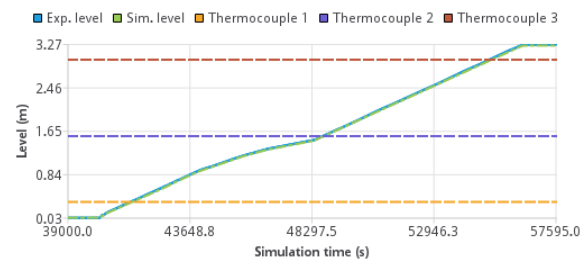


Figure 17. Tank simulation - levels

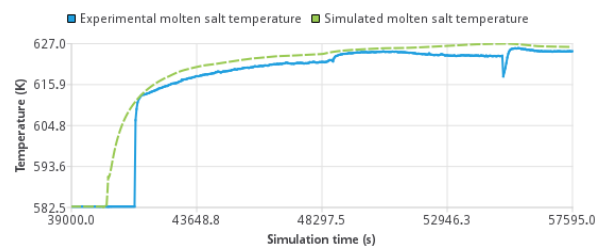


Figure 18. Tank simulation - molten salt temperatures

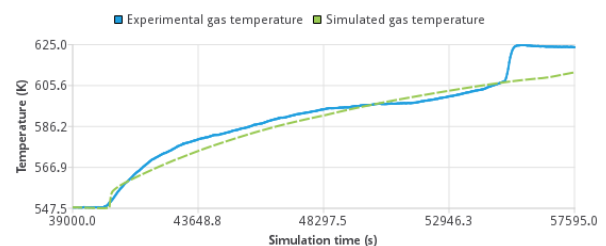


Figure 19. Tank simulation - gas temperatures

- Dakota offers many more algorithms for parameter estimation and optimization. One of the ongoing tasks is to implement several of those algorithms.

As future ideas to improve the software tool, the following will be considered and studied.

- Dakota is a powerful toolkit, other features that could be added to the tool are: parameter studies, sensitivity analyses, design of experiments, uncertainty quantification, and model simplification by means of surrogate models.
- The development of an Application Programming Interface (API) independent of the GUI could be useful for integrating FMI optimization capabilities in other

tools. It could be applied to offline optimization, as well as to online optimization, for example in MPC.

- Parallelization of the software tool could drastically reduce the optimization time. Dakota offers capabilities for parallelization. If we consider parallelization, as well as an API, executable programs could be generated and executed in high-performance clusters to further reduce the computational time.

Acknowledgments

This work has been funded by the National Plan Project DPI2014-56364-C2-1/2-R (ENERPRO-EFFERDESAL) of the Spanish Ministry of Economy, Industry and Competitiveness and ERDF funds.

References

- Brian M. Adams, Mohamed S. Ebeida, Michael S. Eldred, Gianluca Geraci, John D. Jakeman, Kathryn A. Maupin, Jason A. Monschke, Laura P. Swiler, J. Adam Stephens, Dena M. Vigil, and Timothy M. Wildey. Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.5 User's Manual, 2016.
- Karsten Ahnert and Mario Mulansky. Odeint - Solving ordinary differential equations in C++. In *AIP Conference Proceedings*, volume 1389, pages 1586–1589, 2011. ISBN 9780735409569. doi:10.1063/1.3637934.
- Johan Åkesson, Karl-Erik Årzén, Magnus Gäfvert, Tove Bergdahl, and Hubertus Tummescheit. Modeling and Optimization with Optimica and JModelica.org—Languages and Tools for Solving Large-Scale Dynamic Optimization Problems. *Computers and Chemical Engineering*, 34(11):1737–1749, 2010.
- Berkeley Lab. BuildingsPy - Modelica Buildings Library, 2016. URL <http://simulationresearch.lbl.gov/modelica/buildingspy/>.
- Dassault Systemes. Dymola 2017 FD01, 2016. URL <http://www.dymola.com>.
- John Eddy and Kemper Lewis. Effective Generation of Pareto Sets Using Genetic Programming. In *ASME 2001 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, number 1, pages 1–9, Pittsburgh, PA, 2001.
- Hilding Elmqvist, Hans Olsson, Sven Erik Mattsson, Dag Brück, Christian Schweiger, Dieter Joos, and Martin Otter. Optimization for Design and Parameter Estimation. In *Proc. 4th International Modelica Conference*, 2005.
- ESI ITI GmbH. SimulationX 3.8, 2016. URL <http://www.simulationx.com/>.
- Alan C. Hindmarsh, Peter N. Brown, Keith E. Grant, Steven L. Lee, Radu Serban, Dan E. Shumaker, and Carol S. Woodward. SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers. *ACM Transactions on Mathematical Software*, 31(3):363–396, 2005. ISSN 0098-3500. doi:10.1145/1089014.1089020.
- KDE Community. Breeze icons, 2016. URL <https://github.com/KDE/breeze-icons>.
- MapleSoft. MapleSim 2016, 2016. URL <https://www.maplesoft.com/products/maplesim/>.
- Modelica Association. Functional Mock-up Interface for Model Exchange and Co-Simulation, Version 2.0, 2014a. URL <https://www.fmi-standard.org/downloads>.
- Modelica Association. Modelica Specification, version 3.3 Revision 1, 2014b. URL <http://www.modelica.org/documents>.
- OSMC. OpenModelica 1.9.7, 2016. URL <http://www.openmodelica.org/>.
- Andreas Pfeiffer. Optimization Library for Interactive Multi-Criteria Optimization Tasks. In *Proc. 9th International Modelica Conference*, pages 669–680, Munich, Germany, nov 2012.
- Sandia Corporation. Dakota Packages, 2016. URL <https://dakota.sandia.gov/content/packages>.
- Scilab Enterprises. Scilab: Open Source software for numerical computation, 2015. URL <http://www.scilab.org/>.
- SciPy developers. SciPy.org - Python-based ecosystem of open-source software for mathematics, science, and engineering, 2017. URL <http://scipy.org/>.
- Sergey A. Tachenov. QuaZIP - Qt/C++ wrapper for ZIP/UNZIP package, 2016. URL <http://quazip.sourceforge.net/>.
- The MathWorks Inc. MATLAB R2016b, 2016. URL <http://www.mathworks.es/products/matlab/>.
- The Qt Company. Qt - Cross-platform software development for embedded & desktop, 2016. URL <https://www.qt.io>.
- Hubert Thieriot, Maroun Nemer, Mohsen Torabzadeh-Tari, Peter Fritzson, Rajiv Singh, and John John Kocherry. Towards Design Optimization with OpenModelica Emphasizing Parameter Optimization with Genetic Algorithms. In *Proc. 8th International Modelica Conference*, pages 756–762, 2011.
- Luigi Vanfretti, Maxime Baudette, Achour Amazouz, Tetiana Bogodorova, Tin Rabuzin, Jan Lavenius, and Francisco José Gómez-López. RaPIId: A modular and extensible toolbox for parameter estimation of Modelica and FMI compliant models. *SoftwareX*, 5:144–149, 2016. ISSN 23527110. doi:10.1016/j.softx.2016.07.004.
- Michael Wetter. GenOpt - A Generic Optimization Program. *Seventh International IBPSA Conference*, (1):601–608, 2001.
- Edmund Widl, Wolfgang Muller, Atiyah Elsheikh, Matthias Hortenhuber, and Peter Palensky. The FMI++ library: A high-level utility package for FMI for model exchange. *2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems, MSCPES 2013*, 2013. doi:10.1109/MSCPES.2013.6623316.
- Wolfram. SystemModeler 4.3, 2016. URL <http://www.wolfram.com/system-modeler/>.

Innovations for Future Modelica

Hilding Elmqvist¹ Toivo Henningsson² Martin Otter³

¹Mogram AB, Magle Lilla Kyrkogata 24, 223 51 Lund, Sweden, Hilding.Elmqvist@Mogram.net

²Lund, Sweden, toivo.h.h@gmail.com

³Institute of System Dynamics and Control, DLR, Oberpfaffenhofen, Germany, Martin.Otter@dlr.com

Abstract

This paper discusses language innovations for future Modelica versions, on the one hand for generally applicable language elements, and on the other hand to improve modeling of multibody systems with contacts, and media modeling. In a companion paper new algorithms are proposed to handle much larger models than can be treated today. All these innovations are developed and evaluated with the experimental modeling and simulation environment Modia. Modia is based on Julia, a powerful programming language with strong focus on scientific computing, meta-programming and just-in-time compilation that allows very fast development. The modeling language is directly defined and implemented with Julia's meta-programming constructs and is designed tightly together with the symbolic and numeric algorithms. This approach is very well suited for innovation and experimenting with evolutions of modeling capabilities in Modelica.

Keywords: *Modelica, Modia, Julia, modeling, simulation*

1 Introduction

The objective is developing and testing innovations for future Modelica versions with reasonable effort both from a language point of view as well as for new symbolic and numeric algorithms that are tightly designed together with the language elements. To achieve this goal, an experimental modeling and simulation environment called Modia is under development. Modia uses a Modelica-like language. It shall be both simpler and more powerful than Modelica 3.3 (*Modelica Association, 2014*) and takes into account the experience gained with Modelica in the last 20 years.

New algorithms have been already developed and test-implemented in Modia and are described in the companion paper (*Otter and Elmqvist, 2017*). For example, arrays defined in a model stay as arrays in the generated code, even if (array) equations need to be differentiated. This is a pre-requisite to handle much larger models than what can be treated with current Modelica tools.

In addition to equations, Modelica has a function concept for procedural programming of tasks, such as table

look-up, media calculations and control system implementations. The function part of Modelica is, however, not rich enough. There are no advanced data structures such as union types, no matching construct. Type inference is missing with the implication that there are presently separate blocks for adding Reals, Integers and Complex numbers. The evolution of Modelica has slowed down since it's a too large task to make a full algorithmic language. Instead of inventing all such features, it makes sense to use another language as a base.

Julia (*Bezanson, et al., 2017*) is a very promising language design effort with focus on scientific computing and has many of the properties needed to complement the equational style for modeling. Julia also allows definition of real equations (expression = expression). Furthermore, advanced meta-programming features are available which are suitable for symbolic treatment of equations before just-in-time compilation.

Julia allows developing a modeling language together with a public reference implementation so that language features and symbolic/numeric algorithms are designed tightly together. Native Julia functions are used in models and equations use Julia syntax.

Examples of other research oriented language designs for modeling are: SOL (*Zimmer, 2010*), Hydra (*Giorgidze and Nilsson, 2009*) and Modelyze (*Broman and Siek, 2012*). There is also one experimental simulation package for Julia called Sims (*Short, 2012*). Sims does not make any structural and symbolic processing though, but has event handling. It is based on ideas from Modelyze and Hydra.

This paper introduces major language constructs of Modia and proposes new language features for future Modelica versions. Other aspects of Modia and its implementation are given in (*Elmqvist, et al., 2016*). Modia is available from <https://github.com/ModiaSim>.

2 Modia Language Design

2.1 Model with differential equations

Modia is a domain specific language extension of Julia by means of structured macros, that is, the Julia parser is used to parse Modia models.

A simple first order example model is shown below:

```
@model FirstOrder begin
  x = Variable(start=1)
  T = Parameter(0.5, "Time constant")
  U = 2.0
@equations begin
  T*der(x) + x = u
end
end
```

@model is a call to the Modia macro called **model**. The first part after **begin** is used for variable and component declarations by means of calling constructors. The second part inside the **@equations** macro contains differential and algebraic equations as well as connections. **#** starts a Julia comment. Semicolons can be omitted in Julia.

The constructor **Variable** is used to declare **x** with a start value of 1. In general it constructs instances of ordinary variable types and arrays of those. It is a Julia composite type which in addition to its value also allows specifying type, min, max, variability, start value, info, etc. The constructor **Parameter** is a specialization of the **Variable** constructor which sets the variability to parameter, that is, a quantity that is changeable before simulation starts but constant during simulation. There is also a special short hand notation to define parameters by just giving a default value. This notation is used to define the parameter **u**. The operator **der()** denotes the time derivative of its argument.

The corresponding Modelica model is:

```
model FirstOrder
  Real x(start=1);
  parameter Real T=0.5 "Time constant";
  parameter Real u = 2.0;
equation
  T*der(x) + x = u;
end FirstOrder;
```

Modia uses the Julia way to declare variables with constructor calls. The benefit with respect to current Modelica is a simpler syntax since value, variability, info, etc. are all given in the constructor calls. This allows to easily extending the language with new attributes/properties in the future.

2.2 Coupled models

In order to couple models, the interfaces need to be defined. For simplicity of the language and its implementation, this is currently described as a **@model** (and might be improved in the future by a dedicated **@connector** macro):

```
@model Pin begin
  v = Float()
  i = Float(flow=true)
end
```

Float is a specialization of **Variable** with fixed type **Float64**. The flow variable, **i**, is marked with an attribute **flow=true**. Such a **Pin** can be used to define the terminals **p** and **n** of an electrical resistor:

```
@model Resistor begin
  p = Pin()
  n = Pin()
  v = Float()
  i = Float()
  R = Parameter(info="Resistance")
@equations begin
  v = p.v - n.v # Voltage drop
  0 = p.i + n.i # KCL within component
  i = p.i
  R*i = v # Ohm's law
end
end
```

An electrical component library has been developed containing also Capacitor, Inductor, VoltageSource, etc. A low-pass filter can then be defined as a set of connected components:

```
@model LPfilter begin
  R = Resistor(R=100)
  C = Capacitor(C=0.001)
  V = ConstantVoltage(V=10)
@equations begin
  connect(V.p, R.p)
  connect(R.n, C.p)
  connect(C.n, V.n)
end
end
```

The function **connect** has the same meaning as in Modelica. Note, that no ground component is needed because the missing ground can be automatically handled with a new algorithm described in (Otter and Elmqvist, 2017). Modia is used to evaluate whether this simplification is reliable and transparent for the user. The diagram of a corresponding Modelica model is shown in Figure 1.

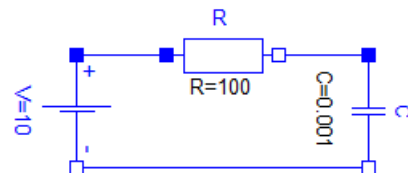


Figure 1. Low pass filter (without ground object)

2.3 Inheritance

There are several electrical components that share the property of having two Pins. Such components are called **OnePorts**. Similarly to Modelica, it is possible to describe the common properties once and inherit them. The common properties are:

```
@model OnePort begin
  p = Pin()
  n = Pin()
  v = Float()
  i = Float()
@equations begin
  v = p.v - n.v # Voltage drop
  0 = p.i + n.i # KCL within component
  i = p.i
end
end
```

The Resistor model can then be simplified:

```
@model Resistor begin
```



```

@extends OnePort()
@inherits i, v
R = Parameter(info="Resistance")
@equations begin
  R*i = v # Ohm's law
end
end

```

The `@extends` macro incorporates all declarations and all equations from `OnePort`. The `OnePort` variables can be accessed by `this.v` and `this.i` in the equations of the `Resistor`. The `@inherits` macro enables to directly use variables `i` and `v`.

2.4 Type and size inference

The Modelica Standard Library (*Modelica Association, 2016*) contains similar models operating on different data types. One example is switches, which based on a Boolean signal select between two Real, two Booleans, or two Complex numbers. There is a desire in the Modelica community to unify this situation by means of type inference.

To experiment with type and size inference, such a feature is included in Modia: Variable constructors do not need to specify type and size. Types and sizes can be inferred from the environment of a model or start values provided, either initial conditions for states or approximate start values for algebraic constraints.

A generic switch that can be applied to matrices and strings as well, can be then defined as:

```

@model Switch begin
  sw = Boolean()
  u1 = Variable()
  u2 = Variable()
  y = Variable()
@equations begin
  y = if sw; u1 else u2 end
end
end

```

2.5 Variable Declarations

There are, however, cases when size and type inference based on start values is not natural, for example, when algebraic equations form a linear system of real simultaneous equations. In such a case, the solution is independent of any start value and only size needs to be given.

It is possible to provide type information in variable declarations using the type parameter `T` in the `Variable` constructor or its short version `Var`:

```
v1 = Var(T=Float64)
```

It is unspecified if the variable `v1` is a scalar or array of `Float64`. It is possible to provide information that a variable is of array type with a certain number of dimensions:

```

array = Var(T=Array{Float64,1})
matrix = Var(T=Array{Float64,2})

```

The size can be fixed using the size attribute:

```

scalar = Var(T=Float64, size=())
array3 = Var(T=Float64, size=(3,))
matrix3x3 = Var(T=Float64, size=(3,3))

```

The size is given with the tuple constructor according to the result of the Julia `size` function. Empty tuple, `()`, means scalar. A vector size is given as a tuple with the size. Such a tuple with one element needs a comma to distinguish it from an expression within parenthesis. When the size attribute is given, `T` denotes the array element type.

There is also a `FixedSizeArrays` module for Julia (*Danish, 2014*) which gives faster code since stack allocation is possible and garbage collection avoided. The corresponding Modia declarations are then:

```

fixedArray3 = Var(T=Vec{3,Float64})
fixedMatrix3x3 = Var(T=Mat{3,3,Float64})

```

SI units can be given using the Julia `SIUnits` module (*Fisher, 2013*). It has predefined types such as: `Meter`, `KiloGram`, `Second`, `Ampere`, `Kelvin`, `Mole`, `Candela`, `Radian`, `Steradian`, `Joule`, `Coulomb`, `Volt`, `Farad`, `Newton`, `Ohm`, `Siemens`, `Hertz`, `Watt`, `Pascal`. A Modia variable with unit `Volt` is declared as:

```
v2 = Var(T=Volt)
```

There is an option in the `SIUnits` module to use units with shorter names (`m`, `kg`, etc) (and `*` is not needed between literal and identifier), for example:

```

m=2.5kg
length=5m

```

However, this feature is not useful since unit `m` would then be in the same name space as the variable `m`. Investigations are being made to allow a local scope for units after literals using a syntax with `[]`:

```

m=2.5[kg]
length=5[m]

```

2.6 Type Declarations

To avoid repeatedly typing type and size information, it's possible to define alternative variable constructors outside the `@model` macro:

```

Float3(; args...) = Var(T=Float64,
  size=(3,); args...)
Voltage(; args...) = Var(T=Volt;
  args...)

```

The notation `"; args..."` denotes a list of keyword arguments which are just passed to the `Variable` constructor using the same notation. This means that a 3-vector with start attribute and a three-phase `Voltage` variable can be declared as:

```

v3 = Float3(start=zeros(3))
v4 = Voltage(size=(3,), start=[220.0,
  220.0, 220.0]Volt)

```


2.7 Redeclaration of submodels

With the "replaceable" language element, Modelica has a powerful concept to exchange submodels on a lower level. However, it is complicated to understand and difficult to implement for tools. Furthermore, it is not powerful enough for certain applications, because redeclarations cannot be controlled by variables and they must be planned in advanced, because only models can be replaced that are marked to be **replaceable**.

A simpler and more powerful concept for redeclarations has been tested in Modia and follows naturally from the constructor style of declaration using expressions, as shown in the following example:

```
MotorModels = [Motor100KW,
               Motor200KW,
               Motor250KW] # Modia models
selectedMotor = motorConfig( ) # Int

@model HybridCar begin
  @extends BaseHybridCar(
    motor = MotorModels[selectedMotor](),
    gear = if gearOption1; Gear1(i=4)
           else Gear2(i=5) end)
end
```

In model `BaseHybridCar` every submodel can be replaced without being marked. In particular new motor and gearbox models are provided. The motor model is selected from an array of Modia models via an integer. The gearbox model is selected based on a logical condition. Such flexible types of redeclarations cannot be formulated in Modelica 3.3.

2.8 Multi-mode Modeling

Several attempts have been made to generalize the semantics of Modelica to allow mode changes, for example (Mattsson, et al., 2015). However, only a limited classes of problems could be handled. One reason is the imposed restriction that the equations are only processed once, code is generated and this code should hold for all mode changes. There are academic simulation prototypes that dynamically process and switch equations during run-time, such as (Zimmer, 2010; Höger, 2014). The question is how to incorporate such ideas in to Modelica and Modelica tools with the goal to solve real-world industrial problems.

First investigations have been carried out in Modia to experiment with changing model structure. Consider the model of an electrical motor with a load in Figure 2. The shaft between motor and load breaks at a certain time.

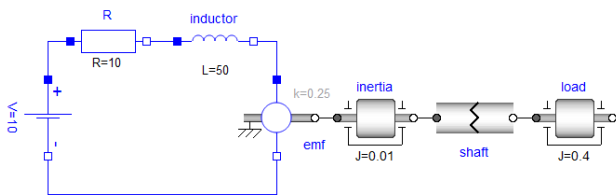


Figure 2. Electrical motor, load and breaking shaft.

The breaking shaft can be modelled as follows using conditional equations:

```
@model BreakingShaft begin
  flange1 = Flange()
  flange2 = Flange()
  broken = Boolean()
@equations begin
  if broken
    flange1.tau = 0
    flange2.tau = 0
  else
    flange1.w = flange2.w
    flange1.tau + flange2.tau = 0
  end
end
end
```

Figure 3 shows the angular speeds of the two inertias when the shaft breaks at time = 100.

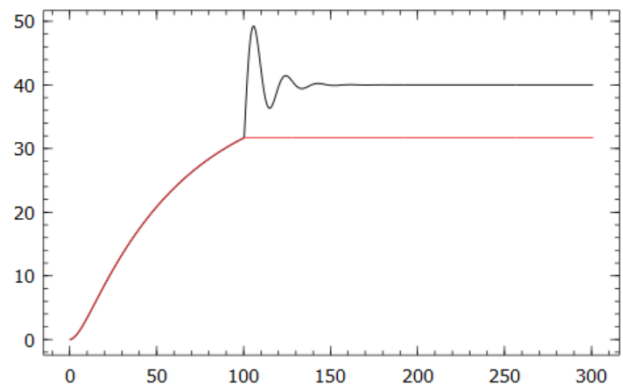


Figure 3. Angular speeds of inertias

The set of model equations and the DAE index is changing when the shaft breaks. The Modia environment makes new symbolic transformations and just-in-time compilation for each mode of the system. The final results of variables before an event is used as initial conditions after the event.

Mode changes with conditional equations might introduces inconsistent initial conditions causing Dirac impulses to occur. This more general problem is treated in (Benveniste, et al., 2017).

2.9 Other features

Other features, such as type and size inference, time events, synchronous controllers, state events, multi domain models are exemplified in (Elmqvist, et al., 2016). There are also ongoing development and experimentation regarding nested simulations, etc.

3 Model Examples

3.1 Multibody modeling

Multibody models uses vector and matrix equations. Below, a tiny multibody library is defined with similarities to package `MultiBody` of the Modelica Standard Library

(Modelica Association, 2016). First, convenient Variable constructors involving SIUnits are defined.

```
Position(; args...) =
  Var(T=Meter; size=(), args...)
Velocity(; args...) =
  Var(T=Meter/Second; size=(), args...)
Acceleration(; args...) =
  Var(T=Meter/Second^2; size=(), args...)
Angle(; args...) =
  Var(T=Radian; size=(), args...)
AngularVelocity(; args...) =
  Var(T=Radian/Second; size=(), args...)
AngularAcceleration(; args...) =
  Var(T=Radian/Second^2; size=(),
    args...)
Force(; args...) =
  Var(T=Newton; size=(), args...)
Torque(; args...) =
  Var(T=Newton*Meter; size=(), args...)
Mass(; args...) =
  Var(T=KiloGram; size=(), min=0,
    args...)
```

Based on these scalar Variable constructors, vector and matrix constructors can be defined.

```
Axis3(; args...) =
  Var(T=SIPrefix; size=(3,), args...)
Position3(; args...) =
  Position(size=(3,); args...)
Velocity3(; args...) =
  Velocity(size=(3,); args...)
Acceleration3(; args...) =
  Acceleration(size=(3,); args...)
Rotation3(; args...) =
  Var(T=SIPrefix; size=(3,3),
    property=rotationGroup3D, args...)
AngularVelocity3(; args...) =
  AngularVelocity(size=(3,); args...)
AngularAcceleration3(; args...) =
  AngularAcceleration(size=(3,); args...)
Force3(; args...) =
  Force(size=(3,); args...)
Torque3(; args...) =
  Torque(size=(3,); args...)
Inertia3(; args...) =
  Var(T=KiloGram*Meter*Meter, size=(3,3);
    property=symmetric, args...)
```

It should be noted that Rotation3, the type for rotation matrices, has a special property:

```
property=rotationGroup3D
```

In particular this means that the element declared in this way is a 3x3 rotation matrix that has 9 elements with 6 implicit constraints between them. In case kinematic loops are present, this property of rotation matrices would lead to redundant constraint equations that are difficult to handle. As discussed in (Elmqvist and Mattsson, 2016), a tool can, however, automatically remove this redundancy of a kinematic loop in a pre-processing step, provided the rotation matrices are marked, as done above. Compared to current Modelica, the benefit is that no special operators Connections.branch/.root/.isRoot

etc are needed anymore. Note, these operators are awkward, difficult to understand and it is easy to make mistakes.

Other properties can be defined as well. In particular, the Inertia3 constructor specifies the matrix to be symmetric. This can enable better user interface for setting parameters.

The coupling semantics is defined by Frames.

```
@model Frame begin
  r_0 = Position3()
  R = Rotation3()
  f = Force3(flow=true)
  t = Torque3(flow=true)
end
```

A Prismatic joint has two Frames. Axis of translation is given by a vector parameter n.

```
@model Prismatic begin
  n = Axis3(value=[1,0,0],
    variability=parameter)

  frame_a = Frame()
  frame_b = Frame()

  s = Position(start=0*Meter)
  v = Velocity(start=0*Meter/Second)
  a = Acceleration()
  f = Force()
@equations begin
  v = der(s)
  a = der(v)

  frame_b.r_0 = frame_a.r_0 +
    frame_a.R'*(n*s)
  frame_b.R = frame_a.R
  frame_a.f = -frame_b.f
  frame_a.t + frame_b.t =
    cross(n*s, frame_b.f)

  # d'Alemberts principle
  f = -dot(n, frame_b.f)
  f = 0*Newton # Not driven
end
end
```

A Revolute joint has similar structure.

```
@model Revolute begin
  n = Axis3(value=[0,1,0],
    variability=parameter)

  frame_a = Frame()
  frame_b = Frame()

  phi = Angle(start=0)
  w = AngularVelocity(start=0)
  a = AngularAcceleration()
  tau = Torque()
  R_rel = Rotation3()
@equations begin
  R_rel = n*n' + (eye(3) - n*n')*cos(phi)
    - skew(n)*sin(phi)

  w = der(phi)
  a = der(w)

  frame_b.r_0 = frame_a.r_0
```

```

frame_b.R = R_rel*frame_a.R
frame_a.f = -R_rel'*frame_b.f
frame_a.t = - R_rel'*frame_b.t

# d'Alemberts principle
tau = -dot(n, frame_b.t)
tau = 0*Newton*Meter # Not driven
end
end

```

The skew function is defined as:

```

skew(x) = [ 0 -x[3] x[2];
           x[3] 0 -x[1];
          -x[2] x[1] 0]

```

Gravity is defined by the following function:

```

gravityAcceleration(r) =
  9.81*[0,-1,0]*Meter/Second^2

```

A Body has one Frame. The parameter r_CM gives the vector from the frame to center of mass.

```

@model Body begin
  r_CM = Position3(variability=parameter)
  m = Mass(variability=parameter)
  I = Inertia3(variability=parameter)

  frame = Frame()

  r_0 = Position3()
  R = Rotation3()
  v_0 = Velocity3()
  a_0 = Acceleration3()
  w_a = AngularVelocity3()
  z_a = AngularAcceleration3()
  g_0 = Acceleration3()
  W = Var(T=Float64, size=(3,3))
@equations begin
  r_0 = frame.r_0
  R = frame.R
  g_0 = gravityAcceleration(r_0 + R'*r_CM)

  # Translational kinematic differential
  # equations
  v_0 = der(r_0)
  a_0 = der(v_0)

  # Rotational kinematic differential
  # equations
  W = der(R)*transpose(R)
  w_a = [W[3,2], W[1,3], W[2,1]]
  z_a = der(w_a)

  # Newton/Euler equations
  frame.f = m*(R*(a_0 - g_0) +
    cross(z_a, r_CM) + cross(w_a,
    cross(w_a, r_CM)))
  frame.t = I*z_a + (cross(w_a, I*w_a) +
    cross(r_CM, frame.f))
end
end

```

The coordinate systems must be fixed for multibody dynamics. This is done by using a World object:

```

@model World begin
  frame = Frame()
@equations begin
  frame.r_0 = zeros(3)*Meter
  frame.R = eye(3,3)

```

```

end
end

```

A simple sliding mass model is shown below:

```

@model TranslationalBody begin
  world = World()
  j = Prismatic(n=[1,1,1]/sqrt(3),
    v = Velocity(start=1*Meter/Second))
  body = Body(r_CM=[0.5,0,0]*Meter,
    m=1.0*KiloGram,
    I=1e-3*eye(3)*KiloGram*Meter^2 )
@equations begin
  connect(world.frame, j.frame_a)
  connect(j.frame_b, body.frame)
end
end

```

3.2 Functions and data structures

One of the reasons for developing Modia on top of Julia is to have direct access to Julia algorithmic features, i.e. much more powerful functions and data structures than available in current Modelica.

One of the limitations of current Modelica is a convenient way of handling collisions of many objects for DEM (Discrete Element Modeling). The problem is that there are $n*(n-1)/2$ potential contacts possible for n objects. The user can of course not explicitly make these connections.

One approach is that each object registers its position. After that, the forces between each pair of objects in contact are calculated. Then each object retrieves the sum of the forces acting on the object. This force is used in the equations of motion. In (*Elmqvist et al., 2015*), the information about each object and the above calculations are handled in C/C++. A problem is that there is no convenient method in current Modelica to make sure all objects have registered their position before forces are extracted. An elaborate scheme involving inner/outer construct together with flow variables was used.

An experimental feature has been included in Modia to solve this problem. The built-in operator `allInstances(v)` creates a vector of all the variables v within all instances of the class where v is declared. It can be seen as a specialization of the proposed Modelica array constructor: `[c.v for c in class Class]`, (*Elmqvist, et al., 2015b*). This construct did not make it into Modelica 3.4 due to concerns about self-reference and mutual recursive loops. The `allInstances` operator is referring to the class where it's used but has a more restricted semantics.

Consider modeling a set of spherical balls moving on a plane. We will assume the same radius for simplicity and a force law of a spring-damper during contact. A Modia model is shown below.

```

@model Ball begin
  r = Var()
  v = Var()
  f = Var()

```

```

m = 1.0
@equations begin
  der(r) = v
  m*der(v) = f
  f = getForce(r, v, allInstances(r),
    allInstances(v), (r,v) -> (k*r + d*v))
end
end

```

The force is dependent on the position and velocity of all Balls, that is, the `allInstances` operator is used on both `r` and `v`. The force law is provided as an anonymous function: `(r,v) -> (k*r + d*v)`.

A set of Balls can easily be modelled by just instantiation. The contact handling is automatic:

```

@model Balls begin
  b1 = Ball(r = Var(start=[0.0,2]),
    v = Var(start=[1,0]))
  b2 = Ball(r = Var(start=[0.5,2]),
    v = Var(start=[-1,0]))
  b3 = Ball(r = Var(start=[1.0,2]),
    v = Var(start=[0,0]))
end

```

In this case with three balls, the operator `allInstances(r)` expands to `[b1.r, b2.r, b3.r]`.

The force contributions from all other balls are calculated according to the spring-damper model by function `getForce`:

```

const k=10000
const d=100
const radius=0.05

function getForce(r, v, positions,
  velocities, contactLaw)
  force = zeros(2)
  for i in 1:length(positions)
    pos = positions[i]
    vel = velocities[i]
    if r != pos
      delta = r - pos
      deltaV = v - vel
      f = if norm(delta) < 2*radius;
        -contactLaw((norm(delta)-
          2*radius)*delta/norm(delta),
          deltaV) else
        zeros(2) end
      force += f
    end
  end
  return force
end

```

The described technique opens up the possibility for further important optimizations. In order to avoid $O(n^2)$ complexity when deciding which objects that are in contact, space partitioning by quad-trees or oct-trees can be used, see (*Elmqvist et al., 2015*). This requires recursive data structures that are available in Julia.

3.3 Media Modelling

The Modelica.Media library within the Modelica Standard Library¹ provides a large set of packages and functions to compute media properties of one and two-phase media dedicated for simulation. Although the Media library is powerful, it has conceptual limitations for the modeling of media with multiple substances that have multiple phases. Furthermore, the details of the library are difficult to understand and difficult to support by Modelica tools due to the extensive use of replaceable packages and functions. There have been several attempts to simplify the approach and making media modeling more powerful.

Julia allows a fresh view on this difficult topic and it seems that *multiple dispatch* and other Julia features allow a surprisingly simple way to model complex media: Following the Modelica.Media library design, a medium has the following orthogonal properties:

1. *Medium states* that define the independent variables of the medium. A medium may have different types of independent variables. For example, it might have as independent variables pressure `p` and temperature `T` or pressure `p` and specific enthalpy `h`. In Julia they would be described as types.
2. *Medium constant data* that defines constants for every instance of a specific medium. For example a simple medium may have a constant `d_const` for the mean density. In Julia constant data would be described as constants in a module.
3. *Medium immutable data* that defines constants specific to an instance of a specific medium that cannot be changed once the medium is instantiated. Typically reference points such as `h_offset` may have a default value, but might be changed for particular medium instances. In Julia such data would be described as immutable types.
4. *Medium functions* that define properties of a medium as function of the medium constant and immutable data and the medium states. For example `density(medium, state)` computes the density for a medium using the given state description.

Below is a sketch of a new Media library design:

```

module Media # Interface of media models
  # Possible medium states
  type State_pT
    p::Float64
    T::Float64
  end

  type State_ph
    p::Float64
    h::Float64
  end

  # Possible medium functions
  density(medium, state)=error(..)
end

```

¹ <http://doc.modelica.org/om/Modelica.Media.html>

```

specificEnthalpy(medium,state)=error(...)
setState_pT(medium,p,T)=error(...)
setState_ph(medium,p,h)=error(...)
...
end

```

In a generic module *Media*, the supported medium states and the supported medium functions are collected. The default implementation of the functions for every medium are error messages. However, also concrete functions could be added here that hold for every medium.

A specific medium is implemented with a Julia module, as shown here for a simple water model:

```

module SimpleWater
import Media

# Constants of medium
const cp_const = 4184.0
const cv_const = 4184.0
const d_const = 995.586
const T0 = 273.15

# Variables specific to an instance
immutable Medium
h_offset::Float64
Medium(;h_offset=0.0) = new(h_offset)
end

# Functions of medium
Media.density(
m::Medium,
state::Media.State_pT) = d_const
Media.specificEnthalpy(
m::Medium,
state::Media.State_pT) =
cp_const*(state.T - T0) + m.h_offset
Media.setState_pT(m::Medium, p, T) =
Media.State_pT(p,T)
Media.setState_ph(m::Medium, p, h) =
Media.State_pT(p,
T0+(h-m.h_offset)/cp_const)
...
end

```

In a Modia model Julia data structures and functions can be used. As a result, it is possible to instantiate a medium model at some place with

```

medium1=SimpleWater.Medium()
medium2=SimpleWater.Medium(h_offset=10.0)

```

and then propagate this medium through all connected fluid component models:

```

@model FluidPort begin
# contains medium, p, h, ...
end
...
port = FluidPort()
...
port.medium = SimpleWater.Medium()

```

Inside a component model, medium properties are computed. The implementation of such a component model neither knows which concrete medium model is used, nor which independent states the medium has, so the

component model can be used for all media that provide an implementation of the used functions:

```

state = setState_ph(port.medium,
port.p,
port.h)
d = density(medium,state)
h = specificEnthalpy(medium,state)

```

Julia selects the concrete functions to be called based on the *medium* type and the *state* type. This is the key innovation that makes media modeling suddenly so simple: a function is (statically) selected based on the types of *several* arguments.

4 Implementation

The Modia implementation is made in Julia which provides meta-programming capabilities which are suitable for symbolic treatment of the equations.

4.1 Meta-programming in Julia

Languages such as Modelica and Modia require symbolic transformations of equations into executable code. A mathematical expression is conveniently represented by an AST (abstract syntax tree). The Julia language (Bezanson, et al., 2017) allows creation of “quoted” expressions encapsulated as, “: (...)”.

```

julia> equ = :(0 = x + 2y)
:(0 = x + 2y)

```

Such an expression is stored as an AST. The AST can be shown by using a built-in function, `dump()`:

```

julia> dump(equ)
Expr
head: Symbol =
args: Array{Any, 2}
 1: Int64 0
 2: Expr
   head: Symbol call
   args: Array{Any, 3}
    1: Symbol +
    2: Symbol x
    3: Expr
       head: Symbol call
       args: Array{Any, 3}
        typ: Any
        typ: Any
        typ: Any

```

`equ` is of type `Expr` which has three fields: `head`, `args` and `typ`. `equ.head` is the `Symbol =` representing the equality of the two expressions of the equation. The right hand side is the sum of two expressions: `x` and `2y`. The operator `+` is represented as a function call: `equ.args[2].head`. Which function to call is defined in `equ.args[2].args[1]`. The operands of the `+` operator are `equ.args[2].args[2]` and `equ.args[2].args[3]`.

A new AST can be built using the `Expr` constructor. For example, solving an equation of the form:

```

0 = x + expression

```

can be done as follows:

```
julia> solved = Expr(:(=),
    equ.args[2].args[2], Expr(:call, :-,
    equ.args[2].args[3]))
:(x = -(2y))
```

It is also possible to create a quoted expression referring to parts of `equ` by the use of “interpolation”, `$ ()`.

```
julia> solved = :($ (equ.args[2].args[2]) =
    - $ (equ.args[2].args[3]))
:(x = -(2y))
```

The result is presented as a quoted expression. By assigning the variable `y`, it's possible to calculate `x` using the `eval` function on the AST `solved`:

```
julia> y = 10
10
julia> eval(solved)
-20
julia> @show x
x = -20
```

4.2 Symbolic Transformations of Modia Models

The following list shows some of the structural and symbolic transformations which are performed by the Modia implementation:

- Instantiation
- Flattening
- Alias elimination
- Type and size inference
- Removal of singularities
- Index reduction and BLT of array equations
- Symbolic differentiation of matrix equations
- Symbolic solution of matrix equations
- Partial state selection and tearing
- Transformation to a special index one DAE
- Determining sparseness structure of Jacobian

Modia supports *type and size inference*, that is, the Variable constructor does not need to specify type and size. However, Pantelides algorithm and removal of singularities require that types and sizes of variables and equations are known. Types and sizes are inferred from the start values provided and by propagation. The left and right hand sides of equations are evaluated with given start values and the type and size inference of Julia is used to determine the size and types of variables and equations.

There are useful application models where structural symbolic algorithms fail and may lead to strange error messages during symbolic processing or to run-time errors. For example, if an electrical circuit is not grounded, the potentials of the electrical Pins can float, that is, the system equations are underdetermined. On the other hand, the equations are overdetermined regarding currents. Such *singularities* needs to be removed before further structural processing. Details of such a technique is described in the companion paper (Otter and Elmqvist, 2017).

The Pantelides algorithm and other structural index reduction algorithms are designed for scalar variables and equations. So Modelica tools typically symbolically expand array equations into a set of scalar equations involving the variable elements. This is not feasible if large array equations are used, for example, for flexible bodies or other discretized partial differential equations. Generalizations of BLT and Pantelides algorithms to directly handle *array equations* can be found in (Otter and Elmqvist, 2017).

Pantelides algorithm determines which array equations that needs to be differentiated. Special care are needed when performing *symbolic operations on array and matrix equations* since matrix multiplication is not commutative. Solving for unknowns are done by a set of rewrite rules. As an example, the right division operator, `/`, or the left division operator, `\`, is used depending on whether the unknown is on the right or left side of a multiplication operator. Special rules can be used for rotation matrices to replace division by multiplication with the transpose of the rotation matrix.

4.3 Numeric Solution of Modia Models

Numeric treatment and transformation of the resulting differential algebraic array equations to index one form is described in the companion paper (Otter and Elmqvist, 2017).

5 Outlook

The Modia experimental language gives new possibilities for creation of new innovative language elements and algorithms to model and simulate more complex models than is possible in current Modelica.

The suggested innovations of the companion paper (Otter and Elmqvist, 2017) can be directly utilized in current Modelica tools. A change in the Modelica language is not needed for them. Part of the proposed innovations in this paper for new language elements, such as type inference, marking of rotational matrices in combination with new algorithms, or the `allInstances(..)` operator, could be included in a fully backwards compatible form in a future Modelica 3.x version.

The use of native Julia for the algorithmic part would simplify the Modelica effort considerably since Modelica does not need to be extended with new features in functions. This means that evolution of Modelica could be focused on the equational modeling aspects.

Contributions to Modia for language design and for improved symbolic and numeric algorithms are welcome.

References

- Benveniste A., Caillaud B., Elmqvist H., Ghorbal K., Otter M., and Pouzet M. (2017): *Multi-Mode DAE Models - Challenges, Theory and Implementation*. Lecture Notes on Computer Science, submitted for review.

- Bezanson J., Edelman A., Karpinski S. and Shah V.B. (2017): *Julia: A Fresh Approach to Numerical Computing*. SIAM Review, Vol. 59, No. 1, pp. 65-98. <http://julialang.org/publications/julia-fresh-approach-BEKS.pdf>; see also: <http://julialang.org/>
- Broman D., Siek J. G. (2012): *Modelyze: a Gradually Typed Host Language for Embedding Equation-Based Modeling Languages*, University of California at Berkeley, No. UCB/EECS-2012-173, www2.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-173.html.
- Danish D. (2014): *FixedSizeArrays*, <https://github.com/SimonDanisch/FixedSizeArrays.jl>
- Elmqvist H., Goteman A., Roxling V., Ghandriz T. (2015): *Generic Modelica Framework for MultiBody Contacts and Discrete Element Method*. Proceedings 11th International Modelica Conference, Versailles. <http://www.ep.liu.se/ecp/118/046/ecp15118427.pdf>
- Elmqvist H., Olsson H., Otter M. (2015b): *Constructs for Meta Properties Modeling in Modelica*. Proceedings 11th International Modelica Conference, Versailles. <http://www.ep.liu.se/ecp/118/026/ecp15118245.pdf>
- Elmqvist H. and Mattsson S.E. (2016): *Exploiting Model Graph Analysis for Simplified Modeling and Improved Diagnostics*. Proceedings EOOLT '16, April 18, Milano, Italy.
- Elmqvist J., Henningsson T. and Otter M. (2016): *System Modeling and Programming in a Unified Environment based on Julia*. Proceedings of ISO/LA 2016 Conference Oct. 10-14, T. Margaria and B. Steffen (Eds.), Part II, LNCS 9953, pp. 198-217.
- Fisher K. (2013): *SIUnits*. <https://github.com/Keno/SIUnits.jl>
- Giorgidze G., Nilsson H. (2009): *Higher-Order Non-Causal Modelling and Simulation of Structurally Dynamic Systems*. In Proceedings of the 7th International Modelica Conference, pages 208–218, Como, Italy. <http://www.ep.liu.se/ecp/043/022/ecp09430137.pdf>.
- Höger C.: *Dynamic structural analysis for DAEs*. In Proceedings of the 2014 SCS Summer Simulation Multiconference, 2014.
- Mattsson S.E., Otter M., and Elmqvist H. (2015): *Multi-Mode DAE Systems with Varying Index*. Proceedings 11th International Modelica Conference, Versailles. <http://www.ep.liu.se/ecp/118/009/ecp1511889.pdf>
- Modelica Association (2014): *The Modelica Language Specification, Version 3.3 Revision 1*, <https://www.modelica.org/documents/ModelicaSpec33Revision1.pdf>
- Modelica Association (2016): *The Modelica Standard Library, Version 3.3.2*, <https://github.com/modelica/Modelica>
- Otter M., and Elmqvist H. (2017): *Transformation of Differential Algebraic Array Equations to Index One Form*. Modelica Conference 2017, Prague, May 15-17.
- Short T. (2012): *Sims - A Julia package for equation-based modeling and simulations*. <https://github.com/tshort/Sims.jl>.
- Zimmer D. (2010): *Equation-Based Modeling of Variable Structure Systems*. PhD Dissertation, ETH Zürich. <http://e-collection.library.ethz.ch/eserv/eth:1512/eth-1512-02.pdf>.

Hierarchical Semantics of Modelica

Christoph Höger¹

¹Institute of Software Engineering and Theoretical Computer Science, Technische Universität Berlin, Germany
christoph.hoeger@tu-berlin.de

Abstract

We present a definition of syntax and semantics for Modelica's hierarchical lookup. By using a context-independent encoding of the static semantics of free variables, it becomes possible to define the evaluation of references within a calculus based on substitution. Hence, all steps of evaluation have a concrete syntactic representation. We augment the calculus with a terminating evaluation and a semantics-preserving translation to a basic λ -calculus.

Keywords: Semantics, Classes, Compilation

1 Introduction

In current Modelica, there is no way to express the definition of a variable as a purely syntactic property, independent of the context in which it might be used. Its definition is obtained as part of the dynamic semantics of the flattening process. This effectively renders static analysis of models and packages impossible. Furthermore, there is no formal method to obtain its meaning from the a found definition in the context of a simulation model, as the dynamic semantics of hierarchical elements are defined only informally.

This paper attempts to improve this situation by the means of a compositional core calculus of classes, MCL. In this language, we define syntactic elements for the expression of static properties of variables in a class. The semantics of Modelica-style hierarchical classes is integrated within the framework of the classic λ -calculus. This integration is inspired by the treatment of modules by Pierce (2005). For the evaluation, we focus solely on the problems mentioned above. For a discussion of the relation between model elaboration and the λ -calculus, we refer to earlier work (Höger 2016). In a final step, we present a translation that replaces the hierarchical elements with semantically identical non-hierarchical terms. This shows how a hierarchical model can be translated into a simpler functional language.

The rest of this paper is organized as follows: An introduction into Modelica's scoping and hierarchical organization leads to the definition of the hierarchical core calculus of MCL. This is followed by a graphical interpretation of the hierarchical environment and consequently its semantics. The paper concludes with a *transformation* of the hierarchical aspects to more basic elements of the language. This transformation is shown to be faithful in the sense that it preserves the evaluation semantics.

2 Modelica Scoping and Hierarchies

In its simplest form, a Modelica class serves as a container for a sequence of *declarations*. These may introduce constants, parameters, unknowns or declare components that are instances of other classes. The meaning of variables in the right-hand sides of these declarations is somewhat intricate as the example in Listing 1 shows.

The declaration of the constant x in class A refers to two free variables, y and z . Class A is a child of class B , which is in turn a child of C in the *class-hierarchy*. Hence it “sees” all declarations¹ of its parent classes. In the classical sense, B is part of A 's lexical scope. Therefore, z is found directly in the surrounding scope. Note that the definition of constant z (the literal 21) is syntactically placed *after* A . The scope of a binding is independent from the order of declarations. Variable y is not defined inside B . The next candidate is C , where it is defined as `modelicaB.z`.

Such a composite name gives access to elements *downwards* the hierarchy. In a first step, B is found as before in the scope of C . The result of this search is then used to search for z , which is defined as 21. Hence the result of evaluating $C.B.A.x$ should yield 42.

Although this kind of scoping might seem pretty standard, there is a subtle difficulty embedded in this seemingly simple principle. In Modelica, there is no (syntactic) difference between looking up a class (e.g. B) and its fields (e.g. z). What might seem like an elegant unification, turns out to be a source of major complication in combination with inheritance.

2.1 Inheritance and Modifications

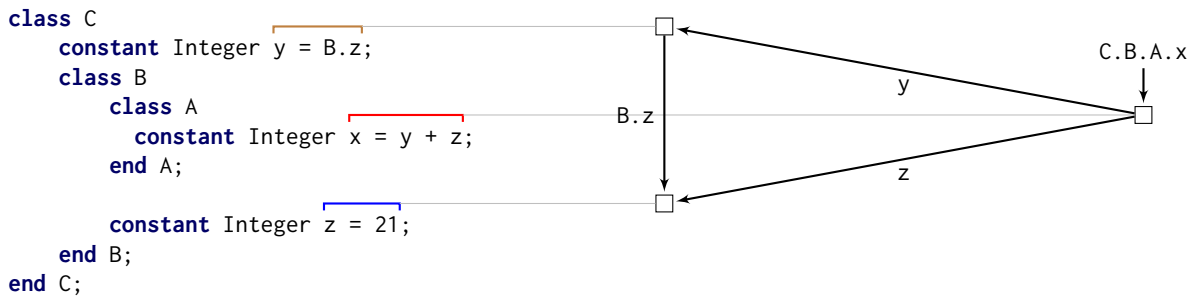
Lexical scoping as it is used above is still a pretty straightforward matter: After all, the environment in which to look for the definition of a variable is determined by the syntactical composition of classes. The complexity rises drastically however, once inheritance (expressed as **extends** statements) comes into play:

```
class D
  extends C.B(z=2);
  constant Integer x = A.x;
end D;

class C ... end C;
```

¹At least the ones with the proper variability

Listing 1. Hierarchical Lookup



In the class D above, with C left unchanged, what is the value of D.x? Since A is inherited in D from C.B, it is tempting to assume the answer is, again, 42. Instead, the returned value is 23^2 .

The reason for this is shown in Listing 2. In the first step, A is found to be inherited from the base class C.B. This lookup succeeds immediately without further involvement of inheritance. Hence, A.x is resolved by looking for x in B. This class contains the same definition of x as before. Accordingly z and y need to be looked up again. Variable z is again looked up in its immediately enclosing scope. This time, this scope is not provided by A, but by the inheriting class D. Therefore the resulting value is 2.

In an interesting twist, y is *not* subject to this modification. Since its lookup passes through C and only then returns to the definition of z the inheritance is discarded. The resulting value is therefore found in the lexical scope of A, and hence yields 21. The overall evaluation yields 23.

This example demonstrates an important fact about Modelica-classes. The site of the definition of a free variable is not a syntactic property of the class. Instead, it depends on the *context* in which this class is used.

2.2 The Principle of Open Recursion

This context is the result of evaluating all relevant super classes. Therefore, the definition of lookup has to be part of the evaluation of classes and vice versa. In Modelica there is no explicit ordering between declarations. Due to the existence of inheritance and because classes are looked up in the same way as other declarations, such an ordering cannot be found without knowledge of the context of the class. Both the construction of the context and the evaluation of class references are recursively linked.

In classical object-oriented languages, this principle is called *open recursion* (Aldrich and Donnelly 2004): Each method has access to a special variable (often called *this* or *self*). Methods are always invoked from a concrete object (sometimes called the receiver of a message). This object then becomes the definition of the special variable during evaluation of the method's body (the special variable is *late bound*). Free variables in the method are interpreted by method invocation on the special variable. This

principle yields an implementation of *recursion*, since the method itself is an element of the receiving object. It is *open*, since the method might be part of different concrete objects (and invoke different siblings on each). Hence, it is possible to change the behavior of all methods of an object by exchanging only one method. The same concept can be used to explain the lookup inside Modelica's classes, when it is applied not only to one, but possibly many special variables.

```

class D
  constant Integer z = 2;
  extends up(1).C.B;
  constant Integer x = this.A.x;
end D;

class C
  constant Integer y = this.B.z;
  class B
    class A
      constant Integer x = up(2).y + up(1).z;
    end A;
  constant Integer z = 21;
end B;
end C;
        
```

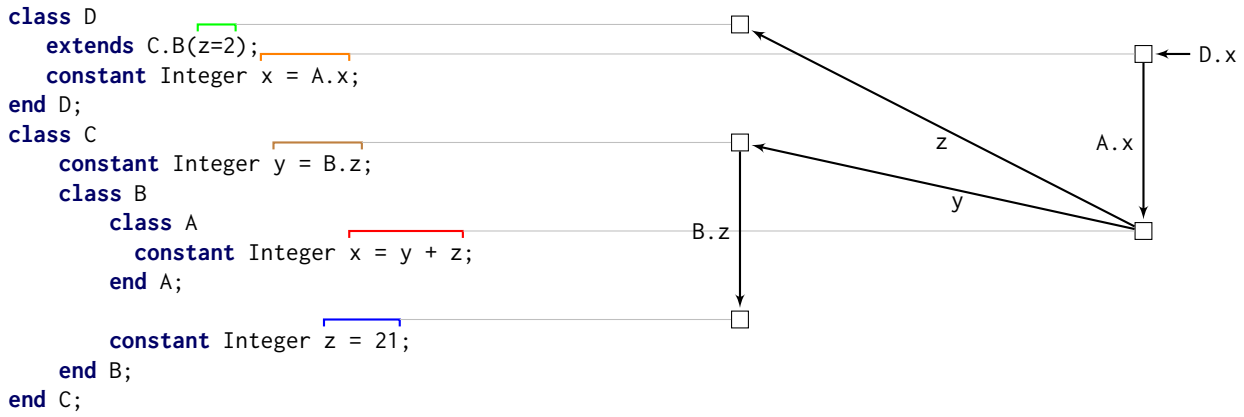
In the listing above, references to the context have been codified by two kinds of special variables: *this* denotes a reference to the immediately enclosing class, while *up(i)* expresses access to the *i*-th enclosing class (hence *up(0)* is the same as *this*, but less readable). The benefit of such a form lies in the fact that it eliminates any free variables and still allows to use the class in different contexts.

3 MCL

In order to focus the discussion on the hierarchical semantics by the means of such special variables, it is useful to define a minimal calculus and ignore the any feature of Modelica that does not directly contribute to the discussion. To this end we define MCL, a small core calculus that embeds hierarchical term into the classical minimal λ -calculus. Besides the more concise representation, such a reduction allows to express the *complete* domain of discourse. In the following sections, all relevant elements can be expressed in the form of expressions in the core

²as discussed in <https://trac.modelica.org/Modelica/ticket/2013>

Listing 2. Hierarchical Lookup with Inheritance



language. There is no need to resort to externally (and imprecisely) defined entities like tables, environments or universes of classes.

3.1 Notational Conventions

Languages are defined in a simple BNF-form: Nonterminals (e.g. t , v are expressed with the same small italic letters as meta-variables of the corresponding syntactic sort (e.g. we will use t to denote both the set of terms and a variable from that set). Productions (e.g. $t ::= \lambda x. t \mid x$) map a nonterminal (to the left of the $::=$) to clauses consisting of nonterminal and terminal symbols. Clauses are separated by a \mid . Each clause is one possible derivation of the left hand side. Terminal symbols (e.g. $,$, true , if) are written in a non-proportional font.

Partial functions are univalent relations $\{x \mapsto y\}$. These relations can be augmented using the \oplus -operator, borrowed from the specification language Z:

$$p \oplus q \triangleq \{x \mapsto y \mid x \mapsto y \in p \text{ and } x \notin \text{dom}(q)\} \cup q$$

$\llbracket t \rrbracket_\Delta$ denotes the function Δ applied to t . In order to enhance readability, these (recursive) functions are defined using pattern matching on their arguments: $\llbracket t_1 \ t_2 \rrbracket_\Delta$ means the application of Δ to *one* term formed by the juxtaposition of two (possible distinct) terms (i.e. the term representing the application of t_1 to t_2). If multiple arguments are passed to a semantic function, they are separated by commas. Meta variables are bound in the patterns or corresponding *where*-clauses. When necessary, we consider each function as *overloaded* on different syntactic sorts, e.g the function Π can be applied to recursive definitions \mathcal{F} as well as the fields of a class \bar{F} .

Sequences are abbreviated by an overline over the name of the contained meta variables, e.g. \bar{t} describes a sequence $t_1 \dots t_n$. The empty sequence is \diamond . Non-empty sequences are written as pairs of a value and the remaining sequence, separated by a double colon, e.g. $s :: \bar{t}$ describes the sequence s, t_1, \dots, t_n . The operator \times maps a semantic function on a sequence, e.g. $f \times \bar{F}$ yields a sequence where

every element is the result of applying f to the corresponding element in \bar{F} .

In order to not confuse meta-level equality (e.g. of terms) with its object-level counterpart (e.g. in an equation), we write $a \stackrel{\wedge}{=} b$ to indicate the former (and $a \neq b$ for the opposite).

3.2 Syntax

The syntax of MCL distinguishes between hierarchical terms h and proper terms t (Figure 1). There are five variants of hierarchical terms: The special variables $\text{up}(i)$ refer to the i -th enclosing class. Literal classes C are a list of fields F bracketed in special curly braces, $\{|\bar{F}|\}$. A class field can either contain a hierarchical class (e.g. a child class) ($L = h$) or a value ($l = t$). We presume that each class-label L can be distinguished from each label l : $L \cap l \stackrel{\wedge}{=} \emptyset$. A hierarchical node v denotes a class containing the fields \bar{F} as a hierarchical child of the enclosing class denoted by $\bar{\pi}$, written $\{|\bar{F} \text{ in } \bar{\pi}|\}$ (the environment is thus encoded as a list of nodes and each node contains its own environment). Explicit Modifications $\{|\bar{h} \text{ with } \bar{F}|\}$ override the fields defined in the class described by h term with the fields in \bar{F} .

Access to the field L of a class requires an explicit notion of the corresponding super class in a reference R : $h_1 . \text{super}(h_2) . L$. Here, h_1 refers to a class extending h_2 which in turn describes the definition-site of the declaration labeled with l . The access reads as: “Get the field labeled with L in the class h_2 extended by h_1 ”. This makes the interface of a class immediately visible (since all inherited fields have to be defined locally as forward references) and is a necessary precondition for a substitution-based semantics. If no super class shall be referenced directly, both parts of a reference are equal. Since this is a common case, we introduce the abbreviation $h . L \stackrel{\wedge}{=} h . \text{super}(h) . L$ to enhance the readability.

Terms t consist of the standard elements of the λ -calculus extended with non-strict conditional, and an explicit fixed point operator fix (which ranges over multiple,

Hierarchical Terms:

$$\begin{aligned}
 h &::= \text{up}(i) \mid C \mid v \\
 &\quad \mid \{ |h \text{ with } \overline{F}| \} \mid R \\
 R &::= h.\text{super}(h).L \\
 v, \pi &::= \{ | \overline{F} \text{ in } \overline{v} | \} \\
 C &::= \{ | \overline{F} | \} \\
 F &::= L = h \mid l = t
 \end{aligned}$$

Core Terms:

$$\begin{aligned}
 t &::= x \mid v \mid t \ t \mid t \circ t \mid r \mid \text{if } t \text{ then } t \text{ else } t \\
 r &::= h.\text{super}(h).l
 \end{aligned}$$

Values:

$$\begin{aligned}
 v &::= b \mid \lambda x. t \mid \text{fix } x \text{ in } \mathcal{F} \mid \text{true} \mid \text{false} \mid \mathbb{Z} \mid \mathbb{Q} \\
 \mathcal{F} &::= \overline{x = \lambda y. t}
 \end{aligned}$$

Figure 1. MCL basic syntax

mutually recursive functions in \mathcal{F}). Values $v \subseteq t$ are the evaluated normal forms. Builtin primitives b are booleans, rational numbers, integers and strings. The corresponding binary operators are summarized in \circ .

A value field can be accessed from a class using a notation similar to the class-selection: A reference $r \triangleq h_1.\text{super}(h_2).l$ refers to the field labeled with l in class h_2 extended by h_1 . Again we allow the convenient abbreviation $h.l \triangleq h.\text{super}(h).l$

Capture-avoiding substitution of variables by a partial function p is written as $[p]t$. The usual conditions for freshness of bound variables have to apply to the codomain of the partial function. Substitutions do not pass over references, i.e. $[p]r \triangleq r$. We write $\llbracket t \rrbracket_{\text{fv}}$ to denote the set of free variables in a term. Variables are bound by abstraction and the mutually recursive functions (plus their arguments) of a fixed-point. In all other cases, the set of free variables is the union of the free variables of all sub terms.

A context is a term with a “hole” into which another term is plugged. This hole is expressed as a special variable \bullet . By convention \bullet is never bound in any term. Plugging a term t into a context s is then obtained via substitution $[\bullet \mapsto t]s$.

4 The Hierarchical Environment

The semantics of hierarchical terms can be seen as the reduction to a normal form of evaluated classes. We will motivate this normal form by a somewhat informal interpretation of the process. For reasons that will become clear in a moment, call an evaluated class a *node* (expressed by the syntactic sort v). Nodes are created as the combination of a literal class C with an environment.

The environment *inside* a node has one additional entry, mapping 0 to the node itself. All other entries link back to the original environment (just one level higher). In a certain sense, this definition forms an inverted view of the syntax tree, as each node gives access to an ordered set of children (which may be its parents in the syntax tree). Environments are forests of such trees and literal classes are node labels. (Hence the name *node* for the elements of this structure.)

Evaluation of hierarchical terms can be defined by the resolution of special variables and the three mutually recur-

sive operations, *selection*, *search* and *evaluation*. During evaluation, a special variable i is *resolved* in a given environment \mathcal{E} to the i -th entry of the environment.

$$\begin{aligned}
 \llbracket \mathcal{E}, \text{up}(n) \rrbracket_{\text{eval}} &\triangleq \mathcal{E}(n) \\
 \llbracket \mathcal{E}, h.L \rrbracket_{\text{eval}} &\triangleq \llbracket \llbracket \mathcal{E}, h \rrbracket_{\text{eval}}, L \rrbracket_{\text{select}} \\
 \llbracket \mathcal{E}, C \rrbracket_{\text{eval}} &\triangleq \{ | C \text{ in } \mathcal{E} | \} \\
 \llbracket \mathcal{E}, t \rrbracket_{\text{eval}} &\triangleq \dots
 \end{aligned}$$

Composite names (e.g. $\text{up}(2).z$) are evaluated from left to right by *selecting* the label. Evaluating a literal class with a given environment yields a context by appending that literal class to the current environment. We ignore the evaluation of proper terms for now.

$$\begin{aligned}
 \llbracket v, L \rrbracket_{\text{select}} &\triangleq \llbracket \llbracket v' \rrbracket_{\text{env}} \oplus \{ 0 \mapsto v \}, h \rrbracket_{\text{eval}} \\
 \text{when} \quad \llbracket v, L \rrbracket_{\text{search}} &\triangleq v', h
 \end{aligned}$$

In order to *select* a field from a class, its definition has to be found in the class itself or in a super class. The resulting term is then evaluated under a new environment (obtained via env from v'). By setting the 0-th environment entry to the receiver, the special variable *this* is given a new meaning. If the definition is found in the receiver itself, i.e. $v \triangleq v'$, the change has no effect.

$$\llbracket v, L \rrbracket_{\text{search}} \triangleq \begin{cases} v, h & \text{if } L = h \in \llbracket v \rrbracket_{\text{class}} \\ v', h' & \text{if } v \text{ extends } h_S \\ & \llbracket h, \llbracket v \rrbracket_{\text{env}} \rrbracket_{\text{eval}} \triangleq v_S \\ & \llbracket v_S, L \rrbracket_{\text{search}} \triangleq v', h' \end{cases}$$

A definition is *searched* recursively: If the field is a literal child, its right hand side is searched. Otherwise, the super classes of the context are evaluated and search continues there.

4.1 Graphical Interpretation

As an example, consider the classes C and D from above and the evaluation of $D.x$. Since all classes in that example have a unique name, this name is used in abbreviations as

VAL	OP		IF-TRUE	IF-FALSE
	$v_3 \triangleq (\text{arithmetic})v_1 \circ v_2$			
$v \Downarrow v$	$t_1 \Downarrow v_1 \quad t_2 \Downarrow v_2$	$t_1 \circ t_2 \Downarrow v_3$	$t_1 \Downarrow \text{true} \quad t_2 \Downarrow v$	$t_1 \Downarrow \text{false} \quad t_3 \Downarrow v$
			$\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \Downarrow v$	$\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \Downarrow v$
APP	FIXAPP		NODE	ROOT
	$t_2 \Downarrow v_1$	$t_1 \Downarrow \text{fix } x \text{ in } \mathcal{F} \quad t_2 \Downarrow v_2$		
$t_1 \Downarrow \lambda x. t_3$	$x \mapsto \lambda y. t_3 \in [\mathcal{F}]_\Pi$	$[[\mathcal{F}]]_\mu([y \mapsto v_2]t_3) \Downarrow v$	$v \Downarrow^h v$	$\{ \bar{F} \} \Downarrow^h \{ \bar{F} \text{ in } \bar{\pi} \}$
$[x \mapsto v_1]t_3 \Downarrow v_2$	$t_1 \quad t_2 \Downarrow v_2$			
HSELECT	SELECT		MOD	
	$h_1 \Downarrow^h v_1 \quad h_2 \Downarrow^h \{ \bar{F} \text{ in } \bar{\pi} \}$	$h_1 \Downarrow^h v_1 \quad h_2 \Downarrow^h \{ \bar{F} \text{ in } \bar{\pi} \}$		
$L \mapsto h_3 \in [\bar{F}]_\Pi \quad [[v_1, \bar{\pi}, h_3]]_\Phi \Downarrow^h v_3$	$l \mapsto t \in [\bar{F}]_\Pi \quad [[v_1, \bar{\pi}, t]]_\Phi \Downarrow v$	$h_1 \text{.super}(h_2) \text{.} l \Downarrow v$	$h \Downarrow^h \{ \bar{F}_1 \text{ in } \bar{\pi} \}$	$\text{dom}(\bar{F}_2) \subseteq \text{dom}(\bar{F}_1)$
$h_1 \text{.super}(h_2) \text{.} L \Downarrow^h v_3$			$[[\bar{F}_3]]_\Pi \triangleq [[\bar{F}_1]]_\Pi \oplus [[\bar{F}_2]]_\Pi$	$\{ \bar{F}_3 \text{ in } \bar{\pi} \} \Downarrow^h \{ \bar{F}_3 \text{ in } \bar{\pi} \}$

Figure 2. Evaluation semantics

4.2 Dynamic Semantics

Figure 2 depicts the rules of the dynamic semantics of MCL in big-step or natural(Kahn 1987) style. A term t evaluates to a value v , iff both are related by a reduction relation $t \Downarrow v$. Erroneous terms are identified by not being related to some value. All elements of \Downarrow are defined inductively by inference rules.

In order to simplify the notation of sequential constructs, it is useful to define a mapping between concrete syntax and partial functions. Each sequence can be seen as a partial function, mapping its left-hand elements to the corresponding right-hand side. This conversion is implemented with the function Π .

$$\begin{aligned}
 [\mathcal{F}]_\Pi &\triangleq \{x_1 \mapsto \lambda y_1. t_1\} \oplus \dots \oplus \{x_n \mapsto \lambda y_n. t_n\} \\
 \text{where } \mathcal{F} &\triangleq x_1 = \lambda y_1. t_1, \dots, x_n = \lambda y_n. t_n \\
 [\Diamond]_\Pi &\triangleq \emptyset \\
 [l = t :: \bar{F}]_\Pi &\triangleq [\bar{F}]_\Pi \oplus \{l \mapsto t\} \\
 [L = h :: \bar{F}]_\Pi &\triangleq [\bar{F}]_\Pi \oplus \{L \mapsto h\} \\
 [\mathcal{F}]_\mu &\triangleq \{x_i \mapsto \text{fix } \mathcal{F} \text{ in } x_i \mid x_i \in \text{dom}([\mathcal{F}]_\Pi)\}
 \end{aligned}$$

Rule APP is the standard application via substitution. OP implements binary operators on builtin primitives; it is actually a family of rules with one element for each builtin operator. Rules IF-FALSE and IF-TRUE implement non-strict conditionals.

Mutually recursive functions $\mathcal{F} \triangleq x_i = \lambda y_i. t_i$ are implemented via the explicit fixed point term $\text{fix } x \text{ in } \mathcal{F}$, the special function μ and rule FIXAPP. In order to evaluate a recursive function, first the argument has to be evaluated. This argument is then substituted into the body of the function, followed by a substitution of the group itself, as defined by μ . Due to the nature of natural semantics, divergence cannot be distinguished from a stuck term.

The hierarchical semantics of MCL is embedded into the proper evaluation (but not vice-versa). In a certain sense, classes play the role of modules. Evaluation of a hierarchical term h to a hierarchical class $v \triangleq \{|\bar{F} \text{ in } \bar{\pi}|\}$ with parents $\pi_1 \dots \pi_n$ is written $h \Downarrow^h v$. Rule SELECT augments the evaluation relation \Downarrow . Hierarchical nodes are already in normal form (rule NODE). An empty literal class evaluates to a root node (rule ROOT).

4.2.1 Selections and Inheritance

Selecting a child class via HSELECT or SELECT relies on the search of the corresponding definition. This is implemented by a partial function from labels to hierarchical terms. Depending on the context, either a class label L or a value label l is looked up. Notably, this definition of the search operation is not recursive. It relies on the encoding of inherited fields as references from this to the corresponding super class.

MCL does not allow for unqualified inheritance of names: Instead of a single **extends** statement, all inherited fields have to be explicitly present in the base class. The definition then forwards to the super class with the second argument of the reference:

```

A = this.super(up(1).Y).A;
a = this.super(up(1).Z).a;
    
```

In the example above, class A and the value a are inherited from classes Y and Z, which are found in the outer scope. The delegation is resolved by either HSELECT or SELECT. Multiple levels of inheritance are then expressed by a chain of such delegations. This style decouples the set of inherited elements from the definitions in the super class and allows for a more selective approach (e.g. it becomes possible to express the resolution of multiple inheritance of fields with the same name).

$$\begin{array}{ll}
\llbracket _, _, v \rrbracket_{\Phi} & \triangleq v \\
\llbracket v, \bar{\pi}, \text{up}(0) \rrbracket_{\Phi} & \triangleq v \\
\llbracket _, \bar{\pi}, \text{up}(n+1) \rrbracket_{\Phi} & \triangleq \pi_{n+1} \\
\llbracket _, \diamond, \text{up}(n+1) \rrbracket_{\Phi} & \triangleq \text{up}(n+1) \\
\llbracket v, \bar{\pi}, \{|\bar{F}|\} \rrbracket_{\Phi} & \triangleq \{|\bar{F}| \text{ in } v :: \bar{\pi}|\} \\
\llbracket v, \bar{\pi}, x \rrbracket_{\Phi} & \triangleq x \\
\llbracket v, \bar{\pi}, \lambda x. t \rrbracket_{\Phi} & \triangleq \lambda x. \llbracket v, \bar{\pi}, t \rrbracket_{\Phi} \\
\llbracket v, \bar{\pi}, t_1 \ t_2 \rrbracket_{\Phi} & \triangleq \llbracket v, \bar{\pi}, t_1 \rrbracket_{\Phi} \llbracket v, \bar{\pi}, t_2 \rrbracket_{\Phi} \dots
\end{array}
\quad
\begin{array}{ll}
\llbracket v, \bar{\pi}, \{|\bar{F}| \} \rrbracket_{\Phi} & \triangleq \{|\llbracket v, \bar{\pi}, h \rrbracket_{\Phi} \text{ with } f \times \bar{F}|\} \\
\text{where} & \llbracket L = h \rrbracket_f \triangleq L = \llbracket v, \bar{\pi}, h \rrbracket_{\Phi} \\
& \llbracket l = t \rrbracket_f \triangleq l = \llbracket v, \bar{\pi}, t \rrbracket_{\Phi} \\
\llbracket v, \bar{\pi}, h_1. \text{super}(h_2). L \rrbracket_{\Phi} & \triangleq \llbracket v, \bar{\pi}, h_1 \rrbracket_{\Phi}. \text{super}(\llbracket v, \bar{\pi}, h_2 \rrbracket_{\Phi}). L \\
\llbracket v, \bar{\pi}, h_1. \text{super}(h_2). l \rrbracket_{\Phi} & \triangleq \llbracket v, \bar{\pi}, h_1 \rrbracket_{\Phi}. \text{super}(\llbracket v, \bar{\pi}, h_2 \rrbracket_{\Phi}). l
\end{array}$$

Figure 3. The Fold Function

In order to adhere to the principle of open recursion between fields of a class, the special variables in a found term are resolved using the fold operators $\Phi : h \times \bar{v} \times h \rightarrow h$ and $\phi : h \times \bar{v} \times t \rightarrow t$ before evaluation. These mutually recursive functions (Figure 3) take three arguments: An evaluated class v represents the tip of the environment (i.e. the *self*-instance), $\bar{\pi}$ is the list of hierarchical parents of the super class containing the definition of the current term, and the third argument is the input that is being folded. Φ expects and returns an hierarchical term h while ϕ works on plain terms t .

In the case of plain terms, the result of folding is simple: ϕ is applied on the sub terms or returns its input unchanged if the argument is primitive. Value references are processed by folding the hierarchical sub terms with Φ .

Folding hierarchical terms resolves the special variables $\text{up}(i)$ (thus implementing the environment directly via substitution): The special variable $\text{up}(0)$ (the *this*-variable) is replaced with the *self*-instance (the first entry of the environment). Other special variables are looked up accordingly. If the environment is empty, the result is left unresolved. References and modified classes are folded by folding their corresponding sub terms. In the case of modifications this ensures that a modified field is evaluated in the context of the modification site (and not in the context of the modified class). Literal classes are turned into nodes by storing the environment alongside their fields. Contrary to modifications, their fields are not subject to further folding. This ensures that child classes retain their own context, when a field is selected from that child class. Nodes are left unchanged by the fold function.

4.3 Modifications and Redeclarations

MCL supports both redeclarations and modifications. The former are implemented via overriding of inherited methods (thus, there all fields are considered replaceable). Modifications differ from overriding in their scope — modifications live outside of the modified class. Each modified field must exist in the modified class. It is not possible to *add* a field via a modification. The modification of a class h with a sequence of fields \bar{F} results in a class containing the modified fields merged with the result of the evaluation (rule MOD). Merging is implemented by lifting the fields into

partial functions, augmenting the original function with the new fields and lowering the result into a sequence of fields.

5 Translation of References

The specification of Modelica require the evaluation of names only when necessary; i.e. the lookup of classes, functions, types and variables is always driven by the attempt to flatten a particular class. We take a slightly different stance, and demand that all references can be looked up strictly. The goal is to replace all references (inside a certain term) with their definitions (and transitively all references in them). The resulting term is then free of any hierarchical references and can be evaluated as usual. This technique allows to consider lookup and flattening as completely separate parts of the semantics (and gives reason to consider the former as part of the static semantics).

5.1 Evaluation of Hierarchical Terms

The definition of \Downarrow^h is algorithmic, a naive implementation will however not always terminate due to the open recursion. In particular, evaluating the subterms of a class reference might require evaluation of the same class reference:

```
{| class A = this; x = this.A.x |}.x
```

In the example above, a naive interpreter will repeatedly attempt to evaluate the class reference `this.A.x`. This is not a particularity of MCL, as the following example shows:

```
class A
  model B extends C; end B;
  model C extends B; end C;
  B.Foo b;
end A;
```

This simple model cannot be flattened (as there is no class definition for the component `b`). Yet, the attempt drives the leading free implementations OpenModelica (in version 1.11.0) and JModelica (version 1.17) into an endless loop, eventually ended by a stack overflow. In a realistically sized model, the user can only speculate what causes such a crash and, should the relevant loop be optimized to a tail-recursive implementation, might not even encounter a crash but a “frozen” implementation.

Thus it is necessary to restrict the computation of references in a way that *guarantees* termination and retains a valid result for a meaningful subset of the terminating nodes. It is hardly constructive to reject *all* recursive relations between classes, as for instance recursive functions would fall under the same rule (functions are specialized classes in Modelica). Instead, the restriction should only prevent divergence during lookup. This can be achieved by only attempting to evaluate an hierarchical reference once for any given environment.

We assume that each literal class is labeled with a unique identifier from a set $\mathbb{L} \subseteq \mathbb{N}$. We write \bar{F}_i to indicate a class with id i . The *syntactic depth* of a literal class is the number of syntactically visible enclosing classes. It is easy to see that this number is invariant during evaluation (otherwise, special variables might be invalidated). Each node is only valid when it has *precisely* the correct amount of enclosing classes for its literal class. A node that fulfills this requirement is called *context correct*. The set of context correct nodes is not finite, though. Consider two distinct literal classes \bar{F}_1 and \bar{F}_2 with depths 0 and 1, then $\{|\bar{F}_2 \text{ in } \{|\bar{F}_1 \text{ in } \diamond|\}\}$ is context correct. But so is $\{|\bar{F}_2 \text{ in } \{|\bar{F}_2 \text{ in } \{|\bar{F}_1 \text{ in } \diamond|\}\}\}$ and so on. Obviously, hierarchies with the repeated occurrence of the same literal class are problematic. It is thus necessary to find a syntactic criterion to rule out such strange loops.

The directed graphs used in Section 4 can be formalized as directed multigraphs (V, E) with vertices $V \subseteq \mathbb{L}$ represented by the labels of literal classes and edges as triples of one outgoing and one incoming vertex together with a natural number $E \subseteq \mathbb{L} \times \mathbb{L} \times \mathbb{N}$. The identity of an edge is defined by its source, its destination and its number. The usual terms from (multi) graph theory (reachability, cycles, etc.) apply.

Definition 1 (Graph Representation of Nodes and Environments). *The directed multigraph of a node is the vertex labeled with the literal class of the node linked to the graphs of all parents by ordered edges. The graph of an environment (i.e. a list of nodes) is the union of the graphs of each node (where the union of graphs is the union of their components).*

$$\begin{aligned} \llbracket \{|\bar{F}_u \text{ in } \bar{\pi}|\} \rrbracket_{gr} &\triangleq (\{u\} \cup V, P \cup E) \\ \text{where } P &\triangleq \{(u, p_i, i) \mid p_i \triangleq \llbracket \pi_i \rrbracket_L\} \\ (V, E) &\triangleq \bigcup_{i \in 1 \dots |\bar{\pi}|} \llbracket \pi_i \rrbracket_{gr} \end{aligned}$$

The set of possible results of this transformation is finite, when both the set of labels and edges are finite. Both conditions are trivially fulfilled by graphs created from context correct nodes, since each node is in itself a finite structure, the syntactic depth of each node is limited by the syntactic structure of the source program, and each source program is labeled by a finite set of labels.

Multigraphs that do not contain any cycles and have a distinguished root node can be unambiguously transformed

into a node, when the outgoing edges of a each node are labeled consecutively with the numbers ranging from 1 to the depth of the corresponding literal class. Such a graph is said to be context correct. This transformation is bijective.

Definition 2 (Admissible Lookups). *A node is admissible, iff its graph representation is a context correct, rooted multigraph. An environment $\bar{\pi}$ is admissible, iff all contained nodes are admissible and the lookup of a label L in an environment is admissible iff the environment is admissible:*

$$\begin{aligned} v \text{ admissible} &\iff \llbracket v \rrbracket_{gr} \text{ is rooted and context correct} \\ \bar{\pi} \text{ admissible} &\iff \forall i \in 1 \dots |\bar{\pi}| \pi_i \text{ admissible} \\ \langle v :: \bar{\pi} \cdot L \rangle \text{ admissible} &\iff v \text{ admissible} \wedge \bar{\pi} \text{ admissible} \end{aligned}$$

If all nodes are rejected that do not meet these simple criteria, the set of admissible nodes is finite. This allows to evaluate any hierarchical term without in a finite amount of steps (by checking for repetitions). As a side effect, all strange loops (i.e. classes that contain themselves) are ruled out, but classes that merely *refer* to each other are still allowed.

Lemma 1 (Finiteness of Admissible Lookups). *For any given finite labeling of literal classes, and a finite maximal depth of classes the set of admissible lookups is finite, admissible nodes is finite.*

Proof. Admissible nodes are finite due to their injective mapping to a context correct, rooted multigraph over the (finite) labeled vertices. Admissible (finite) environments and lookups are products of finite sets. \square

Evaluation of hierarchical terms can be implemented in a terminating, total function \mathcal{H} . This function follows the definition of \Downarrow^h by construction. The sole difference lies in the “memory” G , a set of admissible lookups. No lookup is ever repeated, hence the function terminates.

$$\begin{aligned} \llbracket G, v \rrbracket_{\mathcal{H}} &\triangleq v \\ \llbracket G, \{|\bar{F}| \} \rrbracket_{\mathcal{H}} &\triangleq \{|\bar{F} \text{ in } \diamond|\} \\ \llbracket G, \{|\bar{h} \text{ with } \bar{F}_2|\} \rrbracket_{\mathcal{H}} &\triangleq \{|\bar{F}_3 \text{ in } \bar{\pi}|\} \\ \text{if } \llbracket G, h \rrbracket_{\mathcal{H}} &\triangleq \{|\bar{F}_1 \text{ in } \bar{\pi}|\} \\ \text{dom}(\bar{F}_2) &\subseteq \text{dom}(\bar{F}_1) \\ \llbracket \bar{F}_3 \rrbracket_{\Pi} &\triangleq \llbracket \bar{F}_1 \rrbracket_{\Pi} \oplus \llbracket \bar{F}_2 \rrbracket_{\Pi} \\ \llbracket G, h_1 \cdot \text{super}(h_2) \cdot L \rrbracket_{\mathcal{H}} &\triangleq \llbracket G', \llbracket v_1, \bar{\pi}, h_L \rrbracket_{\Phi} \rrbracket_{\mathcal{H}} \\ \text{if } \llbracket G, h_1 \rrbracket_{\mathcal{H}} &\triangleq v_1 \\ \llbracket G, h_2 \rrbracket_{\mathcal{H}} &\triangleq \{|\bar{F} \text{ in } \bar{\pi}|\} \\ L \mapsto h_L &\in \llbracket \bar{F} \rrbracket_{\Pi} \\ \langle v_1 :: \bar{\pi} \cdot L \rangle \text{ admissible} &\notin G \\ G' &\triangleq G \cup \{\langle v_1 :: \bar{\pi} \cdot L \rangle\} \\ \llbracket G, h \rrbracket_{\mathcal{H}} &\triangleq \text{?} \quad \text{in any other case} \end{aligned}$$

5.2 Lookup of References

The definition of \mathcal{H} immediately yields an algorithm for the lookup of references, \mathcal{L} . A lookup is successful if both the super class and the base class can be evaluated successfully, the resulting environment is admissible and it contains a matching element. The result of a successful lookup maps the environment and looked up label to the folded result term. An error is indicated by the mark $\not\downarrow$.

$$\begin{aligned} \llbracket h_1 \cdot \text{super}(h_2) \cdot l \rrbracket_{\mathcal{L}} &\triangleq \langle v_1 :: \bar{\pi} \cdot l \rangle \mapsto \llbracket v_1, \bar{\pi}, t \rrbracket_{\phi} \\ \text{if} \quad &\llbracket \emptyset, h_1 \rrbracket_{\mathcal{H}} \triangleq v_1 \\ &\llbracket \emptyset, h_2 \rrbracket_{\mathcal{H}} \triangleq \{ \bar{F} \text{ in } \bar{\pi} | \} \\ &l \mapsto t \in \llbracket \bar{F} \rrbracket_{\Pi} \\ \llbracket r \rrbracket_{\mathcal{L}} &\triangleq \not\downarrow \quad \text{otherwise} \end{aligned}$$

This allows to lookup *all* references in a term, including those references that occur transitively as the result of a successful lookup. Such an exhaustive search is achieved by repeated applications of a one-step search function \mathcal{G} to an intermediate result set R :

$$\begin{aligned} \llbracket \not\downarrow \rrbracket_{\mathcal{G}} &\triangleq \not\downarrow \\ \llbracket R \rrbracket_{\mathcal{G}} &\triangleq \begin{cases} \not\downarrow & \text{if } \exists [\cdot \mapsto r]s \in \text{img}(R) \text{ s.t. } \llbracket r \rrbracket_{\mathcal{L}} \triangleq \not\downarrow \\ R \cup \{ \llbracket r \rrbracket_{\mathcal{L}} \mid [\cdot \mapsto r]s \in \text{img}(R) \} & \text{otherwise} \end{cases} \end{aligned}$$

The exhaustive search terminates, when a fixed point is reached. This is guaranteed due to the finite set of admissible environments. \mathcal{G} is also *inflationary*. This guarantees the existence of the conditional fixed point starting from a set R (see Pepper and Hofstedt 2006, chapter 10).

Lemma 2 (Fixed Point of \mathcal{G}). *The ascending Kleene chain of \mathcal{G} has a least fixed point.*

Proof. The partial functions (and error marker) obtained by \mathcal{G} form a complete partial order (cpo) under the subset relation, i.e. $R_1 \leq R_2 \iff R_1 \subseteq R_2$ with $\not\downarrow$ as top element, i.e. $R \leq \not\downarrow$, because the set of admissible environments (the domain of each R) is finite and adding a top element to a cpo yields a cpo. Function \mathcal{G} is also Scott-continuous (the least upper bound of any chain is the set-union in the absence of errors and the error otherwise). \square

5.3 Transformation

A reference must be evaluated in order to look up its corresponding definition. This does not introduce any errors, if the underlying search result is indeed a fixed point, though (as all contained references have already been evaluated at least one). The definition of a reference might (after several steps of lookup) depend on the reference itself. This implicit recursion has to be transformed into a proper fixed point. In order to do so, all definitions have to be regarded

as functions, since MCL does not allow for any other recursive definitions. This is easily achieved by wrapping them into a “thunk” (a function taking an unused argument).

Function \mathcal{C} maps each found definition to a structurally similar term where all references are replaced with their corresponding name. It is assumed that the lookup result is arbitrarily ordered.

Definition 3 (Transformation). *The transformation replaces all references with a recursive function that is obtained by the lookup closure of its definition. The closure replaces each references with the name of its definition.*

$$\begin{aligned} \llbracket R, t \rrbracket_{\mathcal{C}} &\triangleq t \quad \text{if } r \notin t \\ \llbracket R, [\cdot \mapsto r]t \rrbracket_{\mathcal{C}} &\triangleq [\cdot \mapsto x_i \ 0] \llbracket R, t \rrbracket_{\mathcal{C}} \\ \text{where} \quad R &\triangleq \{ L_1 \mapsto t_1, \dots, L_n \mapsto t_n \} \\ \llbracket r \rrbracket_{\mathcal{L}} &\triangleq L_i \mapsto t_i \\ \{x_1 \dots x_n\} &\text{ fresh in } \text{img}(R) \\ \llbracket R \rrbracket_{\mathcal{C}} &\triangleq \{ x_i \mapsto \lambda y. t_i \mid t_i \in \text{img}(R), y \notin \llbracket t_i \rrbracket_{fv} \} \\ \llbracket t \rrbracket_{\gamma} &\triangleq t \quad \text{if } r \notin t \\ \llbracket [\cdot \mapsto r]s \rrbracket_{\gamma} &\triangleq [\cdot \mapsto (\text{fix } x_r \text{ in } \mathcal{F}_R) \ 0] \llbracket s \rrbracket_{\gamma} \\ \text{if} \quad &\llbracket \mathcal{F}_R \rrbracket_{\Pi} \triangleq \llbracket R \rrbracket_{\mathcal{C}} \\ \text{and} \quad &\llbracket R \rrbracket_{\mathcal{G}} \triangleq R \\ \text{and} \quad &R \triangleq \llbracket \{ \langle v :: \bar{\pi} \cdot l \rangle \mapsto t \} \rrbracket_{\mathcal{G}^n} \\ \text{and} \quad &\llbracket r \rrbracket_{\mathcal{L}} \triangleq \langle v_1 :: \bar{\pi}_1 \cdot l_1 \rangle \mapsto t_1 \\ \llbracket [\cdot \mapsto r]s \rrbracket_{\gamma} &\triangleq \not\downarrow \quad \text{otherwise} \end{aligned}$$

5.4 Example

Our running example can be encoded in MCL as $v_{\text{root}} \cdot D \cdot x$. For this term, the exhaustive lookup yields the result:

$$\begin{aligned} R &\triangleq \{ \\ &\langle v_D :: \mathcal{E}_{\text{root}} \cdot x \rangle \mapsto \llbracket v_D, \mathcal{E}_{\text{root}}, \text{this.A.x} \rrbracket_{\phi}, \\ &\langle v_A :: \mathcal{E}_S \cdot x \rangle \mapsto \llbracket v_A, \mathcal{E}_S, \text{up}(2) \cdot y + \text{up}(1) \cdot z \rrbracket_{\phi}, \\ &\langle v_C :: \mathcal{E}_{\text{root}} \cdot y \rangle \mapsto \llbracket v_C, \mathcal{E}_{\text{root}}, \text{this.B.z} \rrbracket_{\phi}, \\ &\langle v_D :: \mathcal{E}_{\text{root}} \cdot z \rangle \mapsto \llbracket v_D, \mathcal{E}_{\text{root}}, 2 \rrbracket_{\phi}, \\ &\langle v_B :: \mathcal{E}_C \cdot z \rangle \mapsto \llbracket v_B, \mathcal{E}_C, 21 \rrbracket_{\phi} \} \end{aligned}$$

After closing this complete result, the translation yields:

```
(x0 in fix
  x0 = y. (x1 0) ;
  x1 = y. (x2 0) + (x3 0) ;
  x2 = y. (x4 0) ;
  x3 = y. 2 ;
  x4 = y. 23 ;
) 0
```

This term then evaluates to 23, as expected.

5.5 Correctness

The correctness of γ depends on the closure of a term by a complete lookup result. The most important step is an observation about a symmetry between the evaluation of a term containing hierarchical references and that of a fixed point constructed from the lookup result R of that term: Both evaluations only differ in the presence or absence of rule SELECT, which is replaced by specific instances of FIXAPP (with a group of recursive definitions generated from R).

Lemma 3 (Correctness of \mathcal{C}). *The closure of a complete lookup result is equivalent to the lookup of references.*

When R is complete, i.e. $\llbracket R \rrbracket_{\mathcal{G}} \triangleq R \neq \perp$, a term t appears on the image of R , i.e. $\langle v :: \bar{\pi} \cdot l \rangle \mapsto [\cdot \mapsto t]s \in R$, and t evaluates with n applications of rule SELECT, i.e. $\llbracket v, \bar{\pi}, t \rrbracket_{\phi} \Downarrow_{n-\text{SELECT}}$. Then the application of R as a fixed point evaluates to an equivalent result n applications of rule FIXAPP to R :

$$(\llbracket R \rrbracket_{\mu \circ \mathcal{C}}) \llbracket R, t \rrbracket_{\mathcal{C}} \Downarrow_{n-\text{FIXAPP}-R} \llbracket R, v \rrbracket_{\mathcal{C}}$$

Proof. By natural induction over n . The base step ($n = 0$) follows by a straightforward induction over \Downarrow , since SELECT is not applied in the derivation. The inductive step also requires a nested induction over \Downarrow . In the case of $t \triangleq r$, the completeness of R is used to apply one step of rule FIXAPP (and thus the outer induction hypothesis). \square

The correctness of the overall transformation is defined as the preservation of the semantics of the transformed term: When a term evaluates and the transformation yields no error, then the transformed term yields a value that is equal to the transformation of the original result.

Theorem 4 (Correctness of γ). *The transformation γ preserves the semantics of terms.*

$$t \Downarrow v \wedge \llbracket t \rrbracket_{\gamma} \triangleq s \implies s \Downarrow \llbracket v \rrbracket_{\gamma}$$

Proof. By induction over t . The fundamental case is $t \triangleq r$. By construction of γ , $\llbracket r \rrbracket_{\mathcal{C}} \triangleq \langle v :: \bar{\pi} \cdot l \rangle \mapsto \llbracket v, \bar{\pi}, s \rrbracket_{\phi} \in R$. Inversion of the evaluation yields $\llbracket v, \bar{\pi}, s \rrbracket_{\phi} \Downarrow v$. The conclusion then follows via rule FIXAPP and Lemma 3. \square

6 Discussion

We have given a definition of an explicit, context-independent syntax and semantics for the lookup of names in Modelica classes. Classes (hierarchical terms) can be translated by a terminating evaluation of all references. This translation maintains the original semantics.

6.1 Related Work

The semantics of Modelica has been subject to surprisingly little research. The work of Kågedal (1998), has a much broader scope. It does however not discuss open recursion

nor redeclarations and is considerably outdated when it comes to modern Modelica. Satabin et al. (2015) use a style comparable to ours, but favor a global environment (called class table) over our substitution based approach. Interestingly, they also notice the difficulty to separate the *static* semantics of a model from its dynamics, but solve this problem by restricting their input language. In particular, no short class definitions or redeclarations are considered. It is also somewhat unclear if their approach allows for the late binding of modifications. Despite these differences, the presented technique may solve the open question of how to obtain the values for our special variables in the first place.

6.2 Conclusion

The definition of Modelica's hierarchical elements by special variables allows to express their static semantics. Treating classes and their interactions like modules with open recursion allows for a precise reasoning of the outcome of redeclarations and modifications. Last but not least, the difficulties that come with the uniform treatment of classes and components are now obvious and might have an influence on the design of future versions of Modelica. The *correct* translation of hierarchical references in a *terminating* process while maintaining the semantics of inheritance, modifications and redeclarations is a feature that, to our knowledge, has not been solved before. It allows a clear separation between the static and dynamic semantics of names in Modelica.

References

- Aldrich, Jonathan and Kevin Donnelly (2004). “Selective open recursion: Modular reasoning about components and inheritance”. In: *SAVCBS 2004 Specification and Verification of Component-Based Systems*, p. 26.
- Höger, Christoph (2016). “Modeling with monads: extensible modeling semantics as syntactic sugar”. In: *Proceedings of the 7th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*. ACM, pp. 15–24.
- Kågedal, David (1998). “A Natural Semantics specification for the equation-based modeling language Modelica”. In: *LiTH-IDA-Ex-98/48, Linköping University, Sweden*.
- Kahn, Gilles (1987). “Natural semantics”. In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer, pp. 22–39.
- Pepper, Peter and Petra Hofstedt (2006). *Funktionale Programmierung – Sprachdesign und Programmieretechnik*. Springer.
- Pierce, Benjamin C., ed. (2005). *Advanced Topics in Types and Programming Languages*. MIT Press.
- Satabin, Lucas et al. (2015). “Towards a formalized Modelica subset”. In: *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21–23, 2015*. 118. Linköping University Electronic Press, pp. 637–646.

Towards a Standard-Conform, Platform-Generic and Feature-Rich Modelica Device Drivers Library

Bernhard Thiele¹ Thomas Beutlich² Volker Waurich³ Martin Sjölund¹ Tobias Bellmann⁴

¹PELAB, Linköping University, Sweden, {bernhard.thiele,martin.sjolund}@liu.se

²ESI ITI GmbH, Germany, thomas.beutlich@esi-group.com

³Chair of Construction Machinery, TU Dresden, Germany, volker.waurich@tu-dresden.de

⁴Institute of System Dynamics and Control, DLR, Germany, tobias.bellmann@dlr.de

Abstract

There are many cases where simulation applications need to interact with their environment. Typical examples are Human-in-the-Loop (HITL) simulators (including flight, driving, and marine training simulators), Hardware-in-the-Loop (HIL) simulators, but also offline process simulators which cannot operate in a completely self-contained manner and therefore need to be coupled to external applications. Embedded control applications are another related area requiring interaction between applications and their environment. The *Modelica_DeviceDrivers* library, which had its first release as open-source library in 2012, tries to cater to such use cases. This paper describes the library for the first time and reports about the numerous challenges that the project experienced to meet its goal of supporting several platforms and tools within a standard-conform, platform-generic, feature-rich, and easy-to-use Modelica library. Furthermore, the paper gives an insight into the inner mechanics of the library's communication and serialization functionalities, the various supported hardware interfaces and the possibilities to generate code for embedded systems.

Keywords: *human-in-the-loop, hardware-in-the-loop, real-time simulation, embedded control application, Modelica external C*

1 Introduction

The most common usage of Modelica models is for offline simulation experiments. However, in many cases simulations need to interact with their environment or other software components. Typical examples are Human-in-the-Loop (HITL) simulators (including flight, driving, and marine training simulators), Hardware-in-the-Loop (HIL) simulators, but also offline process simulators which cannot operate in a completely self-contained manner and therefore need to be coupled to external applications. Furthermore, Modelica can be used for developing (model-based) control applications that also require interaction with their environment.

There are different approaches for enabling the above-mentioned applications in the context of Modelica. Several development environments offer tool chains for real-

time simulation and/or model-based development of embedded control applications. Some of these environments can be coupled with Modelica tools, by wrapping code that is generated from Modelica tools into respective third-party tool-internal representations which can be connected to hardware devices in the respective development environment. For example, such customized solutions are available in Dymola¹ via its DymolaBlock interface to the MATLAB/Simulink² tool chain, OpenModelica³ via customized tool chains (Worschech and Mikelsons, 2012), or SimulationX⁴ via Code Export for Simulink/Simulink Coder² or HIL environments like dSPACE DS1006⁵, NI VeriStand⁶ or ETAS LABCAR⁷ (Blochwitz and Beutlich, 2009). Furthermore, it may also be possible for a Modelica tool to generate Functional Mock-up Units⁸ (FMUs) which can be imported into compatible simulator environments (e.g., the dSPACE SCALEXIO⁵ HIL simulator).

Instead of embedding the (FMI-) compiled Modelica model into a simulator environment, the *Modelica_DeviceDrivers* (MDD) library uses a different approach. The MDD library provides access to external devices by utilizing Modelica's external function interface for interfacing to the C API of various device drivers directly from Modelica models (see Section 2).

Historically, the origins of the MDD library can be traced back to the *ExternalDevices* library (Bellmann, 2009), an internal DLR⁹ Modelica library developed for the interactive simulation and visualization of Modelica models. The *ExternalDevices* library already supported UDP and shared memory communication as well as several

¹Dassault Systèmes, <https://www.3ds.com>

²The MathWorks, <https://mathworks.com>

³Open Source Modelica Consortium (OSMC), <https://www.openmodelica.org>

⁴SimulationX by ESI, <https://www.simulationx.com>

⁵dSPACE, <https://www.dspace.com>

⁶National Instruments, <http://www.ni.com>

⁷ETAS, <http://www.etas.com>

⁸FMI development group, <https://www.fmi-standard.org>

⁹Deutsches Zentrum für Luft- und Raumfahrt (DLR), German Aerospace Center, <http://dlr.de>

input devices (keyboard, 3Dconnexion SpaceMouse¹⁰, and game controller). Additionally, it featured a model-integrated real-time visualization system, the foundation of the later *DLR Visualization* library (Hellerer et al., 2014).

However, the *ExternalDevices* library only supported Microsoft Windows and was developed and tested using only the Dymola tool, which caused unintentional incompatibilities with other Modelica tools. In the further course of development, it was decided to split the *ExternalDevices* library into the commercial *DLR Visualization* library and an open-source cross-platform hardware interface library, the *Modelica_DeviceDrivers* library. The library is available from its GitHub project site at https://github.com/modelica/Modelica_DeviceDrivers/. This paper is based on MDD v1.5.0.

2 Modelica_DeviceDrivers

The MDD library allows Modelica models to access hardware devices by using the Modelica external C interface calling the appropriate C driver functions provided by the underlying operating system (see Section 2.1).

The library is organized in several layers as indicated in Figure 1. It provides two high-level drag & drop block interfaces.

1. The `Blocks` components are compatible to Modelica v3.2, using `when sample()` for periodically calling Modelica functions from the Function Layer.
2. The `ClockedBlocks` components use the synchronous language elements extension introduced in Modelica v3.3 and are compatible with the *Modelica_Synchronous* library (Otter et al., 2012). Due to this support, the MDD library formally depends on the *Modelica_Synchronous* library, but in practice the *Modelica_Synchronous* library (and tool support for the synchronous language elements extension) is only required for this `ClockedBlocks` interface.

2.1 Cross-Platform Support

As of MDD v1.5.0, Windows and Linux are supported as main platforms, but prototypical work also targets popular embedded systems boards directly (see Section 4.2).

When accessing hardware devices, a Modelica model or application calls Modelica functions from the Function Layer (see Figure 1). These Modelica functions provide a generic interface to the underlying C Code Layer, which is accessed by Modelica's external function interface. The platform differentiation is handled in the C Code Layer which uses preprocessor directives for conditional inclusion/exclusion of platform-specific code (`#if`, `#else`, `#endif`, etc.) similar to the code fragment below.

¹⁰3Dconnexion, <https://3dconnexion.com>

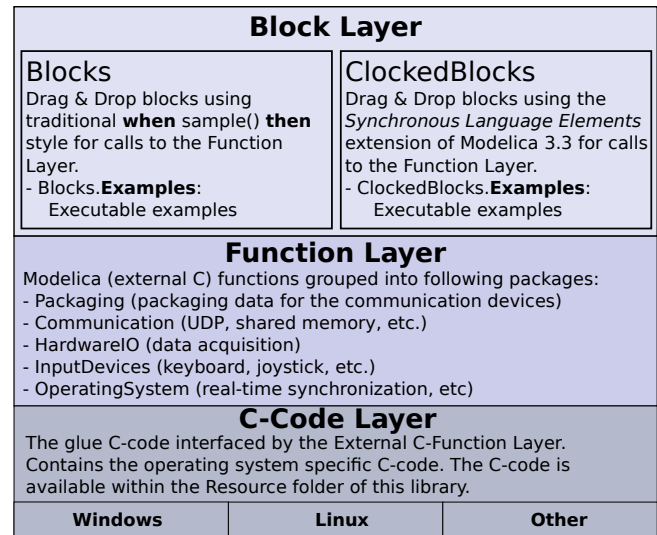


Figure 1. MDD layered architecture.

```
#if defined(_MSC_VER) || defined(__CYGWIN__
) || defined(__MINGW32__)
#include <windows.h>
/* Windows specific code goes here */
#elif defined(__linux__)
#include <unistd.h>
/* Linux specific code goes here */
#else
#error "Modelica_DeviceDrivers: Unsupported
      compiler or platform"
#endif
```

2.2 Extended Tool Support

Back in 2009, the library was developed using the Dymola tool. With MDD v1.4.0, considerable development efforts have been spent on the Modelica compliance of the library in order to better support SimulationX and OpenModelica.

Since OpenModelica v1.11.0 Beta 1 the MDD `SerialPackager` blocks as well as the `Communication` blocks are finally supported by OpenModelica. For achieving this, it was necessary to change parts of the MDD library (under the constraint of maintaining backwards compatibility), and at the same time, to extend the abilities of respective tools (partly by providing support for non-standard Modelica constructs). This is discussed in more detail in Section 3.2.3.

2.3 Library Structure

Figure 2 shows a screenshot of the package browser view with loaded MDD library. The first two sub-packages `Blocks` and `ClockedBlocks` provide the drag & drop blocks which correspond to the Block Layer of Figure 1. The remaining sub-packages (except `Utilities` and `EmbeddedTargets`) provide the Function Layer. Both layers use sub-packages for subdividing the provided functionality into different groups. Package `EmbeddedTargets` contains highly target speci-

fic function and blocks for supporting restricted embedded systems like the Arduino microcontroller (see Section 4).

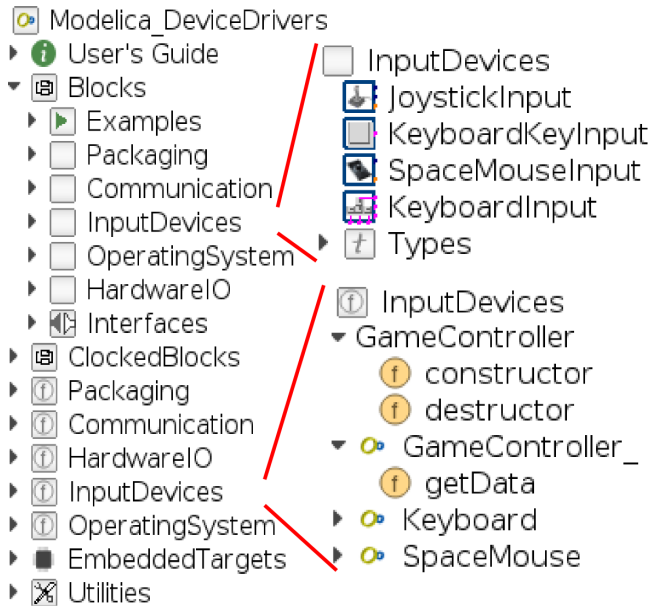


Figure 2. MDD library structure.

Furthermore, Figure 2 gives an indication about the relation between the Block Layer and the Function Layer. Typically, a device driver block will instantiate the corresponding external object from the Function Layer. Modelica's external objects allow external functions to access the internal memory (created by the `constructor`) between calls to external functions, *i.e.*, the device handlers are maintained in memory in order to access them in subsequent simulation phases. Modelica also guarantees that both the `constructor` and `destructor` functions of an external object are called exactly once, enabling a reliable *one-time* initialization and termination of hardware devices, usually during the initialization and termination phase of the Modelica simulation model, respectively. For example, the `JoystickInput` block creates an instance of the external object `GameController`. The package `GameController_` collects functions that can operate on external objects of type `GameController`. This package provides the function `getData`, which takes a `GameController` object as argument and returns the values of the axes and buttons of its associated hardware device.

A good way of learning how to use the Block Layer interface of the library is by exploring the `Examples` package. Care has been taken to provide self-explanatory usage examples for the provided device driver blocks.

2.4 Interfaces

MDD library functionality can be accessed by drag & drop of blocks from the `Blocks` and `ClockedBlocks` sub-packages, or by direct calls to the underlying functions.

An example, which directly uses the Function Layer for accessing a game controller, is given below:

```
model GameControllerExample
import
  Modelica_DeviceDrivers.InputDevices.*;
parameter Integer id = 0 "0 = first
  attached game controller";
GameController gc = GameController(id);
discrete Real axesRaw[6];
Integer buttons[32], pOV;
equation
  when sample(0, 0.1) then
    (axesRaw, buttons, pOV) =
      GameController_.getData(gc);
  end when;
end GameControllerExample;
```

The code above creates an external object named `gc`. The constructor for this object takes the argument `id`. This argument allows specifying which controller to use if several game controllers are attached to the system. The function `getData` is called periodically within a `when`-clause. It takes the external object `gc` as argument and returns vectors which contain the values read from the associated game controller. The vector is pre-dimensioned, so that it can attune to controllers featuring as much as six axes, 32 buttons and a POV (point of view) switch. The actually available data depends on the connected game controller hardware. Tests with the actual hardware are needed for determining which vector entry corresponds to which physical axis or button.

Figure 3 shows how game controllers can be accessed by simply dragging & dropping a `JoystickInput` block into the diagram view of a Modelica tool. While Figure 3a uses the block found in the `Blocks` package, Figure 3b uses the corresponding clocked variant from `ClockedBlocks`. The additional blocks `periodicClock` and `assignClock` are from the `Modelica_Synchronous` library. They associate a periodic clock to the variables and equations within the `JoystickInput` block. As a result, the underlying `getData` function will be called whenever the associated clock ticks (*i.e.*, every 0.1 s in the presented example).

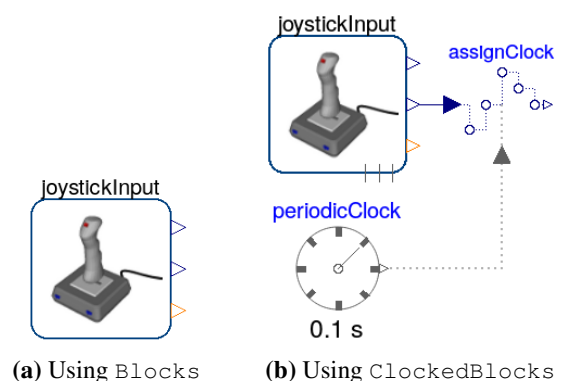


Figure 3. Accessing game controller devices by using the `JoystickInput` block from the `Blocks`, or the `ClockedBlocks` package.

The example models can be simulated, but real-time synchronization is required to slow the simulation speed

down, in order to synchronize the real-time inputs with the simulation progress. The MDD library provides convenient support for (soft) real-time synchronization¹¹. However, a user should consider that Modelica tools might provide better (tool-specific) options for real-time synchronization.

2.5 Features

The MDD library has grown to support a respectable amount of hardware devices and associated features that will be briefly presented in this section.

2.5.1 Input Devices

Standard input devices such as keyboard and game controllers are ubiquitously available on the market, enabling the user to build up interactive simulations quickly. MDD provides blocks for using the generic keyboard and game controller interface of Windows or Linux (see Figure 4).

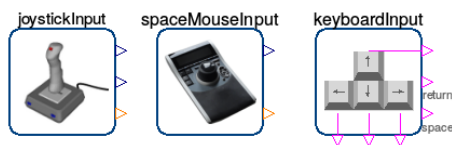


Figure 4. Supported input devices from the Blocks package.

In addition, more specialized hardware like the 3Dconnexion SpaceMouse is supported for both platforms. Often, these blocks will be used for interactive desktop simulations, but they can also become part of more involved (cost-efficient) HITL simulation scenarios.

2.5.2 Communication

The most comprehensive and complex part of the library is related to implementing support for communication devices in Modelica and external C code.

Supported Devices Figure 5 gives an overview over the supported devices.

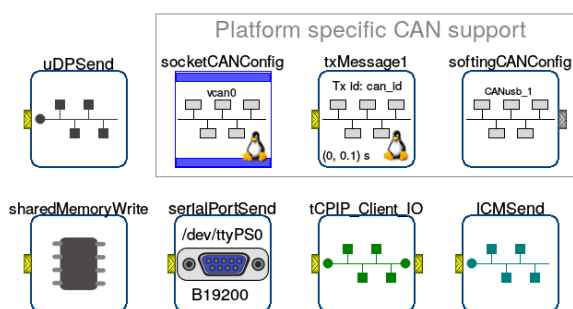


Figure 5. Supported communication devices from the Blocks package.

Cross-platform support for UDP and shared memory was already available in the first released version of MDD. Support for serial port communication is available since

MDD v1.3 (Linux) and v1.4.0 (Windows). A client block for TCP/IP socket communication was added in v1.4.0 (Windows) and v1.5.0 (Linux). Furthermore, support for sending and receiving of Lightweight Communications and Marshalling (LCM) datagrams¹² was added in v1.5.0. LCM is a set of libraries and tools for message passing and data marshalling¹³, which is particularly targeted at low-latency real-time applications for robotic systems (Huang et al., 2010).

Basic support for the Controller Area Network bus (CAN bus) is available by two different block sets. The first is based on the CAN Layer2 API from Softing¹⁴ and restricted to the Windows platform. The second uses the SocketCAN interface provided by the Linux kernel.

Packaging Concept Communication devices like UDP or shared memory use a common packaging concept in order to send or receive data. Therefore, the same packager can be used with different communication devices. Figure 6 shows an example in which a package consisting of three variables of type Real followed by a variable of type Integer is either transmitted using shared memory or UDP blocks. Switching between the two communication devices is achieved by simply replacing the corresponding device block.

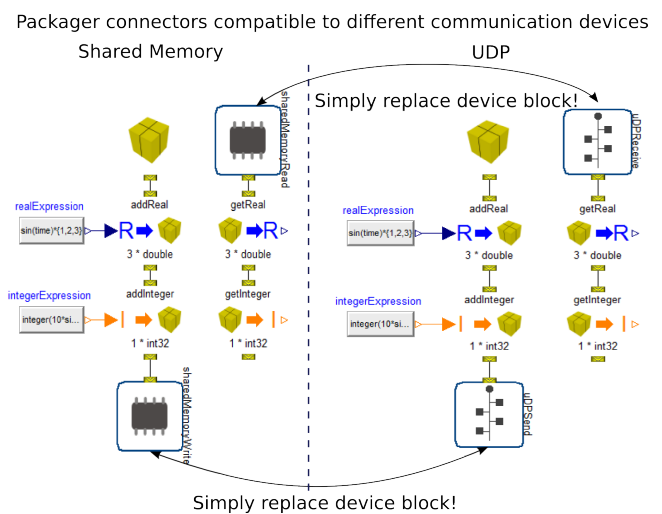


Figure 6. Simple switching of communication devices due to common packaging concept in order to send or receive data.

The packages are constructed by using blocks from the Packaging sub-package (see Figure 2). In the initial design of MDD, it was expected that different packaging concepts would be supported which share a common connector interface. However, as of MDD v1.5.0 the SerialPackager is the only available packager. It allows periodically adding or retrieving fixed size vectors to or from a package, respectively. Figure 7 shows the available blocks for serializing Modelica variables of the

¹¹See documentation to block SynchronizeRealtime.

¹²LCM project, <https://lcm-proj.github.io>

¹³As of MDD v1.5.0, only the communications aspect of LCM is considered.

¹⁴Softing, <http://industrial.softing.com>

predefined types `Boolean`, `Integer`, `Real` and `String` into a “package”. The type `Real` can be packed either as double-precision or (using a C static cast) as single-precision floating-point number.

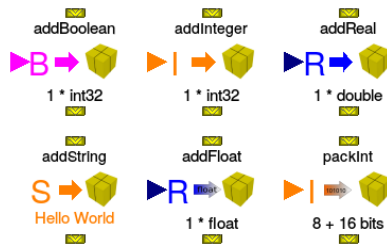


Figure 7. `SerialPackager` blocks for adding variables to a package.

At the C side, a package is a C byte array in which C variables with respectively indicated types are simply successively appended in a data-flow prescribed order. For example, in Figure 6 the resulting byte array starts with three `double` values (3×8 bytes) followed by one `int32` value (4 bytes), resulting in a byte array of size 28. For the sake of providing an illustrative example at the C language level the following C code snippet constructs a structurally equal package named `data` (the example shall shed light on the concept, it does not advocate a coding style using magic numbers for array offsets):

```
double v1[3] = {1.1, 2.2, 3.3};
int v2 = 4;
unsigned char* data = (unsigned char*)
    calloc(28, sizeof(unsigned char));
memcpy(&data[0], &v1[0], sizeof(v1));
memcpy(&data[24], &v2, sizeof(v2));
```

Figure 7 shows the blocks for adding variables to a package, symmetrically, blocks are available for retrieving variables from a package. Using these blocks is deemed to be rather intuitive with the notable exception of the `packInt` block. This block allows packing unsigned integer values at the bit level. The number of bits used for encoding is set by a parameter `width`, therefore the maximum value of the integer signal that can be encoded is $2^{\text{width}} - 1$. A parameter `bitOffset` allows to specify the bit at which the encoding starts relative to the preceding block. Since MDD v1.3 most blocks support specifying the byte ordering (big-endian or little-endian format).

It is simple to use the `SerialPackager` blocks for deserializing data which has been serialized by it (see Figure 6). In practice, however, communication typically needs to be established with a remote station that is unrelated to the Modelica model. As long as this remote station periodically sends or receives structurally static, fixed sized packages, it is usually quite convenient to establish a communication using the MDD blocks. If the remote station uses a more dynamic protocol, it becomes more difficult. In some cases using the Function Layer directly (instead of the Block Layer) can provide additional flexibility for coping with more dynamic protocols. Ho-

wever, the main use-case for the `SerialPackager` concept is periodically sent, structurally static data. These restrictions may be relieved in future versions of the MDD library by providing alternative, well-established “Packagers” that offer support for more flexible means of packaging data, e.g., the data marshalling of the LCM library or the efficient binary serialization format of the MessagePack library¹⁵.

Finally, it turned out that the `SerialPackager` blocks were a major hurdle for extending the number of Modelica tools which support MDD (see Section 3.2).

2.5.3 Hardware I/O

Package `HardwareIO` (see Figure 2) is intended for data acquisition hardware like digital-analog converter (DAC), analog-digital converter (ADC) and other interface hardware. As of MDD v1.5.0, it contains only one sub-package, which provides support for the Linux control and measurement device interface “Comedi”. The Comedi project develops open-source drivers, tools, and libraries for data acquisition¹⁶. The project provides a common interface for accessing supported data acquisition hardware (see the website for supported hardware). The MDD library implements an interface to the Comedi user-space library.

Figure 8 shows an example model, which uses the available blocks. Configuration of the device is performed in the Modelica record named `comedi`. The record contains an external object `dh` of type `ComediConfig` which contains the Comedi device handle and is passed through a parameter to the other blocks (`comedi.dh`). Using external objects in records is not standard-compliant to Modelica v3.3 revision 1 (Modelica Association, 2014), which is further discussed in Section 3.3.

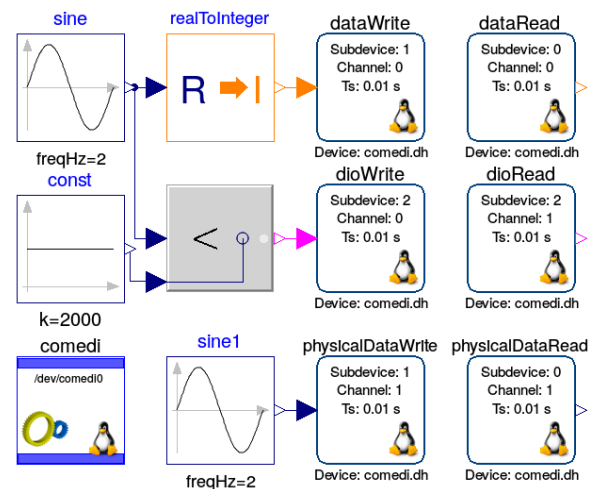


Figure 8. Accessing data acquisition hardware via the Linux control and measurement device interface “Comedi”.

Writing or reading raw integer values to DAC or from ADC channels is provided by the blocks `DataWrite`

¹⁵MessagePack project, <https://msgpack.org>

¹⁶Comedi project, <http://comedi.org>

or `DataRead`, respectively. These blocks have each a variant which works with physical values, instead of the raw integer values (`PhysicalDataWrite` and `PhysicalDataRead`). Blocks `DIOWrite` and `DIORead` support digital input and output (DIO) channels.

2.5.4 Embedded Targets

MDD v1.5.0 introduced the new top-level package `EmbeddedTargets`. The package is intended for platform-specific targets, such as microcontrollers, that cannot so easily share code with other devices due to memory or hardware limitations. There exists first prototypical support for the Atmel¹⁷ AVR family of microcontrollers. A prototype application is described in Section 4.2.

3 Modelica Standard-Compliance

Using a Modelica library-based approach for accessing hardware devices from a simulation started as an experiment, which relied on the Dymola tool and its support for interfacing external C code. However, when trying to extend the number of Modelica tools supporting the MDD library, it became apparent that quite a few constructs that were useful and appreciated by the initial authors of the library were not supported by other tools and were partly problematic in respect of compliance to the Modelica standard.

On one hand, this section reports on important development efforts (starting with MDD v1.4.0) that have been spent on the Modelica compliance of the library for better supporting SimulationX and OpenModelica, and on the other hand it addresses open issues which may be of interest for future improvements to the Modelica standard, or which may require possibly non-backwards compatible revisions of the MDD library for achieving full Modelica compliance.

3.1 Modelica's External Function Interface

As the Modelica standard specification on the external function interface improved over the years, standard-conform libraries with external C code dependencies could be created in a more satisfying way. For example, Modelica v3.2 standardized the search directory structure for the external C header files and libraries (Modelica Association, 2010, p. 153). Having a standardized directory structure facilitated creating cross-platform libraries with external C library dependencies. For example, the Modelica code snippet below declares an include dependency to the header file `MDDKeyboard.h` and linker dependencies to the libraries `X11` and `User32`:

```
function getKey
  input Integer keyCode "Key code";
  output Integer keyState "Key state";
  external "C" MDD_keyboardGetKey(keyCode,
    keyState) annotation(
    Include = "#include \"MDDKeyboard.h\"",
    Library = {"X11", "User32"});
```

¹⁷Atmel, <http://atmel.com>

```
annotation(__ModelicaAssociation_Impure=
  true);
end getKey;
```

A Modelica tool will map this information to compiler- and linker-dependent directives and thereby select the libraries that fit best for the respective platform.

3.1.1 Linking Platform-Dependent System Libraries

Having platform-specific system libraries like `x11` (for Linux only) and `User32` (for Windows only) in one generic `Library` annotation, proofed to be a significant development difficulty. As a remedy, dummy libraries of the Linux system libraries are provided in the Windows-specific library directories `win32` and `win64`, and vice versa. Furthermore, the linking to system libraries on Windows was simplified by the introduction of compiler-specific pragmas, e.g., in `MDDKeyboard.h`

```
#pragma comment(lib, "User32.lib")
```

understood by the Visual Studio compilers only. However, for GCC (including the MinGW and CygWin build environments) the issue remains unresolved¹⁸.

3.1.2 Impure Functions

The above example function `getKey` features an additional (vendor-neutral) annotation which declares the function as “impure”. The intended meaning is that a tool may not expect that the function returns the same output for the same input, which is the typical case for MDD functions that read values from external devices. Indeed, Modelica v3.3 introduced the dedicated keyword “impure” to cater for such cases. However, since not all Modelica tools support this keyword, yet, the MDD library uses the `Impure` annotation which is understood by Dymola, OpenModelica and SimulationX.

3.1.3 Modelica Standard Improvements

Future releases of MDD may benefit from improvements on the external function interface, which are expected in the (future) Modelica v3.4 standard:

- Compiler-specific sub-directories for the platform-specific library directories, e.g., if Visual Studio 2015 is used as a Windows 64-bit compiler a Modelica tool may first search directory `win64/vs2015` for dependent libraries¹⁹.
- The `IncludeDirectory` annotation accepts *multiple* directories enabling a more convenient way to specify several external C header file dependencies distributed over different include directories²⁰.

¹⁸Modelica Issue Tracker, <https://trac.modelica.org/Modelica/ticket/1668>

¹⁹Modelica Issue Tracker, <https://trac.modelica.org/Modelica/ticket/1316>

²⁰Modelica Issue Tracker, <https://trac.modelica.org/Modelica/ticket/2103>

However, a generalized build process¹⁸ of the external code still misses the definition and (future) standardization of build features such as compilation of several C source modules, compiler flags (CFLAGS) or preprocessor defines (CPPFLAGS)²¹.

3.2 The Serial Packager

The `SerialPackager` blocks are the core elements of the block-based communication support provided by the MDD library (see Section 2.5.2). They use a rather intricate approach for propagating a “package” between connected blocks.

3.2.1 Connector Definition

The definition of the `SerialPackager` input connector is given below.

```
connector PackageIn "Packager input
connector"
  input SerialPackager pkg;
  input Boolean trigger;
  input Real dummy;
  output Boolean backwardTrigger;
  output Integer userPkgBitSize;
  output Integer autoPkgBitSize;
end PackageIn;
```

The definition of the output connector is similar, but with reversed input and output causalities. Most notably connector `PackageOut` contains an element `pkg`, which is an external object of type `SerialPackager`. This external object is passed between connected blocks (see Figure 6). Within an “add” or “get” block the passed in external object is used as an argument to external functions which first add or retrieve data from the package and then pass it on to the next block.

Due to the design of the `SerialPackager` connector sharing *both* input and output variables it is impossible to have more than one `connect` equation per connector. However, Modelica offers no option to tell a user already at modeling time about this maximal allowed connector cardinality.

3.2.2 Basic Concept

The following *simplified* Modelica code snippet illustrates the basic idea for adding the (`Integer`) value of an input variable `u` to a package:

```
block AddInteger
  PackageIn pkgIn "Input connector";
  PackageOut pkgOut "Output connector";
  IntegerInput u "Integer input connector";
equation
  when initial() then
    pkgIn.autoPkgBitSize =
      pkgOut.autoPkgBitSize + 32 /* bit
      size of int32 */;
  end when;
  when pkgIn.trigger then
```

```
    pkgOut.dummy = addInteger(pkgOut.pkg,
      u, pkgIn.dummy);
  end when;
  pkgOut.pkg = pkgIn.pkg;
  pkgOut.trigger = pkgIn.trigger;
  pkgOut.backwardTrigger =
    pkgIn.backwardTrigger;
  pkgOut.userPkgBitSize =
    pkgIn.userPkgBitSize;
end AddInteger;
```

The instantaneous equation invoking the `addInteger` function is activated by the event `trigger` which is propagated through the connected packager blocks. The dummy variables are used to establish data-flow dependencies which ensure that the “addValue” functions of connected blocks are invoked in the correct order. The `backwardTrigger` event allows propagating a triggering event in the inverse connector direction. Its supporting logic is omitted here for brevity. A Modelica standard-conform alternative is provided by the variable `userPkgBitSize` that allows propagating a user defined package size, *i.e.*, it is possible for a user to customize the package size of the external data buffer of the communication device block (see Section 2.5.2). However, in the default setting the necessary package size is deduced automatically with the help of the `autoPkgBitSize` variable. This approach is described in Section 3.2.4.

3.2.3 External Object Aliasing

A problem with the Block Layer of the `SerialPackager` is that the `pkg` objects within the connectors are not explicitly created by calling an external object constructor function as required in Modelica v3.3 (Modelica Association, 2014, p. 165). Instead, they rely on aliasing through (connect) equations to access an external object which has been created at another place. In Figure 6 the `pkg` object for the “add” blocks is created in the “Packager” block at the top of the figure, while the `pkg` object for the “get” blocks is created in the device block for reading from shared memory (or UDP, respectively). While the concept of external object aliases does not exist in Modelica v3.3, equating two external objects may be interpreted as an assignment to an external object, which is forbidden. The authors hope that future versions of the Modelica standard will consider use-cases that the Modelica tools Dymola, OpenModelica and SimulationX already support²².

A Modelica standard-conform implementation that avoids the aliasing is to only rely on the Function Layer provided by package `SerialPackager_`.

3.2.4 Automatic Buffer Size

The actual creation of the `SerialPackager` object is performed in the “Packager” block, or, respectively, in the reading device block (see above). The following *simplified* code illustrates the basic concept.

²¹Modelica Issue Tracker, <https://trac.modelica.org/Modelica/ticket/2145>

²²Modelica Issue Tracker, <https://trac.modelica.org/Modelica/ticket/1669>


```

block Packager
  PackageOut pkgOut (
    pkg = SerialPackager(bufferSize),
    dummy(start=0, fixed=true));
  Integer bufferSize;
equation
  when initial() then
    bufferSize =
      if pkgOut.userPkgBitSize > 0 then
        pkgOut.userPkgBitSize else
        pkgOut.autoPkgBitSize;
      end when;
end Packager;

```

The difficulty here is that the `bufferSize` which is needed as an argument for the external object constructor `SerialPackager(bufferSize)` needs to be computed by solving the initial system of equations. This is not supported by all Modelica tools and its Modelica compliance was discussed at the Modelica Issue Tracker with a majority opting to clarify the specification in order to forbid it²³, but on the other hand it was also discussed how the Modelica standard could be extended to allow it²⁴.

In the initial version of the MDD library the external object was actually created within a `when`-clause, which was clearly illegal in Modelica v3.3. As part of improving the Modelica compliance of the library, the creation of the object was moved into the component declaration.

3.3 External Objects in Records

The `SocketCAN` and the `Comedi` blocks use a Modelica record as means for specifying general settings for a hardware device. The idea is that the settings are specified once when creating an instance of the record and this instance is passed as parameter to blocks using this device. For example, the `Comedi` configuration record (stripped from some elements for brevity) is defined as

```

record ComediConfig
  parameter String deviceName =
    "/dev/comedi0" "Name of Comedi device";
  final parameter Comedi dh =
    Comedi(deviceName) "Handle to comedi
    device";
end record;

```

where `dh` is an external object. It is convenient to collect configuration information in a record, since this allows passing a complete set of related configuration settings at once. The problem here is that passing an external object as part of a record can be interpreted as the record returning the object and assigning it to another external object (which is forbidden in Modelica v3.3 but supported by Dymola). However, similarly to the external object aliasing described in Section 3.2.3 it seems highly desirable to consider use-cases as described above in some way, in future versions of the Modelica standard.

²³Modelica Issue Tracker, <https://trac.modelica.org/Modelica/ticket/1907>

²⁴Modelica Issue Tracker, <https://trac.modelica.org/Modelica/ticket/2037>

3.4 Fixed Attribute of Strings

According to Modelica v3.3 the predefined type `String` was designed without the `fixed` attribute (as opposed to other predefined types `Boolean` or `Integer`). However, such a `fixed` attribute is particularly relevant for the `GetString` block of the `SerialPackager` when retrieving sampled `String` data from a package. This issue was resolved by (future) Modelica v3.4 such that future Modelica tools supporting Modelica v3.4 will no longer raise a warning on the `GetString` block²⁵.

4 Applications

This section describes several applications that were implemented with the help of the MDD library.

4.1 Arduino

The Arduino²⁶ is an open-source electronics platform that features easy configurations to read the sensors, process the data and send it to other devices via a serial connection. Therefore, the Arduino can be utilized to provide sensor data in a real-time Modelica model by means of the MDD serial port implementation, as depicted in Figure 9. With the help of potentiometers or other deflection sensors, customized control devices can be built.

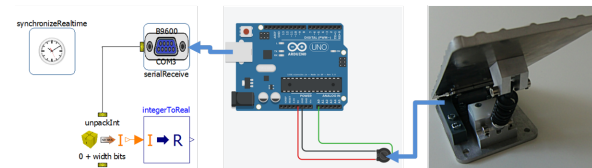


Figure 9. Setup to read potentiometer deflection during real-time simulation with MDD serial port model²⁷.

As an exemplary application, self-built pedals for a driving simulator can be equipped with a sensor in order to measure the displacement. The pedal itself is a steel sheet, mounted on a revolute joint and a shock spring. The measured deflection is transferred via a serial connection to a `Blocks.Communication.SerialPortReceive` in order to drive a virtual vehicle. Therefore, expensive or unavailable input devices can be substituted by custom constructions. By using a Bluetooth module with Serial Port Profile (SPP) a wireless connection between Arduino is handled in the same way as a serial port over USB connection. No further modifications are necessary to implement a wireless control device.

4.2 Embedded Control

The `EmbeddedTargets` package (see Section 2.5.4) contains blocks and functions to directly control I/O or clocks

²⁵Modelica Issue Tracker, <https://trac.modelica.org/Modelica/ticket/1797>

²⁶Arduino, <https://arduino.cc>

²⁷Autodesk screen shots reprinted courtesy of Autodesk, Inc.

in the AVR ATmega microcontroller family²⁸. The advantage of including this code into the MDD library is that it makes it simple to write a model for a microcontroller that works the same way in any Modelica tool since all the OS support, real-time code, *etc.*, is abstracted away. Provided that the Modelica tool produces minimal C-code (uses minimal features outside of standard C: for example, no linear solver included if the system has no linear systems, no OS or I/O functions, no threading models, *etc.*), and the model itself does not use C-code that the embedded target cannot support (such as file I/O), the code generator would work on pretty much any embedded target supporting C.

The Modelica code itself tries to avoid the Integer constants from the data sheets. Instead, enumerations such as `prescaler=1/128` or `clock=2B` are passed from Modelica and the C code for the AVR target depends on function inlining in order to remove dead code. For example, the constructor for the clock takes an enumeration that specifies the clock, which should be manipulated, and after function inlining, the C code for other clocks is removed. The blocks in the MDD library try to take user-friendly constants such as `frequency=100Hz` or `period=0.1s` for real-time synchronization; the Modelica code then has logic to find good clock prescalers to create a matching frequency. The code does not use parameters since they cannot be guaranteed to be evaluated in Modelica, and the C-code depends on the C-compiler (AVR GCC) being able to inline and eliminate dead code from C-code such as the constructor. An example of this is the timer external object in the microcontroller, which becomes one or two bitset instructions when the function is called with a constant input:

```
function constructor "Initialize timer"
  input Types.TimerSelect timerSelect;
  input Types.TimerPrescaler clockSelect;
  input Boolean clearTimerOnMatch;
  output Timer timer;
  external "C" timer = MDD_avr_timer_init(
    timerSelect, clockSelect,
    clearTimerOnMatch)
  annotation(Include = "#include \"
    MDDAVRTimer.h\"");
end constructor;

static inline void* MDD_avr_timer_init(int
  timerSelect, int clockSelect, int
  clearTimerOnMatch)
{
  static const uint8_t
    clockSelectTable0[7] = {...},
    clockSelectTable1[7] = {...},
    clockSelectTable2[7] = {...};
  switch (timerSelect) {
  #if defined(TCCR0)
  case 1: /* Timer 0 */
    TCCR0 |= ...;
    break;
  #endif
  #elif defined(TCCR0B)
  case 1: /* Timer 0 */
    TCCR0B |= clockSelectTable0[clockSelect
      -1];
    TCCR0A |= ...;
    break;
  #endif
  case 2: /* Timer 1 */
    ...
  case 3: /* Timer 2 */
    ...
  default:
    exit(1);
  }
  return (void*)timerSelect;
}
```

²⁸As of MDD v1.5.0, only ATmega16 and ATmega328P (=Arduino Uno) are supported. The code can easily be extended, but requires checking the data sheets in order to write to the correct bits.

```
#elif defined(TCCR0B)
  case 1: /* Timer 0 */
    TCCR0B |= clockSelectTable0[clockSelect
      -1];
    TCCR0A |= ...;
    break;
#endif
#endif
case 2: /* Timer 1 */
  ...
case 3: /* Timer 2 */
  ...
default:
  exit(1);
}
return (void*)timerSelect;
}
```

One of the AVR examples included in MDD is the single board heating system (SBHS²⁹), shown in Figure 10.



Figure 10. The single board heater system running a real-time control algorithm using firmware based on MDD code. There is a programmer attached to the board to upload new firmware, but the code runs without any computer connected to the SBHS.

The SBHS consists of a heater assembly, fan, temperature sensor, AVR ATmega16 microcontroller and associated circuitry. It was developed by IIT Bombay and is used for teaching and learning control systems (Arora et al., 2010). The MDD SBHS example uses pulse width modulation (PWM) blocks to control the heater and fan, and an analog-to-digital converter (ADC) block to read the temperature. It combines these elements with a PID controller with the goal to control the fan such that the temperature settles at a setpoint of 45 °C while a constant voltage feeds the heater assembly.

4.3 DLR Demonstrators

At the DLR Institute of System Dynamics and Control, several simulator systems utilize the MDD library for inter-system communication and querying of input devices.

The DLR Robotic Motion Simulator (Bellmann et al., 2011) is a 7-axis driving and flight simulator based on an industrial robot arm (see Figure 11). The main use of this motion simulator is the evaluation of input devices such as side-sticks, steering wheels, pedals, *etc.*, as well as the test and validation of control algorithms in terms of stability and real-time capability. The control architecture of the simulator uses blocks from the MDD library in several ways:

²⁹SBHS, <http://sbhs.fossee.in/>



Figure 11. The DLR Robotic Motion Simulator.

- Input devices such as force-feedback steering wheels are connected via CAN bus and integrated in the software framework via the CAN blocks; the same applies for a force-feedback side-stick.
- Other, consumer based input devices such as pedals or Airbus styled flight controls are connected via the JoystickInput block.
- The control architecture for the robot consists of two Modelica simulations on two different computers: First, the real-time path planning running on a real-time Linux system controlling the movements of the robot, and second, the control panel running on a standard Windows system. The control panel is used to change parameters such as washout filter modes (the washout filter maps the movement of road vehicles / airplanes to the workspace of the simulator) and gives an overview on the actual robot's position and telemetry. All real-time critical communication (e.g., the simulated road vehicle / airplane forces and angular velocities inputs for the real-time path-planning, or the control panel I/O) are communicated via the UDP blocks and the serial packaging system.

Figure 12 shows the inside of the simulator cabin. The instrumentation package can be adapted for different simulation types or for testing different input concepts. An on-board computer is used to query input devices, to display information on control screens, and to project the pilot's outside view visualization on the embracing concave dome shell. These tasks are performed using Modelica models, where the `SynchronizeRealtime` block is used for real-time synchronization. In addition, communication with the other simulation components is performed partly via the UDP blocks.



Figure 12. View into the simulator cabin of the DLR motion simulator. The instrumentation package is replaceable, so that the simulator cabin can be easily adapted for different simulation types, e.g., for driving or flight simulation.



Figure 13. DLR ROBEX technology demonstrator.

Figure 13 shows the ROBEX demonstrator which was developed as a technology demonstrator for a science exhibition. This demonstrator allows the user to command a rover on a scientific lunar mission. The mission's goal is to pick up a sensor package from a nearby lander and to place it on a marked position on the lunar surface. The user controls the rover via an Android App, which runs on a tablet computer in front of the simulator screen. On the screen, the visualization of the rover is displayed. The underlying Modelica simulation performs the multi-body simulation of the rover and utilizes the *DLR Visualization* library to display the rover and the scenery. It uses the UDP blocks to communicate with the tablet computer and the `SynchronizeRealtime` block to adjust the simulation speed.

In very similar ways, the library is also used in several other simulator and demonstrator systems, e.g., a drilling rig training simulator, several desktop flight simulators, or a rover software-in-the-loop development environment.

5 Outlook

The *Modelica_DeviceDrivers* library is a tried and tested library, which can support a wide range of application scenarios. During its development, valuable experience on interfacing Modelica with external C code has been gained. Thus, the source code can also serve as an example for anybody who is interested in applications, which require a more complex integration of Modelica code with external C code.

Considerable development efforts have been spent on improving the Modelica compliance of the library. Still, there are open issues and one may see the library as a testbed, which stresses Modelica's external function interface to the limit. On one hand, experiences gained thereby can provide inputs for further enhancements to the Modelica standard specification, on the other hand, further efforts in the library development can improve the level of standard-compliance. However, since backwards compatibility is a strong objective in the library development, non-backwards compatible changes for the sake of better standard-compliance will not be introduced lightly.

Naturally, there is a large pool of conceivable feature extensions to the library, due to the myriad number of available external devices and communication protocols. A frequent request is to extend the communication abilities beyond the capabilities of the available *SerialPackager*. There exists a huge choice of data serialization formats that could be utilized for this purpose (e.g., LCM or MessagePack). Particularly, with regard to the Internet of Things (IoT) technology becoming more important, improving communication capabilities is a worthy goal. Similarly, supporting embedded systems beyond the prototypical work is very attractive in that perspective.

Acknowledgements

This work has been supported by Vinnova in the ITEA3 OPENCPs projects, and in the RTISIM project. Support from the Swedish Government has been received from the ELLIIT project, as well as from the European Union in the H2020 INTO-CPS project. The Open Source Modelica Consortium supports the OpenModelica development.

Finally, the authors would like to thank everybody who has contributed to the library, either by providing feedback and suggestions, or by direct contributions to the implementation of the library, particularly, Miguel Neves, Dominik Sommer, Rangarajan Varadan, and Dietmar Winkler.

References

- Inderpreet Arora, Kannan M. Moudgalya, and Sachitanand Malewar. A low cost, open source, single board heater system. In *4th IEEE International Conference on E-Learning in Industrial Electronics (ICELIE)*, November 2010. doi:10.1109/ICELIE.2010.5669868.
- Tobias Bellmann. Interactive Simulations and advanced Visualization with Modelica. In Francesco Casella, editor, *7th Int. Modelica Conference*, Como, Italy, September 2009. doi:10.3384/ecp09430056.
- Tobias Bellmann, Johann Heindl, Matthias Hellerer, Richard Kuchar, Karan Sharma, and Gerd Hirzinger. The DLR Robot Motion Simulator Part I: Design and Setup. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4694–4701. IEEE, May 2011. doi:10.1109/ICRA.2011.5979913.
- Torsten Blochwitz and Thomas Beutlich. Real-Time Simulation of Modelica-based Models. In Francesco Casella, editor, *7th Int. Modelica Conference*, Como, Italy, September 2009. doi:10.3384/ecp09430119.
- Matthias Hellerer, Tobias Bellmann, and Florian Schlegel. The DLR Visualization Library - Recent development and applications. In Hubertus Tummescheit and Karl-Erik Årzén, editors, *10th Int. Modelica Conference*, Lund, Sweden, March 2014. doi:10.3384/ecp14096899.
- Albert S. Huang, Edwin Olson, and David C. Moore. LCM: Lightweight Communications and Marshalling. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, pages 4057–4062, October 2010. doi:10.1109/IROS.2010.5649358.
- Modelica Association. Modelica—A Unified Object-Oriented Language for Physical Systems Modeling v3.2. Standard Specification, March 2010. available at <http://www.modelica.org/>.
- Modelica Association. Modelica—A Unified Object-Oriented Language for Systems Modeling v3.3 Revision 1. Standard Specification, July 2014. Available at <http://www.modelica.org/>.
- Martin Otter, Bernhard Thiele, and Hilding Elmqvist. A Library for Synchronous Control Systems in Modelica. In Martin Otter and Dirk Zimmer, editors, *9th Int. Modelica Conference*, Munich, Germany, September 2012. doi:10.3384/ecp1207627.
- Niklas Worschech and Lars Mikelsons. A Toolchain for Real-Time Simulation using the OpenModelica Compiler. In Martin Otter and Dirk Zimmer, editors, *9th Int. Modelica Conference*, Munich, Germany, September 2012. doi:10.3384/ecp12076839.

modelica.university: A Platform for Interactive Modelica Content

Michael M. Tiller¹ Dietmar Winkler²

¹Xogeny, USA, michael.tiller@xogeny.com

²University College of Southeast Norway, dietmar.winkler@usn.no

Abstract

The World Wide Web was conceived of as a medium for the expression and exploration of scientific and engineering ideas. However, much of the innovation in web technologies is now focused on consumer facing applications. Although science and engineering content is available on the web (Wolfram Alpha, 2017), there are not that many tools that allow engineers and scientists to create and build scientific and engineering applications.

Fundamentally, HTML and HTTP are certainly sufficient for the creation of scientific and engineering content just as they are for the creation of online magazines and websites. But while a number of "content management systems" have been created to facilitate the publication of prose, there are very few such tools that cater to making it easy to create scientific and engineering content.

In this paper, we will present a platform which can be thought of as a content management system for scientific and engineering content. We will start by describing what we believe to be the fundamental requirements for such a system. From there, we will discuss two different applications built on this platform. The first is an interactive tutorial for teaching the basics of the Modelica languages and the other is an example application that involves creating interactive content for use in an engineering course on hydro-electric power generation. This content will be published on the `modelica.university` domain and we are already collaborating with others to contribute additional content to the site.

Keywords: Modelica, web, cloud, education, content management

1 Introduction

1.1 Background

The initial goal of this project was to recreate a previous application entitled "Tour of Modelica" using a newer platform for deploying web-based engineering tools and content. The previous version of the application was written to provide a "tool free" experience for learning the basics of Modelica. Similar efforts involving the OpenModelica tool OMNotebook have also been undertaken (Palanisamy et al., 2016).

Because the tutorial was web-based, it could be used as part of an interactive, introductory tutorial at events

like the North American Modelica Users' Group meetings without requiring participants to install tools. Furthermore, the only prerequisite was a browser. So, the tutorial was not just tool neutral, but OS neutral as well. During live events, the tutorial material was used by participants running Windows, MacOS and even iOS.

However, the tutorial was based on older infrastructure and the decision was made to upgrade the tutorial. At the same time, it was also decided to make the underlying platform available for others to create web-based educational content based on Modelica. The domain name `modelica.university` was registered for this new site.

1.2 Requirements

The underlying platform was created to support the creation of web-based engineering analysis tools. Many lessons from the creation of proprietary tools were factored into the design of the infrastructure that supports the deployment of these applications. In this section, some high level requirements for the platform (based largely on the experience of developing earlier tools) will be enumerated.

1.2.1 Hypermedia

The success of the web is, in part, due to the ability of hypertext to link together content from different sources. For most users and developers of web content, this is most typically associated with HTML (W3C, 2016).

However, it should be noted that the concept of hypertext has since been generalized to the more general term "hypermedia". The concept of hypermedia extends the idea of describing links and relationships not just between text and content within that text, but to data in general. In hypermedia, a URL is used to refer to a "resource". Those resources represent data of some kind and may have potentially multiple different potential representations (*e.g.*, an image resource could be represented as either a JPG or a GIF image). This modern conception of hypermedia and the use of hypermedia as an architectural style for building network based applications was formalized in (Fielding, 2000).

But in order to support this, formats besides HTML are required. This is because HTML is focused on being a declarative way to represent documents (hence the presence of elements like `` (image), `<h1>` (header)

and `<p>` (paragraph). But in order to generalize the approach to data, a whole range of new formats like HAL (Kelly, 2016), Collection+JSON (Amundsen, 2013) and Siren (Swiber, 2016) were developed.

The most essential aspect of these formats is that they allow generalized data (in most cases serialized as either XML (Maler et al., 2008) or, more commonly, JSON (ECMA International, 2011)) to express hypermedia concepts like relationships to other resources and/or actions that can be performed on these hypermedia resources.

At the dawn of the World Wide Web, hypermedia was recognized as an essential component for the expression and exploration of scientific and engineering ideas. Our experience shows that the power of applying hypermedia concepts to science and engineering is still not fully realized and our goal was to not only include it as a requirement for managing scientific and engineering content, but to exploit it even further than most existing platforms.

1.2.2 API

Nearly all web applications require some kind of API to interact with. Generally speaking, the two main functions of an API are to provide information and the carry out tasks. The term “Command Query Responsibility Segregation” (CQRS) refers to an architectural style where these two responsibilities are clearly and cleanly delineated (Fowler, 2011).

As such, it is no surprise that our API requires both of these functions. An API is generally just the “middle man” between the client (*e.g.*, the web application) and one or more sources of information leveraged by the server (*e.g.*, databases, file systems). The *query* functionality allows the web application to request information from those sources via the API. The *command* functionality allows the web application to request tasks to be performed by the server. The main difference between the command and query functionality is that queries are, generally speaking, idempotent, *i.e.*, they don not change the state of the server while the command functionality typically exists solely for the purpose of mutating the server side state. Furthermore, querying functionality generally relies on caching as an optimization to speed up the fetching of information and to ensure its “freshness” while commands frequently invalidate caches as a result of mutation.

For our purposes, we need querying functionality to provide us with text, images, models, simulation results, *etc.*. We need the command functionality mainly to request computational tasks like simulations and optimizations to be performed.

1.2.3 Content Creation

A significant impediment to web and cloud adoption in the world of science and engineering is the fact that there is not much overlap in technical skills between engineers and web developers. As such, engineers need to rely on web developers to help them with creation of web based tools. Of course, HTML is relatively easy. But to move be-

yond simple static markup requires a wider range of skills. Unfortunately, people with those skills tend to be drawn to more “consumer oriented” projects with the potential to reach very large markets (social networking, advertising, search engines, games, *etc.*). As a result, the rate of innovation and adoption in the engineering sector has traditionally been and continues to be slow.

In order to break this cycle, it is essential to develop technologies that make it easy to turn people with specialized scientific or engineering skills into content creators. Of course, this is nothing new. But, again, many of the development resources are focused on empowering broader sections of society and less on science and engineering.

In reducing the learning curve for non-experts, there are two important aspects to consider. The first is easing the creation of content. This means being able to easily make scientific and engineering content accessible through the APIs, *e.g.*, connecting the API to existing data sources or computational capabilities. The other aspect is the visualization of the underlying content in the web browser. For the purposes of this project, we require that both of these are facilitated to some extent.

1.2.4 Third Party Tools

While `modelica.university` is being hosted publicly, the infrastructure it is build on was developed to support proprietary tools and applications. Many of those applications are intended to be hosted on private networks. It is quite common that customers insist that all data remain on private networks. In those cases, it is impossible to rely on third party services hosted on the public Internet (*e.g.*, Amazon EC2, Google Cloud Platform, Digital Ocean).

So none of the software libraries used by the `modelica.university` infrastructure rely on services that are hosted exclusively on the public Internet. However the requirement to avoid public services was relaxed for this project to make deployment easier and more cost effective.

1.2.5 Job Processing

In our earlier discussion on APIs, we mentioned the need to perform “computational tasks”. But for scalability reasons, it is frequently important to delegate these computational tasks away from the API server. Without such delegation, the response of the API server itself could be slowed down considerably by CPU intensive tasks running on the same machine. Furthermore, numerical tools are often written in languages like FORTRAN, C++, Python, Julia, *etc.*, while web servers, databases and other back-end services are written in languages like Javascript, Java and so on. To address both the scalability and interoperability, it is often convenient to introduce message queues or worker queues. These provide a way to link together various services in a scalable way while avoiding the tendency toward monolithic architectures. The term “microservices” (Susan Fowler, 2016) refers to an architectural style which is very much aligned to these require-

ments.

2 Content Management Platform

Now that we have elaborated some of the requirements for the application, we will quickly review how we have addressed those requirements in our implementation.

2.1 Backend

The term “backend” refers to aspects of the application not handled in the web browser. This includes the web server that serves the application, databases, authentication, “memcache”, *etc.*

2.1.1 API

We start our discussion of the backend with the API itself. For `modelica.university`, we leverage the Heisman API framework. Heisman is a proprietary framework developed by Xogeny for creating hypermedia APIs. The main feature of this framework is the ability to define so-called “resources” using an intrinsically hypermedia-oriented structure. Once defined, an HTTP based API can automatically be synthesized for those resources. The emphasis on hypermedia semantics means that resources are able to easily express not just data about themselves but also relations to other resources as well as actions that can be performed by resources.

The fact that an HTTP based API can be automatically synthesized is important because it avoids having to write a great deal of boilerplate code to handle pedantic HTTP specific details like status codes, caching, etags, accept header processing and so on.

We have taken an “API first” approach to application development. As we will discuss shortly, once the resources are defined and the API is automatically generated, a generic API browsing application is already available for the API.

2.1.2 Resources

The resource oriented approach to application development means that resources need to be defined with hypermedia semantics in mind. Our definition of resources is largely inspired by the Siren hypermedia format. Specifically, a resource is described by three distinct types of information.

The first type of information a resource can provide is the “properties” of the resource. This is the true data associated with the resource. For example, if the resource represents results from a time-domain simulation, the “properties” might be the values of the independent and dependent variables.

The second type of information a resource can provide about itself is metadata. The metadata for a resource includes a textual description of the resource as well as zero or more textual “classes” that identify (in some domain specific way) what the resource represents. For example, if the resource represented simulation results, the set of textual classes might include the

string “`simulation_result`”. It may also include the name of a more specialized class, *e.g.*, a resource might include “`drive_cycle_result`” and “`simulation_result`” where the former is a specialized form of the latter.

The final, and arguably most important, type of information associated with a resource is “links”, which convey how one resource relates to other resources. The ability to “link” to other resources is the essence of hypermedia. The link between resources is always associated with one or more “relations”. Relations, like classes, are typically domain specific names although the Internet Assigned Numbers Authority (IANA) has defined a collection of standard link relations (Internet Assigned Number Authority, 2017). For example, the `item` relation is used to define the relationship between a (collection) resource and any other resource “contained” in it. Similarly, the `collection` relation may appear on each item resource to link back to the enclosing container resource.

2.1.3 Domain Specific Resources

The term “resource” is an abstraction used to refer to any kind of data that might be accessed over a network. To help understand what a resource is and how they relate to our application, we will provide several concrete examples for discussion in this section.

Static Content A very common type of resource is a file. In fact, web servers like the Apache or NGINX web servers treat files precisely as hypermedia resources by providing a way to refer to those files as network addressable streams of bytes. Heisman also provides a means to serve files as network addressable resources. However, in our application the contents of the file are only part of the resource. We also allow the metadata and link information to be associated with a file. Just by associating such information with the files, it becomes possible to quickly and easily define a rich range of structural information about the resources associated with an application. This hypermedia oriented information can be supplied within the file itself (by serializing it as a Siren instance) or programmatically via special handler routines registered with the server that add hypermedia annotations to those files.

This ability to annotate files with hypermedia information means that much of the content being managed by the content management system can be represented by files that are statically served directly from a file system. This capability is important because it helps us address the requirement that creation of content should be easy and intuitive for people who are not programmers or web developers. Using this functionality, much of the application can be built simply by dragging and dropping files into directories. We will demonstrate this further in the context of both applications discussed later. It is worth noting that content served from the filesystem is also much easier to version control vs. content stored in a database.

Dynamic Content In addition to static content, most applications depend on the ability to create and manipulate data dynamically in response to user actions. For example, each time a simulation is performed we might wish to store those simulation results away for retrieval later. In some cases, we might want a resource to represent a very specific type of data (*e.g.*, simulations performed by a given user) with specific fields (*e.g.*, model simulated, user who requested the simulation, time request was made, time required to complete the simulation). In other cases, we might require a way to create, manipulate and query arbitrary (schema free) data. While the former often requires specialized resources to be created, Heisman provides a standard collection resource to handle the latter.

Job Brokers The final resource type used in these applications is essential for handling requests for computational work. In both applications, the computational work required is running simulations. Because nearly every scientific or engineering application will require one or more types of computationally intensive analyses, Heisman includes already implemented resources called “job brokers”. These job brokers provide an API for requesting work to be done, tracking the status of that work and reporting back the successful result or an error message. The code is independent of the task to be performed. This means that a job broker can be easily created and associated with one or more specific computational tasks required by the application.

The hypermedia semantics allow us to cross reference job requests with job results. In other words, for a given simulation result we can follow the links associated with that result to find the original request and vice-versa. Such cross referencing of resources can be used for traceability and to determine provenance of data.

2.2 Communication

The capabilities described so far rely on several different communication mechanisms. In this section we will quickly summarize each of these.

The web application running in the browser relies on hypertext transfer protocol (HTTP) (Fielding et al., 1999) for invoking queries and commands. These HTTP requests are received and acted upon by code on the server that maps these requests to the underlying resources referenced in the requests.

The “job broker” resource uses a tool called Redis (Sanfilippo and Noordhuis, 2017) to implement message and worker queues. It is via Redis that messages are sent between the API server and the workers that perform any CPU intensive computations.

2.3 Deployment

Desktop tools are typically compiled into binaries and distributed via “installers”. In contrast, web applications are deployed (often, continuously) to servers where they can then be accessed via a web browser. This simplifies the install process for the user (since they only have to enter a

URL in a web browser), but the process of deploying software to these servers safely and efficiently adds a whole new dimension to the software development process¹

An important technology for the deployment of network services is called “Docker”. Technically, Docker is a tool designed to make it easy to access the special Linux process groups called “containers”. But this explanation does not adequately explain Docker’s role or capabilities.

Conceptually, Docker is a technology for creating extremely resource efficient virtual (Linux) machines. The efficiency comes from Docker’s use of kernel level features in Linux that isolate groups of processes while allowing them to share large amounts of read only data in memory and/or on the file system.

The backend server for `modelica.university` is a Node (Node.js Foundation, 2017) application written in TypeScript (Microsoft, 2017). To generate a Docker image, the `dockergen` Node package (Tiller, 2017) is used. The `dockergen` script creates a `Dockerfile` which specifies how the application should be packaged for deployment to a Docker host. Once a Docker image is built, it can be run as a container on a Docker host. Since this is a public application, we can take advantage of commercial Docker hosting services.

The actual application is made up of several distinct Docker images executed using the “compose” functionality of Docker. In addition to the API server image, the backend consists of several other images. One image runs the Redis server. Another image runs a NGINX web server to act as a reverse proxy. A third image runs the API server. The final image executes the workers for the computational tasks processed via the worker queue. With Docker, it is quite simple to activate multiple containers running the worker image. This allows us to easily scale up the number of workers during periods of high load. Another advantage of Docker that all the machines in a cluster are securely firewalled within the same network. Only ports that have been explicitly opened to machines within the cluster are accessible outside the cluster.

3 Application 1: Tour of Modelica

3.1 Objective

Now that we have discussed how the underlying infrastructure is implemented, let us get into the details of the first application. As mentioned previously, the “Tour of Modelica” application is a reimplement of an earlier web application. The application is structured in the form of chapters and lessons. In each lesson, the user is presented with some introductory material about a specific aspect of the Modelica language and starting from some sample code is asked to carry out several modeling tasks. After completing the exercises, the user moves on to the next lesson and/or chapter.

¹So much so, that the term “DevOps” was coined to refer to the combined set of development and operational skill required to deploy web applications.

To complete each task, the user must be able to edit, compile and simulate Modelica code. The code editing is done in the browser, but the compilation and simulation is requested via the API and performed by a worker that uses OpenModelica (Open Source Modelica Consortium, 2016) to compile and simulate each model.

3.2 Content

The content for the application consists primarily of lessons, chapters, lesson text and sample models. All of these can be represented as static resources using the functionality previously discussed in 2.1.3.

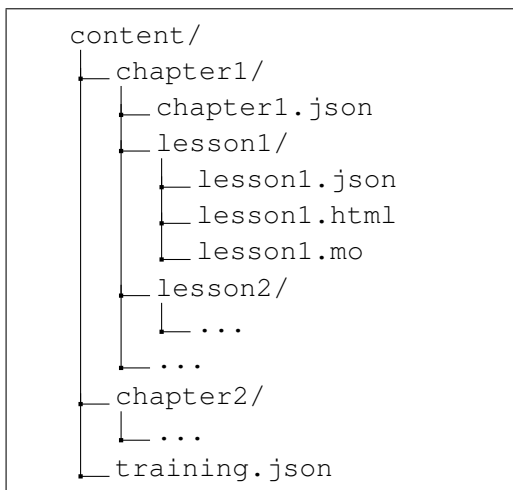


Figure 1. Fragment of the files system.

A fragment of the file system content is shown in Figure 1. All content is rooted in a directory named `content`. The files are organized by chapters and lessons although this is strictly a convention. Files ending in the `.json` suffix are interpreted as hypermedia resource descriptions. These JSON files contain the metadata, properties and links discussed previously. Let us look at the `lesson1.json` file to as an example of how one such resource might be described:

```

{
  "title": "Simplest Model",
  "properties": {},
  "class": ["lesson", "start"],
  "links": [
    { "rel": ["text"],
      "href": "../lesson1.html" },
    { "rel": ["source"],
      "href": "../lesson1.mo" },
    { "rel": ["task"],
      "href": "resource://simulate" },
    { "rel": ["chapter"],
      "href": "../chapter1.json" },
    { "rel": ["training"],
      "href": "../../training.json" }
  ],
  "query": {
    "rel": {
      "training/*": { "embed": true },
      "chapter/*": { "embed": false },

```

```

      "source/data": { "embed": false },
      "text/data": { "embed": false },
      "task": { "embed": true }
    }
  },
}

```

From this description, we can see that this resource is titled “Simplest Model” and has no properties. Because this resource is a lesson, we include the `lesson` class in its description. It also has the `start` class which we can use in our application to locate the first lesson. The `links` section provides (respectively) links to the HTML markup for the lesson text, the initial model source, the job broker that will run the simulation, the chapter that this lesson belongs to and the `training.json` file which describes all the chapters that are part of the “Tour of Modelica” application. The `query` section describes what information about the resource should be returned from each HTTP request². By default, all resources have a “default query” that describes what information about that resource is to be returned for each HTTP request. The `query` section here is defining the default query. Note that clients (*e.g.*, our web application) are free to specify their own query with each request. In this way they can request more or less information to be provided, depending on their needs.

This is a lot of information. Furthermore, nearly all of it is essentially repeated from one lesson to the next where only a few details are changed. Fortunately, Heisman provides a way for us to programmatically augment the contents of resources represented by files on the file system. In this way, we are able to write code to automatically fill in all the information based conventions like the directory structure or the lesson name. In fact, the only thing we cannot figure out automatically is the title. As a result, the task of creating a new lesson resource becomes as easy as creating a file that contains:

```

{
  "title": "Simplest Model"
}

```

A similar process is used to augment information about other types of content on the file system (*e.g.*, chapters). This relatively small amount of upfront work to define specialized handlers greatly simplifies the process of content creation and making the process accessible to non-programmers. In addition, allowing data to describe its relationship to other data means that that information and logic does not need to be coded into the client. This makes development of the client easier and more general.

3.3 Visualization

3.3.1 Generic Browser

There are many aspects about the operation of a web browser that most users are not aware of. One of those

²In our API, the primary response content type is Siren. Because Siren allows related resources to be embedded in a response or simply linked to, our query format must specify which approach to use for each matching resource. Hence the `embed` field.

aspects is the `Accept` header. This is a header included with an HTTP request that lets the server know what types of content it expects back. The default `Accept` header for Google Chrome looks like this:

```
Accept: text/html,application/xhtml+xml,
application/xml;q=0.9,image/webp,*/*;q=0.8
```

This is essentially a list of content types the browser understands. But it also defines the clients order of preference for the different content types. The `Accept` header is useful to the server because it is possible that a given resource could be represented in multiple formats and the `Accept` header provides a clue as to which format is preferred.

The `Accept` header is important in API development because it can be used to determine whether the request that the API is handling is coming from a browser or from Javascript code. If our server sees that the request is for HTML, it will respond to the request by serving up a page that loads an embedded browser application. That web application is actually a generic graphical user interface for Siren APIs that comes bundled with the server. We will talk about the user interface application in greater detail shortly.

This is part of the “API first” philosophy discussed earlier. As a result of following this philosophy, every API developed in this way automatically comes with a graphical user interface. Furthermore, remember that Heisman automatically synthesizes an HTTP API based on the resources that are registered with it. What this means, in practice, is that once you describe your resources, *you immediately and automatically get both an HTTP API and a web application*.

3.3.2 Custom Visuals

As mentioned previously, Simran is the web application that is launched when browsing the API. Simran is a proprietary technology used by Xogeny to create web based UIs for scientific and engineering applications.

Simran is really a browser running in a (web) browser. Generally speaking, web browsers like Chrome or Firefox are used for browsing HTML or other widely used content types. If you are a scientist or engineer, the problem is that web browsers do not understand more technical formats (e.g., Modelica models, `.mat` files, FMUs).

The API browsing application compensates for this by providing a web application that is extensible. Because the browser application is built around the notion of hypermedia (primarily in the form of Siren representations) and not hypertext (i.e., HTML), we can represent many different content types *and* the relationships between them. In a sense, this is a lower level alternative to HTML.

That, by itself, may not sound that useful. But it becomes more useful because of the plugin system. Via the plugin API, it is possible to extend the browsing application with any number of specialized visual components. While the base browser application is a generic browser

that renders all Siren resources essentially the same, when enhanced via plugins the browser application is able to provide custom rendering for different content types based on the metadata, properties or relations of the resource.

For example, using just the base browser, our “Tour of Modelica” application is shown in Figure 2.

There we can see the first lesson and its related resources rendered using metadata. Furthermore, we can click on links to follow the various resources. But each resource will be visualized in the same generic way. However, after we provide a plugin with custom visuals for lessons and chapters, putting the **same URL** in our web browser will yield a rendering of the lesson like the one shown in Figure 3.

The plugin system is based on React (Facebook, 2017). Normally, each React component independently specifies what “properties” it understands when instantiating a component. We turn this around a bit and standardizes these properties to conform to a canonical representation of a hypermedia resource. As a result, all React components are “equivalent” in the sense that they are instantiated with the same set of properties but with different values. But, through the plugin system we have the freedom to customize *which* component to use for each hypermedia resource. In this way, we are essentially creating a browser that can easily be extended to understand any kind of scientific or engineering content instead of being limited to just those standardized in the HTML specification by the W3C.

In the case of the “Tour of Modelica” site, the plugin defines custom renderers for lessons, chapters and the training overview. In addition, it leverages some standard and easily reusable visuals provided by the built in browser for applications and application suites.

For each application, the application developer can decide what types of content the browser should be capable of understanding and then simply add those visuals to their plugin. This modular approach to visualization makes it very easy to create a custom user interface for a particular domain and/or reuse components developed for other applications.

The authors would like to acknowledge the contribution of the `moijs` project for providing syntax highlighting and checking for the embedded Modelica editor as well as the CodeMirror project (Haverbeke, 2017) for the editor widget itself.

3.3.3 Mobile

Consumers of web applications and web content are increasingly consuming this content from mobile devices. Support for phones and tablets mainly involves making sure that layout of content makes sense for small form factor screens. In some cases, some content may be hidden on small displays. With `modelica.university` we have made every effort to support mobile devices.

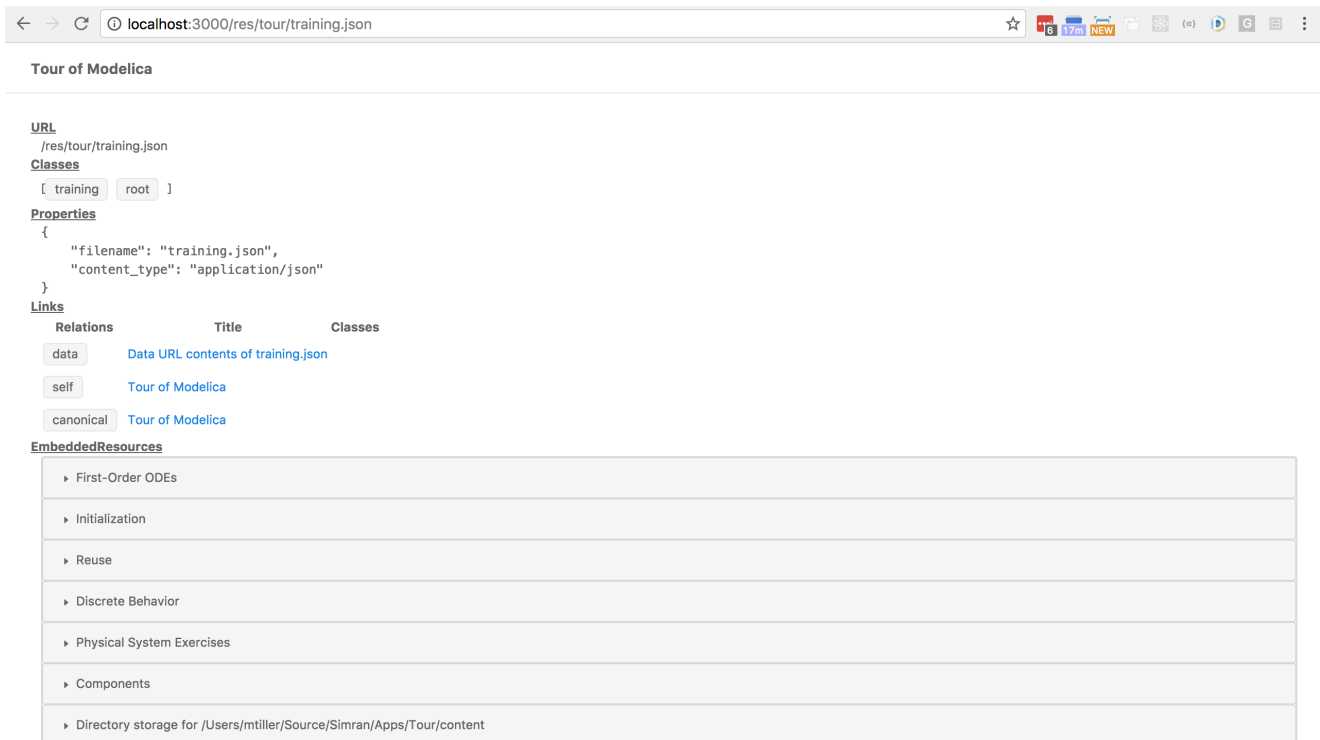


Figure 2. Generic rendering of Tour resources

4 Application 2: Hydro-Electric Power System

4.1 Objective

The second application example is a student exercise that is part of the Master's course "Object-oriented Modelling of Hydro Power Systems" at University College of South-east Norway. The course starts with an introduction to the fundamentals of Modelica. Later on it moves on to model specific parts of a hydro-electric power system.

Typical modeling problems are:

- Waterway configuration
- Water hammer investigations
- Droop control behavior of the turbine governor

Being able to solve such problems interactively using only the browser as a tool without having to immediately understand Modelica code improves the physical understanding of the system. Once the physical understanding is there, creating more complex models and scenarios is easier for the students to achieve.

4.2 Content

The contents of this application are the different main problems and each with multiple configurations. For example, for the *Waterway* application different examples with a number of interconnecting pipes are given where the levels of the pipe ends need to be verified and checked

that they make sense. This is sometimes not as easy as it sounds since pipes might connect to reservoir models which have a different height reference. So the student is given a set of parameters for the different pipe segments of other components of the water way and has to determine if the setup "makes physical sense".

For the *Water hammer* problem, one can investigate the influence of closing time of a valve depending on the pipe diameters and flow rates. The content would also provide certain restrictions like allowable maximum pressure in the pipes.

The *Droop control* (Wikipedia, 2017) problem contains data that describes the droop settings of one or more turbine controllers and lets one investigate the respective frequency dependent power productions.

The typical data structure of the content is shown in Figure 4.

4.3 Visualization

The real benefit for the second application will be the visualizations of the problems and especially solutions.

The *Waterway* problem is much more intuitively solvable when the students is presented with a sketch of the physical setup of the different pipe levels and other waterway components. Here the student can at once see a possible flow in the parameter set.

For the *Water hammer* problem a different method of visualization can be used. For example interactively showing unsuitable closing times by emphasising the pressure plots of setups that violate the restrictions. As the student changes parameters live (e.g., via a slider), they get the plot

Figure 3. Custom rendering of Tour resources

results presented live based on a real simulation done in the background. The executed models can be supplied as Modelica source files or FMUs.

The *Droop control* problem can be visualized by providing interactive droop setting behaviours including limits and again reacting on parameters that can be interactively set.

Figure 5 shows a typical plot of the power sharing behavior of three generators with different droop settings.

5 Related Efforts

The pace of innovation in the web development landscape is breathtaking. It is nearly impossible to keep track of all the new technologies that emerge almost on a daily basis. The authors drew inspiration from many amazing projects, including:

- **Jupyter** A tool for interactive data science and scientific computing across all programming languages (Project Jupyter, 2017)
- **Nextjournal** - An interactive writing and programming environment for every stage of research from experimentation to publication (Nextjournal, 2017)
- **"What Can a Technologist Do About Climate Change? (A Personal View)"** - Bret Victor's sprawling essay on technologies that can help address climate change (Victor, 2015).

- **Modelica in Action** - An interactive notebook for compiling and simulating Modelica (Bonvini, 2017).
- **Modelica by Example** - An interactive book about Modelica (Tiller, 2016).

6 Conclusions

By leveraging the power of hypermedia and a wide array of open source technologies, we were able to build the `modelica.university` site and our two sample applications. We gained several insights as a result of this work.

6.1 Middleware

Creating a site like this involves creation of the underlying content, implementation of the necessary analysis capabilities, an HTTP API and a domain specific web application to support user interaction. But most of the domain specific work here is at the edges, *i.e.*, content creation and visualization. Through their API synthesis and browser architectures, the Heisman and Simran packages allow development resources to remain focused on those domain specific edges. This adds efficiency to the development process while providing a tremendous amount of reusability. Together, these two packages form the foundation of Xogeny's *Aperion* platform.

6.2 Current Status

At this point, `modelica.university` implements the two applications described in this paper. Our experiences

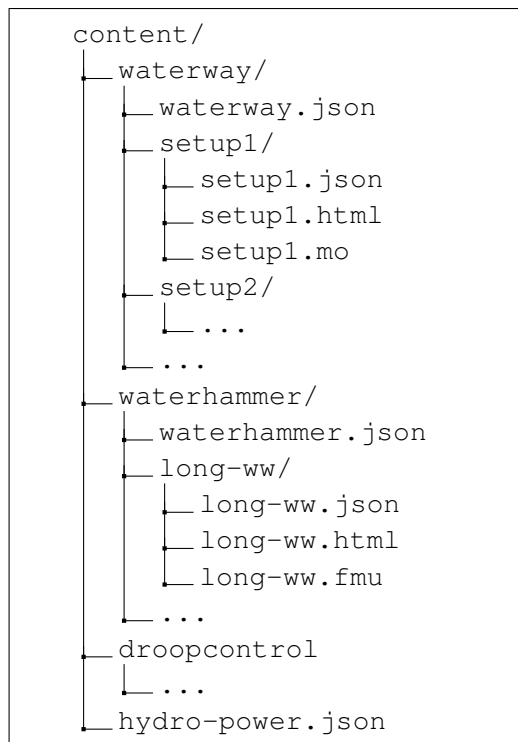


Figure 4. Fragment of the file system.

with these applications further reinforces the importance of the requirements outlined at this start of this paper. We are confident that with each additional application, the platform will gain more and more capability as a browser for scientific and engineering content.

6.3 Future Plans

In terms of content, we hope that others will contribute more content in diverse subject areas to help us further validate our approach, refine our requirements and, ultimately, provide meaningful educational content for science and engineering students.

As for the platform, we feel its further development will be largely driven by use cases involving model-

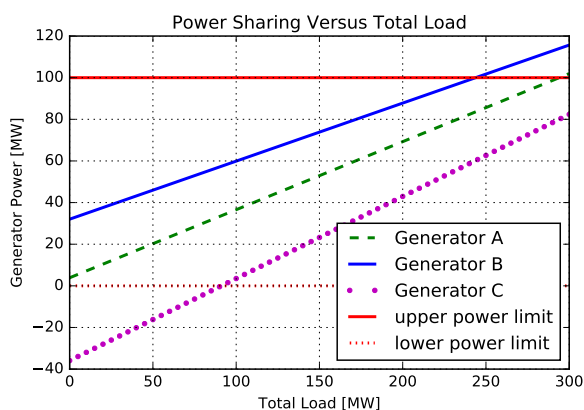


Figure 5. Example of a droop control visualization

ica.university and other proprietary projects. Now that the basic pieces of the architecture are implemented, there are countless optimizations we would like to make to improve responsiveness. There are also many types of content we would like to provide custom visualizations for (e.g., time series data, version trees, diagram authoring).

References

Michael Amundsen. Collection+JSON - Hypermedia Type, 2013. URL <http://amundsen.com/media-types/collection/>.

Marco Bonvini. Modelica in action: compile and simulate models, 2017. URL <http://marcobonvini.com/modelica/2017/01/02/modelica-in-action.html>.

ECMA International. *Standard ECMA-262 - ECMAScript Language Specification*. 5.1 edition, June 2011. URL <http://www.ecma-international.org/publications/standards/Ecma-262.htm>.

Facebook. React - v15.4.2, 2017. URL <https://facebook.github.io/react/>.

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol - HTTP/1.1, 1999. URL <https://tools.ietf.org/html/rfc2616>.

Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, 2000. AAI9980887.

Martin Fowler. CQRS, 2011. URL <https://martinfowler.com/bliki/CQRS.html>.

Marijn Haverbeke. CodeMirror, 2017. URL <https://codemirror.net/>.

Internet Assigned Number Authority. About Us, 2017. URL <http://www.iana.org/about>.

Michael Kelly. JSON Hypertext Application Language, 2016. URL <https://tools.ietf.org/html/draft-kelly-json-hal-08>.

Eve Maler, Tim Bray, Jean Paoli, François Yergeau, and Michael Sperberg-McQueen. Extensible markup language (XML) 1.0 (fifth edition). W3C recommendation, W3C, November 2008. <http://www.w3.org/TR/2008/REC-xml-20081126/>.

Microsoft. TypeScript - Javascript that scales, 2017. URL <https://www.typescriptlang.org/>.

Nextjournal. Nextjournal, 2017. URL <https://nextjournal.com/>.

Node.js Foundation. About Node.js, 2017. URL <https://nodejs.org/en/about/>.

Open Source Modelica Consortium. Openmodelica, December 2016. URL <https://openmodelica.org/>.

- A. Palanisamy, M. Sjölund, and P. Fritzson. Generating OpenModelica Web Books Including Mathematical Typesetting from OMNotebooks, 2016. URL <http://www.modprod.liu.se/filarkiv/1.672879/OpenModelica2016-talk15-Arunkumar-GeneratingOpenModelicaWebbook.pdf>.
- Project Jupyter. Project Jupyter, 2017. URL <http://jupyter.org/>.
- Salvatore Sanfilippo and Pieter Noordhuis. Redis, 2017. URL <https://redis.io/>.
- Susan Fowler. *Production-Ready Microservices: Building Standardized Systems Across an Engineering Organization*. December 2016. URL <http://shop.oreilly.com/product/0636920053675.do>.
- Kevin Swiber. Siren: a hypermedia specification for representing entities, 2016. URL <https://github.com/kevinswiber/siren>.
- Michael M. Tiller. Modelica by Example, 2016. URL <http://book.xogeny.com/>.
- Michael M. Tiller. Generate a Dockerfile for any NodeJS application, 2017. URL <https://www.npmjs.com/package/dockergen>.
- Bret Victor. What Can a Technologist Do About Climate Change? (A Personal View), 2015. URL <http://worrydream.com/ClimateChange/>.
- W3C. HTML 5.1, 2016. URL <https://www.w3.org/TR/html/>.
- Wikipedia. Droop speed control, 2017. URL https://en.wikipedia.org/wiki/Droop_speed_control.
- Wolfram Alpha. Wolfram Alpha, 2017. URL <https://www.wolframalpha.com/web-apps/>.

Object-oriented modelling of a flexible beam including geometric nonlinearities

Davide Invernizzi¹ Bruno Scaglioni² Gianni Ferretti² Paolo Albertelli³

¹Politecnico Di Milano, Dipartimento di Scienze e Tecnologie Aerospaziali
Via La Masa 34, 20156 Milano, Italy

²Politecnico Di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria DEIB
Via Ponzio 34/5, 20133 Milano, Italy

³Politecnico Di Milano, Dipartimento di Meccanica, Via La Masa 1, 20156 Milano, Italy

Abstract

In this paper, an efficient approach for the modelling and simulation of slender beams subject to heavy inertial loads is presented. The limitations imposed by a linear formulation of elasticity are overcome by a second order expansion of the displacement field, based on a geometrical exact beam model. In light of this, the nonlinearities of the elastic terms are shifted as inertial contributions, which yields an expression of the equations of motion in closed form. Thanks to the formulation in closed form, the proposed model is implemented in Modelica, with particular care to the suitability of the model with respect to the Modelica Multibody library. After describing the model formulation and implementation, the paper presents some simulation results, in order to validate the model with respect to benchmarks, widely adopted in literature. In the context of modern multi-domain modelling, the modular and object oriented approaches are the state-of-the-art paradigms upon which complex models are built. In this respect, multibody dynamics is frequently only one of the domains involved, nevertheless several real-world applications can be found where multibody modelling plays a crucial role in the design of systems, analysis and model-based control architectures. In this framework, modelling techniques and tools have evolved towards the insertion of flexible bodies into the models (MSC Software Corporation, 2017; Claytex Services Ltd; Dymore Solutions, 2016; Spacar, 2016; Heckmann et al., 2006; Ferretti et al., 2014).

Flexible multibody systems can be divided in two main branches according to the *linear* or *nonlinear* constitutive laws employed to model flexible elements. In the first case, the strain-displacement relationships are assumed to be linear and strain components to remain small. Nevertheless, several occurrences can be found where elastic bodies may undergo large overall motion while strains are kept small. Traditionally, linear elasticity has been accounted for using the so called *floating frame of reference approach* (short, FFR), which is natural way to include flexibility in the rigid multibody framework (Shabana, 1998). Indeed, the displacement field is

decomposed as the sum of an arbitrary large motion of a suitably selected frame, superposed to an elastic displacement field which is assumed to be small with respect to the overall motion. Thus the elastic displacement field may be computed accurately through a modal expansion, which is extremely efficient from a computational point of view. Furthermore, component mode synthesis techniques, like the well-known Craig-Bampton method (Craig and Bampton, 1968), has been widely adopted in multibody simulation tools and specifically in the context of Modelica, both in commercial (Claytex Services Ltd; Heckmann et al., 2006) and open-source (Ferretti et al., 2014; Bascetta et al., 2015) libraries. By means of this technique, complex geometry can be included in the analysis even though an external finite element modeling tool is required. Although the concept of floating frame is simple, in practice there are several issues to be handled. The selection of the floating frame is not unique and accuracy of the results is strongly affected by the choice of the modal basis (Schwertassek et al., 1999a), (Heckmann, 2010). Furthermore, the use of linear elasticity may lead to erroneous results when the inertial contribution of the floating frame motion is large enough to produce high loads in bodies with high stiffness. A well-known example is the rotating beam around a fixed axis: the coupling among the axial, flexural and torsional motion, neglected by the linearized theory, is crucial in order to correctly predict the behavior of the structure (Berzeri and Shabana, 2002; Sharf, 1995; Ligris et al., 2008; Absy and Shabana, 1997). On the other hand, the flaws of this theory pushed the multibody community toward the development of new approaches, closely related to the nonlinear finite element method (Géradin and Cardona, 2001). In this case the domain is divided in sub-domains or finite elements which are connected at nodes to ensure elements compatibility, the nodal displacements and rotations are referred to a common frame, which is selected as the inertial frame. From a theoretical point of view this approach is challenging when considering structural elements such as beams, shells, plates and in particular when large displacements and rotations are considered. *Geometrically exact models* for these elements have been developed in

the last decades (Pai, 2007) and are the state-of-the-art to deal with large displacements small strains problems. Nonetheless special care is required for the computation of elastic terms and for the interpolation scheme of spatial rotations in a finite elements framework (Jelenić and Crisfield, 1998). Moreover, a large number of degrees of freedom is required to obtain accurate solutions and the expression of the elastic contribution is highly nonlinear even in the case of small strains. The use of such approach within the Modelica multibody library is difficult because the closed form expression of the discretized equations of motion is not manageable even for simple elements like beams.

Within the FFR method, several approaches have been proposed in the reference literature that overcome the shortcomings of the standard linear approach by accounting for geometrically nonlinear effects (Wallrapp and Wiedemann, 2003; Bremer, 2008; Banerjee, 2016). As already mentioned, the classic linear approach may provide erroneous results due to *a-priori* linearization of the kinematics and of the elastic energy, even if the strains are small. This problem has been deeply studied (Absy and Shabana, 1997) and different techniques have been developed to consistently linearize the equations of motion, so that all the relevant terms are retained while keeping the standard linear elastic terms.

In this work, a general framework to include geometrically nonlinear effects within the FFR approach is presented. The approach is based on a second order expansion of the displacement field, which can be derived from geometrically exact models of simple structural elements. Then, the displacement field is written in terms of a set of generalized deformation variables for which the elastic terms are linear. Thanks to this substitution, the standard linear elastic theory can be exploited and the nonlinearities are expressed as inertial contribution, which can be computed in closed form. Hence, the existing FFR formulation can be employed with minor changes, which is particularly efficient when small elastic deformations are expected: few degrees of freedom are usually required.

The proposed formulation has been applied to slender structural elements which are usually modeled as beams. In the standard approach, a linearized model is adopted to describe the deformation field in the floating frame, such as the Euler-Bernoulli or the Timoshenko beam model. This approach greatly simplifies the computation of the elastic terms but limits the correctness of the results by neglecting nonlinear effects, *e.g.* the geometric stiffening induced by the centrifugal acceleration for fast rotating beams. Within the Modelica framework, the standard linear approach has been implemented in (Schiavo et al., 2006), while in (Heckmann et al., 2006) a second order approximation of the deformation field is presented together with a Ritz-Galerkin discretization (Ritz, 1909). With respect to (Heckmann et al., 2006), in this work the deformation field is expanded starting from a geometrically exact description of the beam kinematics ((Schwertassek and

Wallrapp, 1999)) and the model is discretized according to a finite element approach. Finally, the Craig-Bampton reduction is applied to obtain a computationally efficient set of equations. The model is implemented in Modelica following an approach similar to (Ferretti et al., 2014) and two validation benchmarks taken from literature are reproduced, showing trustworthy agreement between simulation results and literature data.

The paper is organized as follows: In section 1 the modelling framework is described for the generic flexible body and the equations of motion are formulated. Section 2 goes into details of the beams by describing the mathematical formulation pointed out in the previous section. In section 3 the implementation and the simulation results are described. In particular, the results are compared with two well known literature benchmarks. Section 4 concludes the paper.

1 Equations of motion of a flexible body

Within the FFR approach, the absolute position \mathbf{p} of a generic point of the flexible body is composed by the sum of three contributions

$$\mathbf{p} = \mathbf{r} + \mathbf{u}_0 + \mathbf{u}_f, \quad (1)$$

where \mathbf{r} is the vector describing the position of the reference frame $\{\mathbf{O}_i, x_i, y_i, z_i\}$ with respect to the inertial frame $\{\mathbf{O}_w, x_w, y_w, z_w\}$, \mathbf{u}_0 is the undeformed position of the point with respect to the local reference frame and \mathbf{u}_f is the deformation field, as shown by Figure 1. The components of \mathbf{u}_0 resolved in $\{\mathbf{O}_i, x_i, y_i, z_i\}$ are named *material coordinates*.

In order to obtain the equations of motion, the *principle of virtual work* is exploited, *i.e.*:

$$\delta W_e = \delta W_i \quad (2)$$

where δW_e and δW_i are the external and internal virtual works. According to the FFR approach, the virtual displacement related to (1) can be computed as follows:

$$\delta \mathbf{p} = \delta \mathbf{r} + \delta \boldsymbol{\phi} \times (\mathbf{u}_0 + \mathbf{u}_f) + \delta \mathbf{u}_f \quad (3)$$

where $\delta \boldsymbol{\phi}$ is the virtual rotations vector of $\{\mathbf{O}_i, x_i, y_i, z_i\}$ while $\delta \mathbf{u}_f$ is the virtual variation of the deformation field with respect to the local reference frame of the body.

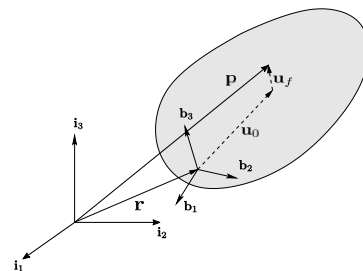


Figure 1. Flexible body reference frames

The external and internal virtual works for a generic flexible body can be defined as:

$$\delta W_e = \int_V \delta \mathbf{p} \cdot (-\rho \mathbf{a} + \mathbf{f}) dV + \int_{A_N} \delta \mathbf{p} \cdot \mathbf{t} dA \quad (4)$$

$$\delta W_i = \int_V \delta \mathbf{B} : \mathbf{J} dV \quad (5)$$

where ρ is the density of the body, $\mathbf{B} \in \mathbb{R}^{3 \times 3}$ and $\mathbf{J} \in \mathbb{R}^{3 \times 3}$ are the Jaumann strain and stress tensor respectively (see (Pai, 2007) for more details). By using the indicial notation, the double inner tensor product ":" is defined by:

$$\int_V \delta \mathbf{B} : \mathbf{J} dV = \int_V \sum_{i=1}^3 \sum_{j=1}^3 \delta B_{ij} J_{ij} dV. \quad (6)$$

Moreover, \mathbf{f} is the body force per unit volume and \mathbf{t} the surface traction per unit area, V is the volume of the body and A is the unconstrained portion of the body surface.

It must be pointed out that the Jaumann strain tensor is related to the deformation gradient $\mathbf{F} \in \mathbb{R}^{3 \times 3}$ as follows (see (Hodges, 2006)):

$$\mathbf{B} = \mathbf{U} - \mathbf{I} \quad \mathbf{U}^2 = \mathbf{F}^T \cdot \mathbf{F} \quad (7)$$

where \mathbf{I} is the identity tensor and \mathbf{U} is the right stretch tensor.

It must be pointed out that the Jaumann strains are an objective strain measure suitable for large displacements-small strains analysis since they are co-rotated engineering strains. As a consequence, in a linear elastic framework the reduced material stiffness matrix can be derived from standard experiments (Pai, 2007). The term of δW_e relative to inertial virtual work can be expanded by substituting (3) in (4), thus obtaining:

$$- \int_V \delta p \cdot \rho (\dot{\mathbf{v}} + \boldsymbol{\omega} \times \mathbf{v} + \dot{\boldsymbol{\omega}} \times (\mathbf{u}_0 + \mathbf{u}_f) + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times (\mathbf{u}_0 + \mathbf{u}_f)) + \ddot{\mathbf{u}}_f + 2\boldsymbol{\omega} \times \dot{\mathbf{u}}_f) dV \quad (8)$$

where \mathbf{v} and $\boldsymbol{\omega}$ are the body translational and angular velocities of the FFR.

In the classic linear approach, the deformation field measured in the FFR is assumed to be infinitesimal such that the computation of the internal virtual work can be approximated with the standard linear theory:

$$\int_V \delta \mathbf{B} : \mathbf{J} dV \approx \int_V \sum_{i=1}^3 \sum_{j=1}^3 \delta \varepsilon_{ij} \sigma_{ij} dV \quad (9)$$

where $\varepsilon_{ij} = \frac{1}{2} (F_{ij} + F_{ji}) - \delta_{ij}$ is the infinitesimal deformation tensor, and σ_{ij} the conjugated stress tensor, both resolved in the undeformed basis $\{\mathbf{O}_i, x_i, y_i, z_i\}$. The corresponding components of the deformation gradient are:

$$F_{ij} = \delta_{ij} + \frac{\partial u_{fi}}{\partial u_{0j}} \quad (10)$$

where δ_{ij} are the components of the identity tensor:

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}. \quad (11)$$

As already mentioned in the introduction, the classic linear approach may lead to erroneous results since several terms are *a-priori* neglected. Instead, a consistent approximation of (5) is based on the decomposition of the deformation gradient such that

$$B_{ij} \approx \frac{1}{2} (F_{ij} + F_{ji}) - \delta_{ij} \quad (12)$$

where $F_{ij} = \sum_{k=1}^3 R_{ik} F_{kj}$ and R_{ik} is the rotation matrix describing the orientation of a suitable frame with respect to the local reference frame. Under the small strains assumption, the aforementioned frame can be selected such that $\hat{F}_{ij} \approx \delta_{ij}$, even if displacements are large (see (Bauchau et al., 2016) for more details).

Within this framework, a second order approximation of the deformation field is considered in this paper. In particular, it is possible to operate a change of variables such that elastic forces can be derived from standard linear theory whereas geometrical effects are computed as an inertial contribution. In order to perform such change, the following assumptions are introduced:

- the deformation field \mathbf{u}_f depends on a finite set of *physical* deformation functions $\mathbf{d}_p = \mathbf{d}_p(\mathbf{u}_0)$, i.e., $\mathbf{u}_f = \mathbf{u}_f(\mathbf{d}_p, \mathbf{u}_0)$. Physical deformations depend only on a reduced set of material coordinates \mathbf{u}_0 , e.g., in beam models, the reference axis displacements and the cross-section rotation angles are functions of the reference axis coordinate alone;
- the physical deformations \mathbf{d}_p can be expressed in terms of generalized deformation functions \mathbf{d}_g , i.e., $\mathbf{d}_p = \mathbf{d}_p(\mathbf{d}_g, \mathbf{u}_0)$, such that the deformation gradient \mathbf{F} (and thus elastic forces) are linear in \mathbf{d}_g when strains (but not displacements) are small;
- the nonlinear relation $\mathbf{u}_f = \mathbf{u}_f(\mathbf{d}_p(\mathbf{d}_g), \mathbf{u}_0)$ is expanded up to the second order in terms of the generalized deformations.

It is worth remarking that these assumptions are not restrictive as they hold true for geometrically nonlinear models of beams, plates and shells. On the other hand, it is not trivial to obtain the relationship among physical and generalized deformation variables (Schwertassek et al., 1999b).

Assuming that the generalized deformations are expanded by means of a Ritz-Galerkin approach (Ritz, 1909), i.e.,

$$\mathbf{d}_g = \Phi(\mathbf{u}_0) \mathbf{q}(t), \quad (13)$$

the displacement field, up to the second order, reads:

$$\mathbf{u}_f = \mathbf{S} \mathbf{q} + \mathbf{G}(\mathbf{q}) \mathbf{q}, \quad (14)$$

where $\Phi(\mathbf{u}_0)$ are spatial mode functions, \mathbf{q} is a vector of generalized coordinates and $\mathbf{S}(\mathbf{u}_0)$ is the standard matrix of shape functions obtained with linear models. The matrix $\mathbf{G}(\mathbf{q})$ linearly depends on \mathbf{q} and allows to account for geometrically nonlinear terms:

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} \mathbf{q}^T \mathbf{G}_1 \\ \mathbf{q}^T \mathbf{G}_2 \\ \mathbf{q}^T \mathbf{G}_3 \end{bmatrix} \quad (15)$$

where $\mathbf{G}_i = \mathbf{G}_i(\mathbf{u}_0)$ is a symmetric matrix depending only on the material coordinates. Adopting the approximation (14) in the definition of the internal and external virtual works (4,5) leads to the derivation of the equations of motion in the following form:

$$m(\dot{\mathbf{v}} - \mathbf{g}) + m\tilde{\mathbf{d}}_C^T \dot{\omega} + \mathbf{C}_t^T \ddot{\mathbf{q}} + \tilde{\omega} m \tilde{\mathbf{d}}_C^T \omega + 2\tilde{\omega} \mathbf{C}_t^T \dot{\mathbf{q}} + \tilde{\omega} m \mathbf{v}_1 = \mathbf{h}_e^r \quad (16)$$

$$m\tilde{\mathbf{d}}_C(\dot{\mathbf{v}} - \mathbf{g}) + \mathbf{J}\dot{\omega} + \mathbf{C}_r^T \ddot{\mathbf{q}} + \tilde{\omega} \mathbf{J}\omega + m\tilde{\mathbf{d}}_C \tilde{\omega} \mathbf{v} + 2\tilde{\omega} \mathbf{C}_r^T \dot{\mathbf{q}} = \mathbf{h}_e^\theta \quad (17)$$

$$\mathbf{C}_t^g(\dot{\mathbf{v}} - \mathbf{g}) + \mathbf{C}_r^g \dot{\omega} + \mathbf{M}_e \ddot{\mathbf{q}} + \mathbf{C}_t \tilde{\omega} \mathbf{v} + (\mathbf{D}_e + \mathbf{D}_{cr}) \dot{\mathbf{q}} + \mathbf{K} \mathbf{q} = \mathbf{h}_e^f - \mathbf{h}_e^{ct} \quad (18)$$

where the terms of the inertia and stiffness matrices include additional terms with respect to the classical linear Newton-Euler approach (Shabana, 1998). In particular, the generalized stiffness matrix is expressed as follows:

$$\mathbf{K} = \mathbf{K}_e + \mathbf{K}_{ct} + \mathbf{K}_g^1 + \mathbf{K}_g^2 + \mathbf{K}_g^r \quad (19)$$

and contains additional contributions relative the motion induced stiffness ($\mathbf{K}_g^1, \mathbf{K}_g^2$) and the external action which account for geometrically nonlinear effects. The motion induced stiffness (\mathbf{K}_g^r) contribution depends on the reference frame motion and accounts for the loss of stiffness induced by the centrifugal acceleration (\mathbf{K}_{ct}). The aforementioned terms appear in the equations of motion as a consequence of the presence of the first order term in the virtual variation of the generalized coordinates formulation:

$$\delta \mathbf{u}_f = \mathbf{S} \delta \mathbf{q} + \mathbf{G}(\mathbf{q}) \delta \mathbf{q} \quad (20)$$

which yields a contribution in the external virtual work formulation

$$\delta W_e = \delta W_e^c + \delta W_g \quad (21)$$

where the standard terms of the external virtual work are contained in δW_e^c and the geometric contribution is described by δW_g . The complete derivation of the terms is not reported here for the sake of brevity, it must be however pointed out that all the terms of the geometric contribution can be computed as function of invariants. The formulation described above enhances the classic linear FFR approach through the addition of further terms, providing a simple solution for including geometric nonlinearities in the equations of motion. This can be considered as a relevant advantage of the proposed formulation. The nonlinearities are isolated in the inertial terms, hence, a closed form expression for the geometrical stiffening effects is derived, which is essential for the application of the proposed approach in the context

of the Modelica framework. The definition of the standard terms ($m, \mathbf{d}_C, \mathbf{C}_t, \mathbf{J}, \mathbf{C}_r, \mathbf{D}_e, \mathbf{K}_e, \mathbf{M}_e, \mathbf{h}_e^r, \mathbf{h}_e^\theta, \mathbf{h}_e^f$) and the computation of the corresponding invariants can be found in (Bascetta et al., 2015).

2 Geometrically exact modeling of slender beams

The method developed in Section 1 is applied to derive the equations of motion of slender beams, for which the cross-section plane is assumed to remain normal to the reference axis during the deformation. The motion of the flexible beam can be described in terms of three reference frames as shown in Figure 2, frame $\{\mathbf{O}_w, x_w, y_w, z_w\}$ is the world reference frame, while the undeformed beam geometry is represented by frame $\{\mathbf{O}_r, x_r, y_r, z_r\}$ where \mathbf{b}_1 is the direction of the undeformed beam axis, \mathbf{b}_2 and \mathbf{b}_3 define the cross section principal axis. Finally, frame $\{\mathbf{O}_d, x_d, y_d, z_d\}$ describes the motion of the beam cross-section. The out-of-plane displacement are assumed to be negligible.

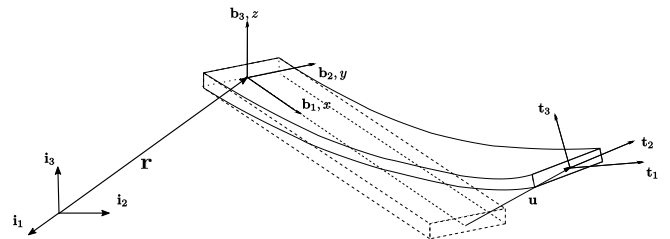


Figure 2. Beam reference frames

According to the reference frames described above, the position of a generic point on the deformed beam, with respect to the inertial frame, (see eq.1) is given by:

$$\mathbf{p} = \mathbf{r} + x\mathbf{b}_1 + \mathbf{u} + \mathbf{R} \cdot \boldsymbol{\xi} \quad (22)$$

where $\mathbf{u} = u(x)\mathbf{b}_1 + v(x)\mathbf{b}_2 + w(x)\mathbf{b}_3$ represents the vector of the cross-section translation dofs, $\boldsymbol{\xi} = y\mathbf{b}_2 + z\mathbf{b}_3$

and $\mathbf{R}(x)$ is the rigid rotation tensor of the cross-section which can be parametrized by means of Euler angles $(\theta(x), \phi(x), \psi(x))$. The deformation field, defined as $\mathbf{u}_f = \mathbf{u} + \mathbf{R} \cdot \boldsymbol{\xi}$, is a nonlinear function of a set of physical displacements $\mathbf{d}_p = (u, v, w, \theta, \phi, \psi)$, as required by the first assumption in Section 1.

As already mentioned, the Jaumann strain tensor \mathbf{B} is chosen as the strain measure. According to the small deformations assumption, \mathbf{B} can be consistently approximated as:

$$\begin{aligned} B_{11} &= e + zk_2 - yk_3 \\ 2B_{12} &= 2B_{21} = -zk_1 \\ 2B_{13} &= 2B_{31} = yk_1 \\ B_{22}, B_{33}, B_{23}, B_{32} &= 0 \end{aligned} \quad (23)$$

where e represents the axial stretch, k_1 the twisting curvature and k_2, k_3 the bending curvatures. These quantities are called generalized strains and are nonlinear functions of the physical displacements and their derivatives, see (Hodges, 2006) for further details. The internal virtual work is expressed in terms of the Jaumann strains B_{ij} in (23) and their work-conjugate stresses J_{ij} as follows:

$$\delta W_i = \int_V (\delta B_{11} J_{11} + 2\delta B_{12} J_{12} + 2\delta B_{13} J_{13}) dV. \quad (24)$$

After substituting (23), the internal virtual work can be compactly written by introducing the generalized axial force F_1 and moments M_1, M_2, M_3 as follows

$$\delta W_i = \int_0^\ell (\delta e F_1 + \delta k_1 M_1 + \delta k_2 M_2 + \delta k_3 M_3) dx. \quad (25)$$

Assuming a linear elastic constitutive law for an isotropic material, the generalized force and moment are related to the corresponding strains as:

$$\begin{Bmatrix} F_1 \\ M_1 \\ M_2 \\ M_3 \end{Bmatrix} = \begin{bmatrix} EA & 0 & 0 & 0 \\ 0 & GJ & 0 & 0 \\ 0 & 0 & EJ_{yy} & 0 \\ 0 & 0 & 0 & EJ_{zz} \end{bmatrix} \begin{Bmatrix} e \\ k_1 \\ k_2 \\ k_3 \end{Bmatrix}$$

where EA, GJ, EJ_{yy}, EJ_{zz} are the axial, torsional, and bending stiffness, respectively.

Thus, by substituting the constitutive law in the internal virtual work expression:

$$\int_0^\ell (\delta e EA e + \delta k_1 GJ k_1 + \delta k_2 EJ_{yy} k_2 + \delta k_3 EJ_{zz} k_3) dx. \quad (26)$$

It is clear that the above expression is linear in generalized strains and has the same mathematical form of the classic linear approach. Nonetheless, it is valid also in the case of large displacements and small strains. In order to adopt the described approach, a suitable change of variables is introduced, such that the elastic forces are linear in these new variables, according to the second assumption in Section 1.

The generalized strain components are written in terms of a set of generalized deformation functions $\mathbf{d}_g = (\bar{u}(x), \bar{v}(x), \bar{w}(x), \bar{\phi}(x))$ by defining:

$$\begin{aligned} e &= \frac{\partial \bar{u}}{\partial x}, & k_1 &= \frac{\partial \bar{\phi}}{\partial x}, \\ k_2 &= -\frac{\partial^2 \bar{w}}{\partial x^2}, & k_3 &= \frac{\partial^2 \bar{v}}{\partial x^2}. \end{aligned} \quad (27)$$

As shown in (Schwertassek and Wallrapp, 1999), by describing the physical variables in terms of generalized strains (27) and expanding up to the second order the corresponding relation, one can compute the components of the deformation field \mathbf{u}_f as:

$$\begin{aligned} u_{f1} &= \bar{u} - y \frac{\partial \bar{v}}{\partial x} - z \frac{\partial \bar{w}}{\partial x} - \frac{1}{2} \int_0^x \left[\left(\frac{\partial \bar{v}}{\partial x} \right)^2 + \left(\frac{\partial \bar{w}}{\partial x} \right)^2 \right] dx + \\ &\quad - y \left[\int_0^x \left(\bar{\phi} \frac{\partial^2 \bar{w}}{\partial x^2} - \frac{\partial \bar{u}}{\partial x} \frac{\partial^2 \bar{v}}{\partial x^2} \right) dx + \bar{\phi} \frac{\partial \bar{w}}{\partial x} \right] - z \left[\int_0^x \left(\frac{\partial \bar{u}}{\partial x} \frac{\partial^2 \bar{w}}{\partial x^2} + \bar{\phi} \frac{\partial^2 \bar{v}}{\partial x^2} \right) dx - \bar{\phi} \frac{\partial \bar{v}}{\partial x} \right] \end{aligned} \quad (28)$$

$$\begin{aligned} u_{f2} &= \bar{v} - z \bar{\phi} + \int_0^x \left[\frac{\partial \bar{u}}{\partial x} \frac{\partial \bar{v}}{\partial x} + \int_0^x \left(\frac{\partial \bar{u}}{\partial x} \frac{\partial^2 \bar{v}}{\partial x^2} - \bar{\phi} \frac{\partial^2 \bar{w}}{\partial x^2} \right) dx \right] dx - \frac{1}{2} y \left[\left(\frac{\partial \bar{v}}{\partial x} \right)^2 + \bar{\phi}^2 \right] + \\ &\quad - z \left[\frac{\partial \bar{v}}{\partial x} \frac{\partial \bar{w}}{\partial x} + \int_0^x \left(\frac{\partial \bar{u}}{\partial x} \frac{\partial \bar{\phi}}{\partial x} - \frac{\partial \bar{w}}{\partial x} \frac{\partial^2 \bar{v}}{\partial x^2} \right) dx \right] \end{aligned} \quad (29)$$

$$\begin{aligned} u_{f3} &= \bar{w} + y \bar{\phi} + \int_0^x \left[\frac{\partial \bar{u}}{\partial x} \frac{\partial \bar{w}}{\partial x} dx + \int_0^x \left(\frac{\partial \bar{u}}{\partial x} \frac{\partial^2 \bar{w}}{\partial x^2} + \bar{\phi} \frac{\partial^2 \bar{v}}{\partial x^2} \right) dx \right] dx - \frac{1}{2} z \left[\left(\frac{\partial \bar{w}}{\partial x} \right)^2 + \bar{\phi}^2 \right] + \\ &\quad + y \left[\int_0^x \left(\frac{\partial \bar{u}}{\partial x} \frac{\partial \bar{\phi}}{\partial x} - \frac{\partial \bar{w}}{\partial x} \frac{\partial^2 \bar{v}}{\partial x^2} \right) dx \right]. \end{aligned} \quad (30)$$

The direction cosine matrix of the cross-section, expanded up to the second order, can be defined accordingly. A full treatise can be found in (Schwertassek et al., 1999b).

Finally, the generalized deformations, namely: $\bar{u}(x)$, $\bar{v}(x)$, $\bar{w}(x)$ and $\bar{\phi}(x)$ can be approximated as a linear combination of shape functions in terms of the generalized coordinates \mathbf{q} by means of the classical Ritz-Galerkin method (Ritz, 1909), in particular:

$$\bar{u} = \Phi_1 \mathbf{q} \quad \bar{v} = \Phi_2 \mathbf{q} \quad (31)$$

$$\bar{w} = \Phi_3 \mathbf{q} \quad \bar{\phi} = \Phi_4 \mathbf{q} \quad (32)$$

where Φ are rows of admissible shape functions. It is worth to remark that the boundary conditions of the generalized displacements are the same of physical displacements. In this work, a finite element approach is used to discretize the beam domain. After the assembly procedure, the Craig-Bampton (Craig and Bampton, 1968) reduction procedure is applied by projecting the equations

on the corresponding modal basis, which include constraint as well as normal modes. This is particularly efficient from a computational point of view since the final model includes only a few degrees of freedom while providing satisfactory results. The deformation field (14) can be written as:

$$\begin{Bmatrix} u_{f1} \\ u_{f2} \\ u_{f3} \end{Bmatrix} = \begin{bmatrix} \mathbf{S}_1 \\ \mathbf{S}_2 \\ \mathbf{S}_3 \end{bmatrix} \mathbf{q} + \begin{bmatrix} \mathbf{q}^T \mathbf{G}_1 \\ \mathbf{q}^T \mathbf{G}_2 \\ \mathbf{q}^T \mathbf{G}_3 \end{bmatrix} \mathbf{q}$$

where

$$\mathbf{S}_1 = \Phi_1 - y \frac{\partial \Phi_2}{\partial x} - z \frac{\partial \Phi_3}{\partial x}$$

$$\mathbf{S}_2 = \Phi_2 - z \Phi_4$$

$$\mathbf{S}_3 = \Phi_3 + y \Phi_4$$

$$\begin{aligned} \mathbf{G}_1 = & -\frac{1}{2} \int_0^x \left(\frac{\partial \Phi_2}{\partial x}^T \frac{\partial \Phi_2}{\partial x} + \frac{\partial \Phi_3}{\partial x}^T \frac{\partial \Phi_3}{\partial x} \right) dx - y \left[\int_0^x \left(\Phi_4^T \frac{\partial^2 \Phi_3}{\partial x^2} - \frac{\partial \Phi_1}{\partial x}^T \frac{\partial^2 \Phi_2}{\partial x^2} \right) dx + \Phi_4^T \frac{\partial \Phi_3}{\partial x} \right] + \\ & -z \left[\int_0^x \left(\Phi_4^T \frac{\partial^2 \Phi_2}{\partial x^2} + \frac{\partial \Phi_1}{\partial x}^T \frac{\partial^2 \Phi_3}{\partial x^2} \right) dx - \Phi_4^T \frac{\partial \Phi_2}{\partial x} \right] \end{aligned} \quad (33)$$

$$\begin{aligned} \mathbf{G}_2 = & \int_0^x \left[\frac{\partial \Phi_1}{\partial x}^T \frac{\partial \Phi_2}{\partial x} + \int_0^x \left(\frac{\partial \Phi_1}{\partial x}^T \frac{\partial^2 \Phi_2}{\partial x^2} - \Phi_4^T \frac{\partial^2 \Phi_3}{\partial x^2} \right) dx \right] dx - \frac{1}{2} y \left(\frac{\partial \Phi_2}{\partial x}^T \frac{\partial \Phi_2}{\partial x} + \Phi_4^T \Phi_4 \right) + \\ & -z \left[\frac{\partial \Phi_2}{\partial x}^T \frac{\partial \Phi_3}{\partial x} + \int_0^x \left(\frac{\partial \Phi_1}{\partial x}^T \frac{\partial \Phi_4}{\partial x} - \frac{\partial \Phi_3}{\partial x}^T \frac{\partial^2 \Phi_2}{\partial x^2} \right) dx \right] \end{aligned} \quad (34)$$

$$\begin{aligned} \mathbf{G}_3 = & \int_0^x \left[\frac{\partial \Phi_1}{\partial x}^T \frac{\partial \Phi_3}{\partial x} + \int_0^x \left(\frac{\partial \Phi_1}{\partial x}^T \frac{\partial^2 \Phi_3}{\partial x^2} + \Phi_4^T \frac{\partial^2 \Phi_2}{\partial x^2} \right) dx \right] dx + \\ & + y \int_0^x \left(\frac{\partial \Phi_1}{\partial x}^T \frac{\partial \Phi_4}{\partial x} - \frac{\partial \Phi_3}{\partial x}^T \frac{\partial^2 \Phi_2}{\partial x^2} \right) dx - \frac{1}{2} z \left(\frac{\partial \Phi_3}{\partial x}^T \frac{\partial \Phi_3}{\partial x} + \Phi_4^T \Phi_4 \right). \end{aligned} \quad (35)$$

The terms of the direction cosines matrix are not reported here for brevity, the procedure for the computation is similar.

3 Model implementation and validation

3.1 Model implementation

The implementation of the model is similar to (Ferretti et al., 2014), a component fully compatible with the stan-

dard Modelica multibody library has been developed. The shape functions and the invariants which assemble the terms of eq.(18) are collected in a Modelica record defined as `replaceable`, in order to exploit the object-oriented approach of the language and possibly instantiate multiple flexible beams in the same model. The aforementioned record has been calculated offline by means of an external script written in Matlab (Mathworks, 2014) starting from the geometric and material parameters of the beam. The symbolic computation toolbox has been used to solve eqs.(28,29,30).

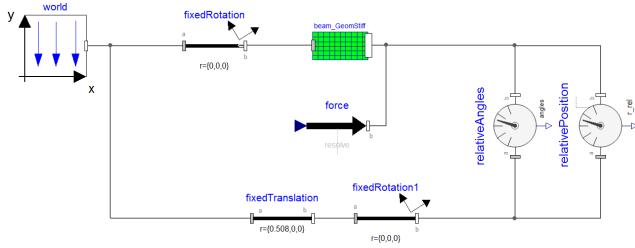


Figure 3. Diagram level scheme of the beam in a simple model

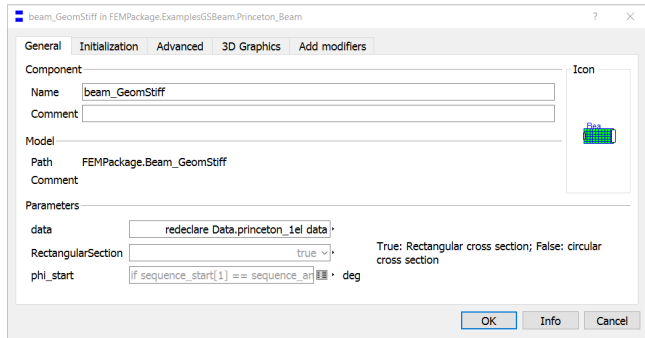


Figure 4. User interface of the beam model

The connectors are placed at the ends of the beam and the number is limited to two. As an example of model usage, fig. 3 shows a simple implementation containing the beam, while fig. 4 shows the GUI of the beam model, where the replaceable data record can be modified.

In the rigid body components of the multibody library, the body coordinates are used as state variables when the component is floating, this is carried out by selecting the component as *root* of one connection tree (see (The Modelica Association, 2009; Otter et al., 2003) for details). Conversely, if the body is connected to a *root*, a branch statement is declared and the kinematic is computed from the states of the joints connecting the component to the root tree. This mechanism is reproduced in the implementation described here, the FFR (placed in the `FrameA` connector) is assigned as root of the connection tree if the body is floating, if the body is part of a kinematic chain a branch is declared between `FrameA` and `FrameB`. Finally, the 3D visualization of the component is provided by means of the `Advanced.Shape` visualizer of the standard multibody library.

3.2 Model validation

The model has been thoroughly validated by means of two simulation scenarios. Initially, the well-known Princeton beam experiment (Bauchau et al., 2016) has been reproduced in order to validate the quasi-static behaviour of the model. The beam is subject to a lateral load in different root orientations ranging from 0 to 90 degrees. The setup is briefly shown in fig. 5 while the geometric and material parameters of the beam are shown in tab. 1. This experiment is particularly effective in order to validate the static deflection of the beam as well as the coupled bending/tor-

sional behaviour, due to the different relative orientation of the load and beam.

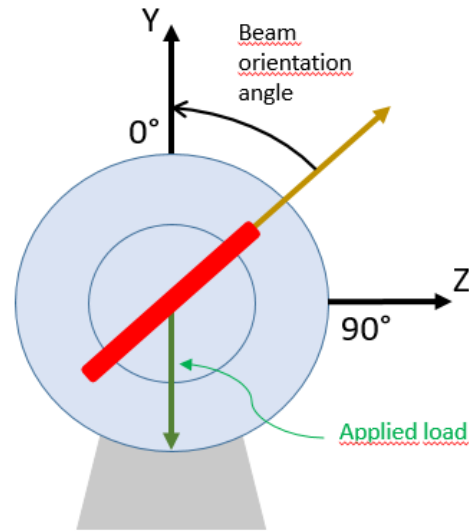


Figure 5. The Princeton beam experiment, reference scheme

Length	0.508 m
Height	3.2024×10^{-3} m
Width	12.377×10^{-3} m
Axial S	2.842×10^6 N
Shearing K_{22}	0.6401×10^6 N
Shearing K_{33}	0.9039×10^6 N
Torsional H_{11}	3.103 Nm ²
Bending H_{22}	36.28 Nm ²
Bending H_{33}	2.429 Nm ²

Table 1. Princeton beam parameters

The simulations have been carried out in three different loading conditions, namely $P_1 = 4448$ N, $P_2 = 8896$ N and $P_3 = 13345$ N and the results have been compared with simulations performed in the software Dymore (Dymore Solutions, 2016) where a geometrically exact beam theory is implemented. A photograph of the animation is shown in fig. 6, where the green arrow on the tip of the bent beam represents the applied load. Moreover, a substructured instance has been tested in order to show the difference in performance and accuracy. The beam has been divided in 5 elements (6 dofs each) placed in series by simply connecting five instances of the model. Figs. 7 and 8 show the absolute displacement of the transverse components of the beam with respect to the beam root orientation, while fig. 9 shows the twisting angle in the same circumstances. The continuous lines represent the Dymore solutions while the triangles represent the simulations performed in Dymola (Dynasim AB). As shown in the figures, the results of the single element model are in good accordance with the exact solution in the first loading case (blue), while five substructuring elements are sufficient to correctly reproduce the exact solution in the other loading cases. Indeed, in the

second and third case, large displacements are expected and a single element with a second order approximation is not adequate. As expected from a theoretical point of view, the proposed model is slightly stiffer because shear deformations are not included. It is also worth to remark that a null twisting angle (fig. 9) would be predicted by a linearized beam model whilst the coupling effect between bending and twisting is correctly captured by the present formulation even with a single element.

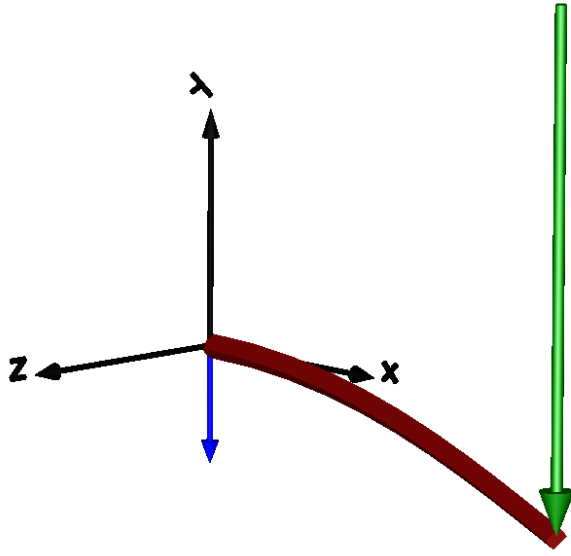


Figure 6. The Princeton beam experiment, simulation visualization

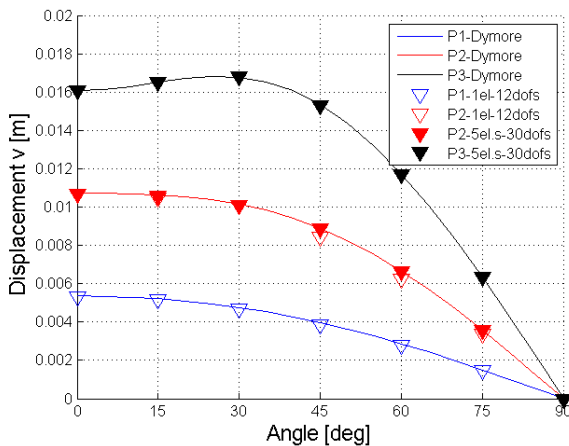


Figure 7. The Princeton beam experiment, transverse tip displacement (v)

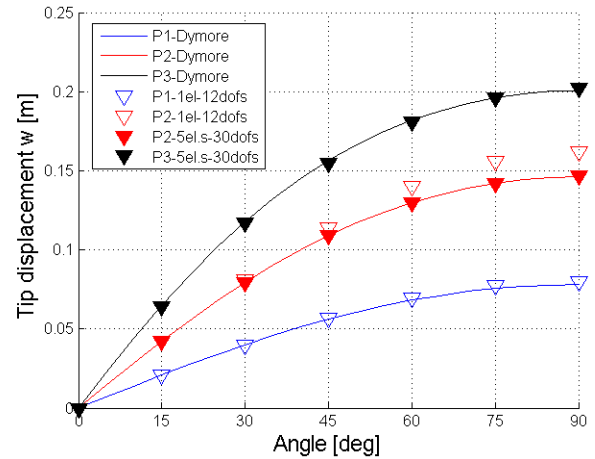


Figure 8. The Princeton beam experiment, transverse tip displacement (w)

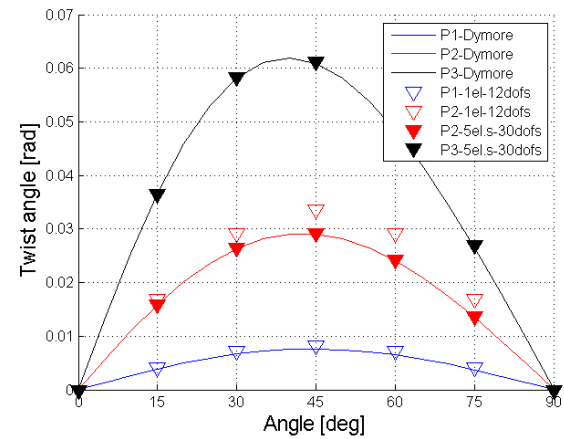


Figure 9. The Princeton beam experiment, twisting angle

In order to validate the dynamic behaviour of the model, the classical planar spin-up manoeuvre benchmark has been considered (Berzeri and Shabana, 2002; Valembois et al., 1997; Shi et al., 2001; Wu and Haug, 1988). A flexible beam rotates about one end with a prescribed angular law, a diagram of the mechanism is reported in fig.10. The law describing the time evolution of the angle θ is the following:

$$\theta(t) = \begin{cases} \frac{\Omega}{T} \left[\frac{t^2}{2} + \left(\frac{T}{2\pi} \right)^2 (\cos(\frac{2\pi t}{T}) - 1) \right], & t < T \\ \Omega(t - T/2), & t \geq T \end{cases} \quad (36)$$

thus, the spin-up starts at $t = 0$ and ends at $t = T$, when a constant angular velocity is reached, where it has been assumed $T = 15s$ in the considered experiment. This benchmark is widely used in literature in order to demonstrate the effectiveness of the substructuring technique as well as the robustness of nonlinear formulations. The following geometrical data were assumed for the beam: length $L = 8m$, cross sectional area $A = 7.3 \cdot 10^{-5}m^2$, modulus of elasticity $E = 1.3359 \cdot 10^{10}N/M^2$, second moment of inertia $I = 8.218 \cdot 10^{-9}m^4$ and density $\rho = 2766Kg/m^3$.

The tip transverse deflection for a 20 seconds simulation is here compared with the results obtained with a completely different approach, thoroughly described in (Boer et al., 2011). A single element beam has been used by retaining two additional normal modes in order to increase accuracy. Thus the number of active dofs is five, considering only the planar components of the end node deformation. Fig. 11 shows satisfactory results in terms of accordance between the two approaches. Moreover, the dy-

namics shown here is in good accordance with the other results in literature (see (Boer et al., 2011; Wu and Haug, 1988)), and the maximum tip deflection is similar to other numerical results as shown in tab. 2. The proposed formulation captures correctly the geometric stiffening effect induced by the rotation and overcomes the shortcomings of the standard linear approach, while keeping low the computational effort.

Model	Number of elements	Max. deflection [m]
Present formulation	1 (5 dofs)	0.536
SPACAR, nonlinear beam	4 (8 dofs)	0.5388
SPACAR, superelement	4 superelements (8 dofs)	0.5375
Wu and Haug (Wu and Haug, 1988)	6 substructure	0.543

Table 2. Maximum tip deflection, comparison with other simulation results

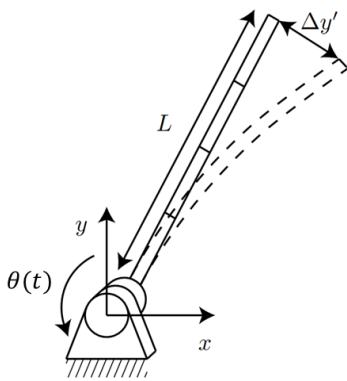


Figure 10. Planar spin-up, scheme of the setup

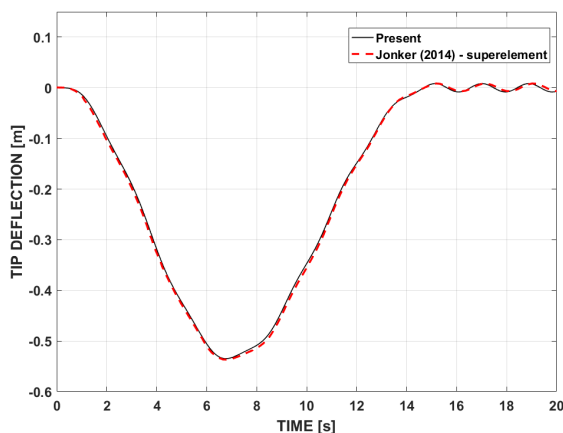


Figure 11. Planar spin-up, tip transverse deflection

4 Conclusion

In this paper, an approximated dynamic model for flexible beams is presented, including geometrically nonlinear

phenomena. The equations of motion for a generic flexible body are developed starting from the approximation of the Jaumann strain tensor under the small strain hypothesis. Assuming that the deformation field of the continuum model can be expanded up to the second order, a closed form expression of the equations of motion is presented including only a few additional terms with respect to the standard floating frame of reference approach. Subsequently, the proposed formulation is applied to slender structures. A second order model is consistently derived from a geometrically exact beam model. The finite element approach is adopted in order to discretize the beam, finally the Craig-Bampton method is applied to reduce the number of degrees of freedom. The theoretical model is implemented in the Modelica framework by adopting an efficient approach where the invariant terms are computed offline and the resulting model is fully integrated in the Modelica multibody library. The model is finally validated by means of comparison with well known literature benchmarks, the numerical results are compared with simulations obtained by means of completely different approaches. The model is suitable to perform small strains, moderate displacements analysis and can be employed in large displacements cases by means of substructuring, which is naturally managed in Modelica. The proposed model will allow to consider the realistic behaviour of slender beams subject to high angular velocities, as well as to correctly consider the geometrical nonlinear phenomena in slender beams. The development of this model constitutes a step forward in the state of the art of the flexible multibody Modelica models, leading to more efficient models for real-world applications. The beam model, as well as the other flexible multibody models developed by the authors are freely available upon request, hence, the author would encourage possible users to contact them.

References

- H.E. Absy and A.A. Shabana. Geometric Stiffness and Stability of Rigid Body Modes. *Journal of Sound and Vibration*, 207(4):465–496, 1997.
- Arun K Banerjee. *Flexible Multibody Dynamics: Efficient Formulations and Applications*. John Wiley & Sons, 2016.
- L. Bascetta, G. Ferretti, and B. Scaglioni. Closed-form Newton–Euler dynamic model of flexible manipulators. *Robotica (in press)*, 2015.
- Olivier A. Bauchau, Peter Betsch, Alberto Cardona, Johannes Gerstmayr, Ben Jonker, Pierangelo Masarati, and Valentin Sonneville. Validation of flexible multibody dynamics beam formulations using benchmark problems. *Multibody System Dynamics*, 37(1):29–48, 2016. ISSN 1573-272X.
- M. Berzeri and A.A. Shabana. Study of the Centrifugal Stiffening Effect Using the Finite Element Absolute Nodal Coordinate Formulation. *Multibody System Dynamics*, 7(4):357–387, 2002.
- S.E. Boer, R.G.K.M. Aarts, J.P. Meijaard, D.M. Brouwer, and J.B. Jonker. A two-node superelement description for modelling of flexible complex-shared beam-like components. In *Multibody Dynamics 2011, ECCOMAS Thematic Conference*, 2011.
- Hartmut Bremer. *Elastic Multibody Dynamics*. Springer, 2008. ISBN 9781402086809.
- Claytex Services Ltd. *FlexibleBody Library*. Coventry, UK.
- R. R. Craig and M. C. C. Bampton. Coupling of substructures for dynamic analyses. *AIAA Journal*, 6(7):1313–1319, 1968.
- Dymore Solutions. *Dymore user’s manual*, 2016.
- Dynasim AB. *Dymola*. Lund, Sweden.
- G. Ferretti, A. Leva, and B. Scaglioni. Object-oriented modelling of general flexible multibody systems. *Mathematical and Computer Modelling of Dynamical Systems*, 20(1):1–22, 2014.
- M. Géradin and Alberto Cardona. *Flexible Multibody Dynamics: A Finite Element Approach*. Wiley, Chichester, Great Britain, January 2001.
- A. Heckmann. On the choice of boundary conditions for mode shapes in flexible multibody systems. *Multibody System Dynamics*, 23(2):141–163, 2010.
- A. Heckmann, M. Otter, S. Dietz, and J. D. López. The DLR FlexibleBodies library to model large motions of beams and of flexible bodies exported from finite element programs. In *5th Modelica Conference*, Vienna, Austria, September 4–5 2006.
- Dewey Hodges. *Nonlinear Composite Beam Theory*. AIAA, 2006. ISBN 1563476975.
- G. Jelenić and M. A. Crisfield. Interpolation of rotational variables in nonlinear dynamics of 3d beams. *International Journal for Numerical Methods in Engineering*, 43(7):1193–1222, 1998.
- U. Lugić, M. A. Naya, J. A. Pérez, and J. Cuadrado. Implementation and efficiency of two geometric stiffening approaches. *Multibody System Dynamics*, 20:147–161, 2008.
- Mathworks. *Matlab*, 2014.
- MSC Software Corporation. *ADAMS/Flex user’s manual*, 2017.
- M. Otter, H. Elmqvist, and S.E. Mattsson. The new Modelica multibody library. In *3rd Modelica Conference*, Linköping, Sweden, November 3–4, 2003.
- P Frank Pai. *Highly flexible structures: modeling, computation, and experimentation*. AIAA (American Institute of Aeronautics & Ast, 2007.
- W. Ritz. Über eine neue Methode zur Lösung gewisser Variationsprobleme der mathematischen Physik. *Journal für die Reine und Angewandte Mathematik*, 135:1–61, 1909.
- F. Schiavo, L. Viganò, and G. Ferretti. Object-oriented modelling of flexible beams. *Multibody System Dynamics*, 15(3): 263–286, 2006.
- R. Schwertassek and O. Wallrapp. *Dynamik flexibler Mehrkörpersysteme*. Vieweg, Wiesbaden, 1999.
- R. Schwertassek, O. Wallrapp, and A. A. Shabana. Flexible multibody simulation and choice of shape functions. *Nonlinear Dynamics*, 20:361–380, 1999a.
- Richard Schwertassek, Oskar Wallrapp, and Ahmed A Shabana. Flexible multibody simulation and choice of shape functions. *Nonlinear Dynamics*, 20(4):361–380, 1999b.
- A. A. Shabana. *Dynamics of Multibody Systems*. Cambridge University Press, New York, 1998.
- I. Sharf. Geometric stiffening in multibody dynamics formulations. *Journal of Guidance, Control and Dynamics*, 18(4): 882–890, 1995.
- P. Shi, J. McPhee, and G.R. Heppler. A deformation field for Euler–Bernoulli beams with applications to flexible multibody dynamics. *Multibody System Dynamics*, 5:79–104, 2001. ISSN 1384–5640.
- Spacar. *user’s manual*, 2016.
- The Modelica Association. *Modelica – A Unified Object-Oriented Language for Physical Systems Modeling*. Language Specification Version 3.1, 2009.
- R. E. Valembois, P. Fiset, and J. C. Samin. Comparison of various techniques for modelling flexible beams in multibody dynamics. *Nonlinear Dynamics*, 12:367–397, 1997. ISSN 0924–090X.
- Oskar Wallrapp and Simon Wiedemann. Comparison of results in flexible multibody dynamics using various approaches. *Nonlinear Dynamics*, 34(1-2):189, October 2003.
- S.C. Wu and E.J. Haug. Geometric non-linear substructuring for dynamics of flexible mechanical systems. *International Journal for Numerical Methods in Engineering*, 26:2211–2276, 1988.

Musculoskeletal Modeling of the Hand and Contact Object in Modelica

Shashank Swaminathan¹ Johan Andreasson²

¹Novi, Michigan, USA, sh.swami235@gmail.com

²Modelon KK, Japan, johan.andreasson@modelon.com

Abstract

The paper's primary goal is to develop a mathematical model that could be used towards the development and improvement of orthotic assist gloves. The model is constructed using component based modeling in the object-oriented declarative language Modelica, specifically the MultiBody Modelica library. Multiple hand models currently do exist; however, they are mainly causal, and require separate development and validation of mathematical solvers before use. By using Modelica, the model is constructed from the system's physical equations, thereby relieving issues regarding validity of the model's computational equations; the acausality inherent in Modelica allows for model development that more closely mirrors relations in the physical world. The model is scoped to be able to model the kinematics and dynamics of the hand when grasping a spherical object – both bone structure and muscle geometry and actuation are simplifications based off anatomy literature. The contact model is developed as a separate component from the hand system. The main design goal of the contact model is to represent the characteristics of a relatively rigid object that still maintains a degree of friction and pliability on the surface layer.

The main two grasps tested in the paper are the prehensile and precision grasps (powerful and dexterous grasps). The muscle actuation profiles per each finger are adjusted until the desired dynamic profile is achieved for each type of grasp. The main data points of interests are the joint angles and contact forces for each finger. Further verification of the model is done using the animation automatically generated by the tool. Simulation testing results indicate that the model can successfully simulate contractions at all levels of abstraction of the hand's components (basic bone-joint components, finger components, and the overall hand system). The results also indicate that both prehensile and precision grasps are possible, given appropriate muscle actuation and finger orientation parameter values.

Keywords—*musculoskeletal model of hand; Modelica; grasp model; orthotic gloves*

1 Introduction

1.1 Relevant Background and Definitions

Patients recovering from a stroke, or those that have Parkinson's disease, amongst many others, typically experience muscle weakness in the upper extremities. The use of orthotic devices in such situations is an effective method of returning a modicum of motor control to patients. Multiple such orthotic devices have been developed, including, but not limited to, gloves (Radder et al, 2015), (Adler, 2016), braces (Linn et al, 2012), and soft-muscle pneumatic tubes (Yanchev, 2015; Polygerinos, 2015). However, many of these devices, must be specially constructed per each patient, and requires multiple rounds of testing and data acquisition before completion. Constructing a mathematical model would enable a better understanding of the orthotic device, as well as optimize its construction. The prerequisite to developing a model of an orthotic device, is the development of a model for the underlying system, the hand.

Hand modeling has been typically done as a system of rigid bodies connected through revolute joints e.g. (Griffin et al, 2000). The papers derive the full set of equations of motion of the hand from this physical concept e.g. (Tarmizi, 2009).

From (Marieb, 2000), neural impulses trigger protein-based reactions that leads to overall muscle contraction, proportional to the neural impulse strength. Since the purpose of this paper is not to model the neural aspects, we will abstract this as an actuation request for a percentage of total muscle force.

In this paper also, the hand is modeled as composed of rigid bodies connected by revolute joints. The joints have restrictions on the total angle of rotation, and muscle actuation is added to the fingers appropriately.

(Hicks et al, 2015) observes that mathematical modelers have a dual responsibility of verifying and validating both the physical equations in the model, and the mathematical solving components of the model. We aim to significantly reduce this challenge by keeping the physics of the system well-removed from the mathematics required to solve the models. This is achieved by using Modelica (Modelica®, 2013) as the modeling language to describe the physical equations of

the hand, and Modelica-supporting tools – specifically Dymola (Dymola, 2017), OpenModelica (OpenModelica, 2016), JModelica (JModelica, 2016), and Wolfram SystemModeler (SystemModeler, 2015) – to mathematically solve the model. The choice of language was made due to Modelica's object-oriented unique nature; the systems can be broken into components, and each component's behavior can be represented solely through its physical equations. The separation of the physical and mathematical aspects of the system, with the user only interacting with the physical equations, and the tool handling the mathematical portion, enables focus only on the validity of the physical equations of the model.

The model utilizes the MultiBody library (Otter et al., 2003), which contains many components dealing with three-dimensional rigid bodies, further reducing the user burden.

To check the performance, the hand model performs prehensile (powerful) and precision (gentle) grasps around spherical objects. The grasps are derived from the taxonomy of grasps defined in (Cutkosky, 1990). To investigate such motion, a model of a contact object is also required; given the various levels of potential abstraction available while developing the contact object, this is addressed separately in the paper.

1.2 Objectives

The goal for the work described in this paper was to build a prototype mathematical model of the hand, in Modelica, that can describe the kinematic and dynamic interaction between the bones, joints, and natural or artificial muscles and tendons, such that it can be used to:

1. Simulate the curling and extension motion of the finger based on activation of the posterior muscles and anterior muscles.
2. Simulate different types of grasping motions; specifically simulate prehensile and precision grasping motions around a spherical object.
3. Visualize the simulation of the finger motions through three-dimensional animation.
4. Capture the contact forces on the fingers resulting from muscle actuation around the spherical object.

In Section 2, the physiological considerations in modeling the hand are discussed, including the necessary assumptions made. In Section 3, a closer look is taken at the hand model itself, involving both a component-by-component inspection, as well as a broader view at the package hierarchy. Section 4 follows with detail on the structure of the contact model developed in this paper. Section 5 handles the simulation of the models, as well as the corresponding analysis. Section 6 provides the final remarks and closes the paper.

2 Approach to the Physiology

The bones in the hand are treated as rigid bodies, and joints are modeled as a set of revolute joints, the number depending on the degrees of freedom in the joint's motion.

The muscles in the hand are composed of numerous sarcomeres (muscle fibers); these muscle fibers actuate in unison to produce the overall muscle force. The model of the muscle abstracts this actuation process into one total force - the input to the muscle component is the percent of the total muscle being actuated, and the output is the product of the percent value and the parameter value for the total muscle force (Marieb, 2000). This is done as sarcomeres actuate in an all-or-nothing manner; hence, for the muscle to vary the force of contraction, it must vary the total amount of sarcomeres firing – in essence, activating a portion, or percentage, of the total possible muscle force.

The muscle's complex structure is broken down into multiple line segments moving between attachment points, as an approximation to the curve, demonstrated in Figure 1.

The attachment points function as the skin, limiting the muscle to conform to the physiology of the hand itself. The nature of skin as a dividing middle layer between a bone and an object is included in the contact model. It acts as a buffer layer between the direct contact between the bone and object, serving to add a degree of compliancy. The tendons are assumed to act in conjunction with the muscles as massless bodies that connect contracting muscles with appropriate bone structures.

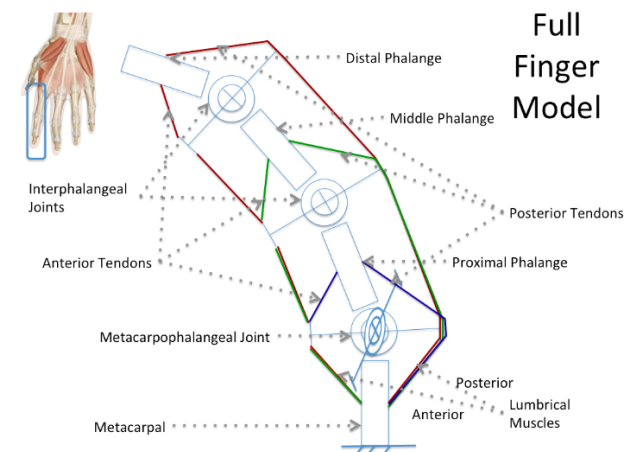


Figure 1: Finger Model Approximation schematic

3 Hand Model

3.1 Modeling Approach

The musculoskeletal aspect of the hand can be broken down into component-based construction using bones and muscles. The basic component considered to have similar functional properties to the hand is called the

Bone-Joint-Bone component; it is constructed using two bones, a connecting revolute joint, and actuating muscles on both the anterior and posterior side.



Figure 2: Bone-Joint-Bone Component Structure

A finger can be considered an extension of this idea; rather than having two bones and one joint, there are multiple bones, and multiple joints, between joints for normal flexing motion as well as sideways motion.

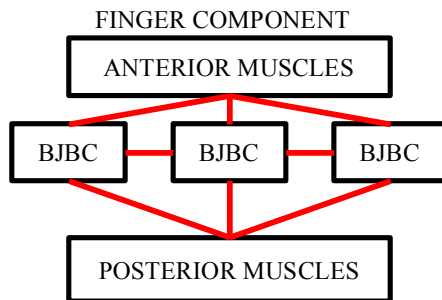


Figure 3: Finger component structure

The hand itself can be thought of as the joint workings of multiple fingers in unison, connected through a bone structure representing the wrist.

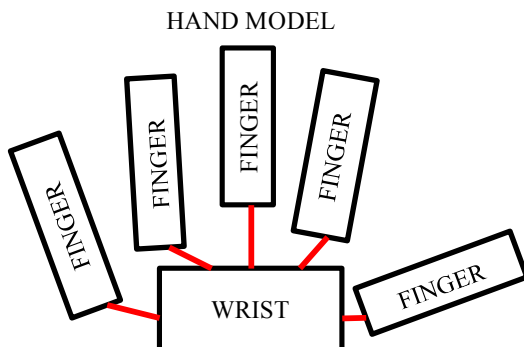


Figure 4: Hand Model Structure

The modeling approach relies on the component breakdown detailed above. By relying on the basic Bone-Joint-Bone component structure, the finger bones and the overall hand are constructed. Muscle components are added as appropriate to actuate the joints present.

3.2 Bone-Joint-Bone Component

This component (BJBC) represents the basic structure of the bones and joints in the hand. The component is constructed using two rigid bodies representing bones, connected by a revolute joint representing a finger joint; there are attachment points designated on the bones as areas the muscle will actuate upon. The Double-Joint-Bone (DBJBC) component is an extension of this idea, with an additional degree of motion added to the joint, to allow sideways motion.

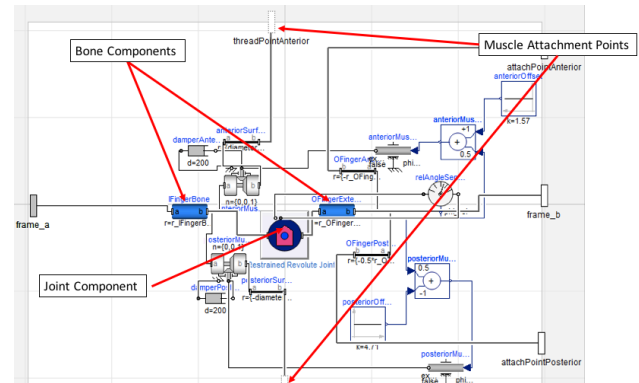


Figure 5: Schematic of a Basic Muscle-Joint component

3.3 Finger Component

The finger is constructed by fusing two BJBC's and one DBJBC, to make four bones (metacarpal and phalanges) connected by three joints (metacarpophalangeal and interphalangeal joints) – as seen in Figure 6A.

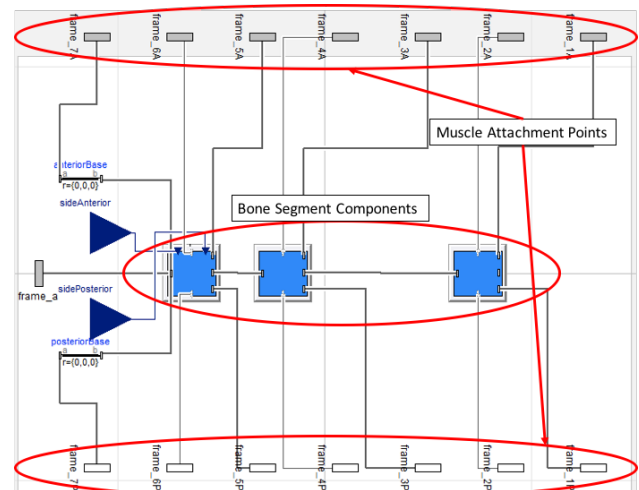


Figure 6A: Schematic of the Finger Bone Model

There are muscle components for both the anterior and posterior side, connected to the bone at the attachment points (as seen in Figure 6B).

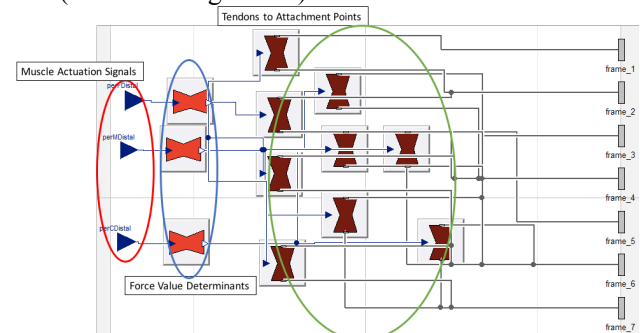


Figure 6B: Schematic of the Finger Muscle Model

The finger model additionally contains elements that model interface to a contact object, and is discussed in the next section; this is shown in Figure 6C.

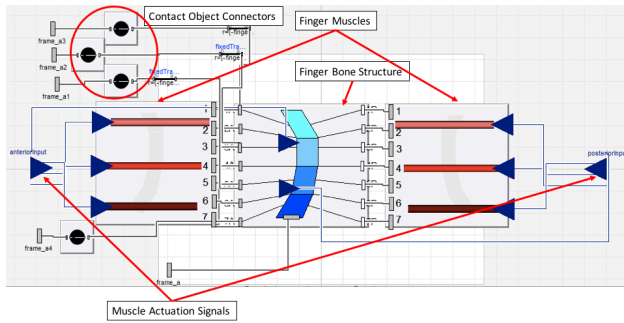


Figure 6C: Schematic of the Full Finger Model

3.4 Hand Component

The model is created by instantiating multiple finger components, each at a different position and orientation relative to the inertial frame. The non-opposable fingers each have axes of rotation rotated slightly (about 15 degrees) relative to each other, while the opposable finger's axis of rotation is almost opposite to the axes of the other fingers.

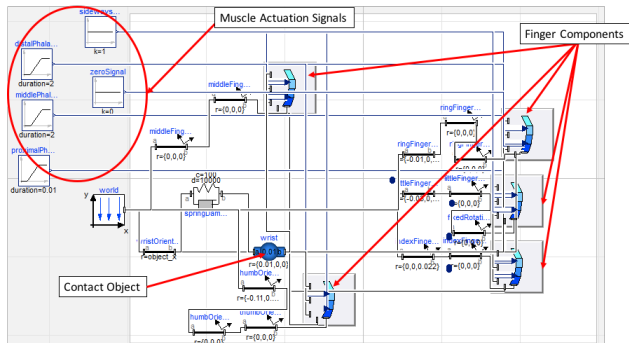


Figure 7: The Hand Component

3.5 Package Structure

The overall PowerGrab library consists of one main package, PowerGrabStructure (as seen in Figure 8), and a separate package for test models, named PowerGrabTestingRig (not shown in the figure). The division was made so that the main models can be assuredly independent of the testing models and other older versions.

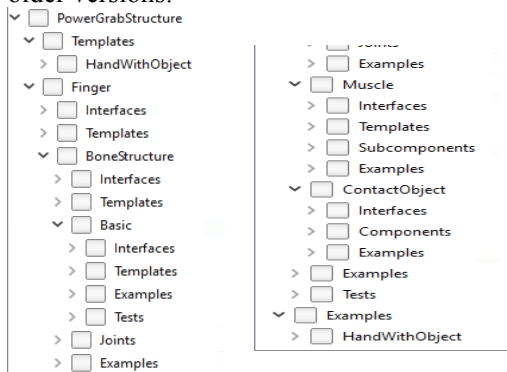


Figure 8: Package Structure

The PowerGrabStructure package contains the main components of the library, including the bone structures,

the muscle components, and contact object models. There also exist examples for each type of system, namely the basic bone-joint system, the finger-and-contact system, and the hand-and-contact system.

4 Contact Object

The contact object is modeled essentially as a semi-rigid sphere – a combination of nonlinear spring and damper systems that only exerts a force on the bone when a contact event occurs (below is a diagram of the contact object).

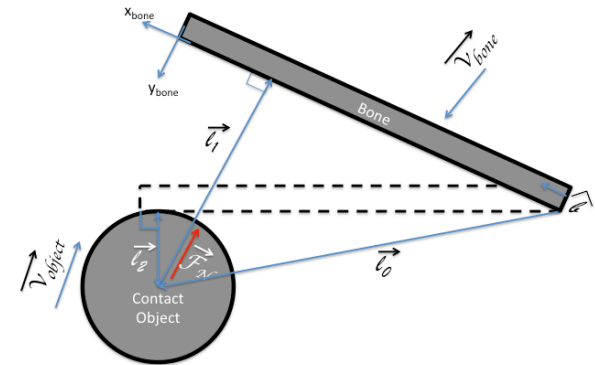


Figure 9: Spherical Object Contact Model

The object is represented as the combination of a point defining the center of the contact object, and a connector component between the object center and the potential point of contact on the bone. Each connector component details the contact dynamics between the contact object and the specific bone segment the component is connected to. Having separate connectors per each bone segment allows there to be multiple contact points per finger, one per each bone segment (for a maximum of 4 points per finger). However, as the connector follows a straight line, the contact is restricted to occur at a single point per each bone segment. Furthermore, as the contact model is designed for a spherical object, later models for other object shapes must be independently developed.

4.1.1 Determining the point of contact

We define vector \vec{l}_0 as the direct path from the base of the bone to the center of the contact object. Next, we define a vector perpendicular to the bone, \vec{l}_1 , by subtracting the projection of the vector \vec{l}_0 along the length of the bone from \vec{l}_0 . Should the magnitude of \vec{l}_1 fall below the radius of the object, we can then determine that contact has occurred.

$$\vec{l}_1 = \vec{l}_0 - (\vec{l}_0 \cdot \hat{i}_{bone})\hat{i}_{bone}$$

The actual implementation of this strategy in Modelica is simplified using a relative position sensing

component and a prismatic joint. A prismatic is attached to the base of the bone, and is free to slide along the x-axis, as resolved in the frame of the bone. Using a relative position sensing component between the center of the contact object and the nonmoving base end of the prismatic joint, resolved in the frame of the bone, the vector for the relative position of the center of the object is found. The component of that vector along the bone is determined (as the x-component of the vector, as it is resolved in the frame of the bound), and is subtracted from the overall relative position vector. This leaves us with the component of the vector that is perpendicular to the bone, which is the vector desired.

If and as the finger slips along the object, both the number and location of contact points will be updated accordingly.

4.1.2 Determining the force of contact

As the contact object is spherical in nature; this allows for the following abstraction: the object is a spring that has a relaxed length of 0, with a nonlinear stiffness that becomes nonzero only when the stretched length is below a certain threshold (thus creating a zone of nonzero stiffness described by a threshold radius tR). The force equation is thus as follows:

$$F_N(\vec{l}_1) = \begin{cases} k(tR - |\vec{l}_1|) * \hat{l}_1, & |\vec{l}_1| \leq tR \\ 0, & |\vec{l}_1| > tR \end{cases}$$

Apart from the normal contact force between the object and the bone, an additional force representing the effect of skin on contact is also applied. This is considered as a "buffer layer", as the skin will meet the contact object before the bone, thus acting as a buffering between the two. The skin is considered to have some pliability, and is therefore modeled as a spring connection between the contact object and the point of contact on the bone. Due to modeling purposes, the skin is assumed to be layered around the contact object rather than the bone itself, as it allows for the approximation that the skin-caused buffering force $F_{Buffer}(|\vec{l}_1|)$ acts in the same manner as the normal contact force, albeit at a larger threshold range. This extension in force and range is reflected in the parameters bC and bR , respectively.

$$F_{Buffer}(|\vec{l}_1|) = \begin{cases} bC, & |\vec{l}_1| \leq tR + bR \\ 0, & |\vec{l}_1| > tR + bR \end{cases}$$

4.1.3 Determining the friction due to contact

The friction due to the contact between the bone and contact object is represented as a damping on the sliding motion across the surface of the contact object. The magnitude of damping is $|\vec{F}_N| * |\vec{v}_{surface}|$. The normal force magnitude is equal to the magnitude of the contact force on the bone, and the surface velocity is determined as the magnitude of the result of subtracting the vector

component of the relative velocity between the bone and contact object that is parallel to the radius from the overall relative velocity vector.

$$\vec{v}_{rel} = \vec{v}_{bone} - \vec{v}_{object}$$

$$\vec{v}_{surface} = \vec{v}_{rel} - \left(\vec{v}_{rel} \cdot \frac{\vec{l}_1}{|\vec{l}_1|} \right) \frac{\vec{l}_1}{|\vec{l}_1|}$$

5 Simulation

5.1 Component Testing

The purpose of the component tests is to determine if the component's performance conforms to the expected result. To test the bone structure components, muscle components are instantiated in the test models, to actuate the bone structures.

5.1.1 Bone-Joint-Bone Component Test

The muscle actuation profile alternates between actuating the anterior muscle and actuating the posterior muscle, with small intervals of overlap. As seen in Figure 10, the limits on rotation are -0.5 and 1.6 radians, and the system can successfully reach those limits following sustained muscle activation. As the desired functionality is for the component to be able to undergo such motion, we conclude that the Bone-Joint-Bone Component can adequately support our needs.

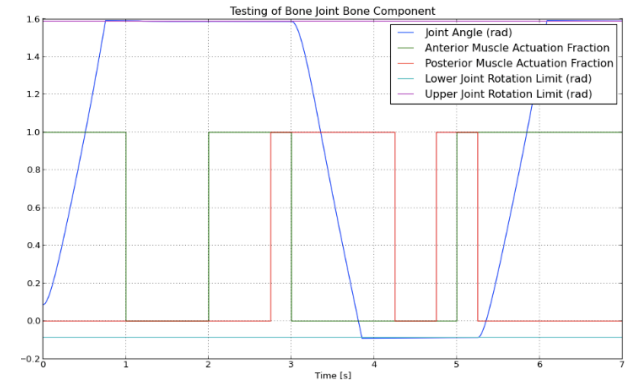


Figure 10: (Unit Level) test of BJB component

5.1.2 Double-Bone-Joint-Bone Testing

Like the Bone-Joint-Bone component's test, the DBJBC component also utilizes muscle components to actuate the bones in the system. As seen in Figure 11, we actuate the side muscles using the same muscle activation profile used for testing BJB component, while keeping the other muscles inactive. The following angular displacement occurs in the Side Joint (note that the limits on the angle of rotation is different between the joints).

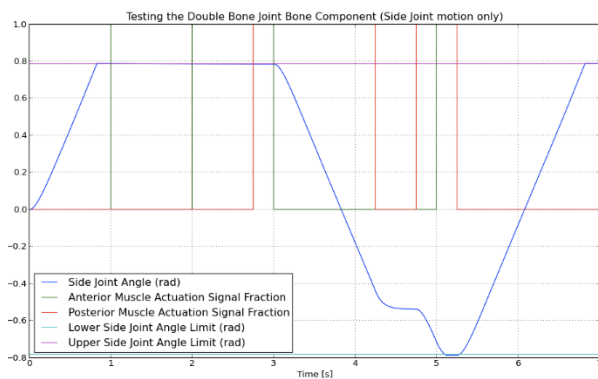


Figure 11: (Unit Level) test of the DBJB component: Side joint actuation only

Of special interest is the relationship between sideways motion and forward motion. It is reasonable to expect that, due to coupled dynamics, forward motion will cause motion sideways, and vice versa. In Figure 12, both the Forward Joint and the Side Joint are actuated. For the Forward Joint, it is sequential activation of the anterior and posterior muscles. The Side Joint follows the same activation profile used in the previous tests, and correspondingly experiences motion as seen in Figure 12 (bottom graph). There are also slight additional movements in the side joint, in conjunction with change in direction of movement in the forward joint, which can be attributed to the coupled dynamics.

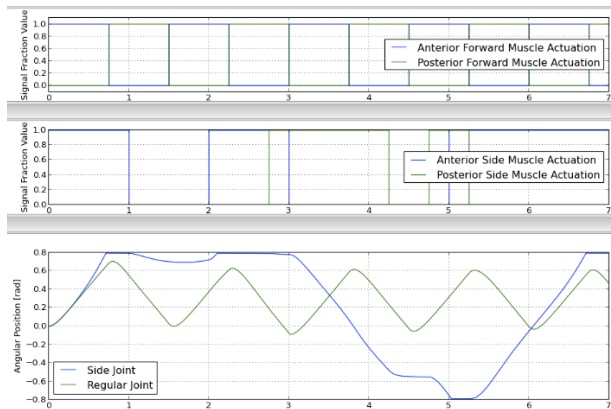


Figure 12: (Unit Level) test of the DBJB component, with simultaneous side and forward actuation

Similar to the requirements for the Bone-Joint-Bone component, the requirement for this component is to be able to undergo such movement given appropriate actuation, without too much deviation from smooth motion. As such, we determine that this component is suitable for use.

5.1.3 Finger Component Test

5.1.3.1 Finger Testing without Contact Object - Results:

The finger muscles were sequentially actuated to enable flexion and extension. The activation is done in

a square wave pattern, as seen in Figure 14 (bottom), and alternates the anterior actuation with the posterior actuation. The resulting joint angles (Figure 14 top) indicate that the model successfully captures contracting motion, with the finger curling when the anterior muscles are actuated, and extending when the posterior muscles are actuated. A screen capture of the animation of the testing is seen in Figure 13.

These test results, in conjunction with the test results shown in the previous section, indicate that the first goal of the paper has been satisfied (to simulate the curling and extending motion of the fingers through actuation of the muscles).

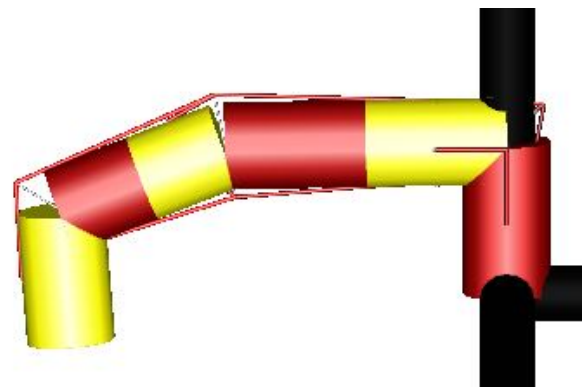


Figure 13: Animation of No Contact Finger Component Test

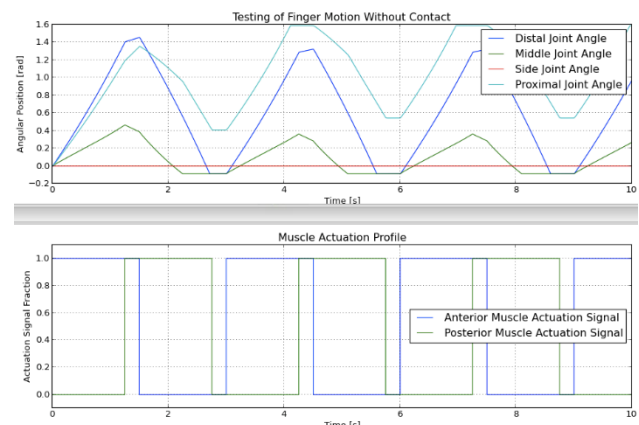


Figure 14: Curling and extension tests for Finger model

5.1.3.2 Finger Testing with the Contact Object - Results:

The previous test was repeated, but with the addition of a spherical contact object placed in front of the finger. The actuation profile is a staggered sequential activation from the proximal phalanx to the distal phalanx. The test's goal is to have the finger curl around the contact object when the object is positioned both directly in front of the finger, and positioned in front with a small offset to the side. As seen in Figure 15 below, when the object is directly in front, the finger curls around the object without slipping to the side. (Note that the middle

phalange and distal phalange contact the object at the same time, and that the side joint's angle is constant at zero).

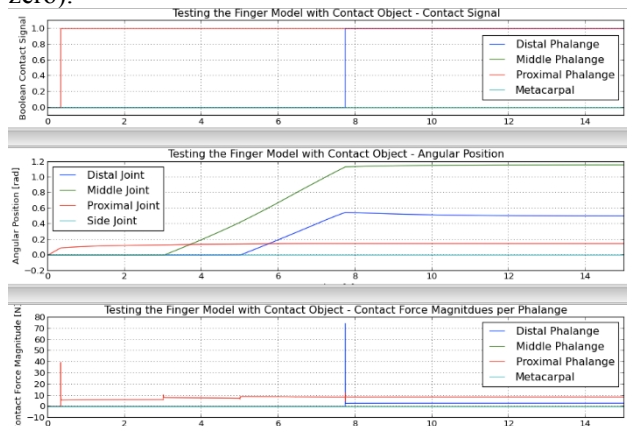


Figure 15: Finger contacting spherical object “head-on”

Figure 16 shows a screen capture of the animation of the head-on contact with the spherical object.

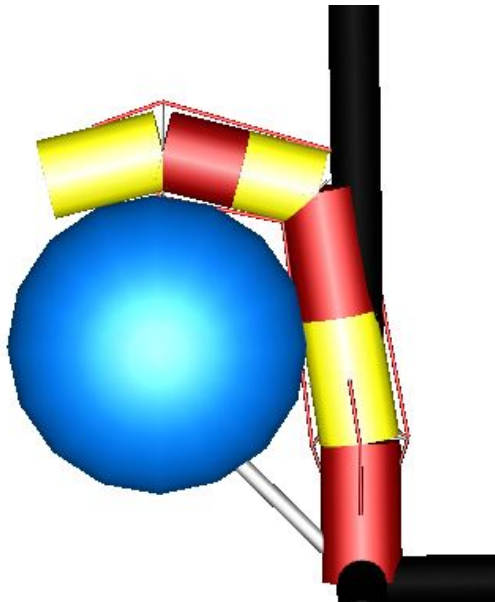


Figure 16: Animation of the Finger contacting spherical object “head-on”

The experiment is repeated with the object placed with a small offset. As seen in Figures 17 and 18 below, the finger still contacts the object, but proceeds to slide across the surface for a short period.

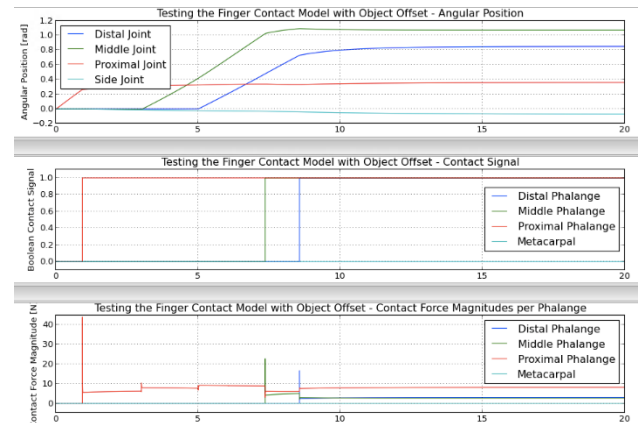


Figure 17: Finger contacting spherical object with offset

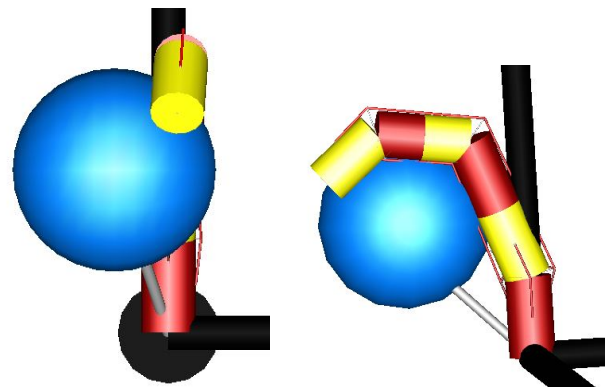


Figure 18: Animation of Finger contacting spherical object (front and side views)

The finger, during each trial, undergoes motion that conforms to expectation on how it should behave, and so is considered successfully tested.

A limitation observed is that the frictional force model, along with the high stiffness associated with the object's normal force, causes computational strain on the numerical solver during model simulation.

5.2 Hand Grasping Tests

The two grasps tested for in simulation were the prehensile and precision grasps. The precision grasp is a grasping motion that relies on relatively minimal muscle actuation to lightly hold the contact object; a prehensile grasp is when the muscles in the hand actuate to fully grab, and squeeze, the contact object in question (Cutkosky et. al., 1990).

The testing of the hand model, consisting of five finger component instantiations, is similar to the contact object test for the individual finger component. The muscles of the hand actuate, and cause the hand to contract. The purpose of the test is to determine if the hand can both perform a prehensile circular grasp around the ball, or a precision circular grasp around the ball. Separate actuation profiles were used for the prehensile grasp and the precision grasp.

Correspondingly, the object was placed at two different locations, depending on the type of grasp.

The results of the prehensile grasp (Figure 19) indicate that the fingers make and maintain contact with little sliding through the testing – all except the little finger – as indicated by the minimal movement in the side-joint's angle. The contact object's small radius of only 2.75 cm, in comparison to the fingers of average length 15 cm with diameter 2 cm, was chosen to demonstrate a typical hand grasp. The opposable fingers can maintain a firm grasp on the object, like how actual hands maintain holds on small objects. Some slipping occurs because there is not enough friction between the surfaces. The sliding motion could not be further reduced by increasing the friction, as doing so caused numerical issues; however, the slipping did not occur indefinitely, due to the side joint's resistance to motion (as mentioned in Section 3.2.1).

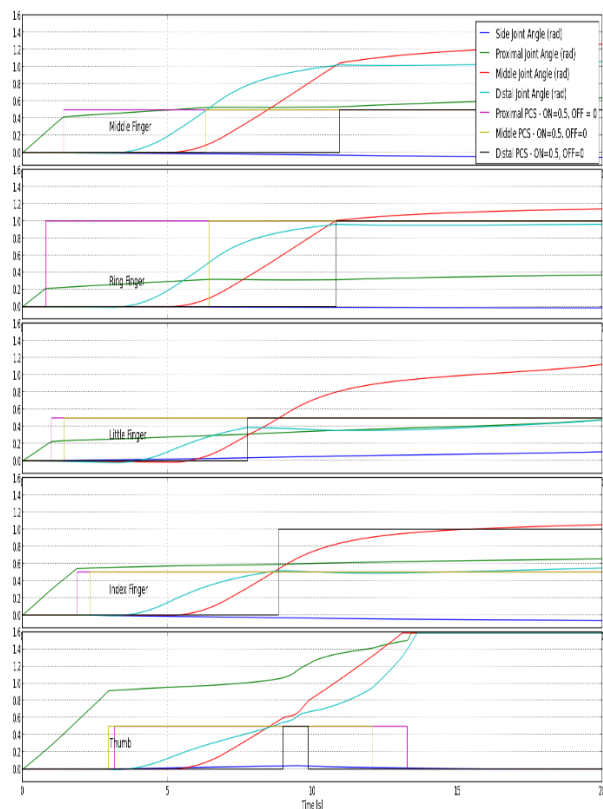


Figure 19: Prehensile Grasp Simulation

For the precision grasp, only the proximal phalanges were actuated, and the object was located closer to the distal phalanges. As seen from the results in Figure 20, contact occurs only as the distal phalanges – the little finger does not contact the object entirely, as it does not reach the object. Looking at the joint angles, we see that the proximal joints' motion stops soon after the distal phalange contacts the object. The distal phalanges' start to bend backwards upon with the object, as reasonably expected – once the distal phalanges stop bending backwards, the proximal phalanges' motion stops as

well. During the entire grasp, there is minimal slipping exhibited at all the contact points.

The hand testing, for both the prehensile and the precision grasps, displayed both the contracting motion and the grasping characteristics desired.

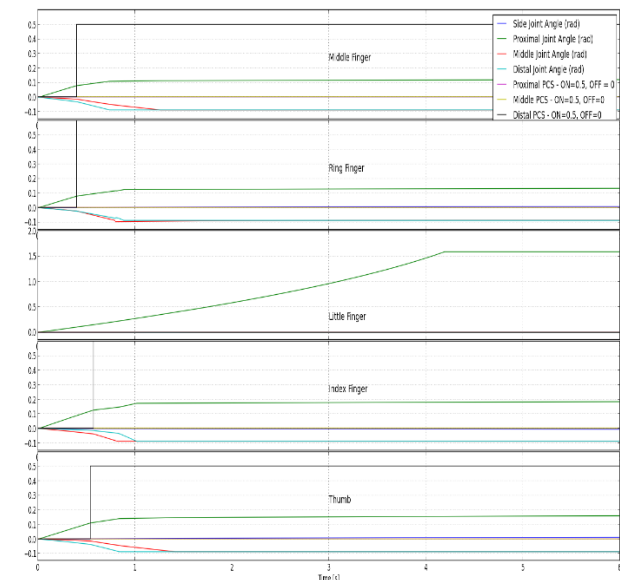


Figure 20: Precision Grasp Simulation

6 Conclusions

6.1 Results Summary

Implication with respect to Paper Goals:

The primary goals of this study were to be able to:

1. Simulate the curling and extension motion of the finger based on activation of the posterior muscles and anterior muscles.
2. Simulate different types of grasping motions; specifically - specifically simulate prehensile and precision grasping motions around a spherical object.
3. Visualize the simulation of the finger motions through three-dimensional animation.
4. Capture the contact forces on the fingers resulting from muscle actuation around the spherical object.

As seen from the results, goals 1, 2, 3 and 4 were successfully achieved.

6.2 Development Review

The initial construction of the model was done in Wolfram SystemModeler; the later models were developed in Dymola, leveraging its user-friendly refactoring and model creation capabilities. For compilation, debugging, and simulation capability, JModelica was used. OpenModelica was then separately utilized for animation (with simulation). All three services, apart from OpenModelica's lack of a

CVODES solver, did not require major tool-specific code alterations to run. This allowed for smooth transition between tools during model development and testing. The non-tool-specific functionality of Modelica, and the support for Modelica from the tools SystemModeler, Dymola, JModelica, and OpenModelica, were extremely useful for this package's creation.

6.3 Further Work

Much of this model was based off simplifications to get the models and simulations running – future work should focus on refining these model assumptions.

6.3.1 Musculoskeletal Model Improvement

Models of both the bone structure and the muscle geometry can be improved from its current state. The bone's physical parameters, including bone lengths and finger orientations, should reflect the actual structure of the bones. Currently, most parameters were chosen from an average of measurements taken of a group of volunteers; parameters not gathered from measurements were chosen arbitrarily. Using data selected from studies on hand dimensions and appropriately large ranges of volunteers would improve the validity of the resulting forces and motion involved in grasping. The muscles are currently modeled through linear line force segments between a limited number of attachment points to approximate the muscle's curve – this could be expanded by better representing the multiple interacting muscles and tendons, and their corresponding muscle geometries. Further improvements with the muscle actuation dynamics are also possible.

6.3.2 Contact Model Improvements

The current contact model is based around a spherical object; this should become more generalized, for multiple geometrical shapes and surfaces. Furthermore, the model currently assumes that each interacting bone will have a maximum of one tangential contact point, and that skin acts as a mere buffer – the models should account for the hand's relative flexibility and pliability. Lastly, the friction model is constructed as a type of surface damping, but it would be more appropriate to include Coulombic friction as well. As the contact object was developed in an ad-hoc manner, improvements can be made by integrating standard existing contact model approaches.

6.3.3 Testing Improvements

The testing of the hand grasping motion should be improved such that the muscle actuation profiles are not arbitrary pulses, but an imitation of natural activation profiles. Moreover, future work should also integrate experimental datasets from muscular grasps to make testing result analysis more accurate.

References

- Cutkosky, M. R., & Howe, R. D. (1990). Human Grasp Choice and Robotic Grasp Analysis. In S. T. Venkataraman & T. Iberall (Eds.), *Dextrous Robot Hands* (pp. 5-31). Springer-Verlag.
- Griffin, W. B., Findley, R. P., Turner, M. L., & Cutkosky, M. R. (2000). Calibration and Mapping of a Human Hand for Dexterous Telemanipulation. *ASME IMECE 2000 Conference Haptic Interfaces for Virtual Environments and Teleoperator Systems Symposium*. Retrieved from http://www-cdr.stanford.edu/DML/publications/griffin_asme00.pdf
- Adler A, "GM-NASA Space Robot Partnership brings "Power" Glove to Life", GM Corporate News announcement, 2017 July 6th, <http://media.gm.com/media/us/en/gm/home.detail.html/content/Pages/news/us/en/2016/jul/0706-gm-nasa.html>
- Linn D. M., Ihrke A. C., Diftler M. A., "Human grasp assist device and method of use", US Patent No. 8255079 B2, 2012.
- Polygerinos P, , Galloway K. C., Savage E., Herman M., O' Donnell K, and Walsh J. C., "Soft Robotic Glove for Hand Rehabilitation and Task Specific Training", 2015 IEEE International Conference on Robotics and Automation, May 2015, doi: 10.1109/ICRA.2015.7139597
- Yanchev T, "Power Assist Gloves", <https://www.youtube.com/watch?v=gzfZCTYREww> 2015
- van Nierop, O. A., van der Helm, A., Overbeeke, K. J., & Djajadiningrat, T. J.P. (2007). A natural human hand model. *The Visual Computer*, 24(1). <http://dx.doi.org/10.1007/s00371-007-0176-x>
- Gustus, A., Stillfried, G., Visser, J., Jörntell, H., & van der Smagt, P. (2012). Human hand modelling: kinematics, dynamics, applications. *Biological Cybernetics*, 106(11). <http://dx.doi.org/10.1007/s00422-012-0532-4>
- Wan Tarmizi, W. F. B., Elamvazuthi, I., & Begam, M. (2009). Kinematic and Dynamic Modeling of a MultiFingered robot Hand. *International Journal of Basic & Applied Sciences*, 9(10). Retrieved from <http://ijens.org/index.htm>
- Marieb, E. N. (2000). *Essentials of human anatomy and physiology* (6th ed.). San Francisco: Benjamin Cummings.
- Otter, M., Elmquist H, Mattson S. E., "The New Modelica Multibody Library", *Proceedings of the 3rd International Modelica Conference*, Linköping, 2003
- Tiller, M. (2014). *Modelica by Example*. Retrieved from <http://book.xogeny.com/>
- Modelica® (2013) - A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification <https://modelica.org/documents/ModelicaSpec32Revision2.pdf>
- Hicks J. L., Uchida T.K., Seth A., Rajagopal A., Delp S.L., "Is my model good enough? Best practices for verification and validation of musculoskeletal models and simulation environment", *Journal of Bioengineering*, Vol 137, Feb 2015.
- Radder, B., Kottink AIR, van der Vaart N, et. al, "User-centred input for a wearable soft-robotic glove supporting hand function in daily life", 2015 IEEE International

Conference on Rehabilitation Robotics (ICORR), Singapore, 2015, doi: 10.1109/ICORR.2015.7281249

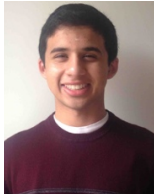
Dymola (2017) Copyright © Dassault Systèmes, 1992-2016
<http://www.3ds.com/products-services/catia/products/dymola/>

SystemModeler (2015) Copyright © 2015 Wolfram Research, Inc. <http://wolfram.com/system-modeler/>

OpenModelica (2016) Copyright Open Source Modelica Consortium (OSMC) <https://www.openmodelica.org/>

JModelica (2016) from <http://jmodelica.org/>

Author's Notes



I was first introduced to Modelica during an internship at Xogeny in the summer of 2014, as a freshman in high school. This led to me modeling a flat linear motor I had previously built by hand. The linear motor consisted of multiple solenoids sequentially activated by an Arduino, to propel a cart down metal tracks. It was developed in an ad hoc manner, and adjusting the design was difficult; I felt that modeling was a nicer way of doing these kinds of tasks.

When noticing elderly persons having trouble opening doors and holding jars, I began thinking of ways to devise an orthotic glove to assist them. To gain a better understanding of the system, I decided to first develop a mathematical model of the hand. I began working on the PowerGrab project as a summer intern at Modelon KK, under the supervision of Dr. Andreasson. Since then, I have continued to work on developing the code for the PowerGrab library, as well as writing the corresponding paper, with the continued guidance of Dr. Andreasson's comments and feedback.

My Background:

Shashank was born in Michigan, USA, and is currently a senior in high school. He was a summer intern at Modelon KK and Modelon, Inc., and is currently in the Wolfram Mentorship Program.

Modelica Spur Gears with Hertzian Contact Forces

Markus Dahl¹ Håkan Wettergren¹ Henrik Tidefelt¹

¹Wolfram MathCore, Linköping, Sweden, {markusd,hakanw,henrikt}@wolfram.com

Abstract

To be able to capture the dynamics of entire systems is one of the strengths of the Modelica language. This article will examine the possibility of modeling spur gears in the Modelica environment Wolfram System-Modeler, and integrating them with other rotating machinery elements, such as roller bearings and flexible shafts. The contact forces between spur gears are based on the Hertzian Theory of Contact¹

Keywords: Spur Gear, Hertz Contact Theory, Rotating Machinery

1 Introduction

Gear contact forces can be accurately modelled by FEM programs, but usually at a high computational cost. The focus is usually on solving the force equations statically, where some dynamics might be lost. A common way of calculating the dynamics is to add a so called *application factor* to the static solution, approximating the dynamic result. By using a Modelica model instead, the dynamics can be included in the model, replacing the application factor. To be compatible with other libraries, the models here are based on the MultiBody library from the Modelica Standard Library. The choice of using a 3D mechanical library instead of libraries such as PlanarMechanics (Zimmer, 2012), is to be able to keep building on these models to handle helical and other type of gears.

Another benefit of Modelica models is that they can be used with other rotating machinery elements. Let's say that a wind turbine gear box should be analyzed. This gear box contains inner and outer spur gears, flexible shafts, and roller bearings. If a Modelica model is used for this purpose, we can see both how the spur gears affect the bearings, the shafts, as well as how the shafts and bearings affect the gears. Therefore, the dynamics of the entire system can be captured and analyzed.

In Section 2, the gear geometry of a spur gear will be introduced. Following that, Section 3 will be an introduction to the Hertzian Contact theory. Section 4 will explain how this was implemented in Modelica. Section 5 will go through some examples where the gears were used. Finally a discussion of the results will follow in Section 6.

1.1 Previous work

Several papers have been written regarding modeling gears in the Modelica language. Special attention has been seen in the area of powertrains. One of the original papers is (Otter et al., 2000) where the PowerTrain package was presented.

The package PowerTrains contains 1-dimensional, rotational mechanical systems. I.e. a lot of simplifications have been made to be able to model the complete driveline. At that time it was a state of the art approach. However, the very idealized components in that library cannot be used for any advanced dimensioning or root cause analysis. The description of the components are very simplified compared to the special simulation tools that exist in each specific machine element area.

During the years, several of the components have been refined. One work is (van der Linden and de Souza Silva, 2009) where a 3-DOF elastic model was used which included the elasticity of the support bearings in the load direction, which was not possible in the standard gear model. The model was then extended in (van der Linden, 2012) to also include a Gear Contact Model. A later paper, (van der Linden, 2015) compared the results from a Modelica model that investigated gear contact to tests.

A much more detailed approach was taken by (Kosenko and Gusev, 2011) and further improved in (Kosenko and Gusev, 2012), where the forces between gears were modelled with high detail in a Modelica environment.

2 Gear Geometry

This section describes the geometrical modeling of two gear wheels that are in contact or in close proximity. Starting with the geometry of a single gear wheel, we then proceed with the geometry of the interaction between two gear wheels, before going into more advanced topics such as tip relief and the geometry involved in triggering events in a Modelica model.

2.1 Geometry of a gear wheel

A gear is basically a toothed wheel aimed to transmit rotation from one shaft to another. Spur gears, that is the focus in this article, can be described as parallel-axes gears without a helical angle. The gears can be of

¹As found in Roark's Formulas for Stress and Strain, 2002

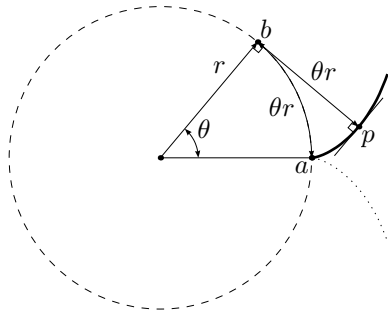


Figure 1. Involute of a circle. The involute (thick line) is traced by mapping each angle θ to the point p by going the distance θr along the circle from point a to point b , and then going the same distance back along the tangent to point p . Note the right angle at point p . The involute obtained by this procedure creates an involute corresponding to a rear flank of a gear tooth. Mirroring the procedure creates a front flank, drawn with a dotted line.

Table 1. Gear wheel geometry parameters.

Parameter	Description
z	Number of teeth in gear
m	Gear module
$\alpha_0 = 20^\circ$	Reference profile angle
x	Profile shift factor

two types, inner gear (a ring with teeth on the inside) and outer gear (a wheel with teeth on the outside). For clarity of presentation, only outer gears are considered in this section.

One of the main reasons for the broad use of gears is the efficiency of the transmission, which depends on shape of the teeth. The most common shape of a tooth is a circle involute. A circle involute or simply *involute*, is a curve following the end point of a tangent that is rolled up from a circle. It is defined by the geometry in Figure 1. The right angle at point p , between the tangent of the circle and the tangent

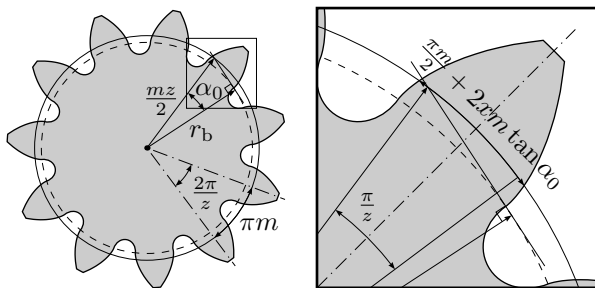


Figure 2. Gear wheel geometry parameters in an outer spur gear. The radius of the base circle, r_b is easily derived from the gear wheel parameters z , m , and α_0 . The effect of the profile shift, x , is best understood in relation to a straight gear reference profile, but the derivation is out of scope in the current presentation.

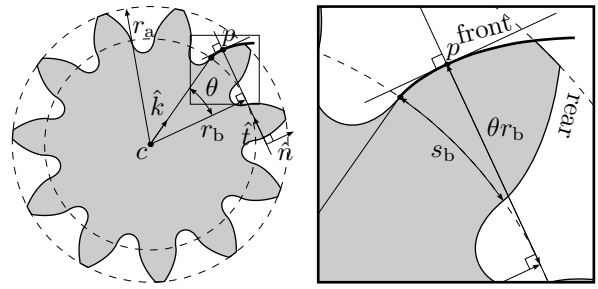


Figure 3. Intersection between gear flank and a tangent of the base circle. Also showing the derived quantity s_b , and the additional parameter r_a . Vector \hat{k} points to the point on the tooth where the involute begins. The point p is then obtained from the rolling angle, θ , and r_b . The front flank is where force is transmitted when applying torque in counter-clockwise direction, and the rear flank when applying clockwise torque.)

of the circle involute, is a fundamental property of involute curves.

There are four geometrical parameters that need to be specified for a spur gear in our case, listed in Table 1. These parameters are shown in Figure 2. From these user-specified parameters, many other quantities are derived, see Figure 3. For example, one can derive the tooth base thickness, s_b , and also express a standard choice of r_a , as

$$s_b = \left(\frac{\pi}{2} + 2x \tan \alpha_0 + z \operatorname{inv} \alpha_0 \right) m \cos \alpha_0 \quad (1)$$

$$r_a = m \left(\frac{z}{2} + x + 1 \right) \quad (2)$$

where $\operatorname{inv} \alpha_0 = \tan \alpha_0 - \alpha_0$.

The signed rolling angle, θ , is related to the inner product of the unit vectors \hat{k} and \hat{t} ,

$$\langle \hat{k}, \hat{t} \rangle = \cos \left(\frac{\pi}{2} - \theta \right) \quad (3)$$

from which it can be solved reliably.

2.2 Line of contact

Contact between two gears always occur for either front-front flank contact, or rear-rear flank contact. For each of the two contact cases (front or rear), there is a *line of contact* (LoC), along which the contact between the teeth will be located. The front and rear contact cases are analogous, and to avoid going into details about sign conventions, only the front case will be considered from here on. Using the wheel positions, the distance between these points, a_w can be related to the angle of LoC, α_w . See Figure 4.

$$a_w = \frac{m}{2} (z_1 + z_2) \frac{\cos \alpha_0}{\cos \alpha_w} \quad (4)$$

Here, the indices 1 and 2 mean gear wheel 1 and gear wheel 2 respectively. The LoC normal (in two

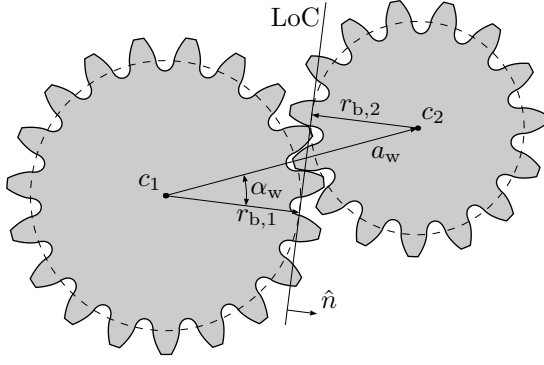


Figure 4. Line of contact of two gears, for contact between front flanks on the teeth (transmitting force when either gear is driving with positive direction of rotation). The LoC for contact between rear flanks is obtained by reflection in the line through the gear centers.

dimensions), \hat{n} , can then be expressed using the angle α_w , as

$$\cos \alpha_w = \frac{(r_{b1} + r_{b2})}{\|a_w\|} \quad (5)$$

$$\sin \alpha_w = \pm \sqrt{1 - \cos^2 \alpha_w} \quad (6)$$

$$\hat{n} = \begin{pmatrix} \cos \alpha_w & \sin \alpha_w \\ -\sin \alpha_w & \cos \alpha_w \end{pmatrix} \frac{a_w}{\|a_w\|} \quad (7)$$

where the sign of $\sin \alpha_w$ reflects the choice between LoC for clockwise or counter-clockwise rotation.

A point p on the flank of a tooth can now be described using α_w , the LoC, r_b , and the center position of the gears, c_i .

In order to determine contact forces between two gear flanks, we define an indentation depth, δ measuring the amount of intersection between the teeth. The depth is modeled using the points p_1 and p_2 , where the gear flanks intersect with the LoC.

2.3 Tip relief

To get a smooth contact force, a modification of the tip is usually done, called a *tip relief*. This can be done by removing a small portion of the tooth, as shown in Figure 5. The LoC will be affected by this modification, but the change is on the scale of 0.01

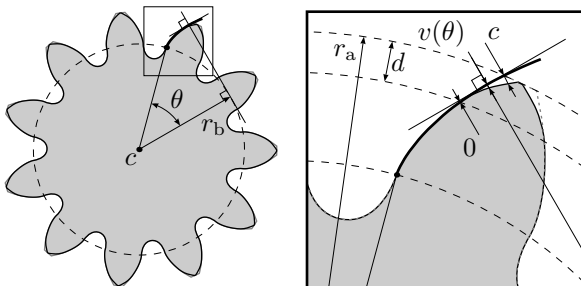


Figure 5. Gear with tip relief (exaggerated).

modules, which makes the effect on the LoC negligible. The new tooth shape with tip relief can then be calculated by the following standard equation:

$$v(\theta) = q(\theta - \theta_a) \quad (8)$$

where q is a coefficient to ensure that the amplitude is obtained correctly, and θ_a is the maximum value for the roll angle. How much of the tooth that is removed can be specified by setting the distances c and d , defining the tip relief amplitude and tip relief length, respectively.

2.4 Tooth pair activation

A pair of teeth in contact are identified by one integer index on each gear, i_1 and i_2 . Together with the rotations of the axes on which the gears are mounted, φ_1 and φ_2 , the directions k_1 and k_2 pointing at the starting points of the involutes on the base circles follow, which in case of front flank contact are given by

$$\hat{k}_1 = \begin{pmatrix} \cos \beta_1 \\ \sin \beta_1 \end{pmatrix} \quad \beta_1 = \frac{s_{b,1}}{2r_{b,1}} + i_1 \frac{2\pi}{z_1} + \varphi_1 \quad (9)$$

$$\hat{k}_2 = \begin{pmatrix} \cos \beta_2 \\ \sin \beta_2 \end{pmatrix} \quad \beta_2 = \frac{s_{b,2}}{2r_{b,2}} + i_2 \frac{2\pi}{z_2} + \varphi_2 \quad (10)$$

As was shown in Figure 3, the rolling angles follow for any given direction of the LoC. In Figure 6, the front flanks corresponding to indices i_1 and i_2 are marked with a thick line. As the gears rotate, the current tooth pair will become disengaged, while other pairs will become engaged. An index skip, i_Δ , is chosen by upward rounding of the gear contact ratio. Depending on the direction of rotation, the next tooth pair to follow when the current pair has become disengaged is selected as

$$i'_1 = i_1 \pm i_\Delta \quad (11)$$

$$i'_2 = i_2 \mp i_\Delta \quad (12)$$

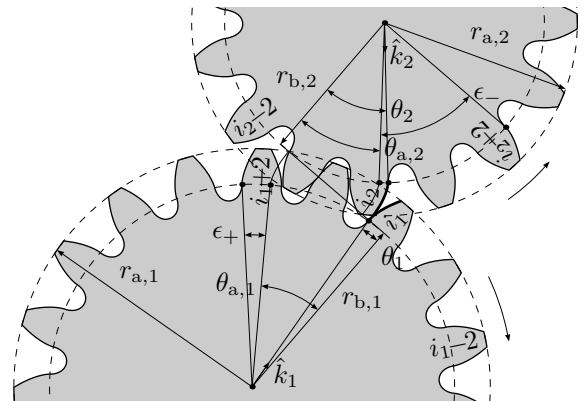


Figure 6. Slack variables used to control tooth pair activation for front flank contact. Here, $i_\Delta = 2$, corresponding to a configuration for a contact ratio between 1 and 2.

In Figure 6 where gear 1 rotates clockwise, the change in i_1 is with positive sign, while the change in i_2 is with negative sign. The slack angle ϵ_+ measures how much the rolling angle θ_1 may decrease until it is absolutely necessary to consider interaction for the tooth pair i'_1 and i'_2 . Analogously, when gear 2 rotates clockwise, ϵ_- measures how much the rolling angle θ_2 may decrease until it is absolutely necessary to consider interaction for the tooth pair in the opposite direction. The slack angles are given by

$$\epsilon_+ = \theta_1 + i_\Delta \frac{2\pi}{z_1} - \theta_{a,1} \quad (13)$$

$$\epsilon_- = \theta_2 + i_\Delta \frac{2\pi}{z_2} - \theta_{a,2} \quad (14)$$

In the Modelica model presented in Section 4, the tooth indices are updated such that the slack angles are positive at all times. For robust simulation, it is desirable to update indices with some margin until it is absolutely necessary according to the slack variables. That is, we should avoid triggering the update at the lower bound 0. When indices are updated because one of the slacks is getting too close to zero, that slack variable will be reset to a large value, while the other slack variable will be significantly reduced. We will trigger based on the following conditions, where γ is a positive constant which remains to be determined,

$$\epsilon_+ < \gamma \left(\epsilon_- - i_\Delta \frac{2\pi}{z_2} \right) \quad \text{Trigger positive change in } i_1 \quad (15)$$

$$\epsilon_- < \gamma \left(\epsilon_+ - i_\Delta \frac{2\pi}{z_1} \right) \quad \text{Trigger negative change in } i_1 \quad (16)$$

In order to avoid endless event iteration when updating the indices, it must be ensured that triggering a change in one direction does not reduce the other slack so much that it satisfies the condition for re-triggering a change in the opposite direction. By equating the margin to the lower bound of zero slack, with the margin to re-triggering a change in the opposite direction, a natural choice of $\gamma = \frac{\sqrt{5}-1}{2}$ is obtained.

3 Hertz Contact Stress

To calculate the force between two gear teeth, Hertzian Contact Theory has been used. At the point of contact, the two teeth are approximated by two cylinders with parallel axes, see Figure 7.

The indentation depth, δ , is related to the contact force by

$$\delta = \frac{2F(1-\nu^2)(\frac{2}{3} + \log(\frac{4R_1}{b}) + \log(\frac{4R_2}{b}))}{\pi L E_{\text{mod}}} \quad (17)$$

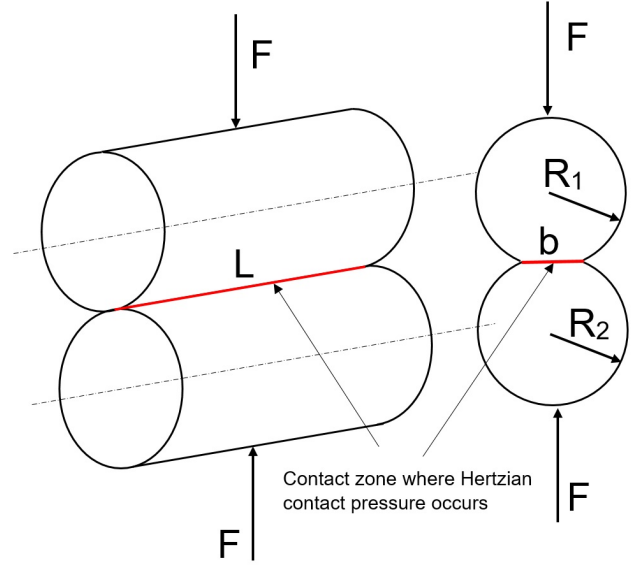


Figure 7. Contact between two cylinders with variables.

where F is the force, ν is the Poisson ratio, E_{mod} is Young's modulus, L is the length of the cylinder, R_i are the radii of the cylinders, and b is the contact width². The contact width is modeled by

$$b = \sqrt{\frac{32F}{\pi L E_{\text{red}} (\frac{1}{R_1} + \frac{1}{R_2})}} \quad (18)$$

Here, E_{red} is a combination of the two gear wheels' material parameters

$$\frac{1}{E_{\text{red}}} = \frac{1}{2} \left(\frac{1-\nu_1^2}{E_{\text{mod}1}} + \frac{1-\nu_2^2}{E_{\text{mod}2}} \right) \quad (19)$$

From (17) and (18), the force can be calculated as a function of the indentation depth. The geometry of the cylinders in contact will change when moving on the LoC. This means that the curvature radii R_1 and R_2 will be changing so that one of them will start with a small radius and increase when moving along the LoC, and the other will start with a big radius and decrease when moving along the LoC.

The conditions for applying a force at the appropriate time is to check if the distance between the wheel center c_i and the point p_i is less than the top radius of the gear $r_{a,i}$. Additionally, we also check if δ is larger than zero, i.e.

$$|c_1 - p_1| < r_{a,1} \quad (20)$$

$$|c_2 - p_2| < r_{a,2} \quad (21)$$

$$\delta > 0 \quad (22)$$

The effects of gear damping has not been included in this model.

²As found in Roark's Formulas for Stress and Strain, 2002, Table 14.1.2

4 Modelica Model

The implementation of the spur gear model in Modelica is using the Modelica MultiBody library. This means that there will be 3D animations for all models. For example, the contact force is visualized by arrows between two teeth, as seen in Figure 8.

The topology of the model consists of a top layer diagram where the layout is specified, i.e. how the gear wheels are positioned and if a force should be calculated between them. Figure 9 shows the Modelica diagram layer of a simple two wheel model with contact between them.

The force between the two gear wheels is calculated inside the `gearForceCalculation` component. The system of non-linear equations (17) and (18) has multiple solutions, and the correct one is not

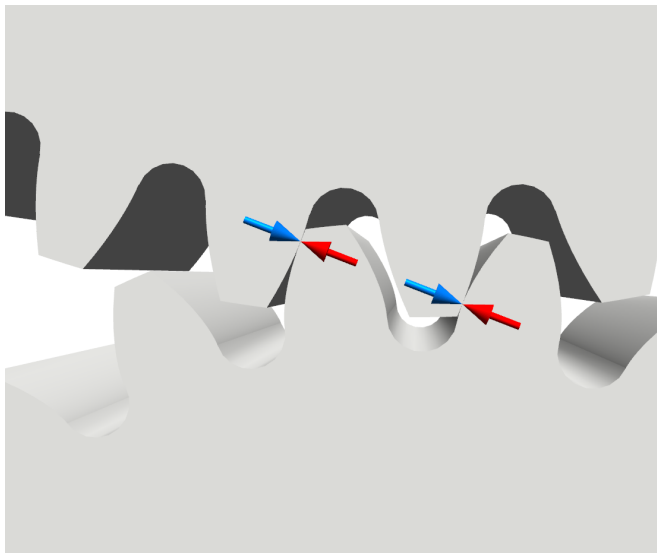


Figure 8. Two outer spur gears in contact, with arrows representing the contact forces acting on the gears. At this moment, two teeth pairs are in contact.

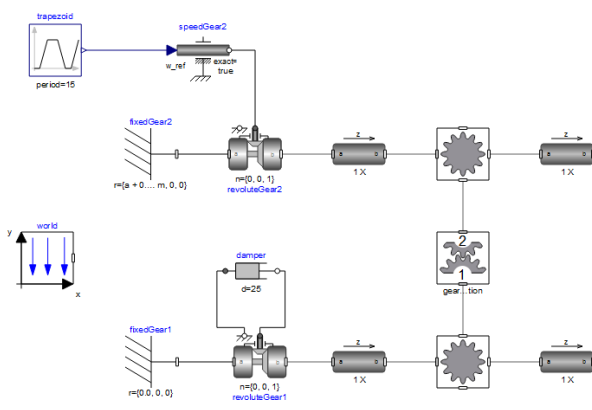


Figure 9. Model of two gear wheels in contact.

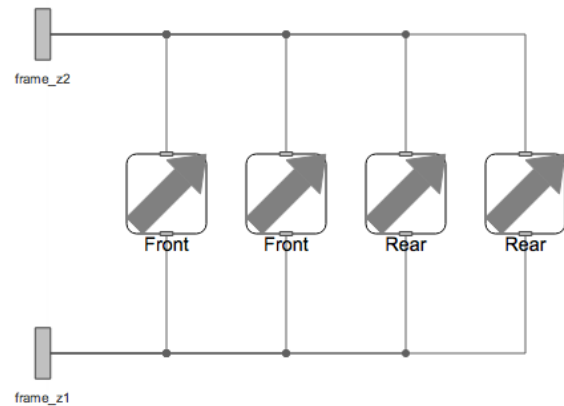


Figure 10. Model of the component where gear forces are calculated, containing four `ContactForcePoints`.

differentiable at $\delta = 0$. To handle this, the solution is approximated by a closed-form expression. The `gearForceCalculation` class contains at least four `ContactForcePoint` components, as seen in Figure 10. Each of these `ContactForcePoints` is responsible for calculating one force pair on one matching teeth pair. Since the gear ratio should be above one, but less than two in the case of two outer spur gears, two force pairs are needed. Two more force pairs (`ContactForcePoint` components) are needed due to the two flanks on each tooth. If a gear ratio over 2 is possible, as in the case of a planetary gear between the ring and a planet, more `ContactForcePoints` can be added to handle this.

The contact ratio will be calculated automatically, depending on the geometry of the two gear wheels in contact. This is used to assert that the `GearForceCalculation` component is set up correctly.

Many parameters can be set by set user on the `GearForceCalculation` component that will affect the `ContactForcePoints` inside. The parameters are listed in Table 2.

Table 2. `GearForceCalculation` parameters.

Parameter	Description
m	Gear module
L	Contact width of wheels
z_i	Number of teeth in gear wheel i
x_i	Profile shift factor for gear wheel i
ν_i	Poisson's ratio for wheel i
E_{mod_i}	Young's Modulus wheel i
c_i	Tip relief amplitude for wheel i
d_i	Tip relief length for wheel i

5 Example Models

Figure 9 showed a simple example with two outer gears in contact, and no other components in the system. This can easily be expanded.

5.1 Effect of tip relief

Using the presented gear model, it is possible to compute dynamic effects in gears such as the variation in contact stresses. Figure 11 shows the contact stress for one tooth during a contact. In this case the gear wheel support is fixed in all translational directions, which means that there are no external vibrations affecting the result.

The contact starts with a transient, stabilizing to a lower force level, where two pairs of teeth in contact. Then the other pair of teeth goes out of contact and only the current tooth takes all force. After a while, a new teeth pair will go into contact, and the force level will drop again. Finally the current teeth pair goes out of contact, and the force drops to zero. The reason for the transient is that all pair of teeth in contact will have the same indentation depth, δ . Hence, when a new pair of teeth goes into contact, the initial indentation depth at the tip of the incoming tooth will be that of the one pair of teeth currently in contact. The gears will then quickly adjust to the same total torque of two pairs of teeth. To avoid the transient, the tip of the involute shape is modified with a *tip relief*, as was shown in Figure 5.

Figure 12 shows the much smoother contact pressure with tip relief.

5.2 Wind turbine gear box

A planetary gear box can be created by combining inner and outer spur gears with flexible shafts. A two-stage gear box is then connected to the planetary gear box. A screenshot of the animation is shown in Figure 13 and the model diagram is shown in Figure 14. This setup is capable of changing the angular velocity from the slow rotation of the blades, around

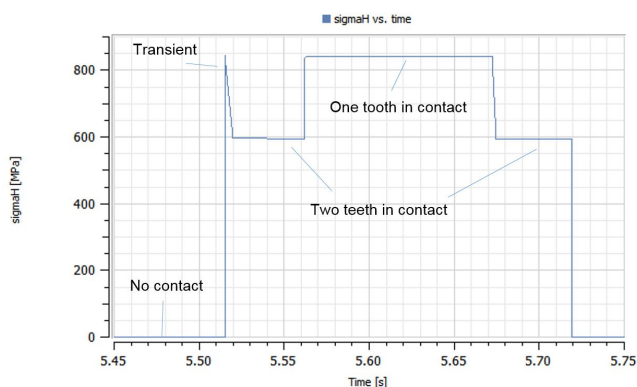


Figure 11. Contact pressure between two teeth without tip relief.

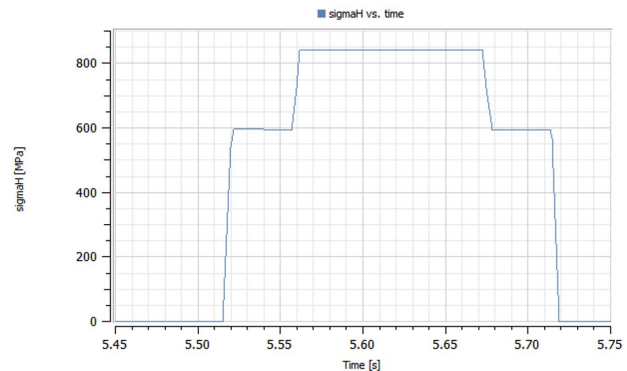


Figure 12. Contact pressure between two teeth with tip relief.

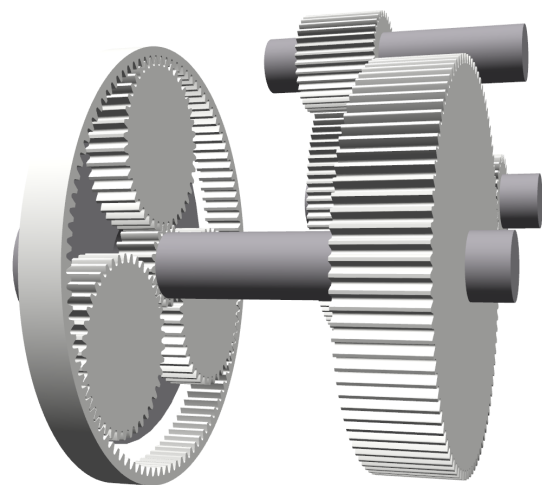


Figure 13. Animation of the wind turbine gear box model.

10 rpm, to the fast rotation of the generator, around 1500 rpm.

The contact forces are calculated at many different points in this system. In the planetary gear there are forces between the center gear wheel (the sun) and the three outer wheels (the planets), and also between the three planets and the inner gear wheel (the ring). With a possible contact ratio above 2, the `GearForceCalculation` components between inner and outer gears (planet and ring) contain 6

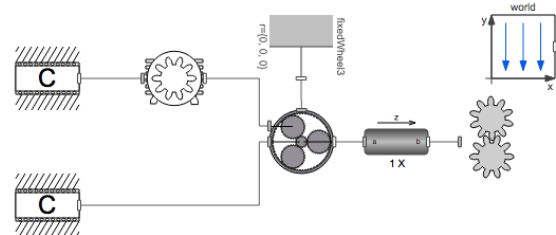


Figure 14. A wind turbine gear box.

ContactForcePoints. In total, the system contains 38 **ContactForcePoint** components, making it possible to simulate this system with gear forces at all points and all rotational directions. The gear can be driven by any wheel, both clockwise and counter-clockwise. In addition to this, the wheels are connected to flexible shafts that are fixed in a support.

In this setup, one could detect if there are any peaks in forces between the gears, or if the flexible shafts affect the gear in any significant way if the material parameters for the beams changes. An example of contact pressures in a steady state of the gear box can be seen in Figure 17 on page 9.

5.3 Gear and bearings

Since rotating machinery elements affect each other in various ways, it is important to be able to study different elements together. An example of such and interaction is when a system containing rotating flexible shafts with bearings, that sets the gear wheels in motion. The bearings contain cylindrical rollers, which have Hertzian contact forces between the inner and outer ring of the bearing. A 3D visualization of the setup is shown in Figure 15.

Two flexible shafts are supported by roller bearings. The bearings on the lower shaft are mounted on flexible supports, that are fixed to the ground. The lower shaft rotates at 600 rpm. One of the bearings has an outer ring defect (top right bearing in figure 15). The visualization of the outer ring in this bearing has been removed to give a better view of the rollers inside the outer wheel. This defect will in this case

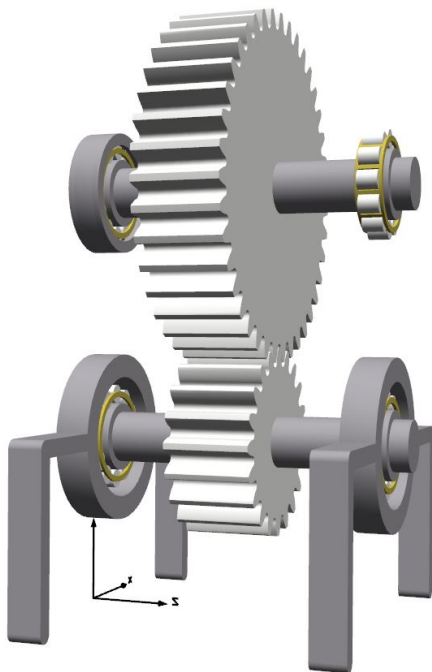


Figure 15. Two shafts with bearings, connected by spur gears.

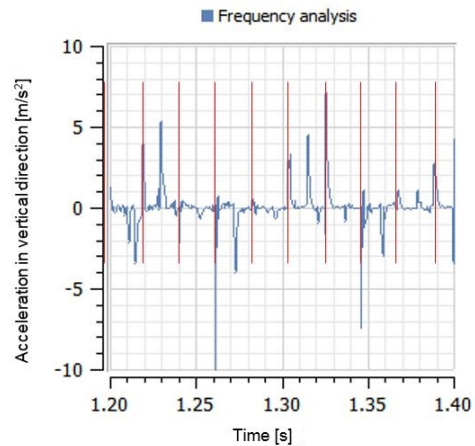


Figure 16. Acceleration in the vertical direction, with a red highlighting at impact points, as predicted by the shock pulse method.

cause an extra downward force to be applied at the "12 o'clock" position when a roller passes this point.

Vibration analysis of a damaged bearing is usually a quite complicated task. A frequency spectrum will normally not show a small bearing defect. Instead different kinds of shock pulse methods have been developed. The signals are normally at rather high frequencies. In this example the time of impact has been marked with red lines. As can be seen in figure 16, the accelerations and impacts align at most points. This analysis can also be done to investigate a damaged gear.

The benefit of being able to do this simulation analysis is huge. Understanding where and how large a damage is gives a better picture of what and when an overhaul should be done, often saving a lot of money as well as improving the overhaul. A typical example of where this is vital is in paper machines where a carefully planned overhaul may save millions of dollars. Another example is in cruise ship machines, where a dry-dock needs to be available for an overhaul and passengers might need to rebook their trip to other ships, depending on overhaul time.

6 Conclusions

With this implementation of a gear model, a high accuracy causal model of contact forces can be put in a multi-domain simulation environment. This can then be used to find weak points in complex rotating machinery systems. Key points are summarized below.

- The physical accuracy is at the level where factors like shaft position, clearances, tip relief, misalignment and vibrations by default are taken into account. Essentially, this approach combines the simplicity of drag-and-drop, and multi-domain modeling in Modelica, as in (van der Linden, 2012), with high accuracy calculations of gears, as in (Kosenko and Gusev, 2012).
- The drag-and-drop capability of Modelica and Wolfram SystemModeler, makes the creation of a custom model very easy. All effort required from a user is to parametrize the models.
- The formulation of the gear contact shown in this paper is designed to be simple to expand into more complex contacts. In other words it is easy to take deviations from the ideal involute gear into account. Tip relief was used here as an example. Another effect of this is that the gear formulation can also easily be expanded to helical gears, bevel gears and worm gears. Effects such as contact roughness, and manufacturing errors can also be included in the future.
- Domain specific software, such as gear or bearing design software, gives very accurate results for a specific machine element, but are limited when an extension outside the domain is needed. The examples presented in this paper shows how efficient several different machine elements can be combined, as well as coupling to other domains. For instance, the applied torque in the models was obtained using a PID controller.
- The wind turbine example showed how important tip relief is for avoiding excessive wear. Using a software not able to include tip relief may lead to a bad geometrical design and high costs in the correction process. This is particularly true if the wear is detected after some time in operation.
- What is missing and can be included in future work, is testing and verification of the modeling results as done in (van der Linden, 2015), as well as a speed up of the simulation time. Today, the simulation time is around real-time for simple models (2 spur gears in contact, both connected on flexible shafts), and slower for more complex models, depending on speed of rotation. Tooth

bending has been neglected in this model and should be included in the future

References

- Ivan Kosenko and Il'ya Gusev. Implementation of the spur involute gear model on modelica. *Proceedings of the 8th International Modelica Conference*, 2011. URL https://www.modelica.org/events/modelica2011/Proceedings/pages/papers/13_3_ID_117_a_fv.pdf.
- Ivan Kosenko and Ilya Gusev. Revised and improved implementation of the spur involute gear dynamical model. *Proceedings of the 9th International Modelica Conference*, 2012. doi:10.3384/ecp12076311.
- M. Otter, M. Dempsey, and C. Schegel. Package PowerTrain: A Modelica library for modeling and simulation of vehicle power trains. *Modelica Workshop 2000 Proceedings*, pages 22–32, 2000. URL https://www.modelica.org/events/workshop2000/index_html/proceedings/Otter.pdf.
- F.L.J. van der Linden. Modelling of elastic gearboxes using a generalized gear contact model. *Proceedings of the 9 International Modelica Conference*, 2012. doi:10.3384/ecp12076303.
- F.L.J. van der Linden. Modeling of geared positioning systems: An object-oriented gear contact model with validation. *Proceedings of the Institution of Mechanical Engineers*, 2015. doi:10.1177/0954406215592056.
- F.L.J. van der Linden and P.H. Vazques de Souza Silva. Modelling and simulating the efficiency and elasticity of gearboxes. *Proceedings 7th Modelica Conference*, 2009. doi:10.3384/ecp09430052.
- Dirk Zimmer. A planar mechanical library for teaching modelica. *Proceedings of the 9th International Modelica Conference*, 2012. doi:10.3384/ecp12076681.

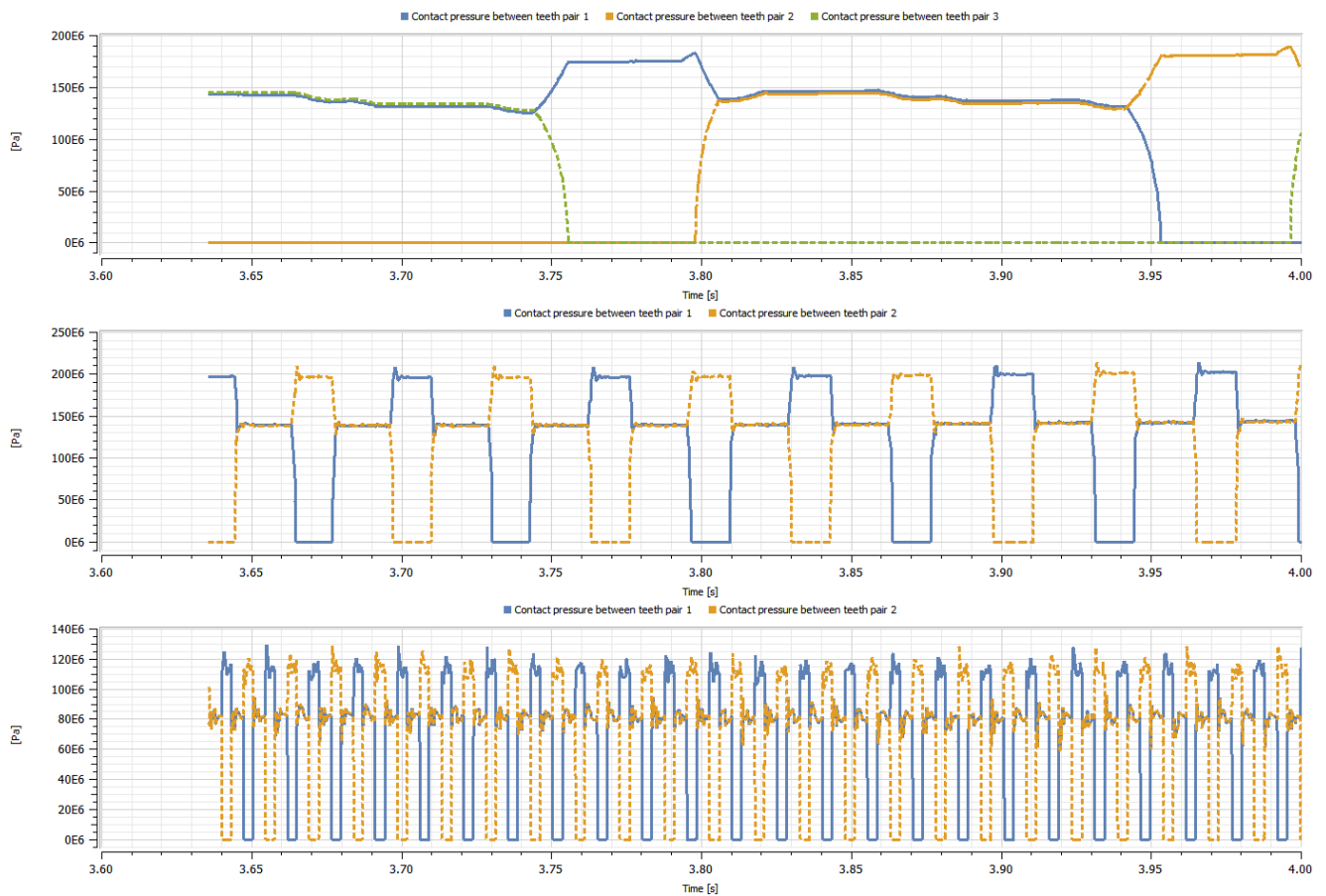


Figure 17. Contact pressure on teeth at three places in the wind turbine gear box. The first plot is showing the pressure of one planetary wheel to the ring. The second plot shows the pressure between teeth in the first step in the two-step gear box. Finally the third plot shows the pressure between teeth in the last step before the generator. One can easily see how the frequency changes from the planetary gear to the last step before the generator.

Modeling of Roller Bearings

Dipl.-Ing. Tobias Weiser¹ Univ.-Prof. Dr.-Ing. Burkhard Corves²

¹KUKA Robotics GmbH, Germany, Tobias.Weiser@kuka.com

²Department of Mechanism Theory and Dynamics of Machines, RWTH Aachen, Germany,
corves@igm.rwth-aachen.de

Abstract

Modeling of multi-body system mechanics plays a central role in the design of mechatronic systems. Roller bearings contribute stiffness and damping to the system dynamics of a mechatronic system. This article shows the stiffness modeling of selected types of roller bearings. The kinematics of deformation of a roller bearing are shown. Based on the principle of Hertz'ian contact stress the elastic forces and torques are calculated. These forces are considered and implemented in the MultiBody Library.

Keywords: *Bearing Stiffness, Bearing Modelling, Multi-Body Library*

1 Introduction

This document discusses the modeling of stiffness for various types of rolling-contact bearings for use in the simulation of multi-body systems. In addition to this macroscopic perspective, the modeling of stiffness is used for rotordynamics. Beyond this, research in bearing modeling deals with the effects on structure-borne noise. The study (Ghalamchi et al., 2013) puts forward a simple model based on Hertz’ian contact stress to calculate rotordynamics for barrel roller bearings. A second application for bearing modeling is damage diagnostics and the identification of the causes of bearing damage. In the publication (Tadina and Boltežar, 2011), the system-dynamic effects of damage to the balls and running surfaces of ball bearings are analyzed. The inner ring, the outer ring and the rolling elements are modeled as rigid bodies. The balls are elastically connected to the inner ring and the outer ring. To test design measures to improve the contact pattern of the bearing and its rolling elements, system-dynamic investigations are carried out. To calculate the optimal profiles for cylindrical roller bearings, the rollers are discretized in (Qian and Jacobs, 2014) using the slice model, and the stiffness is modeled as a Hertz’ian contact.

The scope of this paper is to create a model for the stiffness of selected types of single and double row roller bearings. It will be used for modeling mechanisms like a robot arm. The target is to simulate macroscopic system dynamics of a mechanism.

Modelica provides a powerful library for simulating multi-body systems like mechanisms - the MultiBody library. Therefore this approach extends the range of this library. An internal bearing analysis is out of scope of this

model. Effects considering the rolling elements e.g. mass effects are neglected.

2 Single-row bearings

For the modeling of roller bearings or ball bearings, the elastic forces and torques are determined for both types using the Hertz'ian contact stress.

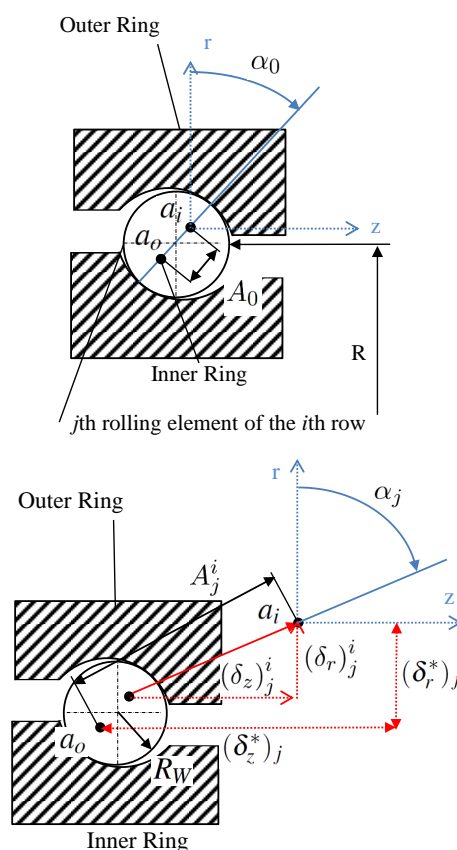


Figure 1. Deformations of a bearing

For simulating the stiffness of a bearing in Modelica the forces and torques in a bearing have to be derived.

First the change in angular position of the rolling element j when rotating the bearing about the Z-axis with the angle φ is calculated using the bearing geometry of the pitch radius of the rolling elements R and the rolling element radius R_W .

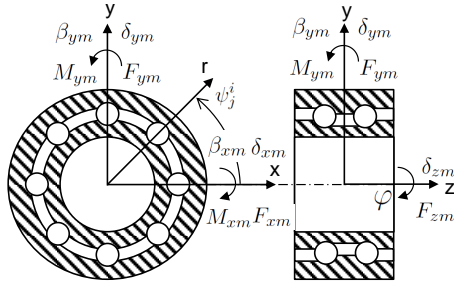


Figure 2. Coordinate system of a double row bearing

$$\Delta\psi_j = \begin{cases} \frac{R - R_W \cos(\alpha_0)}{R} \cdot \varphi, & \text{fixed outer ring} \\ \frac{R + R_W \cos(\alpha_0)}{R} \cdot \varphi, & \text{fixed inner ring} \end{cases} \quad (1)$$

We simplify in the next equation the angle ψ_j of the roller position of the j th roller according to the X-axis to the angle $\tilde{\psi}_j$:

$$\tilde{\psi}_j = \psi_j + \Delta\psi_j \quad (2)$$

For every rolling element j , the axial δ_{zj} and radial deformation δ_{rj} is calculated from the translational deformations $\delta_{xm}, \delta_{ym}, \delta_{zm}$ and the angular deformations β_{xm}, β_{ym} of the bearing (Fig. 2) with the pitch radius R and the bearing clearance r_L .

$$\delta_{zj} = \delta_{zm} + R(\beta_{xm} \sin(\tilde{\psi}_j) - \beta_{ym} \cos(\tilde{\psi}_j)) \quad (3)$$

$$\delta_{rj} = \delta_{xm} \cos(\tilde{\psi}_j) + \delta_{ym} \sin(\tilde{\psi}_j) - r_L \quad (4)$$

The contact angle under load α_j of the rolling element j can be calculated with the contact angle of the bearing α_0 , the net effective radial $(\delta_r^*)_j$ and axial $(\delta_z^*)_j$ displacement and the relative distance A_0 between the raceway groove curvature centers of the inner a_i and outer a_o bearing ring in case of no load:

$$(\delta_z^*)_j = A_0 \sin(\alpha_0) + \delta_{zj} \quad (5)$$

$$(\delta_r^*)_j = A_0 \cos(\alpha_0) + \delta_{rj} \quad (6)$$

$$\tan(\alpha_j) = \frac{(\delta_z^*)_j}{(\delta_r^*)_j} \quad (7)$$

Then we obtain the distance between the raceway groove curvature centers under load A_j :

$$A_j = \sqrt{(\delta_z^*)_j^2 + (\delta_r^*)_j^2} \quad (8)$$

Depending on the type of the rolling element W , the total elastic deformation is $\delta_W(\psi_j)$, ($W = B : \text{ball}, R :$

roller).

Ball :

$$\delta_{B,j} = \begin{cases} A_j - A_0, & \delta_{Bj} > 0 \\ 0, & \delta_{Bj} \leq 0 \end{cases} \quad (9)$$

Roller :

$$\delta_{R,j} = \begin{cases} \delta_{rj} \cos(\alpha_j) + \delta_{zj} \sin(\alpha_j), & \delta_{Rj} > 0 \\ 0, & \delta_{Rj} \leq 0 \end{cases} \quad (10)$$

The load Q_j on each rolling element j depends on the type of roller bearing W and is calculated with the Hertz'ian exponent n . For elliptical Hertz'ian contact (ball - elastic half-space) we assume $n = 3/2$. For a rectangular contact (cylinder - elastic half-space) we assume $n = 10/9$. Depending on the geometry and the material properties of the roller we denote the load Q_j on each rolling element j with the Hertz'ian stiffness constant K_n for ball (B) and roller bearings (R):

$$Q_j = K_n \cdot \delta_W(\tilde{\psi}_j)^n \quad (11)$$

Then we yield the forces and torques on a bearing with Z rolling elements and $\tilde{\psi}_j$ as the angular position of each rolling element (Lim and Singh, 1990a,b; Gunduz, 2012).

$$\begin{bmatrix} F_{xbm} \\ F_{ybm} \\ F_{zbm} \\ M_{xbm} \\ M_{ybm} \\ M_{zbm} \end{bmatrix} = \sum_{j=1}^Z Q_j \begin{bmatrix} \cos(\alpha_j) \cos(\tilde{\psi}_j) \\ \cos(\alpha_j) \sin(\tilde{\psi}_j) \\ \sin(\alpha_j) \\ R \sin(\alpha_j) \sin(\tilde{\psi}_j) \\ -R \sin(\alpha_j) \cos(\tilde{\psi}_j) \\ 0 \end{bmatrix} \quad (12)$$

For roller bearings we obtain with $\alpha_j = \alpha_0$ (Lim and Singh, 1990a) the forces and torques:

$$\begin{bmatrix} F_{xbm} \\ F_{ybm} \\ F_{zbm} \\ M_{xbm} \\ M_{ybm} \\ M_{zbm} \end{bmatrix} = K_n \sum_{j=1}^Z (\delta_{R,j} \cos(\alpha_0))^n \begin{bmatrix} \cos(\alpha_0) \cos(\tilde{\psi}_j) \\ \cos(\alpha_0) \sin(\tilde{\psi}_j) \\ \sin(\alpha_0) \\ R \sin(\alpha_0) \sin(\tilde{\psi}_j) \\ -R \sin(\alpha_0) \cos(\tilde{\psi}_j) \\ 0 \end{bmatrix} \quad (13)$$

The rotation about the Z-Axis is free because the torque about the Z-axis M_{zbm} in equations 12 and 13 is zero.

3 Double-row bearings

Besides single-row the force and torque balances for double-row ball and roller bearings are determined in this section. Three possible configurations for the double row bearings exist (Figure 3). The bearing arrangement coefficient c_3 considers these configurations. The calculations of the radial $(\delta_r)_j^i$ and axial displacement $(\delta_z)_j^i$ of

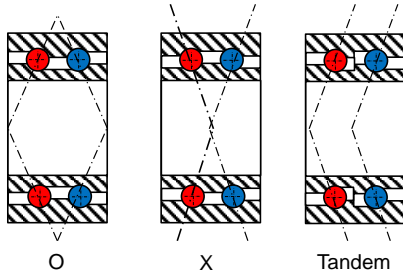


Figure 3. Bearing Arrangements (Matek et al., 2013) (Gunduz, 2012)

each rolling element j in row i (equations 3,4) have to be extended by the kinematics of the double row bearing with the row coefficient c_1 , the bearing clearance r_L , the distance between the two bearing rows e , the translational deformations $\delta_{xm}, \delta_{ym}, \delta_{zm}$, the angular deformations β_{xm}, β_{ym} of the bearing and the angular position $\tilde{\psi}_j^i$ of each rolling element.

$$c_1 = \begin{cases} 1, & \text{for } i = 1, \text{ left row} \\ -1, & \text{for } i = 2, \text{ right row} \end{cases} \quad (14)$$

$$(\delta_z^*)_j^i = [\delta_{xm} + c_1 \beta_{ym} e] \cos(\tilde{\psi}_j^i) + [\delta_{ym} - c_1 \beta_{xm} e] \sin(\tilde{\psi}_j^i) - r_L \quad (15)$$

$$(\delta_r^*)_j^i = \delta_{zm} + R [\beta_{xm} \sin(\tilde{\psi}_j^i) - \beta_{ym} \cos(\tilde{\psi}_j^i)] \quad (16)$$

Next the equations of the net effective radial $(\delta_r^*)_j^i$ and axial $(\delta_z^*)_j^i$ displacement of equation 5 and 6 are extended for the double row bearings. With the bearing arrangement coefficient c_3 (Figure 3) we calculate the net radial $(\delta_r^*)_j^i$ and axial $(\delta_z^*)_j^i$ effective displacements and the loaded distance A_j^i between the raceway groove curvature centers of the inner a_i and outer a_o bearing ring of the rolling element j in the row i .

$$c_3 = \begin{cases} [\text{left row, right row}], \text{ arrangement} \\ [1, -1], & \text{Back-To-Back, O} \\ [-1, 1], & \text{Face-To-Face, X} \\ [1, 1], & \text{Tandem} \end{cases} \quad (17)$$

$$(\delta_r^*)_j^i = A_0 \cos(\alpha_0) + \delta_r^i \quad (18)$$

$$(\delta_z^*)_j^i = \delta_z^i + c_3 (A_0 \sin(\alpha_0) + \delta_z^i) \quad (19)$$

$$A_j^i = \sqrt{(\delta_z^*)_j^i)^2 + ((\delta_r^*)_j^i)^2} \quad (20)$$

In general, the following relation applies to the forces $F_{xbm}, F_{ybm}, F_{zbm}$ and torques $M_{xbm}, M_{ybm}, M_{zbm}$ on the double-row ball bearing. For simplification we introduce the parameter R^* .

$$\begin{bmatrix} F_{xbm} \\ F_{ybm} \\ F_{zbm} \\ M_{xbm} \\ M_{ybm} \\ M_{zbm} \end{bmatrix} = \sum_{i=1}^2 \sum_{j=1}^Z Q_j \begin{bmatrix} \cos(\alpha_j^i) \cos(\tilde{\psi}_j^i) \\ \cos(\alpha_j^i) \sin(\tilde{\psi}_j^i) \\ \sin(\alpha_j^i) \\ R^* \sin(\tilde{\psi}_j^i) \\ -R^* \cos(\tilde{\psi}_j^i) \\ 0 \end{bmatrix} \quad (21)$$

$$R^* = R \sin(\alpha_j^i) - c_1 e \cos(\alpha_j^i) \quad (22)$$

For bearings with rollers as rolling elements, the angular offset α_j^i assumed to be zero. Therefore, the following applies:

$$\alpha_j^i = \alpha_0 \quad (23)$$

The equations for the forces $F_{xbm}, F_{ybm}, F_{zbm}$ and torques $M_{xbm}, M_{ybm}, M_{zbm}$ are thus as follows for each rolling element j in row i introducing the parameter \tilde{R} for simplification.

$$\begin{bmatrix} F_{xbm} \\ F_{ybm} \\ F_{zbm} \\ M_{xbm} \\ M_{ybm} \\ M_{zbm} \end{bmatrix} = \sum_{i=1}^2 \sum_{j=1}^Z Q_j \begin{bmatrix} \cos(\alpha_0) \cos(\tilde{\psi}_j^i) \\ \cos(\alpha_0) \sin(\tilde{\psi}_j^i) \\ \sin(\alpha_0) \\ \tilde{R} \sin(\tilde{\psi}_j^i) \\ -\tilde{R} \cos(\tilde{\psi}_j^i) \\ 0 \end{bmatrix} \quad (24)$$

$$\tilde{R} = R \sin(\alpha_0) - c_1 e \cos(\alpha_0) \quad (25)$$

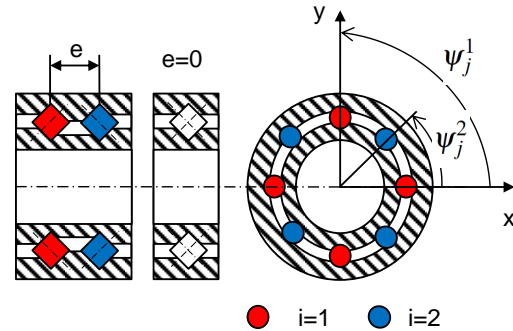


Figure 4. Conversion of a double row roller bearing to a cross roller bearing

A cross-roller bearing is modeled as a double-row bearing for which the distance e between both bearing rows is zero (see Fig. 4). For each row the correct angle offset $\tilde{\psi}_j^i$ for each rolling element has to be considered.

4 Implementation

The integration is carried out in the MultiBody library. In the preceding chapters, the relationship of the forces and torques on a roller bearing has been described. The motion equations and the elastic forces and torques of the bearing are determined using Lagrange's equation of the

second kind. Here T is the kinetic energy, \mathbf{Q} are the generalized forces (Hardtke et al., 1997). The generalized forces \mathbf{Q} consist of conservative \mathbf{Q}_k and non-conservative forces \mathbf{Q}_n . In this model non-conservative forces don't exist. With the vector of generalized coordinates \mathbf{q} and the potential energy $U(\mathbf{q})$ we denote:

$$\frac{d}{dt} \left(\frac{\partial T}{\partial \dot{q}_i} \right) - \frac{\partial T}{\partial q_i} = \mathbf{Q} \quad (26)$$

$$\mathbf{Q} = \mathbf{Q}_k + \mathbf{Q}_n \quad (27)$$

$$\mathbf{Q}_n = \mathbf{0} \quad (28)$$

$$\mathbf{Q}_k(\mathbf{q}) = -\frac{\partial U(\mathbf{q})}{\partial \mathbf{q}} \quad (29)$$

The vector of generalized coordinates \mathbf{q} are the deformations on the bearing and we yield the vector of the forces on a rolling element $\mathbf{F}(\mathbf{q})$ with the auxiliary parameters for coding $\mathbf{F}_{\text{Bearing}}, \mathbf{M}_{\text{Bearing}}$:

$$\mathbf{q} = [\delta_{xm}, \delta_{ym}, \delta_{zm}, \beta_{xm}, \beta_{ym}]^T \quad (30)$$

$$\mathbf{F}(\mathbf{q}) = [\mathbf{F}_{\text{Bearing}}, \mathbf{M}_{\text{Bearing}}]^T \quad (31)$$

$$\mathbf{F}_{\text{Bearing}} = [F_{xbm}, F_{ybm}, F_{zbm}]^T \quad (32)$$

$$\mathbf{M}_{\text{Bearing}} = [M_{xbm}, M_{ybm}, 0]^T \quad (33)$$

The force and torque relationships for single-row and double-row bearings are described in chapter 2 and chapter 3, respectively. For calculation of the potential energy $U(\mathbf{q})$, the following applies in general for the energy over the force $\mathbf{F}(\mathbf{q})$ with the vector of generalized deformations \mathbf{q} and finally we yield for the conservative force \mathbf{Q}_k :

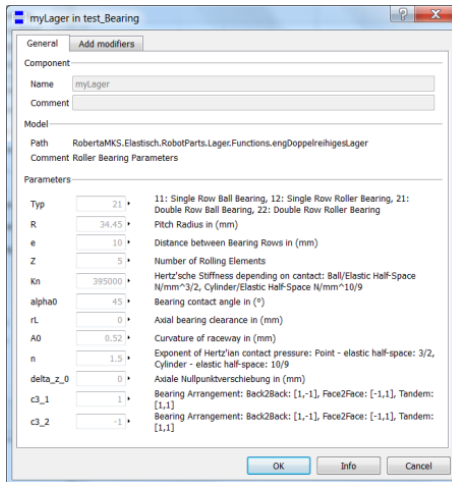


Figure 5. Input parameter mask in Dymola

$$\frac{\partial U}{\partial q_i} = \frac{\partial}{\partial q_i} \left(\int_0^q \mathbf{F}(q_i) d q_i \right) \quad (34)$$

$$\frac{\partial U}{\partial q_i} = \mathbf{F}(q_i) \quad (35)$$

$$\mathbf{Q}_k = -\mathbf{F}(\mathbf{q}) \quad (36)$$

The equations 35 and 36 show that the relationships of the forces and torques can be directly incorporated into the equations of motion.

The force and torque equilibrium then is:

```
import Modelica.Mechanics.MultiBody.Frames;
// Force and torque equilibrium
frame_a.f = -F_Bearing;
frame_b.f = -Frames.resolve2(
    Frames.relativeRotation(frame_a.R,
        frame_b.R),
    frame_a.f);

if fixedRotationAtFrame_a then
    Connections.root(frame_a.R);
    frame_a.R = Frames.nullRotation();
else
    frame_a.t = -M_Bearing;
end if;

if fixedRotationAtFrame_b then
    Connections.root(frame_b.R);
    frame_b.R = Frames.nullRotation();
else
    frame_b.t = -Frames.resolve2(
        R_rel, frame_a.t);
end if;
```

The calculation of the forces $F_{xbm}, F_{ybm}, F_{zbm}$ and torques $M_{xbm}, M_{ybm}, M_{zbm}$ on a bearing is realized in the sub-function *calculateBearingForce()*. The deformations $r_{\text{rel_a}}, R_{\text{rel}}$ between the *frame_a* and *frame_b* of the bearing block are required for calculating bearing forces and torques. The angles $\beta_{xm}, \beta_{ym}, \varphi$ are calculated as consecutive rotations out of the rotation matrix R_{rel} . The vector *angles* represents these angular deformations.

$$\text{angles} = [\beta_{xm}, \beta_{ym}, \varphi]^T \quad (37)$$

```
r_rel_a = Frames.resolve2(frame_a.R,
    frame_b.r_0 -
    frame_a.r_0);
R_rel = Frames.relativeRotation(
    frame_a.R, frame_b.R);
angles = Frames.axesRotationsAngles(R_rel,
    {1, 2, 3}, 0);

/* Determine forces and torques at frame_a
and frame_b */
q = {r_rel_a[1], r_rel_a[2], r_rel_a[3],
    angles[1], angles[2]};
(Fx, Fy, Fz, Mx, My) =
    Functions.calculateBearingForce(
        myLager, angles[3], q);

f_Bearing = {Fxbm, Fybm, Fzbm};
m_Bearing = {Mxbm, Mybm, Mz};
Mz = 0;
```

For the transformation of the forces and torques in the equations of motion the Jacobi matrix \mathbf{J} is required. This matrix describes the kinematics on the bearing and is determined by means of the deformation \mathbf{q} . The Cartesian deformations $\delta_{im}, i = x, y, z$ are modeled as sliders and the

rotations $\beta_{im}, i = x, y$ are modeled as consecutive rotation about x, y with the rotation axis vector \mathbf{z}_i using the rotation matrices R_x, R_y (see Fig. 2).

$$\mathbf{z}_i = \mathbf{e}_x, \mathbf{e}_y \quad (38)$$

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{vi} \\ \mathbf{J}_{\omega i} \end{bmatrix} \quad (39)$$

$$\mathbf{J}_i = \begin{cases} \mathbf{z}_i & , \text{translational motion} \\ \mathbf{z}_i \times \mathbf{p}_i^* & , \text{rotational motion} \end{cases} \quad (40)$$

$$\mathbf{J}_{\omega i} = \begin{cases} \mathbf{0} & , \text{translational motion} \\ \mathbf{z}_i & , \text{rotational motion} \end{cases} \quad (41)$$

The possible singularity of the Jacobi matrix for rotation about the x and y axes by 90° is ignored since small deformations are expected.

The mass and inertia of the inner and outer ring of the bearing are modeled with the block "Body" of the Multi-Body.Parts library. Each body is attached to *frame_a* and *frame_b* respectively. The dynamics of the rolling elements are neglected since only macroscopic effects of the system dynamics will be modeled.

Deriving a stiffness matrix $\mathbf{K}(\mathbf{q})$ is not required for the equations of motion in equation 26. If required, the stiffness matrix $\mathbf{K}(\mathbf{q})$ can be calculated by means of the partial differentiation of the force and torque vectors $\mathbf{F}(\mathbf{q})$ according to the generalized coordinates of the bearing deformation \mathbf{q} in equation 30.

$$\mathbf{K}(\mathbf{q}) = \frac{\partial \mathbf{F}(\mathbf{q})}{\partial \mathbf{q}} \quad (42)$$

5 Simulation Results

The parameters from table 1 are used to simulate the stiffness forces and torques with varying preload F_{xm} and a partial rotation $\varphi = 360/Z$ of the bearing.

Parameter	Value	Unit
R	34.45	mm
e	10	mm
Z	15	
K_n	395000	$N/mm^{\frac{3}{2}}$
α_0	45	
r_L	0	mm
A_0	0.52	mm
n	1.5	
$\delta_{z,0}$	0	mm

Table 1. Parameters of a double row roller bearing

Figure 6 shows a periodicity for the forces and torques within a bearing rotation. For each bearing rotation, the force and torque progression is repeated by the number of rolling elements. To assess the effects of parameter excitations on the system dynamics of the manipulator, further

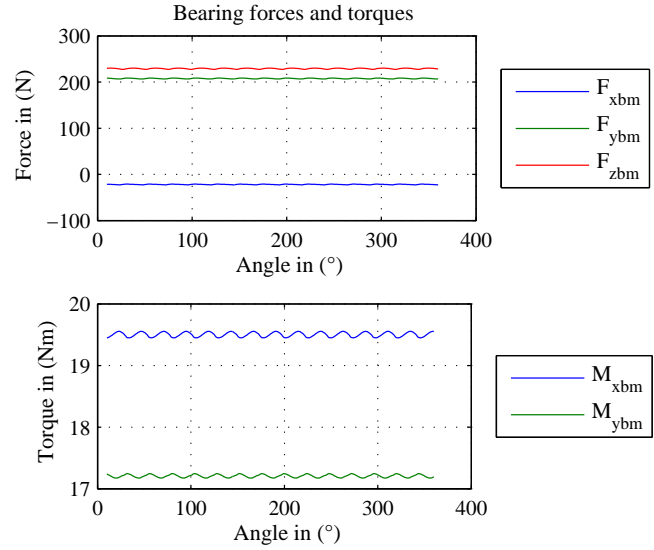


Figure 6. Bearing stiffness forces, bearing angle position varying

testing of the overall system of the manipulator is necessary. Next a simple example of a pendulum is shown. A stiff revolute joint and a bearing with its parameters in table 1 as a revolute joint are compared. The point mass is $1kg$ and the length of the pendulum is $1m$. Figure 7 shows the model in initial configuration. The reaction forces and torque in figure 8. show the difference between the rigid and the elastic suspended joint of the pendulum.

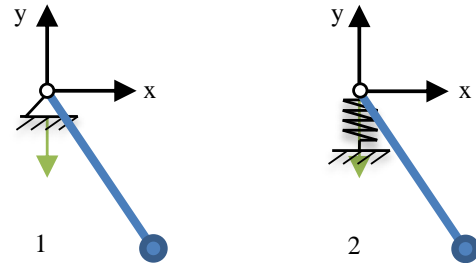


Figure 7. Pendulum, 1: rigid suspension 2: elastic suspension

6 Conclusion and Outlook

In this work the calculation of the stiffness forces and torques of single- and double-row bearings are shown. The following bearing types are considered:

- Single row ball bearing
- Single row roller bearing
- Double row ball bearing
- Double row roller bearing
- Cross roller bearing

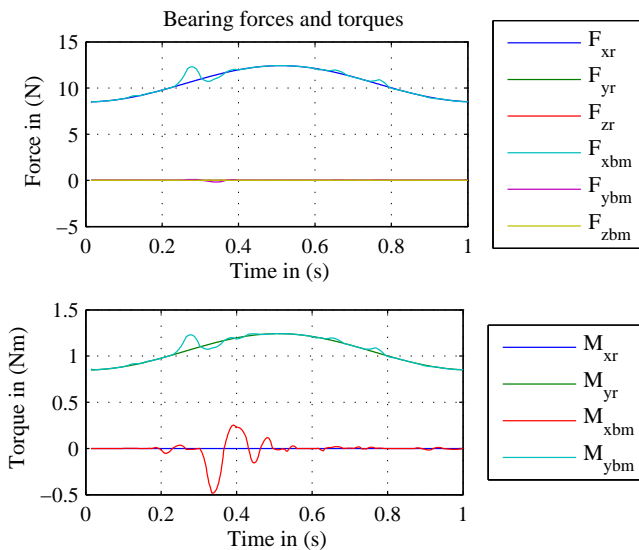


Figure 8. Forces and torques on the bearing and on the revolute joint and the displacement

The stiffness forces and torques are implemented in the MultiBody library of Modelica. A simulation shows the influence of loads and bearing position.

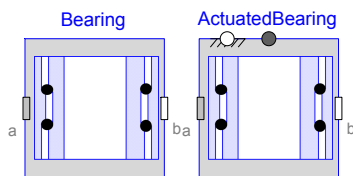


Figure 9. Bearing with 1 DOF and actuated bearing

According to equation 12 and figure 2 the bearing model yields a free rotation about the Z-axis. For multi-body systems and actuated mechanisms it will be important to consider the drivetrain. The scope of the future work is to develop an actuated bearing similar to the actuated joint in the Modelica MultiBody library. Further modeling and simulation of different mechanisms like a robot arm will be performed and validated.

References

- Behnam Ghalamchi, Jussi Sopanen, and Aki Mikkola. Simple and versatile dynamic model of spherical roller bearing. *International Journal of Rotating Machinery*, 2013, 2013.
- Aydin Gunduz. *Multi-dimensional stiffness characteristics of double row angular contact ball bearings and their role in influencing vibration modes*. PhD thesis, The Ohio State University, 2012.
- Hans-Jürgen Hardtke, Bodo Heimann, and Heinz Sollmann. *Lehr- und Übungsbuch Technische Mechanik Bd. II, Kinetik*. Kinetik-Systemdynamik-Mechatronik, Fachbuchverlag Leipzig, 1997.

Teik C. Lim and Rajendra Singh. Vibration transmission through

rolling element bearings, part i: bearing stiffness formulation. *Journal of sound and vibration*, 139(2):179–199, 1990a.

Teik C. Lim and Rajendra Singh. Vibration transmission through rolling element bearings, part ii: system studies. *Journal of sound and vibration*, 139(2):201–225, 1990b.

Wilhelm Matek, Dieter Muhs, Herbert Wittel, and Manfred Becker. *Roloff/Matek Maschinenelemente: Normung Berechnung Gestaltung*. Springer-Verlag, 2013.

Weihua Qian and Georg Jacobs. Dynamic simulation of cylindrical roller bearings. Technical report, Lehrstuhl und Institut für Maschinenelemente und Maschinengestaltung, RWTH Aachen, 2014.

Matej Tadini and Miha Boltežar. Improved model of a ball bearing for the simulation of vibration signals due to faults during run-up. *Journal of sound and vibration*, 330(17):4287–4301, 2011.

Cabin Thermal Needs: Modeling and Assumption Analysis

Florent Brèque Maroun Nemer

MINES ParisTech, PSL Research University, Center for energy Efficiency of Systems
5 rue Léon Blum, Palaiseau, 91120, France
florent.breque@mines-paristech.fr

Abstract

Interest for cabin thermal needs has strongly increased for the past 10 years, particularly due to heating. Indeed, the development of electric and hybrid vehicles stressed the need to rethink the cabin thermal design and the HVAC, this high-consuming auxiliary, which can dramatically decrease the vehicle electric range. Thus, this paper presents a detailed transient and mono-zonal model of a car cabin in order to predict the thermal needs. The model is developed using the MODELICA language via the DYMOLA environment. It considers conduction, convection, radiation heat transfers as well as the HVAC and water vapor impacts. The different assumptions of the model are discussed and important considerations usually not discussed are highlighted. The thermal loads are also analyzed. Finally, the heating and cooling thermal needs are computed for steady state mode and for convergence mode as well as for varying recirculation ratios. This work is useful to better understand the whys and wherefores related to the cabin thermal needs.

Keywords: *thermal model, vehicle cabin, cabin thermal needs, HVAC, heating, Air-conditioning, electric vehicle.*

1 Introduction

For the past ten years, the electric vehicle developments and deployments have strongly accelerated. However, one of the major concerns with those vehicles is their low range. Hence, a lot of effort has been made to improve the vehicle range. It appeared that a major load for the battery is actually the HVAC system, which can consume as much energy as the motor in some conditions. Thus, additional development is necessary to decrease the HVAC energy consumption. One solution is to use a heat pump instead of the electric heaters typically used.

The work presented in this paper is part of a project which aims at developing a heat pump technology for electric vehicles. The project focuses particularly on the frosting issue with regards to the evaporator. Another important aspect of the project is to evaluate the gains that can be obtained from using heat pumps

and evaluate other thermal strategies. In order to do so, a detailed thermal model of the cabin is required. Different models have already been presented in the literature. They are of different types.

A first category is composed of the CFD models (Versteeg and Malalasekera, 2007). Some authors use those models to evaluate the thermal comfort (Fujita et al., 2001; Kataoka and Nakamura, 2001; Sevilgen and Kilic, 2012). Others study the impact of specific aspects such as the windows opening, the glazing properties or the air quality (Al-Kayiem et al., 2010; Fujita et al., 2001; Zhu et al., 2010).

On the other hand, lumped models, also called mono-zonal model, have been developed. Here, the cabin air is modeled by a single node and is therefore considered homogeneous (Marcos et al., 2014; Wischhusen, 2012). Those types of models are mainly used to study the impact of some factors on the thermal load (Li and Sun, 2013; Torregrosa-Jaime et al., 2015) but can also be used for studying HVAC control (Sanaye et al., 2012).

Finally, an intermediary approach between the two previous exists and is called the zonal approach. It consists in defining several air lumped nodes in a single air volume and linking them in order to exchange mass (Boukhris et al., 2009). The challenge here is then to use or develop a proper flow model between the nodes (Daoud and Galanis, 2008; Inard et al., 1996). It can be seen as a simplified CFD approach in some cases. For vehicle applications, this approach is sometimes used but usually only with two nodes in the cabin (Torregrosa-Jaime et al., 2015; Wischhusen, 2012).

Based on this review, it clearly appears that the most relevant category of models to study the cabin thermal needs takes the mono-zonal approach. It is therefore the one that will be developed in this paper.

Regarding the thermal need analysis, several authors have studied it (Iskandar, 2010; Li and Sun, 2013; Marcos et al., 2014; Mezrhab and Bouzidi, 2006; Torregrosa-Jaime et al., 2015). However, they focused on air-conditioning needs (since it was the important aspect for conventional cars) but neglect heating needs. Furthermore, they have studied a limited number of cases and the assumptions have not been discussed.

Consequently, the aim of this paper is to address those issues.

To do so, a model is developed using the MODELICA language via the DYMOLA environment. First, an overview of the top level model is presented in this paper. Then the main models are presented. Through the model descriptions, the different assumptions of the model are discussed and important considerations usually not discussed are pointed out. Then, in the result section, the steady state mode is first analyzed, followed by the convergence mode, which correspond to the initial transient warm-up or cool-down phase. The thermal loads are analyzed for the steady state case and, for both mode, a sensitivity analysis is performed. Finally, cooling and heating thermal needs are computed.

2 Mathematical model

2.1 Model overview

Figure 1 is a view of the cabin top level model in the DYMOLA environment. It is composed of several models. First, the cabin model itself includes the thermal network and the cabin air node (moist air). The HVAC model handles the recirculation air flow and the heating/cooling of the air. It is operated by a controller which either adjusts the thermal power to reach the targeted cabin air temperature or imposes a constant thermal power depending on the user choice. The atmosphere model imposes the weather conditions. Finally 5 records are used here as an interface for model parameterization.

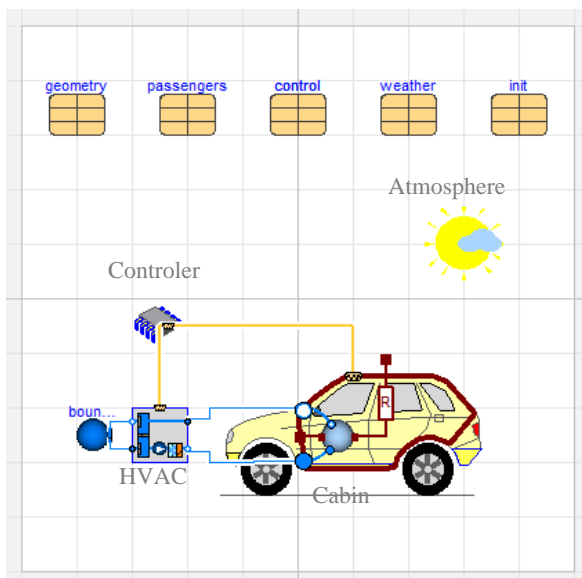


Figure 1. Cabin top level model in DYMOLA.

Figure 3 is a schematic of the complete model. It can be observed that the model is divided in two parts.

First, there is the thermal network. It includes the heat transfer exchanges with the exterior. In addition, two cabin internal nodes are defined and connected

between each other and to the walls. The first corresponds to the cabin air and the second to the internal solid mass (seats, dash board...).

Second, a fluid flow network is represented. It represents the moist air flows. It is particularly of interest since it computes the properties of the recirculation air, it determines the water vapor condensation rate in the evaporator and also computes the water vapor mass balance in the cabin taking into account passenger water vapor generation.

This model is described in more details in the following sections.

2.2 Atmosphere model

Basically, the atmosphere model is quite simple. It is here to provide 5 inputs: outside air temperature, outside air humidity, solar direct and diffuse flux as well as the sun direction vector. One can directly provides this information as parameters. Instead of giving directly the solar vector, it is also possible to write a latitude and longitude with the date and time. Then, using calculation from (ASHRAE, 2009), the solar vector is computed.

It can be noted here that the atmosphere variables are transferred to the other models via the 'inner'/'outer' method.

2.3 Wall model

The wall model presented in Figure 2 is the thermal network between the cabin interior and the outside environment (all thermal components are from the MSL Thermal library).

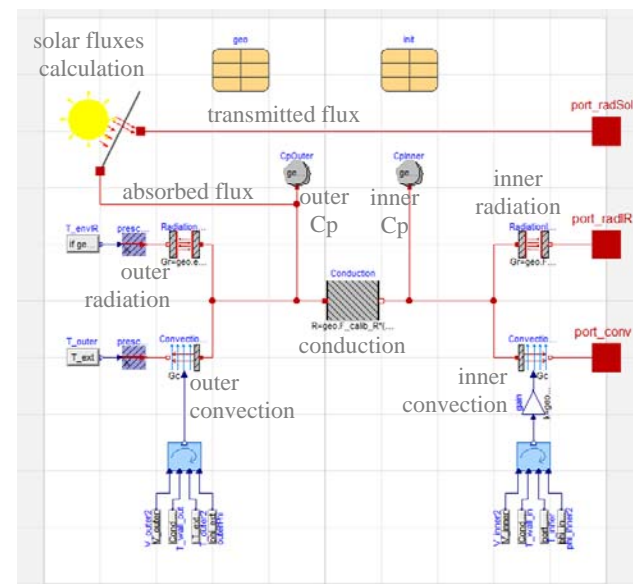


Figure 2. Wall model in DYMOLA.

Based on Figure 2, the thermal balance at the outer wall node T_{wall}^{out} applies as follow (by convention, heat flow is positive when heat goes from outside into the cabin):

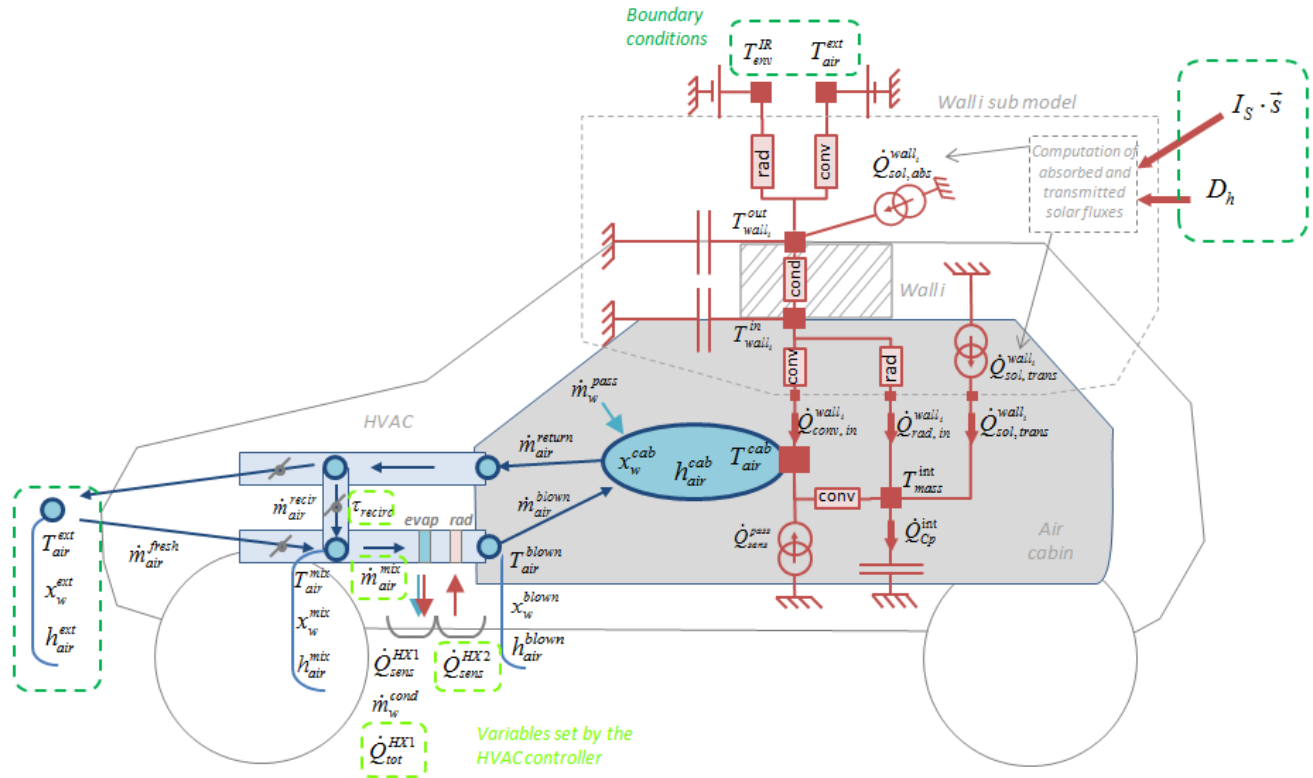


Figure 3. Schematic of the model.

$$\begin{aligned}
 & C_{p \text{ out}}^{\text{wall}} \cdot \frac{d(T_{\text{wall}}^{\text{out}})}{dt} \\
 &= \dot{Q}_{\text{conv, out}}^{\text{wall}} + \dot{Q}_{\text{rad, out}}^{\text{wall}} - \dot{Q}_{\text{cond}}^{\text{wall}} + \dot{Q}_{\text{sol, abs}}^{\text{wall}} \\
 &= h_{\text{conv}}^{\text{out}} \cdot A_{\text{wall}} \cdot (T_{\text{air}}^{\text{ext}} - T_{\text{wall}}^{\text{out}}) \\
 &+ \varepsilon_{\text{wall}}^{\text{out}} \cdot A_{\text{wall}} \cdot \sigma \cdot (T_{\text{env}}^{\text{IR}^4} - T_{\text{wall}}^{\text{out}^4}) \\
 &- \frac{1}{R_{\text{cond}}^{\text{wall}}} \cdot (T_{\text{wall}}^{\text{out}} - T_{\text{wall}}^{\text{in}}) + \dot{Q}_{\text{sol, abs}}^{\text{wall}}
 \end{aligned} \quad (1)$$

Similarly, the thermal balance at the inner wall node $T_{\text{wall}}^{\text{in}}$ is expressed as follow:

$$\begin{aligned}
 & C_{p \text{ in}}^{\text{wall}} \cdot \frac{d(T_{\text{wall}}^{\text{in}})}{dt} \\
 &= \dot{Q}_{\text{cond}}^{\text{wall}} - \dot{Q}_{\text{conv, in}}^{\text{wall}} - \dot{Q}_{\text{rad, in}}^{\text{wall}} \\
 &= \frac{1}{R_{\text{cond}}^{\text{wall}}} \cdot (T_{\text{wall}}^{\text{out}} - T_{\text{wall}}^{\text{in}}) \\
 &- h_{\text{conv}}^{\text{in}} \cdot A_{\text{wall}} \cdot (T_{\text{wall}}^{\text{in}} - T_{\text{air}}^{\text{cab}}) \\
 &- \varepsilon_{\text{wall}}^{\text{in}} \cdot F_{\text{wall-int}} \cdot A_{\text{wall}} \cdot \sigma \cdot (T_{\text{wall}}^{\text{in}^4} - T_{\text{mass}}^{\text{int}^4})
 \end{aligned} \quad (2)$$

In eq. (1), the solar flux $\dot{Q}_{\text{sol, abs}}^{\text{wall}}$ is computed via a sub-model described in section 2.4. In addition, the convection coefficient $h_{\text{conv}}^{\text{in}}$ and $h_{\text{conv}}^{\text{out}}$ are given in

section 2.5. The wall thermal resistance $R_{\text{cond}}^{\text{wall}}$ and the thermal capacitances $C_{p \text{ in}}^{\text{wall}}$ and $C_{p \text{ out}}^{\text{wall}}$ are determined by basic calculations taking into account the multilayer structure of the wall. It is done in a generic way such that both opaque body (with an inner skin, a lining material and a outer skin) and glazing surfaces can be handled by the same model.

An interesting aspect here is that the radiation heat transfer has been considered both for inner and outer wall surfaces. The outer surface exchanges by radiation with a so called environment IR temperature $T_{\text{env}}^{\text{IR}}$. Determining this temperature is not trivial. Depending on the surfaces, it could be equal to the outside air temperature, but it also could be equal to the sky temperature, which can be far lower than the outside temperature for clear skies. In addition, during summer, the floor sees the road which can have a very high temperature compared to the air. In the model, those aspects are configurable. For the clear sky temperature, the correlation from (Swinbank, 1963) is used.

Furthermore, in eq. (2), one can observe that the internal radiative heat exchange is between the wall and the internal mass (the seats, the dashboard, ...) via a view factor. First, this view factor is not easy to determine. In addition, the radiative heat exchanges between the walls themselves are not considered.

The impact of those different assumptions are evaluated in the results section.

2.4 Solar fluxes computation sub-model

This sub-model included in the wall model aims at computing the absorbed solar flux $\dot{Q}_{sol,abs}^{wall}$ by the wall and the transmitted solar flux through the wall $\dot{Q}_{sol,trans}^{wall}$ (for glazing). The following equations, extracted from (ASHRAE, 2009) are used.

The direct solar flux I_n received by a surface with a normal \vec{n} is given by (with p_{ns} the scalar product between the solar vector \vec{s} and \vec{n} and I_s the solar flux):

$$I_n = p_{ns} \cdot I_s \quad (3)$$

Then, the total diffuse flux D_n received is:

$$D_n = D_h \cdot \left(\frac{1 + \cos\theta}{2} \right) + \rho \cdot G_h \cdot \left(\frac{1 - \cos\theta}{2} \right) \quad (4)$$

With D_h the horizontal diffuse flux, θ the angle between \vec{n} and the vertical, ρ the albedo and G_h the global flux.

Finally, the solar fluxes are given by eq. (5) and (6).

$$\dot{Q}_{sol,abs}^{wall} = \alpha_{sol} \cdot A_{wall} \cdot (I_n + D_n) \quad (5)$$

$$\dot{Q}_{sol,trans}^{wall} = \tau_{sol} \cdot A_{wall} \cdot (I_n + D_n) \quad (6)$$

2.5 Convection sub-model

The wall model also includes a convection sub-model. The convection in this thermal problem is quite complicated. Indeed, the geometry is complex and natural and forced convection can be mixed. A detailed analysis of the convection correlations has been conducted here.

First, it can be noted that the cabin walls are often assumed to be flat plates. Hence, the general correlations for flat plates (see (Bergman et al., 2011)) are often used. For instance, eq. (7) is for forced parallel and turbulent flow and eq. (8) is for natural convection over a vertical plate. All applicable flat plate correlations have been implemented in the code.

In addition, for internal convection, correlations of the form of eq. (9) (Abou Eid, 2016) and used in building applications have also been evaluated. The coefficients c and n are adjusted depending on the situations (floor, ceiling, vertical walls, mixed convection...).

Finally, for external correlations around the vehicle (car, bus or train), several authors (Fujita et al., 2001; Kataoka and Nakamura, 2001; Li and Sun, 2013; Mezrhab and Bouzidi, 2006; Zhang et al., 2009) use their own specific correlation in the general form of eq. (10) (with a , b and c constants). Most of the authors define a minimum value which is applied for low velocities.

$$Nu_L = (0,037 \cdot Re_L^{0.8}) \cdot Pr^{1/3} \quad (7)$$

$$Nu_L = \left(0.825 + \frac{0,387 \cdot Ra_L^{1/6}}{1 + (0,492/Pr)^{9/16}} \right)^2 \quad (8)$$

$$h_{conv} = c \cdot |T_{in}^{wall} - T_{air}^{cab}|^n \quad (9)$$

$$h_{conv} = a + b \cdot v^c \quad (10)$$

In order to give an idea on how the correlations compare, **Error! Reference source not found.** plots the computed coefficients. It appears that the general trends are similar except for the laminar correlations and Li's correlation. However, the coefficients can be multiplied by 1.5 from one correlation to the other. The impact of those correlations is evaluated in the result section. By default, the (Fujita et al., 2001) correlation and the eq. (9) with $c=3$ and $n=1/3$ (mixed convection) are used.

2.6 Passenger model

As shown in Figure 3, the model takes into account the passenger thermal loss \dot{Q}_{sens}^{pass} and the passenger water vapor generation \dot{m}_w^{pass} . According to (ASHRAE, 2009), it is assumed that a person emits 70W of sensible heat and 35W of latent heat. The latent heat is converted to a mass flow rate in the model and not added in the thermal network. Only the mass flow rate is injected in the cabin volume and is then considered for the mass and energy balances. The number of passengers can be adjusted.

2.7 Cabin model

The cabin model is shown in **Error! Reference source not found.** It includes the wall and passenger models that just have been presented. In the figure, only one wall model appears. However, an important feature here is that it is actually a vector of walls. Thus, the number of walls, opaque or transparent walls, can be varied. This allows adjusting to the exact vehicle configuration.

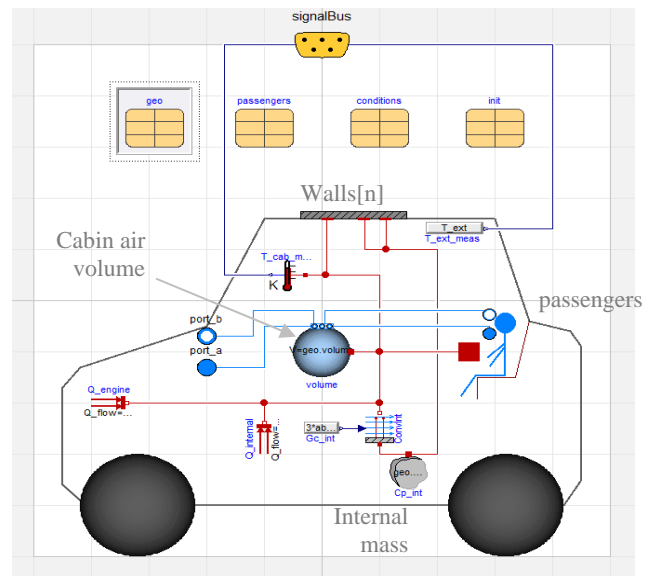


Figure 4. Cabin model in DYMOLA.

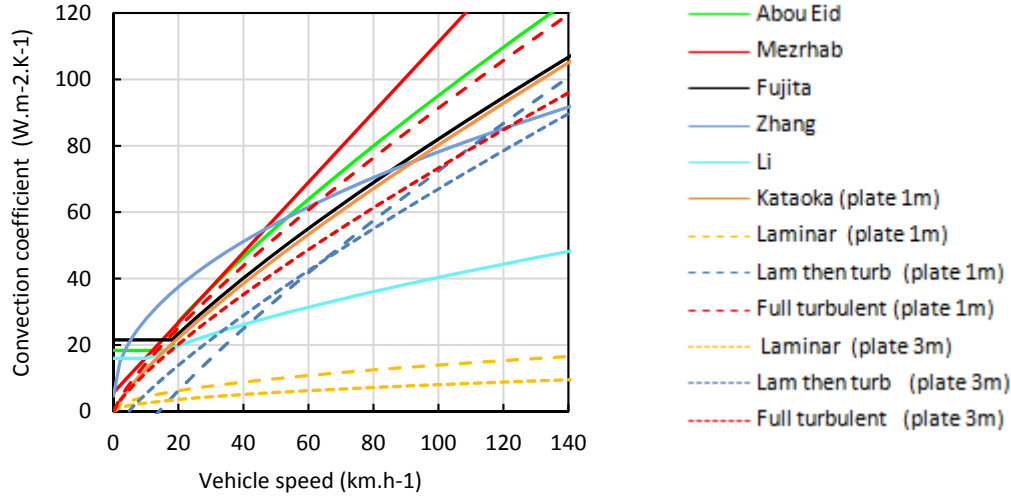


Figure 5. Convection coefficients from the different correlations for external forced convection.

In this model, the energy balance applies for the internal mass node T_{mass}^{int} :

$$C_p^{int} \cdot \frac{d(T_{mass}^{int})}{dt} = \dot{Q}_{conv}^{int} + \sum_{i=1}^{nWall} (\dot{Q}_{sol,trans}^{wall_i} + \dot{Q}_{rad,in}^{wall_i}) \quad (11)$$

With :

$$\dot{Q}_{conv}^{int} = h_{conv}^{int} \cdot A_{mass} \cdot (T_{air}^{cab} - T_{mass}^{int}) \quad (12)$$

An assumption that can be discussed here is the approach with the internal mass and the way the radiation problem is treated. It follows the approach used in (Marcos et al., 2014) where a “base” node is defined. The internal mass is assumed to be a black body and to receive all the radiation heat fluxes: the solar transmitted flux as well as the IR flux from the inner surface of the walls. It is useful to consider this internal mass since it has a strong impact on transient results. In addition, due to the complex shape of the internal masses and their high emissivity and absorptivity, it makes sense to assume a black body here. Those points are discussed in the result section.

Then, the energy balance is applied to the air volume:

$$\begin{aligned} V_{air}^{cab} \cdot \left(u_{air}^{cab} \cdot \frac{\partial \rho_{air}^{cab}}{\partial t} + \rho_{air}^{cab} \cdot \frac{\partial u_{air}^{cab}}{\partial t} \right) \\ = \dot{m}_{air}^{blown} \cdot h_{air}^{blown} - \dot{m}_{air}^{return} \cdot h_{air}^{cab} \\ + \dot{m}_w^{pass} \cdot h_{w,vap}^{37^\circ C} + \dot{Q}_{sens}^{pass} \\ + \sum_{i=1}^{nWall} \dot{Q}_{conv,in}^{wall_i} - \dot{Q}_{conv}^{int} \end{aligned} \quad (13)$$

With $h_{w,vap}^{37^\circ C}$ the enthalpy of water vapor at $37^\circ C$.

In addition to the energy balance, the mass balances also apply:

$$V_{air}^{cab} \cdot \frac{\partial (\rho_{air}^{cab})}{\partial t} = \dot{m}_{air}^{blown} - \dot{m}_{air}^{return} + \dot{m}_w^{pass} \quad (14)$$

$$V_{air}^{cab} \cdot \left(x_w^{cab} \cdot \frac{\partial \rho_{air}^{cab}}{\partial t} + \rho_{air}^{cab} \cdot \frac{\partial x_w^{cab}}{\partial t} \right) = \dot{m}_{air}^{blown} \cdot x_w^{blown} - \dot{m}_{air}^{return} \cdot x_w^{cab} + \dot{m}_w^{pass} \quad (15)$$

A key point here is of course the water vapor mass balance which will determine the humidity ratio in the cabin air and impact the HVAC either by requiring additional fresh air to avoid mist and/or by adding a cooling load to the evaporator.

2.8 HVAC model

The HVAC can be seen as an external component of the cabin. Therefore, one could want to focus on the cabin and not to consider at all the HVAC for the determination of thermal cabin needs. However, considering the HVAC is actually required here to properly compute the thermal cabin needs, which translate into the HVAC thermal loads. Indeed, knowing the water vapor condensation rate at the evaporator is important since it can strongly impact the total A/C needs. Moreover, this condensation rate is required when addressing the cabin dehumidification needs, which also translate into cabin needs. In addition, the thermal needs are strongly dependant on the recirculation ratio, which is controlled by the HVAC.

Figure 6 is a view of the HVAC model. It is composed of a recirculation box and a cooling/heating element. The recirculation box is made of valves (from MSL Fluid library) whose openings are adjusted to

control the recirculation ratio as desired (a PI is used here in the controller of the Figure 1). The model handles the mixing of the fresh air with the recirculated air via basic mass and energy balances.

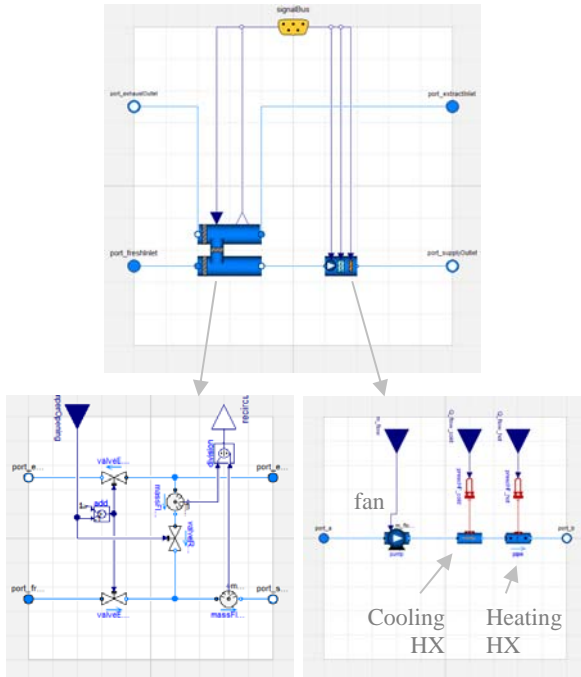


Figure 6. HVAC model in DYMOLA.

The cooling/heating element is composed of a fan, a cooling HX and a heating HX. The fan model is the ‘PrescribedPump’ model from MSL Fluid library and the heating HX is modeled with a ‘DynamicPipe’ from the same library. Only the cooling HX is modeled via an ad hoc pipe which is able to handle water vapor condensation. To do so, the sensible heat exchange \dot{Q}_{sens}^{evap} is computed as follow:

$$\dot{Q}_{sens}^{evap} = h_{conv}^{evap} \cdot A_{evap} \cdot \left(\frac{T_{air}^{mix} - T_{air}^{evap\ out}}{\ln \left(\frac{T_{air}^{mix} - T_{HXi}^{wall}}{T_{air}^{evap\ out} - T_{HXi}^{wall}} \right)} \right) \quad (16)$$

Then, the water vapor condensation rate \dot{m}_w^{cond} is given by (when $x_{air}^{mix} > x_{sat}(T_{evap}^{wall})$; null otherwise):

$$\dot{m}_w^{cond} = \frac{h_{conv}^{evap} \cdot A_{evap}}{\rho_{air}^{mix} \cdot c_{p,air}} \cdot \left(\frac{x_{air}^{mix} - x_{air}^{blown}}{\ln \left(\frac{x_{air}^{mix} - x_{sat}(T_{evap}^{wall})}{x_{air}^{blown} - x_{sat}(T_{evap}^{wall})} \right)} \right) \quad (17)$$

Then, it comes:

$$\dot{Q}_{tot}^{evap} = \dot{Q}_{sens}^{evap} + \dot{m}_w^{cond} \cdot L_{lv} \quad (18)$$

At this point it can be noted that the controller of the HVAC (see Figure 1) can adjust the evaporator sensible cooling power and the heating power to obtain proper cabin air temperature and humidity (and hence to satisfy the cabin needs: heating, cooling, dehumidifying). It is also possible to impose both sensible cooling power and heating power. Then, using eqs. (16) to (18), the T_{evap}^{wall} is computed followed by \dot{m}_w^{cond} and \dot{Q}_{tot}^{evap} .

Here, it can be noted that the heat-pickup (or loss) in the air distribution system (dashbord and recirculation channel) has been neglected.

2.9 Model parameters

The model requires a full list of parameters. There are not presented in detail here due to conciseness considerations. Basically, the car geometry is described by the wall areas and orientations. In addition, thicknesses and thermal parameters are required. For the following results, parameters from a mid-size car are applied (main parameters given in Table 1).

Table 1. Geometry and thermal parameters.

Parameter	Value
Glazing areas	2 m ²
Opaque areas	9.9 m ²
Lateral insulation thickness	75 mm
Floor insulation thickness	15 mm
Roof insulation thickness	13 mm
Internal thermal capacity	75 kJ.K ⁻¹
Total wall thermal capacity	155.4 kJ.K ⁻¹
Glazing transmittivity	0.85
Wall outer absorptivity	0.85

In addition to those parameters, when one wants to compute the cabin thermal needs, operating condition parameters are required. Those parameters are presented in Table 2. As we will see, some of those parameters strongly influence the results. They are in a sense arbitrary and depend more on the manufacturer philosophy (specifications), but they have to be carefully considered by one who wants to study thermal needs due to their strong influence.

In addition to the parameters given in Table 2, the target cabin temperature is set to 23°C. The number of passengers is 0 for heating and 4 for AC. No sun is considered for heating and the sun is defined by $I_s = 700 \text{ W.m}^{-2}$ and $D_h = 117 \text{ W.m}^{-2}$ for A/C. The vehicle speed is set to 45 km/h. And the initial condition is a cabin at T_{air}^{ext} (all thermal nodes).

Table 2. Operating conditions.

T_{air}^{ext}	φ	t_{targ}	\dot{m}_{air}^{blown} Stab.	\dot{m}_{air}^{blown} Conv.	\dot{m}_{air}^{fresh}	Deshum. ΔT_{air}^{evap}
(°C)	%	min	(kg.h ⁻¹)	(kg.h ⁻¹)	(kg.h ⁻¹)	(°C)
-20	85	20	245	400	Same as \dot{m}_{air}^{blown} (no recirc)	No
-15			236	390		
-10			225	360		
-5			210	330		
0	95	15	200	310		3
5			185	290		
10		10	180	285		10
15			187	290		
20			195	300		
15	95	10	187	290	187	min 10
20			195	300	195	
25			248	390	124	
30	85		330	533	88	N/A
35	65	15	378	608	59	
40	55		400	632	48	
45	35	20	408	640		

2.10 Model validation

The experimental validation of the model is being carried out at the moment. Hence, the results were not available for this paper but they will be presented in a following paper. The first tests conducted with a crane cabin (and not a car cabin, due to industrial partner needs) were very encouraging since the cabin temperature was predicted within $\pm 1^\circ\text{C}$. The first tests suggested that the thermal parameters need to be filled carefully and that the analysis conducted in this paper are valid.

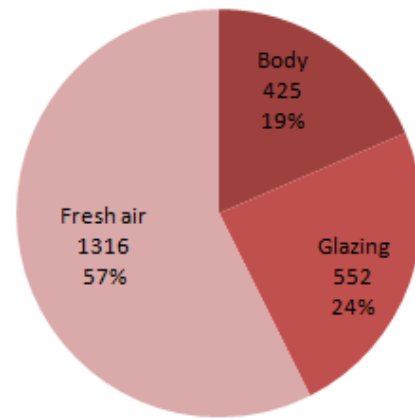
3 Model results

3.1 Reference cases – steady state

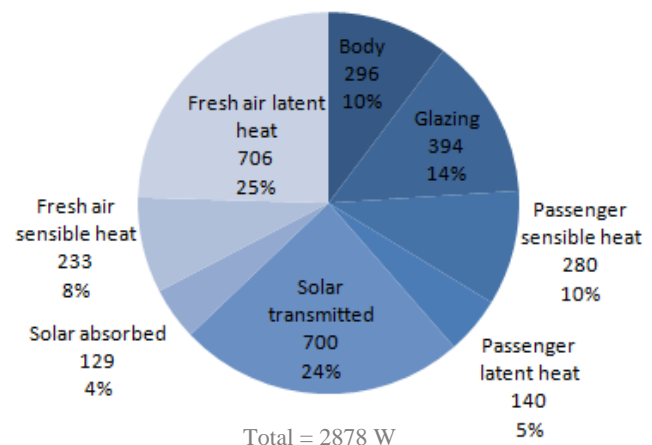
To start the analysis, let's have look at the thermal loads at 0°C and 40°C (the other parameters are as given in section 2.9) in steady state.

Figure 7 presents the thermal loads at 0°C . The total heating need is 2.3 kW. It can be observed that the fresh air is responsible for more than half of the needs. Then, the heat transfer occurs more through glazing.

For the air-conditioning case at 40°C presented in Figure 8, the load split is more complex. The total cooling load is 2.9 kW. 1/3 is due to the solar, 1/3 due to fresh air and the last third due to heat exchange with the exterior temperature through wall and due to the passengers. Because the recirculation ratio is high, the fresh air sensible load is low. However, the high humidity level results in an important fresh air latent load. It is important to take into account this load since the A/C system will have to overcome it.



Total = 2293 W

Figure 7. Thermal loads (W) at 0°C .

Total = 2878 W

Figure 8. Thermal loads (W) at 40°C .

3.2 Sensitivities – steady state

In this section different assumptions were varied for the two reference cases (at 0°C and 40°C).

First, the different correlations presented in section 2.5 were tested. Using flat plate natural convection correlation for the internal surfaces can decrease by 10% the thermal needs. For external surfaces, the sensitivity to the correlations is less than $\pm 3\%$.

Several cases were tested to evaluate the dependence on radiative assumptions. The IR environment temperatures have been assumed to be all the sky temperature or all the outside air temperature. A case with a road at 80°C has also been considered. The view factors between each wall and the internal mass have been varied from default value to 1. For all those cases, the sensitivity was less than $\pm 3\%$. A case with no radiative heat exchange inside the cabin has been considered. It decreased by 7% the thermal heating need and 5% the cooling needs for the reference case using the mixed convection correlation for the inner convection. However, if pure natural

convection conditions prevail, neglecting the inner radiation decreases by 17% the heating and cooling needs. Hence, internal radiation needs to be taken into account.

The impacts of the conditions selected in section 2.9 have also been analyzed via the following cases:

- Case 1: 25% increase in blown air flow rate (with same recirculation rate as default case)
- Case 2: recirculation rate at 50%
- Case 3: target cabin temperature set at 20°C
- Case 4: stationary vehicle

The results are presented in Table 2. It is clear that those operating conditions have strong impacts on the results, far more important than the majority of previous modeling sensitivities. Therefore, it means that, when talking about the cabin needs, the related considered conditions should be clearly stated, particularly the recirculation rate considered and the blown air flow rate.

Table 3. Sensitivities to operating conditions

<i>Thermal needs (W) and relative gap (%).</i>				
Text	Case 1	Case 2	Case 3	Case 4
0°C	2622	1638	1982	2079
	14.3%	-28.6%	-13.6%	-9.3%
40°C	2786	4126	3102	2937
	-3.2%	43.4%	7.8%	2.1%

3.3 Cabin needs – steady state

The thermal cabin needs in steady state conditions are presented in Figure 9. Between 5 and 20°C, the dehumidification (cooling and heating at the same time) is taken into account.

Based on the previous discussion, needs for varied recirculation ratio from 0 to 100% is added to the curves (without taking into account dehumidification needs). It corresponds to the grey bands.

For the dehumidification zone, the required heating is between 1 to 2 kW. Then, at -10°C, it increases up to 3.6 kW. Those numbers are the same order of magnitude as the average traction power for an urban trip (taking into account stops), though lower. Thus, it appears clearly here why the heating is so damageable for electric vehicle range for a basic case using electric heaters. The A/C needs are also high but they increase less with respect to the temperature above 25°C because of the high recirculation ratio considered and the A/C system has a better COP than the electric heaters.

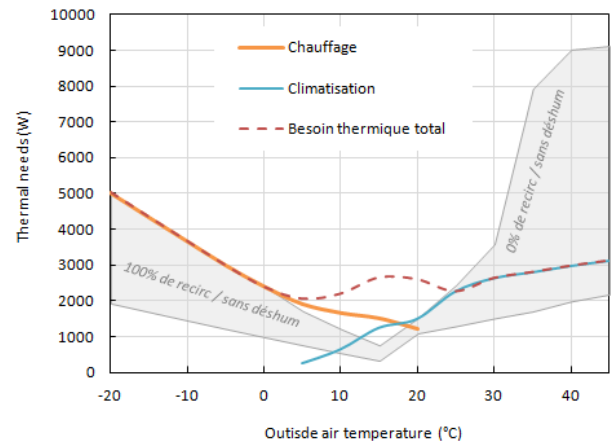


Figure 9. Steady state thermal needs.

3.4 Reference cases – convergence

Similarly to what has been done for steady state, two reference cases, based on operating conditions described in section 2.9 at 0°C and 40°C, are analyzed. Figure 10 is a plot of the cabin air temperature during the warm-up. 4 kW heating is required to reach the targeted temperature within the targeted duration. The heating power is applied continuously during the simulation and that is why the temperature exceeds the target at the end of the simulation. For normal cases, once the targeted temperature is reached within a certain interval, a real HVAC control algorithm would decrease the thermal power and converge approximately towards the steady state required power (values presented previously). Here, the controller is not modeled in details and just a constant-power transient is analyzed.

The results for the cool-down at 40°C are presented in Figure 11. Here, 3.5 kW were required to reach the proper temperature within the required duration.

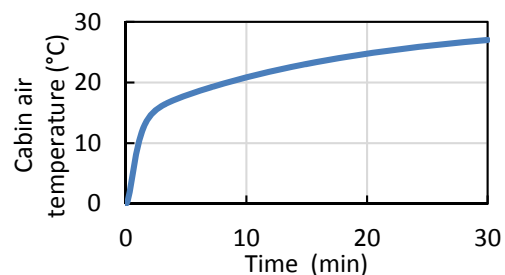


Figure 10. Warm-up transient cabin air temperature.

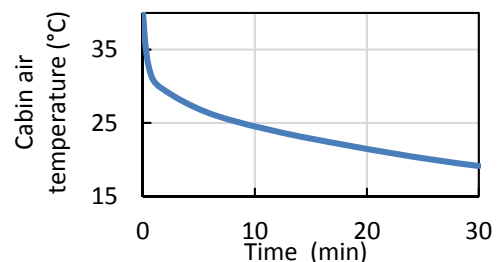


Figure 11. Cool-down transient cabin air temperature.

A rapid change of slope can be observed on both curves. It separates two transient phases. The first one corresponds to the air capacity and the second one to the wall and internal capacities. This separation is not so obvious in practice since the HVAC power also integrates a transient phase, which smoothes the curves.

3.5 Sensitivities – convergence

A sensitivity analysis with regards to the thermal capacitances has been conducted. It appears that the internal mass has 3 times more influence than the wall capacitances, but the dependency is not so important since an increase by 50% of the internal mass capacitance results in an increased less than 7.5 % of the required power.

On the other hand, the assumption with an internal mass separated from the air cabin by a convection resistance has been analyzed. Integrating directly the internal mass capacitance to the air (i.e. without convection resistance) increases by 9% and 28% respectively the heating needs and the cooling needs. In addition, if the inner radiative couplings are considered between the walls and the air (instead of between the walls and the internal mass), the cooling needs are increased by 15%. Of course, those two approaches do not represent the reality, but this analysis emphasizes the fact that the internal mass has to be considered properly.

Furthermore, a sensitivity analysis on the conditions given in Table 2 is conducted with the following cases:

- Case 1: targeted temperature at the end of the convergence 2°C lower than steady state.
- Case 2: required duration to reach proper cabin temperature is 10 min instead of 15 min initially.
- Case 3: initial condition: vehicle parked under sun (and so initial cabin temperatures higher than outside temperature).

Results are presented in Table 3. The two first cases have impacts around 10%. On the other hand, taking into account a case with the cabin under sun for hot cases increased by more than 40% the thermal needs.

Table 4. Sensitivities to operating conditions.

<i>Thermal needs (W) and relative gap (%).</i>			
Text	Case 1	Case 2	Case 3
0°C	-10.0%	12.5%	N/A
40°C	-8.6%	8.6%	42.9%

3.6 Cabin needs – convergence

The thermal needs for convergence are presented in Figure 12. In addition to the ‘with or without recirculation’ grey band, a point at 40°C corresponding to the case with a parked vehicle under the sun is added since it increases a lot the needs.

For the convergence mode, the heating needs are roughly twice the needs for the steady state mode. The increase is less important for the cooling needs.

Those results suggest that, for an electric vehicle, the usage will have a strong influence on the thermal needs and, thus, on the vehicle range. Indeed, if the vehicle is used only for urban trip, short distances but lots of time at low speed or even stopped, the HVAC consumption will be high due to convergence mode.

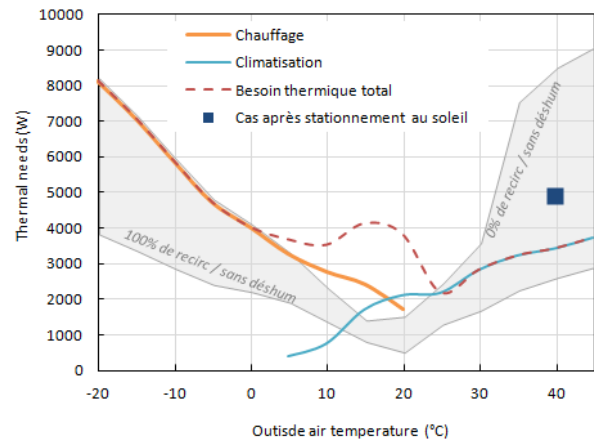


Figure 12. Convergence thermal needs

4 Conclusion

A transient thermal model of a vehicle cabin has been developed. Different model assumptions have been discussed and analyzed. The results showed that the model is not so sensitive to some non trivial modeling aspects such as the convective correlations. However, it appeared that neglecting the inner radiation or the internal thermal node can have an important impact in some cases (up to 15 to 25% approximately). On the other hand, it appeared that, if one wants to determine the thermal cabin needs, the considered arbitrary conditions actually dramatically influence the results. In addition, it is important to consider the dehumidification needs and the latent heat generated in the evaporator.

The model allowed establishing the thermal needs for the steady state case as well as for the convergence mode. The heating needs for the convergence mode are roughly twice the needs for the steady state mode. In addition, the thermal needs are of the same order of magnitude as the traction needs. The results clearly indicate that the thermal needs are damageable for electric vehicle range, particularly the heating, and that the vehicle usage is determinant.

In future work, the proposed model will be integrated in an electric vehicle model in order to study different thermal strategies such as the use of a heat pump, pre-conditioning and thermal storage. The model can also be used to evaluate improved cabin thermal design.

Acknowledgements

This work received financial support from the French National Research Agency (ANR) under the ELEC-HP project (grant number ANR-11-VPTT-005).

Nomenclature

<i>Symbol</i>	<i>Name [unit]</i>
A	Area [m ²]
c_p	Specific heat [J.K ⁻¹ .kg ⁻¹]
F	View factor [-]
h	Enthalpy per unit mass [J.kg ⁻¹]
L	Length [m]
L_v	Latent heat of vaporization [J.kg ⁻¹]
\dot{m}	Mass flow rate [kg.s ⁻¹]
Nu	Nusselt number [-]
Pr	Prandtl number [-]
\dot{Q}	Heat transfer rate [W]
R	Thermal resistance [K.W ⁻¹]
Ra	Rayleigh number
Re	Reynolds number [-]
T	Temperature [K]
t	Time [s]
u	Internal energy per mass [J.kg ⁻¹]
V	Volume [m ³]
v	Velocity [m.s ⁻¹]
x_i	Mass ratio of the species i in a mixture [kg.kg ⁻¹]
<i>Greek symbols</i>	
α	Absorptivity [-]
φ	Relative humidity [-]
ε	Emissivity [-]
ρ	Density [kg.m ⁻³]
σ	Stefan's constant [-]
τ	Recirculation [-] or transmissivity [-]

Subscripts or Superscripts

<i>abs</i>	Absorbed
<i>air</i>	Related to moist air
<i>blown</i>	Blown
<i>cab</i>	Cabin
<i>cond</i>	Conduction
<i>conv</i>	Convection

<i>env</i>	Environment
<i>evap</i>	Evaporator
<i>ext</i>	Exterior
<i>IR</i>	Infrared
<i>In</i>	Inner side
<i>HX</i>	Heat exchanger
<i>mix</i>	Mixed
<i>out</i>	Outer side
<i>pass</i>	Passengers
<i>return</i>	Return
<i>sat</i>	Saturation
<i>sol</i>	Solar
<i>trans</i>	Transmitted
<i>w</i>	Related to water
<i>wall</i>	At or through wall surface

References

- Abou Eid, R., 2016. Rapport - Passenger comfort and HVAC thermal load in a tramway.
- Al-Kayiem, H.H., Sidik, F.B.M., Munusammy, Y.R., A., L., 2010. Study on the Thermal Accumulation and Distribution Inside a Parked Car Cabin. *Am. J. Appl. Sci.* 7, 784–789.
- ASHRAE, 2009. ASHRAE Handbook—Fundamentals.
- Bergman, T.L., Lavine, A.S., Incropera, F.P., Dewitt, D.P., 2011. Fundamentals of Heat and Mass Transfer, 6th ed. John Wiley & Sons.
- Boukhris, Y., Gharbi, L., Ghrab-Morcos, N., 2009. Modeling coupled heat transfer and air flow in a partitioned building with a zonal model: Application to the winter thermal comfort. *Build. Simul.* 2, 67–74. doi:10.1007/S12273-009-9405-8
- Daoud, A., Galanis, N., 2008. Prediction of airflow patterns in a ventilated enclosure with zonal methods. *Appl. Energy* 85, 439–448. doi:10.1016/j.apenergy.2007.10.002
- Fujita, A., Kanemaru, J. ichi, Nakagawa, H., Ozeki, Y., 2001. Numerical simulation method to predict the thermal environment inside a car cabin. *JSAE Rev.* 22, 39–47. doi:10.1016/S0389-4304(00)00101-6
- Inard, C., Bouia, H., Dalicieux, P., 1996. Prediction of air temperature distribution in buildings with a zonal model. *Energy Build.* 24, 125–132. doi:10.1016/0378-7788(95)00969-8
- Iskandar, B.S., 2010. Study on the Thermal Accumulation and Distribution Inside a Parked Car Cabin Hussain H . Al-Kayiem , M . Firdaus Bin M . Sidik and Yuganthira R . A . L Munusammy Department of Mechanical Engineering , University Technology PETRONAS ., Ashrae Stand. 7, 784–789.
- Kataoka, T., Nakamura, Y., 2001. Prediction of thermal sensation based on simulation of temperature distribution in a vehicle cabin 30, p, 195–212.

- Li, W., Sun, J., 2013. Numerical simulation and analysis of transport air conditioning system integrated with passenger compartment. *Appl. Therm. Eng.* 50, 37–45. doi:10.1016/j.applthermaleng.2012.05.030
- Marcos, D., Pino, F.J., Bordons, C., Guerra, J.J., 2014. The development and validation of a thermal model for the cabin of a vehicle. *Appl. Therm. Eng.* 66, 646–656. doi:10.1016/j.applthermaleng.2014.02.054
- Mezrhab, A., Bouzidi, M., 2006. Computation of thermal comfort inside a passenger car compartment. *Appl. Therm. Eng.* 26, 1697–1704. doi:10.1016/j.applthermaleng.2005.11.008
- Sanaye, S., Dehghandokht, M., Fartaj, A., 2012. Temperature control of a cabin in an automobile using thermal modeling and fuzzy controller. *Appl. Energy* 97, 860–868. doi:10.1016/j.apenergy.2012.02.078
- Sevilgen, G., Kilic, M., 2012. Three dimensional numerical analysis of temperature distribution in an automobile cabin. *Therm. Sci.* 16, 321–326. doi:10.2298/TSCI1201321S
- Swinbank, W.C., 1963. Long-wave radiation from clear skies. *Q. J. R. Meteorol. Soc.* 89, 339–348. doi:10.1002/qj.49708938105
- Torregrosa-Jaime, B., Bjurling, F., Corberan, J.M., Di Sciullo, F., Paya, J., 2015. Transient thermal model of a vehicle's cabin validated under variable ambient conditions. *Appl. Therm. Eng.* 75, 45–53. doi:10.1016/j.applthermaleng.2014.05.074
- Versteeg, H., Malalasekera, W., 2007. An introduction to computational fluid dynamics: The finite volume method, PEARSON Pr. ed.
- Wischhusen, S., 2012. Modelling and Calibration of a Thermal Model for an Automotive Cabin using HumanComfort Library. *Int. Model. Conf.* 253–263. doi:10.3384/ecp12076253
- Zhang, H., Dai, L., Xu, G., Li, Y., Chen, W., Tao, W.Q., 2009. Studies of air-flow and temperature fields inside a passenger compartment for improving thermal comfort and saving energy. Part II: Simulation results and discussion. *Appl. Therm. Eng.* 29, 2028–2036. doi:10.1016/j.applthermaleng.2008.10.005
- Zhu, S., Demokritou, P., Spengler, J., 2010. Experimental and numerical investigation of micro-environmental conditions in public transportation buses. *Build. Environ.* 45, 2077–2088. doi:10.1016/j.buildenv.2010.03.004

Simulative Comparison of Mobile Air-Conditioning Concepts for Mechanical and Electrical Driven Systems

Arnim von Manstein¹ Dirk Limperich¹ Shivakumar Banakar²

¹ Daimler AG, Germany, {arnim.von_manstein, dirk.limperich}@daimler.com

²Mercedes Benz R & D India Pvt. Ltd., India, Shivakumar.banakar@daimler.com

Abstract

Ever increasing energy demand and the stringent emission norms have resulted in the need for developing more efficient automotive systems. Fuel economy and emission targets are the two important driving factors in the development of an automobile. Efficiency of a Mobile Air-Conditioning system (MAC) has a considerable impact on the fuel economy of an automobile. This study involves simulative comparison of MAC concepts for mechanical & electrical driven systems. System models are developed for MAC concepts using Dymola simulation tool. Drive cycles considered in this study correspond to the real time driving scenarios and ambient conditions. From this study the conclusions are drawn about the most efficient ways to reach the thermal comfort for the passenger cabin in an automobile.

Keywords: *Energy Efficiency; MAC; HVAC; LV; HV; MHEV; PHEV; BEV; Compressor; Dymola.*

1 Introduction

MAC systems were considered to be an optional equipment in the past for the automobiles. Nowadays they have become an integral part of all the cars that are produced. In the recent time momentum has gained for the development of hybrid & electric vehicles which means that vapor compression refrigeration systems will become a necessity for passenger cabin cooling as well as battery cooling. The UN estimated the sale of more than one billion cars with MAC system in 2015 which has resulted in 2.3 gigatons of carbon dioxide adding into the environment (Lemke et al. 2011). In order to reduce the world wide emission caused by automobiles there are two ways. One is to reduce the direct emissions of cars. For example implementing stringent emission norms, use of alternative powertrain concepts, reducing weight by replacing heavy cast iron components are very simple and effective methods (Slattery et al. 2010). The second way would be to reduce the indirect emissions by changing for example refrigerants of the refrigeration cycle. Especially for the second way the European Union set in the directive 2006/40/EC the way to reduce the Global Warming Potential (GWP) of refrigerants. It is set that from January 2017 only refrigerants with a GWP of lower than 150 are allowed

to be sold in Europe (Europäische Union 6/14/2006). In order to fulfill this directive there are two alternatives, one is R1234yf (2, 3, 3, 3-Tetrafluorpropen) with a GWP of 4 and the other is R744 (CO₂) with a GWP of 1.

The refrigerant loop in a MAC system consists of an evaporator, compressor, condenser and expansion valve. The low temperature, low pressure vapor is compressed by a compressor to a high temperature and high pressure vapor. This vapor is condensed into high pressure liquid in the condenser, by rejecting heat to a low temperature ambient air and then passes through the expansion valve. Here, the high pressure liquid is throttled down to a low pressure liquid and passed on to an evaporator, where it absorbs heat from the cabin air and vaporizes into a low pressure vapor. Then it reaches compressor suction line and the cycle repeats. In order to increase the efficiency, Daimler AG, Germany, introduced at early stages for example internal heat exchanger (IHX). There the refrigerant is subcooled after it exits the condenser. Therefore you can reach lower temperatures after expansion to increase cooling capacity on the one hand and ensure compressor safety on the other side due to its superheating effect after the refrigerant exits the evaporator. By ensuring superheat at the compressor suction, we can eliminate liquid lock.

A key component of a MAC system is the compressor. This refrigerant compressor is driven in state of the art automobiles with a belt drive directly from an Internal Combustion Engine (ICE). With hybrid powertrain such as Mild Hybrid Electric Vehicle (MHEV), Plug-in Hybrid Electric Vehicle (PHEV) and Battery Electric Vehicle (BEV), the compressor is driven by an electric motor via the necessary voltage level. Especially for MHEV there are changes in the automotive industry to reduce this voltage level limit up to 60V. The further reduction in voltage level with electrification of the powertrain is known and also the new voltage level already described in detail (Coppin, Potteau 2015).

With this electrification and with the implementation of stringent emission norms to reduce emission of carbon dioxide, have resulted in the tremendous changes for MAC.

2 Experimental Procedure

System simulation techniques play a very important role in the development and evaluation of various concepts for MAC system. So far the simulation predictions were mainly used to determine whether the designed MAC system is able to achieve thermal comfort for the customer at certain boundary conditions.

Going a step further the car manufacturers are interested in studying the system behavior under real time operating conditions of the customer. For sure testing with prototypes is very important testing set up, but lot of what if scenarios can be addressed through simulation studies in the early stages of concept development and concept evaluations.

In this study we will limit ourselves to a comparison of MAC system driven by a mechanical compressor and an electrical compressor. Because, already there are large number of scientific papers that discuss the overall efficiency of ICE and hybrid powertrain, such as (Carpetis 2000). These concepts are evaluated using Dymola simulation tool. In general this study will result in an overall discussion about what is the most efficient way to reach thermal comfort in an automotive cabin.

In our special use case we evaluate two types of compressor that come along in automotive application:

1. Regular mechanical compressor driven by belt drive
2. HV electrical compressor with a voltage level around 400V

Fortunately, there are several reports about World's climate conditions such as the FAT 224. In order to reduce complexity we just take MAC system operation into consideration and will provide data for the ambient air temperature of 22°C, 29°C and 35°C.

As mentioned, these systems are operated by customers in different ambient and driving conditions. Most severe conditions for MAC systems are higher ambient temperatures at low car speed.

Daimler AG has selected one drive cycle (MBVT), which was created using the real time data from vehicles operating in various driving scenarios and ambient conditions all over the world.

The MBVT is a mixed drive cycle with $\approx 50\%$ inner city driving, $\approx 35\%$ interstate and $\approx 15\%$ highway (Autobahn) profile. Additional parameters in the drive cycle include time/speed gradients, acceleration, deceleration and the pitch. The stop phases are calculated to be roughly about 4.5 mins.

The MAC system investigated in this study is based on a series production S-Class system which is operated in both systems with a hydrofluoroolefin refrigerant (such as R134a/R1234yf). This set up is described in detail in Section 3.

3 System Description and Simulation Model

The vapor compression refrigeration system components configurations that we have used for investigation in our study corresponds to the series production S-Class car. Table 1 describes geometrical parameters of all components used in the system.

Table 1. Component details & geometrical parameters

Component	Description	Geometry	
Condenser	Cross flow, Fin & Tube heat exchanger with 2 layers	Height (mm)	453.10
		Width (mm)	640
		Depth (mm)	12
		HTA (m ²)	2.0158
		Volume (m ³)	5.552E-4
Evaporator	Air cooled, Cross flow, Fin & Tube heat exchanger	Height (mm)	223
		Width (mm)	303
		Depth (mm)	50
		HTA (m ²)	1.199
		Volume (m ³)	0.001178
Internal Heat Exchanger	Concentric tube in tube heat exchanger	Length (mm)	498
		HTA (m ²)	0.08372
Expansion Valve	Thermal Expansion Valve with superheat feedback	Capacity (ton)	2.0
Compressor 1	Swash plate variable displacement compressor	Capacity (cc)	170.0
Compressor 2	Scroll compressor	Capacity (cc)	33

Simulation models of the components of refrigerant loop are developed using a multi-engineering dynamic simulation tool Dymola (version 2015 FD01) and Air-conditioning Library (version 1.9). The snapshot of the system model developed in Dymola is as shown in Figure 2. In the first step heat exchangers like Condenser, Evaporator & IHX (Internal Heat Exchanger) are modelled using the templates from air-conditioning library and geometrical data. The developed heat exchanger models are then calibrated and validated using the calibration toolbox within

Dymola and experimental data from suppliers. The compressor model is as described below.

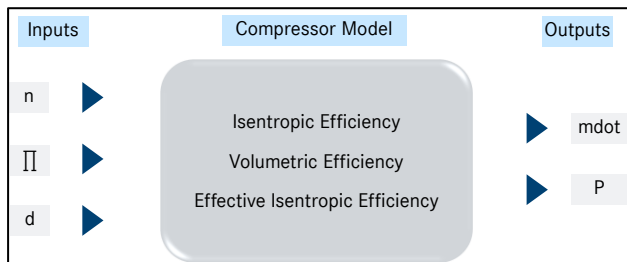


Figure 1. Compressor model variables

Where,

n = Compressor speed

π = Compressor pressure ratio

d = Relative displacement

\dot{m} = Refrigerant mass flow rate through compressor

P = Compressor power consumption

Efficiencies of the compressor i.e., isentropic efficiency, volumetric efficiency and effective isentropic efficiency are modeled as a function of compressor speed, relative displacement and pressure ratio. Correlations for the compressor efficiencies are developed using the measured data from the supplier. Range of the compressor measurement data used for the modelling is shown in Table 2. The standard deviations in isentropic efficiency & volumetric efficiency models are found to be 4.81% & 2.9% respectively. Compressor performance parameters like mass flow rate, power consumption and compressor discharge temperature are computed using these efficiency values.

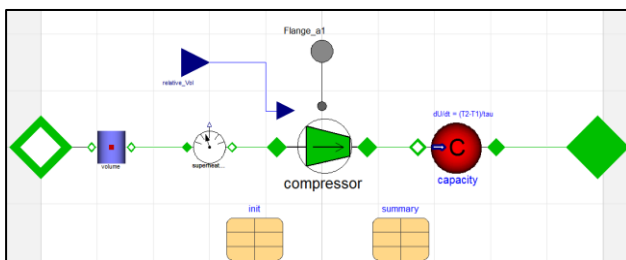


Figure 2. Representation of a compressor model developed in Dymola.

Table 2. Details of the compressor measured data used for modelling.

Parameters	π	n (rpm)	d
Range	2.5 to 8.4	700 to 8000	0.35 to 1.0

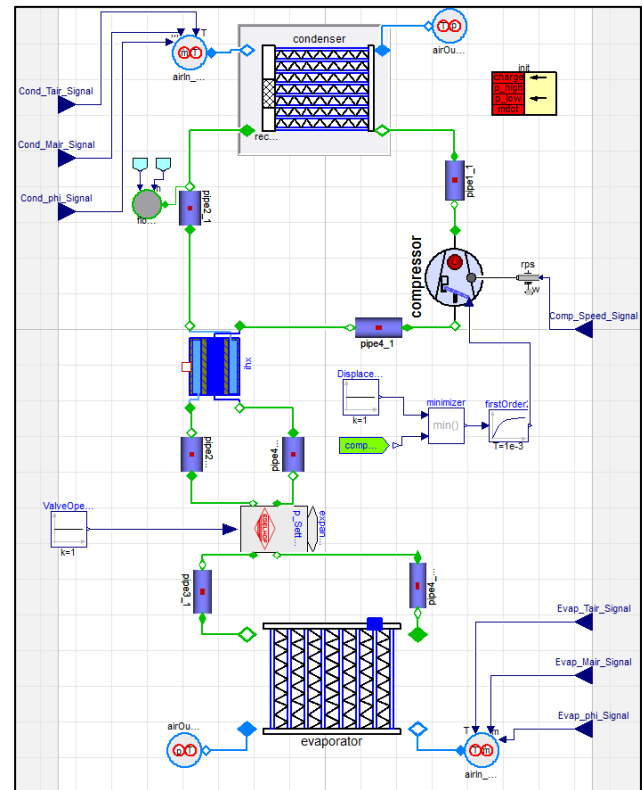


Figure 3. System model developed in Dymola simulation tool.

The validated component models were then integrated to develop the system model as shown in Figure 2. Two variants of the system models were developed; “System A” with engine driven mechanical swashplate compressor and “System B” with battery driven electric scroll compressor; keeping all other components same. The system simulations were carried out in the subsequent stage to evaluate and compare the performance of these two systems.

As previously described (Dermont et al. 2016) the main challenges in simulating the real time conditions are the stop phases, where the mechanical swash plate compressor, during a vehicle stand still at a traffic light, is shut off. In order to mimic the stop phase of the compressor in simulations, the compressor speed was limited to 10 rpm. Because of the very low rpm of the compressor, refrigerant mass flow in the system is of the order of $10\text{E-}4$ kg/s which results in a zero-mass flow scenario. Flow reversal was observed in some of the components of the system during the stop phase which resulted in lots of numerical problems in the simulation.

Numerical problems in the simulations were caused by the flow reversal that was observed at the outlet of the condenser. This was found to be because of the dynamic enthalpy value that was assigned in the flow source charge, which is connected at the outlet of condenser. The flow source charge is used to ensure fixed quantity of charge inside the system. To fix these numerical problems, the enthalpy input in the flow

source charge was assigned a constant value. Simulation cases were re-simulated again using Dymola 2017 FD01, where the handling of zero flow simulation was found to be better. It was observed that the simulations were faster and smoother in the new version of Dymola as compared to the previous versions.

4 Results & Discussion

After developing the system models, simulations were done using the MBVT drive cycle for 3 different ambient temperatures and 2 different air mass flows over the evaporator at each ambient temperature. The details of the boundary conditions are as shown below in Table 3.

Table 3. Details of the comparison study for each temperature and air massflow rate

MBVT	T_ambient = 22°C		T_ambient = 29°C		T_ambient = 35°C	
	[kg/min]		[kg/min]		[kg/min]	
	1,5	2	2	3	4	6

Results are described in the following section.

4.1 Ambient Temperature 22°C

The simulations at 22°C ambient temperature and 55% relative humidity were done with an air mass flow of 1,5 kg/min over the evaporator. At 29°C ambient (40% humidity) with a massflow of 2 kg/min and at 35°C (40% humidity) with 4 kg/min. The following simulation results are showing data of the mechanical and electrical system in one graph for each evaluation criterion. The relevant criteria for this paper are:

- Pressure (suction and discharge) [bar]
- Evaporator air outlet temperature [°C]
- Refrigerant Massflow [kg/h]
- Cooling capacity [kW]
- Coefficient of performance (standardized) [-]

In Figure 4, we can see that the suction and discharge pressure reach their aimed ratios for both the systems. For the electrical system both suction & discharge pressures run more stable, especially for the suction pressure. This fluctuating pressure from the mechanical system can result for example in pulsation which have a negative impact on the acoustics of such a system.

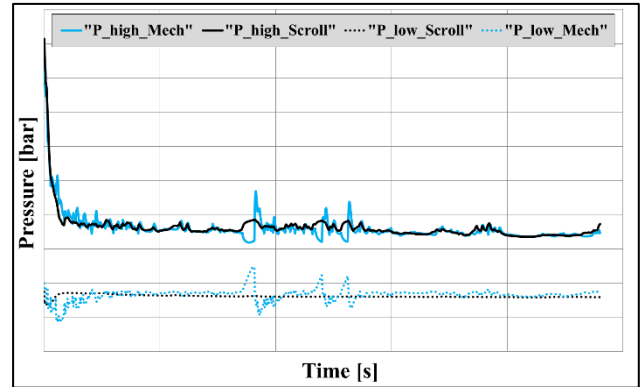


Figure 4. Suction and discharge pressure of both system as a function of time

There are two main aspect visualized in Figure 5. The first one is that the massflow for mechanical system experiences large variations. This is caused by the fact that the compressor is being driven via the belt drive of the ICE. Also in the simulation model, the compressor speed was reduced to a minimum of 10 rpm at stop phases. A complete stand still of the mechanical compressor was not achievable with this Dymola model. This leads us to the second main aspect of Figure 5. Due to the reduction of compressor speed resulting in a reduction of massflow, the air outlet temperature of the evaporator couldn't be kept constant. For longer periods, for example during long phases of traffic signal, it is observed that the temperature increases dramatically. This will result in a massive discomfort in the passenger cabin as compared to the electrical system. The electrical scroll compressor was operated during all stop phases in the drive cycle that results in maintaining the thermal comfort of the passengers.

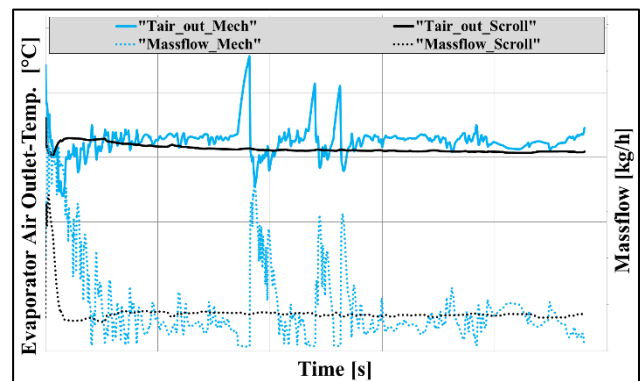


Figure 5. Evaporator air outlet temperature and massflow as a function of time

For the electrical system in general it is evident that it runs more stable compared to the system with the mechanical compressor at 22°C ambient temperature. This hypothesis is affirmed in Figure 6. The massive variation of the cooling capacity of the mechanical system is again linked to the stop phases. During the

stop phase it will decrease because of the reduced compressor speed. Additionally, the temperature of the evaporator increase, as the ambient air at 22°C is blown over it. Once the system starts again, the system tries to reach the evaporator air set point temperature of 3°C.

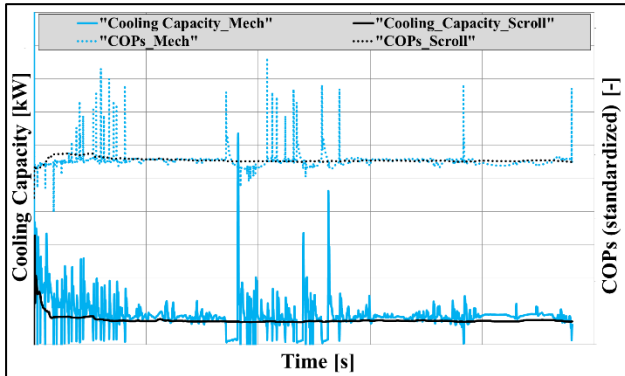


Figure 6. Cooling capacity and COPs as a function of time

Also the variation in COP of the mechanical system can be explained with the stop phases. There it is important to know that mechanical compressors in automotive application have a variable displacement which is controlled via parameters of the refrigeration loop. So once the car starts after a stop phase the loop tries to reach the evaporator air set temperature. The speed of the compressor is fixed to the given profile so the only chance to increase the capacity is to increase the compressor displacement. This is also shown in Figure 4 with the graph of the massflow. There it is noticeable how the massflow is changing due to the changes in the displacement. Because of the design of the variable displacement compressor it is found to be operating in a more efficient mode at lower speed and full displacement. After the set point is achieved the displacement decreases and it runs afterwards in a rather inefficient mode in comparison to the scroll compressor. In Figure 6 the COP is standardized because at stop phases when the mechanical compressor is shut of we still gain the cooling capacity which is stored in the refrigerant loop. But in the meantime you lose thermal comfort in the vehicle cabin so it is needed to take this also into consideration in calculation of the COP. So the factor ε is introduced for calculating COPs. Its graph is plotted for the air outlet temperature over the evaporator in Figure 7.

$$COPs = COP \times \varepsilon$$

where,

$$\varepsilon = \frac{\vartheta_{evap_set}}{|\vartheta_{evap_out} - \vartheta_{evap_set}| + \vartheta_{evap_set}}$$

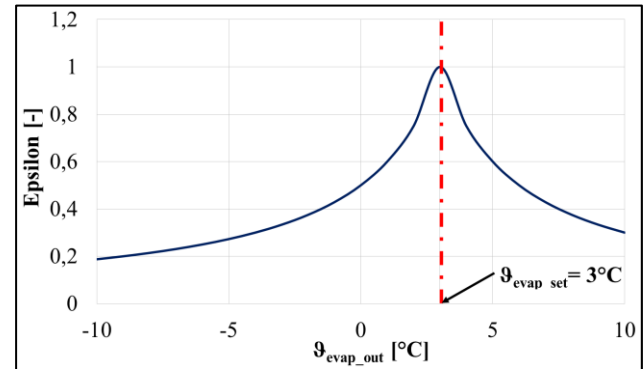


Figure 7. Correction factor (ε) for each temperature of the air at the evaporator outlet

4.2 Ambient Temperature 29°C

The following graphs for 29°C ambient are pretty similar to those of 22°C ambient. In Figure 8 we see that both systems attaining the same high pressure level and due to the constant speed of the electrical compressor the suctions pressure is in stable conditions.

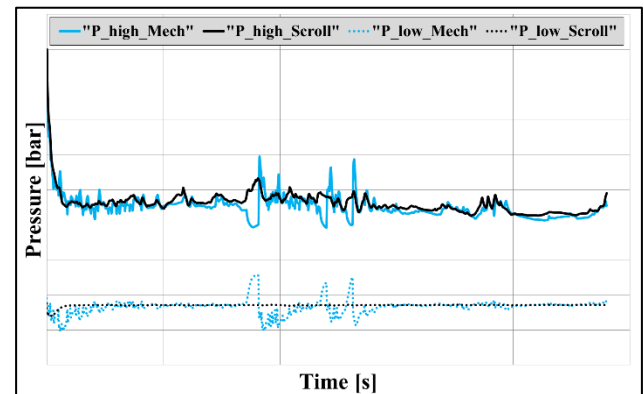


Figure 8. Suction and discharge pressure of both system as a function of time

In Figure 9 the advantages of an electrical system are evident. Constant refrigerant massflow and evaporator air exit temperatures are observed for most part of the system operation. Thermal comfort in the cabin is also maintained for most part of the drive cycle.

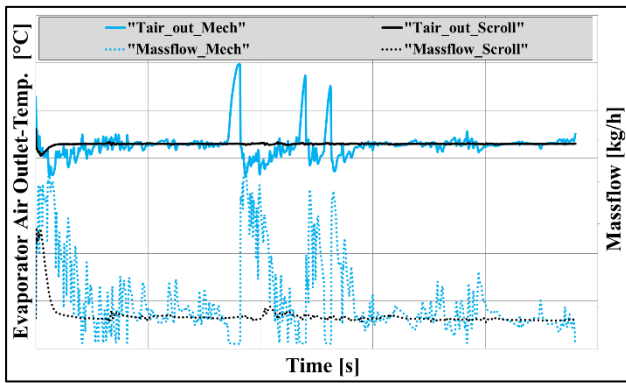


Figure 9. Evaporator air outlet temperature and massflow as a function of time

By visual comparison the system with an electrical compressor operates with a better COPs for most part of the drive cycle. The large peaks for the system with mechanical compressor are due to the stored cooling capacity in the system. Additionally, it is observed from Figure 8 that the high pressure for the mechanical system decreases during the stop phases.

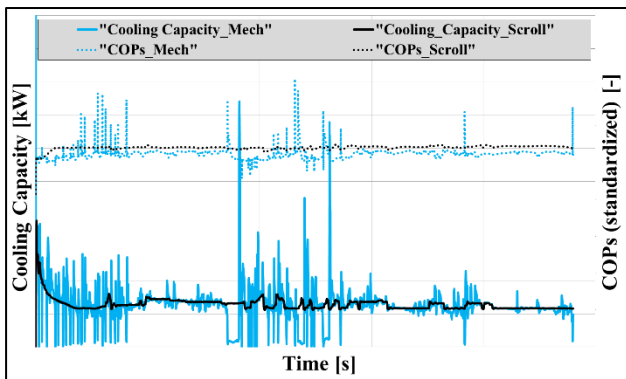


Figure 10. Cooling capacity and COP as a function of time

In order to compare both systems in terms of efficiency we compare the classic COP without standardization regarding cabin comfort:

$$\overline{COP_e} = \frac{\sum_{n=1}^t COP_{e_n}}{t} ; \overline{COP_{s_m}} = \frac{\sum_{n=1}^t COP_{m_n}}{t}$$

$$\rightarrow COP_v = \frac{\overline{COP_e}}{\overline{COP_m}} = 1,2851 \approx 128,5\%$$

So even under severe conditions the electrical compressor system runs at 29°C in a higher COP by 28,5% compared to the mechanical compressor system. The work that is needed over the driving cycle for the mechanical conferred to the electrical is higher by 57%. Although there is just an overall advantage of 28% the difference is with variation of the cooling capacity.

4.3 Ambient Temperature 35°C

In the end the simulations were done at an ambient temperature of 35°C. In statistics this might be the most severe temperature for Germany for example. But in southern parts of Europe and for example in the western region of the United States of America these temperatures occur more often. It is easy to detect that now both systems operate at the upper end of their capacities. In Figure 11 the suction pressure stays in the same regions except during the stop phases, but the high pressure differs a lot. This can be explained by the different compressor types. As previously described the mechanical compressor is a piston type and the electrical is a scroll one.

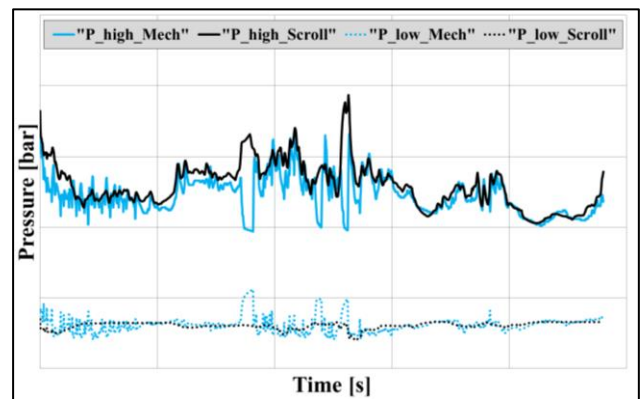


Figure 11. Suction and discharge pressure of both systems as a function of time

Although the evaporator air temperature over the evaporator is almost kept constant for the electrical system you can easily see by analyzing the plotted massflow that the compressor is also varying the speed. The massive differing shows that even if the scroll compressor is independent of the belt drive of the ICE it regulates dynamically for this high load case.

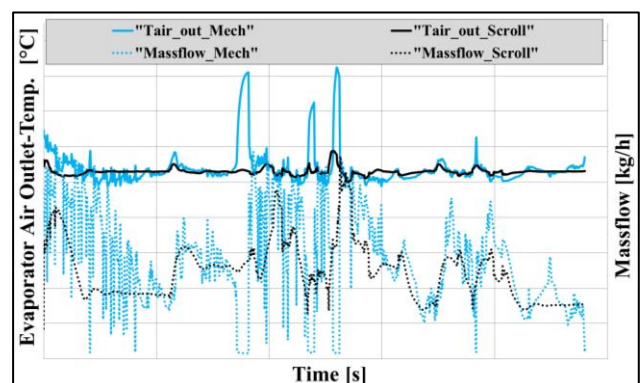


Figure 12. Evaporator air outlet temperature and massflow as a function of time

As a last the COPs in Figure 13 interestingly are no longer higher for the electrical system. This is also due

to the compression technique of the mechanical compressor. Piston type compressors have an optimum operation ratio at almost full stroke with 100% displacement. The average displacement over this driving cycle for the mechanical system is at 50%.

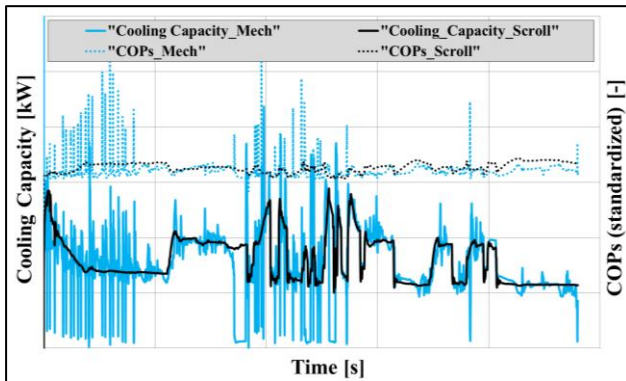


Figure 13. Cooling capacity and COPs as a function of time

Simulations plots are described for one evaporator air massflow rate at each ambient temperatures. All the plots aren't added to this paper but they do show similar behavior in general. The conclusion all in all stays the same.

5 Conclusion

For future investigation simulation tools such as Dymola will be mainly used for efficiency analysis and system optimizing operations. The comparative study outlined this strategy and justified this procedure. This simulation will be moreover expand in an even wider temperature range to complete the virtual behavior of an automotive A/C-loop. Especially transient simulations will become more and more sufficient. By simulating a mechanical compressor at all ambient temperatures and even an electrical at severe conditions the changes in the massflow are tremendously. By simulating just steady state conditions you can't visualize those changes that are also important for NVH (Noise Vibration and Harshness) analyses and cabin comfort for the customer. The shut-off phases of the mechanical system are the main issue as stated in chapter 3. For this point of view we could show that the system driven with an electrical scroll compressor has 20% higher COPs at 29°C than a system with a mechanical (belt driven) compressor. In terms of increasing the system efficiency, the hybridization of passenger cars is also a big chance for the thermal management of the cabin and powertrain. A validation of the simulation model is of course applicable and will be done as a next step.

Acknowledgements

The authors would like to thank and gratefully acknowledge the support received from RD/KIT and RD I/CCS departments at Daimler AG & MBRDI respectively.

Publication bibliography

Carpetis, C. (2000): Globale Umweltvorteile bei Nutzung von Elektroantrieben mit Brennstoffzellen und/oder Batterien im Vergleich zu Antrieben mit Verbrennungsmotor. Edited by Deutsches Zentrum für Luft und Raumfahrt e.V. Institut für Technische Thermodynamik. Stuttgart (STB-Bericht, 22), checked on 10/12/2016.

Coppin, Oliver; Potteau, Sébastien (2015): 48-V-Hybrid-Systemarchitektur zur Reduzierung der CO₂-Emissionen. In *ATZ elektronik* 10 (02).

Dermont, Pieter; Limperich, Dirk; Windahl, Johan; Prölss, Katrin; Kübler, Carsten (2016): Advances of Zero Flow Simulation of Air Conditioning Systems using Modelica. In : Deployment of high-fidelity vehicle models for accurate real-time simulation, 2011-02-05: Linköping University Electronic Press (Linköping Electronic Conference Proceedings), pp. 139–144.

Europäische Union (6/14/2006): Richtlinie 2006/40/EG des Europäischen Parlaments und des Rates vom 17. Mai 2006 über Emissionen aus Klimaanlage in Kraftfahrzeugen und zur Änderung der Richtlinie 70/156/EWG des Rates. 2006/40/EG, revised L161/12.

Lemke, Nicholas; Mildenerberger, Julia; Graz, Martin (2011): Unterstützung der Markteinführung von Pkw-Klimaanlagen mit dem Kältemittel CO₂ (R744). Prüfstandsmessungen und Praxistest. Im Auftrag des Umweltbundesamtes. Edited by Umweltbundesamt. Dessau-Roßlau (Texte, 64). Available online at <http://www.uba.de/uba-info-medien/4184.html>, checked on 6/25/2015.

Slattery, B. E.; Edrissy, A.; Perry, T. (2010): Investigation of wear induced surface and subsurface deformation in a linerless Al–Si engine. In *Wear* 269 (3-4), pp. 298–309. DOI: 10.1016/j.wear.2010.04.012.

Nomenclature

A/C	Air Conditioning
BEV	Battery Electric Vehicle
COP	Coefficient of Performance
COPs	Coefficient of Performance standardized
GWP	Global Warming Potential
ICE	Internal Combustion Engine
IHX	Internal Heat Exchanger
MAC	Mobile Air Conditioning
MBVT	Special Daimler Driving Cycle
MHEV	Micro Hybrid Electric Vehicle

NVH	Noise Vibrations and Harshnes
PHEV	Plug-In Hybrid Electric Vehicle
RPM	Rounding Per Minute
n	Compressor speed
π	Compressor pressure ratio
d	Relative displacement
mdot	Refrigerant mass flow rate through compressor
P	Compressor power consumption

Duty Cycle for Low Energy Operation of a Personal Conditioning Device

Rohit Dhumane Jiazhen Ling Vikrant Aute Reinhard Radermacher

Center for Environmental Energy Engineering, University of Maryland, College Park, 4164 Glenn L. Martin Hall Bldg., MD 20742, USA

{dhumane, jiazhen, vikrant, raderm}@umd.edu

Abstract

The Roving Comforter (RoCo) is an innovative personal thermal management technology that provides ultimate personal thermal comfort for individuals in inadequately or even unconditioned environments. It is a miniature heat pump system mounted on a robotic platform capable of autonomously following individuals to deliver comfort by directing hot or cold air through automatically controlled nozzles. This allows buildings to relax their thermostats up to 4°F (2.2°C), leading to energy savings anywhere between 10 to 30% depending on climatic conditions. Since RoCo is a portable device, it needs to be operated on battery. A smaller battery pack will require frequent charging making it inconvenient for the users, while a bigger battery pack will add to the weight of the device leading to higher power consumption during motion. To address this problem, a multi-physics model for the operation that incorporates thermodynamics, electricity and mechanics of RoCo is developed and two duty cycles analyzed. Strategies for the operation of RoCo are provided from the observations of results.

Keywords: Battery, Air-conditioner, Duty-cycle

1 Introduction

Climate change and global warming have been hot topics of discussion over past few decades. The greenhouse gas emission from human activities has lead to disruption of several natural systems leading to rising sea-levels, increased ground instability in mountains and change in seasonal winds. The United Nations IPCC has identified the building industry as the one with the most climate mitigation potential (Intergovernmental Panel on Climate Change Fourth Assessment, 2007).

Building Heating, Ventilation and Air Conditioning (HVAC) account for 13% of energy consumption in the United States (United States Department of Energy, 2011). Much of this energy goes into maintaining narrow indoor temperature ranges that building operators consider necessary for comfort but are really not necessary for occupant comfort (Zhang et al., 2011).

Hoyt et al. (2015) demonstrated the potential of energy savings from extending thermostat set-points in the building. They concluded that if it were possible to relax the

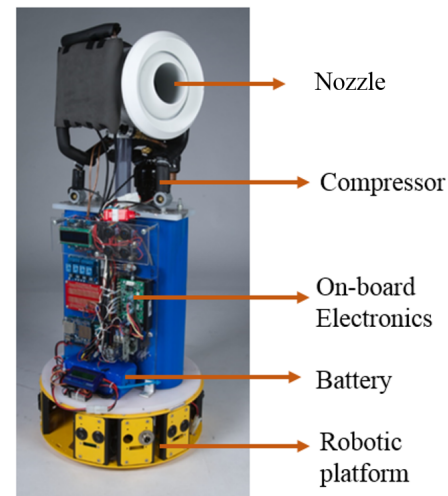


Figure 1. Current prototype of RoCo

temperature range in either the hot or cold direction, total HVAC energy is reduced at a rate of 10% per °C. To enable expansion of building set-point temperatures, it is necessary to provide supplementary Personal Conditioning System (PCS) operating at significantly lower energy consumption.

PCS offer dual benefits of energy saving and increased comfort. As a result, several PCS have been developed and are summarized in the review articles by Zhang et al. (2015) and Veselý and Zeiler (2014). However, except for the desk and ceiling fans, they are not commercially available. This can be attributed to a variety of factors unique to the designs like poor thermal performance, low energy efficiency, high cost and poor aesthetics. To address these issues and to achieve the benefits of PCS, an innovative robotic personal conditioning device called the Roving Comforter (RoCo) shown in Figure 1, is being developed (Du et al., 2016).

2 System Description

In technical terms, RoCo is a vapor compression system mounted on an autonomous robotic platform and delivering comfort by directing hot or cold air using its automat-

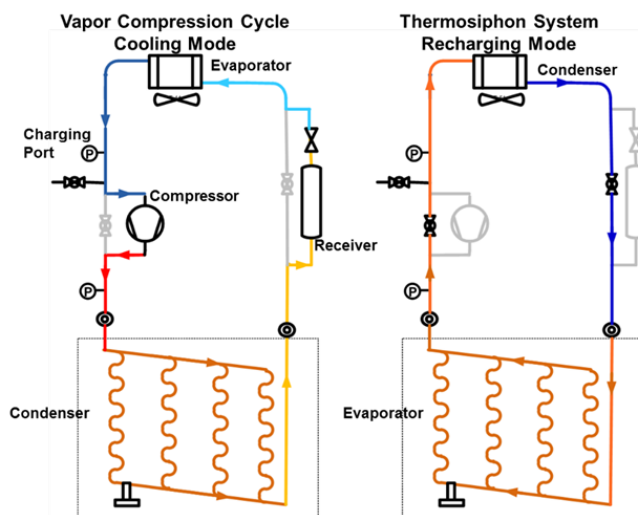


Figure 2. Conceptual representation of RoCo in operation

ically controlled nozzles. It is a heat pump on wheels.

RoCo stores its condenser heat in a compact phase change material based thermal storage which needs to be recharged before its next cooling operation. Thus RoCo operates in two modes and it is necessary to size the battery to deliver multiple operating cycles. Figure 2 shows the schematic of the two alternating modes of RoCo. The left schematic in Figure 2 shows the onboard vapor compression system using R134a as the refrigerant. The cooling operation is terminated when the PCM surrounding the condenser is completely melted. Before the next cooling operation, there is a need to re-solidify the PCM. This PCM recharge is achieved by a gravity assisted thermosiphon operation. Details of this recharge operation and its modeling are discussed in Dhumane et al. (2016).

The current prototype has separate battery packs for its robotic platform and vapor compression system. For the new prototype, a single unified battery pack is desired. An innovative nozzle design which permits nozzle rotation using motors in both horizontal and vertical plane is being developed for the new prototype. To understand various power draws for a single unified battery pack a duty cycle, representative of a worst-case operation of the new RoCo prototypes incorporating all the modifications is conceptualized. Simulations are carried out for the operation of various components to come up with strategies for increased the battery operation time.

3 Component Modeling

The model for the current simulation is shown in Figure 3. The top portion of the model contains components for modeling the vapor compression cycle. The compressor pumps the refrigerant in the circuit. A non-adiabatic tube element accounts for heat losses from the refrigerant between the compressor and the condenser. The condenser consists of inlet and outlet headers, and four refrigerant tubes immersed in PCM (See Figure 2 to understand the tube layout). The headers are modeled by lumped refriger-

ant control volumes. The four refrigerant circuits formed by each of the refrigerant tube immersed in the PCM are assumed to be symmetric. To avoid computational expense, only one of these refrigerant circuits is modeled. Flow splitter and flow mixer components are used to accomplish this. The splitter component divides the refrigerant mass flow rate into four equal parts and the mixer merges it back to resume the original mass flow rate for the next component in the refrigerant circuit. The PCM blocks interact with the refrigerant control volume using the `HeatPort` interface. The refrigerant then flows through receiver, valve and evaporator before reaching the compressor. Component models for refrigerant tube connecting various components are also included.

The bottom left portion contains the battery model and a power load component. The latter needs power draw as an input. The components which draw current from the battery are Compressor, fan, nozzle, robotic platform and on-board electronics. The power consumption from all of them is added and provided as input for the power load component. It then determines the current draw from the battery. This section discusses equations involved in calculating the power consumption from all these components.

3.1 Vapor Compression System Components

RoCo is a heat pump on wheels and detailed modeling for vapor compression cycle is carried out using components from CEE Modelica Library (CML) (Qiao et al., 2015). Components used from the library are compressor, evaporator, pipe, receiver and fixed orifice expansion device.

3.2 Phase Change Material Heat Exchanger

The condenser consists of helical refrigerant tubes surrounded by the phase change material (PCM). The PCM melting is a complex phenomenon due to the fact that the solid-liquid boundary moves depending on the rate of heat transfer and hence its position with time forms part of the solution. The rate of heat transfer varies progressively during the melting due to varying effects of conduction and natural convection. It decreases in at early times, attains a minimum, then rises again to a maximum and subsequently decreases (Sparrow et al., 1977). The helical nature of the refrigerant tube further increases the complexity by making the problem 3D. Due to strong non-linear nature of the problem, a simple 2D problem of melting in a square cavity may take several days on a personal computer (Wang et al., 2010). Finally, the two-phase refrigerant circuit exchanging heat with the PCM adds difficulty in convergence.

The model used in the current work is a trade-off for accuracy, complexity and usability. The PCM block is taken as a lumped control volume to eliminate the momentum equation. Two components are used to model PCM: `PCMConductor` and `PCM Capacitor` which are PCM analogous versions of `HeatCapacitor` and `ThermalConductor` from the Thermal package of Modelica Standard Library. Temperature transforming model by Cao and Faghri (1990) is used to model the energy equation since it also captures temperature glide over melting without much oscillatory effects. A heat transfer coefficient vs melt fraction profile based on the various heat transfer regimes discussed in Sparrow et al. (1977) is used and fitted to match experimental data.

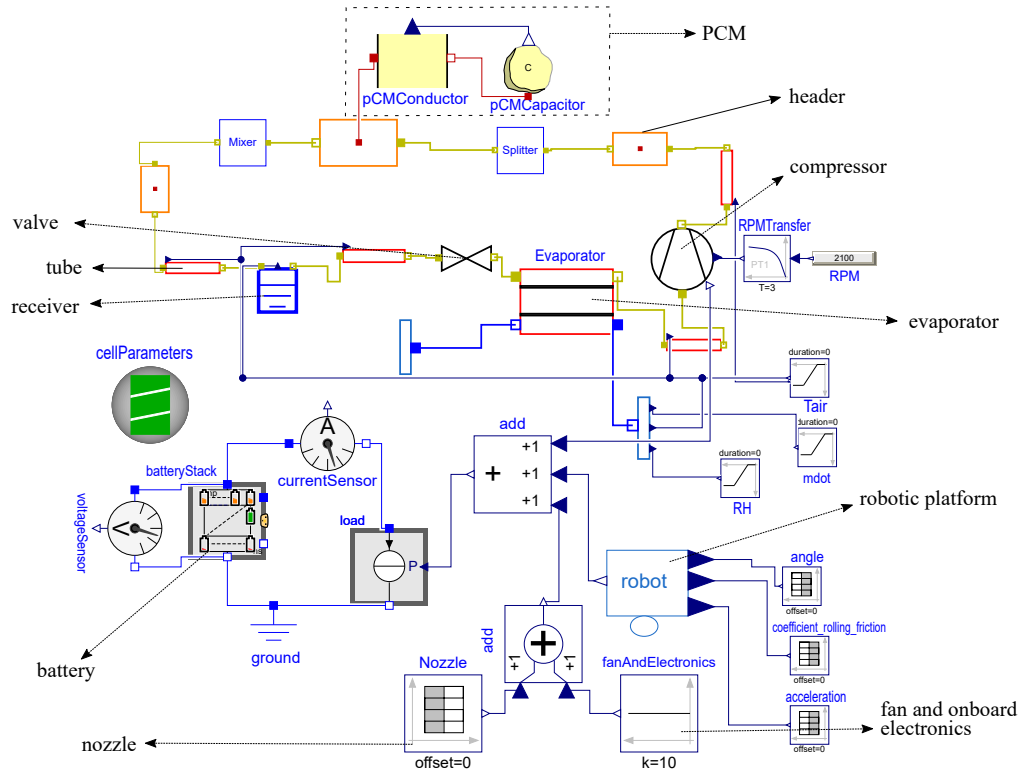


Figure 3. Schematic of System Model for RoCo.

3.2.1 PCM Capacitor

The PCM Capacitor block includes a HeatPort and models the heat storage of PCM. The equations for temperature transforming model use scaled temperature (T^* [K]) as input. It is defined as:

$$T^* = T - T_m \quad (1)$$

where T_m [K] is the mid-point of the temperature glide and T [K] is the lumped PCM temperature. The specific enthalpy (h [Jkg⁻¹]) is calculated by:

$$h = c(T^* + s) \quad (2)$$

The specific heat capacity (c [Jkg⁻¹K⁻¹]) and the source term (s [K]) are defined as:

$$c = \begin{cases} c_s, & T^* < -\delta T \\ \frac{c_s + c_l}{2} + \frac{h_{sl}}{2\delta T}, & -\delta T \leq T^* \leq \delta T \\ c_l, & T^* > \delta T \end{cases} \quad (3)$$

where, δT [K] is the temperature range over which the PCM melts.

$$s = \begin{cases} \delta T, & \text{if } T^* \leq \delta T \\ \frac{c_s}{c_l} \delta T + \frac{h_{sl}}{c_l}, & T^* > \delta T \end{cases} \quad (4)$$

The if-else loops are implemented using NoEvent operator as shown below for the specific heat capacity block.

```

if noEvent(T_star < -deltaT) then
  c = c_s;
elseif noEvent(T_star <= deltaT) then
  c = (c_s + c_l)/2 + h_sl/(2*deltaT);
else
  c = c_l;
end if;

```

The melt fraction (λ) of PCM is calculated from its enthalpy value as:

$$\lambda = \max(0, \min(1, \frac{h}{h_l})) \quad (5)$$

where h_l [Jkg⁻¹] is the enthalpy at the point where the PCM just turns liquid. The equation is simplified because of the fact that the enthalpy scale is defined as zero for the point where the PCM starts to melt. The melt fraction is made available for the PCM capacitor block through the RealOutput interface.

3.2.2 PCM Conductor

PCM Conductor block connects the refrigerant control volume of the condenser to the PCM Capacitor block. It extends Modelica.Thermal.HeatTransfer.Interfaces.Element1D block and provides for the heat flow, which is calculated using CombiTable1D fitted function for heat transfer coefficient as a function of melt fraction. The RealInput interface is used to obtain melt fraction input from PCM Capacitor.

Table 1 contains the anchor points given to the CombiTable block used as input for the normalized heat transfer coefficient

as a function of melt fraction. The constant value used to multiply the normalized function to obtain heat transfer coefficient (HTC) is $116 \text{ W m}^{-2} \text{ K}^{-1}$. These numbers are obtained by matching the condenser pressure from simulation to the experiment since there are no correlations to capture the behavior in literature. Pal and Joshi (2001) discusses the heat transfer variation in the four regimes captured by Table 1. The initial heat transfer occurs in a conduction dominated regime. Then there is a reduction in heat transfer coefficient with the appearance of small melt layer because the velocity of the liquid PCM due to buoyancy force is low. The melting then progresses to a convection dominated regime where the velocity of liquid PCM increases causing a higher rate of heat transfer. Finally, the magnitude of velocity decreases as the temperature in the molten PCM becomes more uniform with time due to natural convection stirring, leading to reduced buoyancy force for convection.

Table 1. Input table for PCM Conductor block.

Melt Fraction	Normalized HTC
0	1
0.2	0.9
0.4	1
0.7	0.9
1	0.8

3.3 Battery

Battery modeling is necessary to reduce the total weight and cost, which are critical parameters in the design of RoCo. Accurate prediction of the state of charge (SOC) of the battery is essential to determine how long the battery will last with a typical user case scenario. The battery capacity should be sufficient to run the whole cooling operation with charging and discharging operations without entering regions of overcharging and over-discharging for longer operation.

There are a variety of methods for mathematical modeling of the battery which vary in complexity, computational requirements and reliability of the prediction. The models based on electrochemical principles which model first-principle phenomena require significant computational resources and detailed datasets for input (Marco et al., 2015). Equivalent circuit models have a good trade-off between exactness, complexity and usability while still providing some insights into the battery state (Einhorn et al., 2011b). As a result, the equivalent circuit approach is used for the current research. Modeling for the battery is carried out using the Electrical Energy Storage Library (Einhorn et al., 2011a). The battery pack is modeled using the model `LinearDynamicImpedance` from the battery stack sub-package.

The battery model in the library involves modeling a single cell as an effective resistance capacitor [R-C] circuit. By taking inputs of the number of cells in series and number of cells in parallel of the battery, the behavior of the battery can be modeled by appropriate scaling of the cell model.

The state of charge of a cell (SOC) is calculated as:

$$SOC = SOC_0 - \frac{\Delta Q}{C} \quad (6)$$

where, ΔQ [C] is the charge removed and C [C] is the capacity of the cell. SOC_0 [C] is the initial state of charge of the cell.

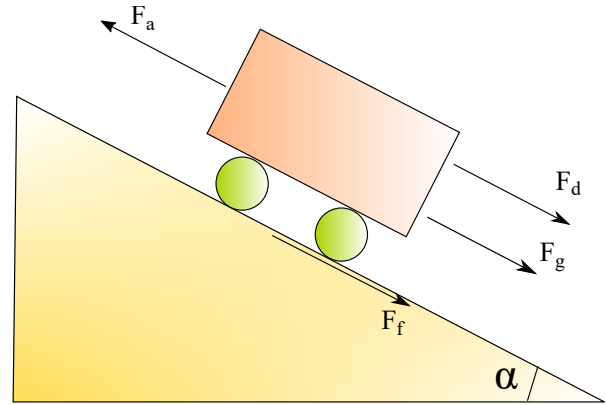


Figure 4. Free Body Diagram for Robotic platform

The model provides inputs for modeling the capacity of the cell, resistances and capacitances as variables using parameters for calendaric aging and aging due to cycling. The charge removed from the battery is calculated as shown in Equation (7).

$$\Delta Q = \int_0^t I dt \quad (7)$$

Various components are available to model different types of loads in the circuit, which determines the current I [A]. For the present study, the component `SignalPower` is used. This component takes power draw as input and creates loads on the battery accordingly.

3.4 Platform

The power consumption from the motion of Robotic platform is estimated from first-principles based approach used by (Gonullu, 2013). The free body diagram of the platform is shown in Figure 4. The total force required for motion (F_t [N]) is the sum of gravitational force acting along the incline (F_g [N]), the drag force by the air (F_d [N]), the force to overcome friction (F_f [N]) and the force required to produce acceleration (F_a [N]).

$$F_t = F_g + F_d + F_f + F_a \quad (8)$$

Since the platform operates at speeds of around 1 ms^{-1} , the drag force (F_d) can be neglected. If m [kg] is the total mass of RoCo, g [ms^{-2}] the acceleration due to gravity, the remaining

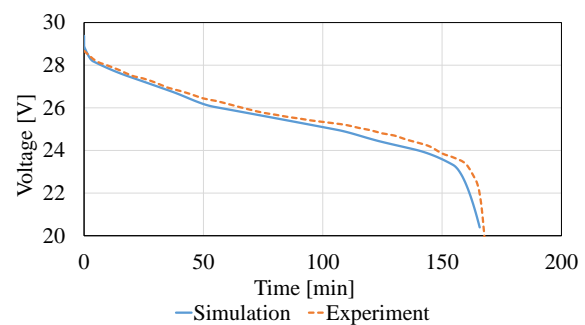


Figure 5. Comparison of simulation results to experimental for the battery discharge test

forces can be calculated as:

$$F_g = mg \sin(\alpha) \quad (9)$$

$$F_f = f_r mg \cos(\alpha) \quad (10)$$

$$F_a = ma \quad (11)$$

Finally, the net power for the motion of the platform (P [W]) is obtained as:

$$P = F_t v \quad (12)$$

where v [ms^{-1}] is the velocity of the platform.

Typical variations in the motion are changes in velocity, motion over different surfaces and on surfaces with different inclinations. A `TimeTable` block for each of acceleration (a [ms^{-2}]), rolling friction resistance of surface (f_r) and inclination angle of surface with horizontal (α) is used to capture these variations in the motion.

3.5 Fan

The fan operates continuously during the operation of RoCo and can be modeled by `Modelica.Blocks.Sources.Constant`. The power measured from the experiment is given as input. The power draw from on-board electronics is also constant and is lumped together with the fan power.

3.6 Nozzle

The stationary nozzle modeled in Figure 1 can be housed with two motors to rotate it along horizontal and vertical directions. The power consumed by the rotary nozzle is taken as the maximum power draw of two DC motors used for its motion. The transient power draw from the nozzle is given by `Modelica.Blocks.Sources.TimeTable` block. The nozzle is assumed to move continuously for a period of 10 seconds, for every 5-minute interval.

4 System Model

The screenshot for the system model is shown in Figure 3. The inputs for various components are discussed in this section. The displacement volume and RPM for the compressor are provided from manufacturer's data. The efficiency is adjusted to match the power consumption measured from experiment (Du et al., 2016). For pressure drop calculations, values are calculated using various correlations (McAdams et al., 1942; Friedel, 1979; Lockhart and Martinelli, 1949; Müller-Steinhagen and Heck, 1986) and nominal values selected based on the range calculated from them. For refrigerant heat transfer coefficient, single phase heat transfer coefficients are evaluated using Dittus and Boelter (1985) correlation. Two phase heat transfer coefficient in the evaporator is evaluated using Shah (1982) correlation. Condenser consists of helical coils inside the PCM. The single phase liquid only heat transfer coefficient is calculated using Schmidt (1967) correlation to be used as input to Shah (2016) correlation for two phase heat transfer coefficient calculation. The air side heat transfer coefficients are calculated using Wang et al. (2000).

The battery used in the prototype shown in Figure 1 consists of 21 cells, consisting of 3 parallel lines of 7 cells of Samsung ICR18650-26F in series (7s3p). Its capacity is 7.8 Ah. Obtaining the input parameters for battery requires results from extensive battery testing carried out primarily to obtain parameters for the model which were not available in open literature. To

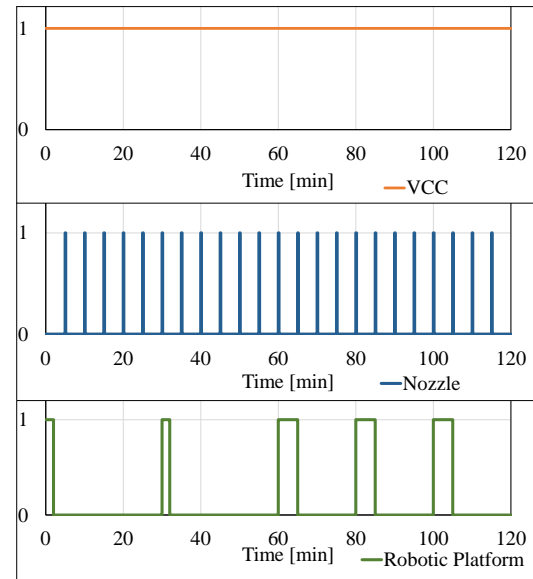


Figure 6. Typical operation of RoCo with VCC always ON

address this, the resistance and capacitance values for the model are taken from Muenzel et al. (2015) for Sanyo UR18650FM since it has similar cell capacity and initial internal impedance as Samsung ICR18650-26F. These two parameters are most important for modeling overall battery performance as can be seen from results of Einhorn et al. (2011b). For accurate prediction of battery performance, a battery discharge test is conducted using a constant power drawing circuit. The power draw of this circuit is selected to be similar to that of RoCo during a steady operation. The SOC vs OCV (Open Circuit Voltage) table is modified in the model to match the experimental discharge profile. The comparison of voltage discharge profile of simulation and experimental case is shown in Figure 5.

The weight of the system applied to the platform is taken to be 30 kg. RoCo is assumed to be moving upwards a slope with 10° incline. The coefficient of rolling friction is taken to be 0.05 which applies for poor condition stone paving to represent the worst conditions.

Two duty cycles are considered for the operation. The logic for the operation of various power draw sources in a two hour time duration is shown in Figure 6 and Figure 7. In the first one, the fan and compressor continue to operate for the whole duration (hereby referred to as Cycle 01, see Figure 6) while in the second one, they are stopped when RoCo is in motion (hereby referred to as Cycle 02, see Figure 7). This is done to avoid peaks in the power draws for battery. In the simulation, the compressor and fan are first turned off. The platform motion is started after a delay of 1 second.

5 Results and Discussion

The model is simulated using Dymola 2017 with Radau Ila - order 5 stiff solver. The tolerance selected for the solver is $1e-6$. The simulation time on a PC with 16 GB RAM, 64-bit Operating System and 3.5 GHz is 1159 seconds.

Figure 8 and Figure 9 show comparison of the simulated results with the experimental data. It can be observed that the simulation predicts the experimental trends to a reasonable extent.

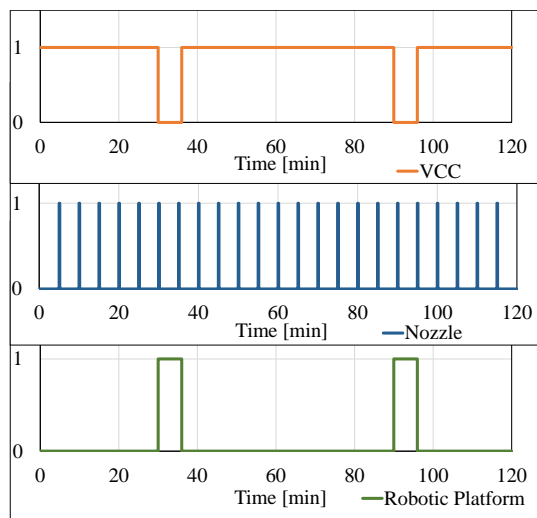


Figure 7. RoCo Operation with VCC turned off during motion

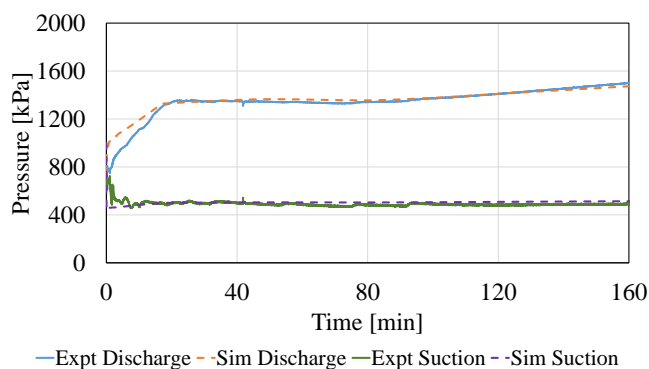


Figure 8. Pressures at suction and discharge of compressor

The experimental setup uses a variable expansion valve which is modulated by an operator. The valve model used in the simulation assumes fixed opening and constant discharge coefficient. These two effects lead to the deviations in measured pressures and mass flow rates in the initial part of the operation. Another factor for the deviation of discharge pressure in the initial 20 minutes (Figure 8) is from the inaccuracy of the heat transfer coefficient.

The dynamic modeling for cycle 02 is complicated due to cycling. So to model it, the power consumption profile of compressor is extracted from cycle 01 results. For the portion where RoCo is in motion, the values are set to zero in the profile. This load is now given as an input to the battery using a `Modelica.Blocks.Sources.TimeTable` block.

The results of power consumption can be seen in Figure 10. It can be observed that in cases where all the components operate power exceeds 100 W. However, by turning off the fan and compressor when RoCo is in motion, it is possible to limit the power draw to 70 W.

For improved results, a few parameters from the simulation are calibrated for a better match with the experimental data (Du et al., 2016). A fixed opening valve is used in the model. Its

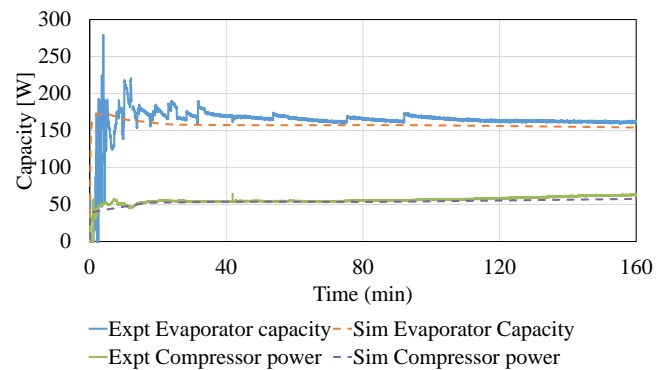


Figure 9. Cooling capacity and power consumption of RoCo

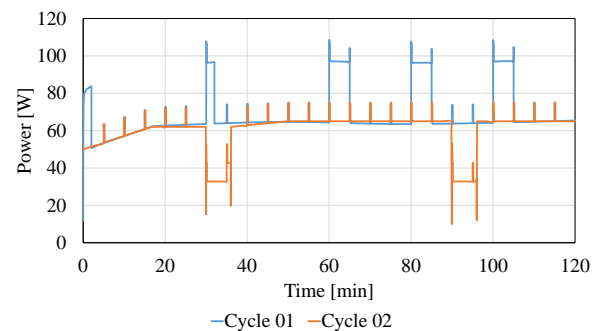


Figure 10. Power draw during the two operating cycles

opening and flow coefficient are adjusted using the experimental refrigerant mass flow rate. The suction and discharge pressure from simulation also need to be noted during the txv parameter calibration. Since the RPM and displacement volume of compressor are available from manufacturer's data, the valve as the only component to significantly affect the refrigerant mass flow rate, permitting the aforementioned adjustment. Calibration is also needed for the battery, compressor power and PCM, which is already discussed in the respective sections.

6 Conclusions

A first principle based multi-physics model is developed to model the behavior of a portable air conditioning device. The model is used to capture power consumption of new version of the device for two different operating cycles. Based on the results of power consumption, it is observed that turning off the VCC during the motion can reduce peak load on the battery by up to 34%.

7 Acknowledgment

This research was supported by the Advanced Research Projects Agency - Energy (ARPA-E) with Award Number DE-AR0000530. We thank the members of Center for Environmental Energy Engineering (CEEE) and team members of the Roving Comforter Project for their support.

References

Y. Cao and A. Faghri. A numerical analysis of phase-change problems including natural convection. *Journal of heat transfer*, 112(3):812–816, 1990. doi:10.1115/1.2910466.

- R. Dhumane, Y. Du, A. Mallow, K. Gluesenkemp, J. Ling, V. Aute, and R. Radermacher. Transient Modeling of a Thermosiphon based Air Conditioner with Compact Thermal Storage: Modeling and Validation. In *16th International Refrigeration and Air Conditioning Conference*, Purdue, Indiana, USA, 2016.
- F. W. Dittus and L. M. K. Boelter. Heat transfer in automobile radiators of the tubular type. *International Communications in Heat and Mass Transfer*, 12(1):3–22, 1985. doi:10.1016/0735-1933(85)90003-X.
- Y. Du, J. Muehlbauer, J. Ling, V. Aute, Y. Hwang, and R. Radermacher. Rechargeable Personal Air Conditioning Device. In *ASME 2016 10th International Conference on Energy Sustainability collocated with the ASME 2016 Power Conference and the ASME 2016 14th International Conference on Fuel Cell Science, Engineering and Technology*. American Society of Mechanical Engineers, 2016. doi:10.1115/ES2016-59253.
- M. Einhorn, F. V. Conte, C. Kral, C. Niklas, H. Popp, and J. Fleig. A modelica library for simulation of electric energy storages. In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical University; Dresden; Germany*, number 63, pages 436–445. Linköping University Electronic Press, 2011a. ISBN 1650-3740. doi:10.3384/ecp11063436.
- M. Einhorn, V. Conte, C. Kral, and J. Fleig. Comparison of electrical battery models using a numerically optimized parameterization method. In *2011 IEEE Vehicle Power and Propulsion Conference*, pages 1–7. IEEE, 2011b. ISBN 1612842488. doi:10.1109/VPPC.2011.6043060.
- L. Friedel. Improved friction pressure drop correlations for horizontal and vertical two-phase pipe flow. In *European two-phase flow group meeting, Paper E*, volume 2, page 1979, 1979.
- M. K. Gonullu. Development of a mobile robot to be used in mobile robot research. Master's thesis, Department of Mechanical Engineering, Middle East Technical University, 2013.
- T. Hoyt, E. Arens, and H. Zhang. Extending air temperature set-points: Simulated energy savings and design considerations for new and retrofit buildings. *Building and Environment*, 88: 89–96, 2015. doi:10.1016/j.buildenv.2014.09.010.
- I. Intergovernmental Panel on Climate Change Fourth Assessment. Contribution of Working Groups I, II and III to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change, 2007. ISSN 14764687. URL http://www.ipcc.ch/publications{_}and{_}data/ar4/syr/en/spms2.html{#}footnote5.
- R. W. Lockhart and R. C. Martinelli. Proposed correlation of data for isothermal two-phase, two-component flow in pipes. *Chem. Eng. Prog.*, 45(1):39–48, 1949.
- J. Marco, N. Kumari, W. D. Widanage, and P. Jones. A cell-in-the-loop approach to systems modelling and simulation of energy storage systems. *Energies*, 8(8):8244–8262, 2015. doi:10.1049/cp.2011.0421.
- W. H. McAdams, W. K. Woods, and L. C. Heroman. Vaporization inside horizontal tubes-II-benzene-oil mixtures. *Trans. ASME*, 64(3):193–200, 1942.
- V. Muenzel, A. F. Hollenkamp, A. I. Bhatt, J. de Hoog, M. Brazil, D. A. Thomas, and I. Mareels. A Comparative Testing Study of Commercial 18650-Format Lithium-Ion Battery Cells. *Journal of The Electrochemical Society*, 162(8):A1592–A1600, 2015. ISSN 0013-4651. doi:10.1149/2.0721508jes.
- H. Müller-Steinhagen and K. Heck. A simple friction pressure drop correlation for two-phase flow in pipes. *Chemical Engineering and Processing: Process Intensification*, 20(6):297–308, 1986. ISSN 0255-2701. doi:10.1016/0255-2701(86)80008-3.
- D. Pal and Y. K. Joshi. Melting in a side heated tall enclosure by a uniformly dissipating heat source. *International Journal of Heat and Mass Transfer*, 44(2):375–387, 2001. ISSN 0017-9310. doi:10.1016/S0017-9310(00)00116-2.
- H. Qiao, V. Aute, and R. Radermacher. Transient modeling of a flash tank vapor injection heat pump system—part I: model development. *International journal of refrigeration*, 49:169–182, 2015. doi:10.1016/j.ijrefrig.2014.06.019.
- E. F. Schmidt. Wärmeübergang und Druckverlust in rohrschlangen. *Chemie Ingenieur Technik*, 39(13):781–789, 1967. doi:10.1002/cite.330391302.
- M. M. Shah. Chart correlation for saturated boiling heat transfer: equations and further study. *ASHRAE Trans.:(United States)*, 88(CONF-820112-), 1982.
- M. M. Shah. Comprehensive correlations for heat transfer during condensation in conventional and mini/micro channels in all orientations. *International journal of refrigeration*, 67: 22–41, 2016. doi:10.1016/j.ijrefrig.2016.03.014.
- E. M. Sparrow, S. V. Patankar, and S. Ramadhyani. Analysis of melting in the presence of natural convection in the melt region. *Journal of Heat Transfer*, 99(4):520–526, 1977. ISSN 0022-1481. doi:10.1115/1.3450736.
- United States Department of Energy. *Energy Efficiency and Renewable Energy*. 2011.
- M. Veselý and W. Zeiler. Personalized conditioning and its impact on thermal comfort and energy performance – A review. *Renewable and Sustainable Energy Reviews*, 34:401–408, 2014. doi:10.1016/j.rser.2014.03.024.
- C.-C. Wang, K.-Y. Chi, and C.-J. Chang. Heat transfer and friction characteristics of plain fin-and-tube heat exchangers, part II: Correlation. *International Journal of heat and mass transfer*, 43(15):2693–2700, 2000. doi:10.1016/S0017-9310(99)00333-6.
- S. Wang, A. Faghri, and T. L. Bergman. A comprehensive numerical model for melting with natural convection. *International Journal of heat and mass transfer*, 53(9-10):1986–2000, 2010. doi:10.1016/j.jheatmasstransfer.2009.12.057.

- H. Zhang, E. Arens, and W. Pasut. Air temperature thresholds for indoor comfort and perceived air quality. *Building Research & Information*, 39(2):134–144, 2011. doi:10.1080/09613218.2011.552703.
- H. Zhang, E. Arens, and Y. Zhai. A review of the corrective power of personal comfort systems in non-neutral ambient environments. *Building and Environment*, 91:15–41, 2015. doi:10.1016/j.buildenv.2015.03.013.

A Platform for the Agent-based Control of HVAC Systems

Roozbeh Sangi Felix Bünning Johannes Fütterer Dirk Müller

Institute for Energy Efficient Buildings and Indoor Climate, E.ON Energy Research Center, RWTH Aachen University, Germany, rsangi@eonerc.rwth-aachen.de

Abstract

Attempts to develop efficient and environmentally friendly building energy systems have led to modern complex energy concepts for buildings, which have consequently initiated a need for new control strategies for them. Multi-agent control, which is known with other names like agent-based control, offers a promising solution to this challenge. To the knowledge of the authors, there are 96 platforms for multi-agent systems in different programming languages available, which are mostly java-based and mainly used in logistic applications, but there is no platform in the modeling language Modelica, which is widely used for simulation of dynamic systems, especially buildings performance simulation. This lack motivated the authors to develop a platform for agent-based control of HVAC systems. The platform eliminates the dependency of models developed in Modelica on an extra interface, which is usually required to couple the models to the platforms written in any programming languages other than Modelica. This paper presents the structure of the platform and explains how the agents' communications work. The flexibility of the optimization objective is ensured through the definition of readily interchangeable cost functions. The applicability and functionality of the platform are proved by applying the platform in the control of building energy systems examples.

Keywords: *Agent-based control, Building energy systems, Control, HVAC, Modelica, Multi-Agent System*

1 Introduction

The amount of energy used for heating and cooling in the building sector is about one third of the total energy consumed in the world. The finiteness of natural energy resources on the one hand, and the ever-increasing demand for energy in the world on the other hand, necessitate the development of systematic approaches for improving the efficiency of building energy systems as well as minimizing the usage of primary energy resources and the damaging impacts and harmful effects on the environment (Sangi et al., 2014). Taking into account that renewable energy sources for the building sector such as photovoltaics, heat pumps and combined-heat-and-power units are becoming profitable, such components are installed in private and commercial buildings with increasing quantity. Often more than one of such components are operated in parallel to increase cost effectiveness and the security of

supply.

Consequently, the complexity of building energy systems has severely increased in the recent past. Together with the need for controlling concepts that can handle such complexity has arisen. Besides concepts like Model-Predictive control and Artificial Neural Networks (Afram and Janabi-Sharifi, 2014), the concept of agent-based control realized through Multi-Agent Systems (MAS) promises good results in the area of HVAC control (Huber et al., 2015).

Multi-Agent Systems were successfully applied in the areas of logistics and telecommunication in the past (Verein Deutscher Ingenieure, 2010). Inherently this concept is suited to solve complex control problems and is therefore predestined for the control of complex energy systems. For the development of such systems, tools for multiple programming languages and programming environments are available (Allan, 2010), but not for the object-oriented language Modelica.

Modelica is a modelling language commonly used for the dynamic simulation of thermo-hydraulic systems. It is receiving growing attention in the use of modeling and simulation of building energy systems, as recent studies indicate: In (Wetter et al., 2014) a Modelica library for the simulation of building energy systems is introduced. (Ali et al., 2013), (Perera et al., 2016), (Sangi et al., 2016) and (Fuchs et al., 2016) use Modelica in order to model, simulate and investigate in building energy systems as well as district heating systems. MAS will play an important role in the control of future building energy systems (see 2.2). Consequently, a Modelica library for MAS will be needed.

In the course of this work a library for the agent-based control of building energy systems in the modeling language Modelica is developed, implemented and finally validated in a case study. The library allows "plug-and-play" implementation of MAS into any model of a building energy system in the Modelica environment, thus allowing the investigation in agent-based building energy system control through dynamic simulation. It depicts a solution that through UDP/IP-communication can be run on distributed machines, enabling the user to integrate both software and hardware into the optimization problem. Furthermore, the cost functions can be changed without interfering with the agent system leading to a flexible solution in which the individual user can optimize their energy system for an individual optimization goal with only

minor engineering effort.

In the following an overview on multi-agent system is presented and the system structure is described. The developed library for agent-based control is also introduced following by an example that demonstrates the application of the library in building energy systems.

2 Overview on Multi-Agent Systems

2.1 The Concept of Agents and Multi-Agent Systems

Agents The concept of agent-based control is a concept which allows to control complex systems by splitting the main objective of the system into smaller objectives which so-called agents try to obtain by interacting with each other. Although the concept is widely spread in the scientific area, especially in the field of computer science and information technologies, there is no unified definition of the term agent.

After the term first appeared in the context of a dissertation in 1985, in which the term agent is connected with the attributes of autonomy and problem-solving behaviour (Rosenschein, 1985), further attributes such as proactivity and the ability to work towards higher goals (Wernstedt, 2005), the ability to perceive the changes of their surroundings and to react on them (Divenyi, 2013), the ability of rational calculation and organization of actions to achieve higher aims as well as permanent activeness (Kirn, 2002), socialness and truthfulness (Bellifemine et al., 2007) were defined by various authors.

In VDI 2653 agents are defined as encapsulated entities, hardware or software, with specified objectives. An agent attempts to achieve these objectives through its autonomous behaviour, in interacting with other agents and their surrounding. In addition, several characteristics such as autonomy, scope of action, interaction, encapsulation, persistence, goal-orientation and reactivity are defined.

Multi-Agent Systems VDI 2653 describes Multi-Agent Systems (MAS) as a set of agents interacting to fulfil one or more tasks. Bellifemine describes MAS as entities that can model complex systems and introduces the possibility of agents having common or conflicting goals. These agents are able to interact with each other both indirectly, by acting on the environment, or directly via communication and negotiation. Depending on their task they may cooperate to reach a common goal or compete to achieve their own aims. (Bellifemine et al., 2007)

An MAS can be used to control complex systems. One advantage over a holistic control concept is the possibility of splitting the often very complex control problem into sub-problems and -tasks and dividing them between the agents. This approach is beneficial for the developer as the analysis of those sub-problems is more accessible than the analysis of the holistic problem and thus also the implementation of the systems solving these problems. Furthermore, an agent-based approach has the advantage of being

more easily adjustable during the runtime of the system as new agents can be implemented and added to the system.

2.2 Use of Multi-Agent Systems in energy-system control

MAS have received growing recognition in various fields over the past few years. Beginning in the fields of computer science, such as Human Computer Interaction, where agents help the user depending on their already existing experience with the software, or Information Retrieval, where agents search the Internet for specific information for their user, now agent-based systems have also reached the field of logistics and telecommunication (Verein Deutscher Ingenieure, 2010). As a consequence of growing complexity in the various fields of science, MAS also receive growing attention in the fields of chemistry, biology, physics, sociology and economics (Kirn, 2002). In recent years the field of energy generation and distribution has become much more complex due to the increase of renewable energies and the concept of smart- and micro-grids. MAS depict a promising technology to control the described energy systems.

Regarding the use of MAS to control classical smart- and micro-grids, i.e. systems which generate and distribute electricity, a lot of research has been conducted (for example (Jiang, 2006), (Kok et al., 2012), (Kuznetsova et al., 2014), (Ye et al., 2015), (Karavas et al., 2015), (Khan et al., 2016), (Radhakrishnan and Srinivasan, 2016), (Rahman et al., 2016), (Xydas et al., 2016), (Ansari et al., 2016)). However, also the use of MAS for complex energy systems for the generation and distribution of heat or cold, such as building energy systems, HVAC systems and district heating grids, has recently gained growing attention.

In (Huberman and Clearwater, 1995) a market-based MAS is used to distribute warm and cold air in an office building. The system uses a double-blind auction procedure in which agents bid to buy and sell warm and cold air. The auction is managed by a central auctioneer. Experiments with a real office building show an even temperature distribution in the building without leading to excessive actuator movement.

(Qiao et al., 2006) introduces an MAS which combines the control of a building energy system with user interaction. The system is built of personal agents, local agents and central agents. Personal agents act as teachable assistants which carry personal user information, such as preferred room temperature, humidity and the current location of the user. Local agents act as mediators, policy enforcers and information providers. The tasks of the central agent are decision aggregation and interfacing services.

A similar system based on personal agents, local agents and central agents is used in (Yang and Wang, 2013). Personal agents are developed to predict user preferences by learning their behaviours. Local agents act as mediators, information providers, decision makers and control executors while the central agent facilitates collaboration be-

tween the local agents while regarding the overall system goal. The functionality of the system regarding effective control of the building energy system while satisfying occupants' demands is proven with simulations and case studies.

In (Wang et al., 2011) a system using only central and local agents is used. The central agent contains the main intelligence of the system. It calculates set point of temperature, illumination and humidity based on user preferences and outdoor information using particle swarm optimization. The local agents use fuzzy controllers to control the actuators in order to reach said set points.

In (Wang et al., 2012) an indoor energy and comfort management system based on information fusion and a multi-agent control system is proposed. A multi-agent building control system with particle swarm optimization is built to achieve the smart building control goal, which is to maximize the comfort index using minimum power consumption.

(Davidsson and Boman, 2000) uses an MAS consisting of personal comfort agents, room agents, environmental parameter agents and badge system agents to control temperature and illumination in a building. Following simulation results, the system is able to reduce energy use while maintaining user satisfaction.

In (Mokhtar et al., 2013) an already existing MAS for building heat distribution control is updated with an ARTMAP, a type of artificial neural network with learning capabilities. Simulation results show the proposed intelligent MAS is able to maximize the use of a ground source heat pump effectively by profiling, predicting and coordinating its usage with other energy resources.

(Mokhtar et al., 2014) uses a similar MAS based on ARTMAP to control a building energy system based on learned user preferences. Simulation results show that the system provides better energy control and thermal comfort management than a reference rule-based MAS.

In (Hurtado et al., 2015) an agent-based approach to optimize the interaction of smart-grids and building energy systems is developed. Particle swarm optimization is used to maximize comfort and energy efficiency. It is shown that the operation of the building energy system with the MAS allows the support of the voltage control in the smart-grid.

(van Pruissen et al., 2014) presents a solution based on electronic market principles called HeatMatcher. HeatMatcher is a P2P system based on PowerMatcher (Kok et al., 2012), which is a general purpose coordination mechanism for balancing supply and demand in electricity networks. The system features trading of heat on two different time scales depending on the inertia of the components involved in trading. The MAS is tested with a floor heating system connected to a heat pump and a boiler and shows more energy efficient operation than a reference centralized controller.

In (Huber et al., 2015) an MAS based on consumer agents and supply agents is introduced. Consumer agents

recognize heating or cooling demands and request them from supply agents. Supply agents estimate the corresponding costs. The control of the energy system is governed by negotiation between those agents. The system is tested with a hardware-in-the-loop test bench consisting of a central air handling unit and four rooms. Results show basic functionality of the system.

The developed MAS within the course of this research offers a flexible approach to the control of building-energy systems. The MAS can, without any adaptation necessary, be applied to any building energy system in the simulation environment of Modelica. With only minor effort it can also be used to control any energy systems, e.g. electrical grids or district heating networks. Furthermore, the MAS can also be used for real life applications beyond the scope of simulation thanks to agent communication via UDP/IP network protocol with only minor adaptation to the real life building energy system necessary. Moreover, the separation of agent logic and cost functions allows changing of the optimization goal for each individual user without interfering with the agent system and empowers other developers to design own cost functions for their individual optimization goal. To the best of the authors knowledge, no other MAS offers this functionality.

3 System architecture

In the following sections the roles of different types of agents, the system architecture, and the communication architecture of the Modelica library will be discussed.

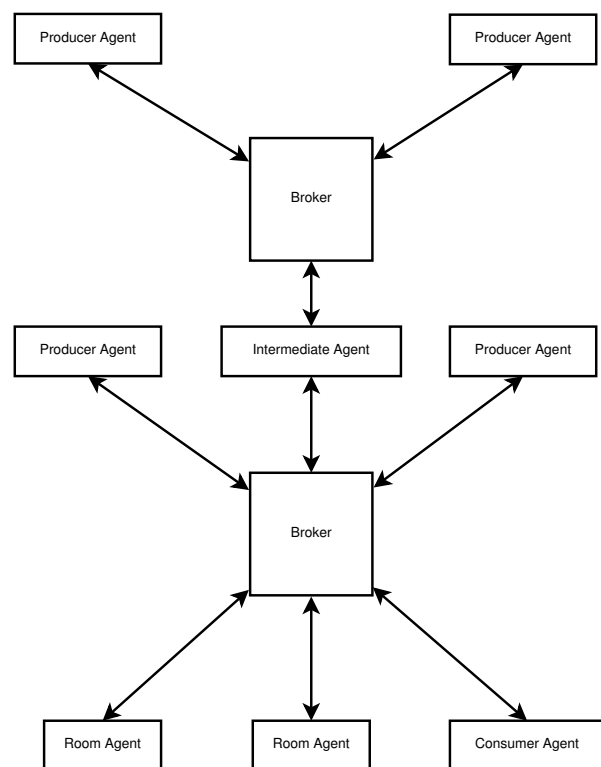


Figure 1. Structure of the Multi-Agent System

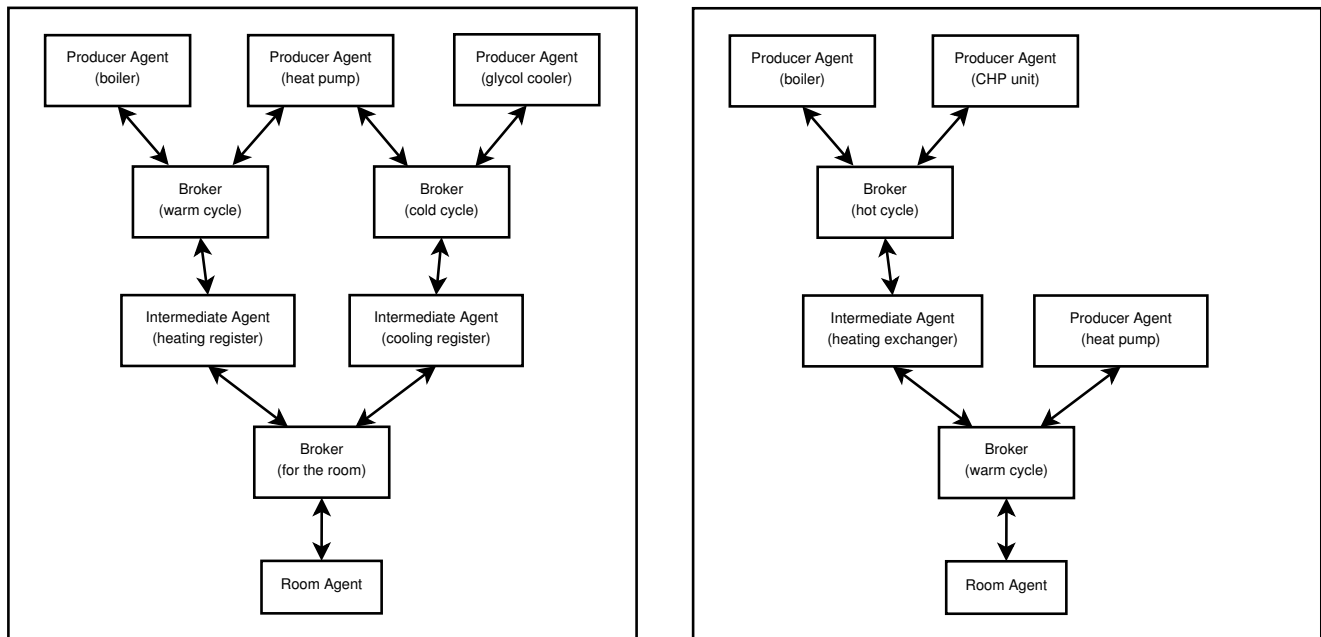


Figure 2. Cascading of the agent system with the help of the intermediate agent

3.1 Fundamentals

The agents are designed as state machines, which is a type of event discrete system, meaning that one agent can take on one state and change to a different state when a transition condition becomes true. For the implementation of the state machines, the Modelica StateGraph library (Otter et al., 2005) was used.

The contentual agent communication is based on the FIPA ACL Message Structure Specification (FIPA, 2002a) and FIPA Communicative Act Library Specification (FIPA, 2002b), which depict a common standard for agent communication. Physical agent communication is established via UDP/IP standard under the use of the Modelica DeviceDrivers library (Thiele and Bellmann, 2015). The UDP/IP standard allows communication outside of the simulation environment of Modelica, giving the opportunity to operate agents on different machines and potentially hardware-in-the-loop simulation. As UDP does not guarantee message delivery, measurements to guarantee delivery are implemented on the application level.

3.2 MAS structure

Figure 1 shows the structure of the MAS. The system is a partially centralized market-based approach in which capacity adjustments of heat and cold are traded. There are four different types of agents in the system: room/consumer agent, producer agent, intermediate agent and broker.

Room/consumer agents represent entities in the system which require heat or cold for their proper functioning within a building energy system, for example rooms, storage tanks or lab equipment which requires process heat. Their task is to estimate a need for heat or cold and to make a corresponding request to the broker. Room agents

use PID controllers to calculate a capacity request from a deviation between the current room temperature and the set point. The set point and an allowed deviation from the set point can be set for each individual room agent in the system, allowing the occupants of the room to adjust the system according to their needs. Consumer agents can use a variety of strategies to determine the capacity request.

Producer agents represent entities in the system which supply heat or cold, for example boilers, CHP units, heat pumps or chillers. Their task is to sell heat or cold to the broker and to adjust capacity when a deal was successfully made. In order to make an offer, they use a cost-function which matches a capacity adjustment with a corresponding price. The cost function is exchangeable, rendering the optimization of the system performance towards different optimization goals possible.

Intermediate agents are a hybrid of producer agents and consumer agents. They act as a consumer in one market and as a producer in another market. They can represent heat exchangers in building energy systems. With the help of the Intermediate agent, cascaded energy systems can be controlled with the MAS system.

The broker is a purely virtual agent and is not connected to any physical entity of the building energy system. It facilitates the trade of heat and cold by collecting requests from room/consumer agents and asking for offers from Producer agents. After collecting all offers, it chooses the most cost effective supplier and requests a capacity adjustment.

3.3 Trading procedure

In the following, the working principle of the whole controlling process will be explained. The room agent notices a temperature which is out of a certain range around the set

temperature and decides to take action. It sends a heating or cooling request to the broker asking for a certain surplus of heat or cold during a specified period of time. The broker waits for requests of other rooms for a certain time to bundle individual requests to one big request. When a specific waiting time for more requests has expired, the broker calls for proposals from each of the producer agents. The producer agents check if they can supply the requested amount of heat or cold and calculate a corresponding price with the help of a cost function. Afterwards, all producer agents send either an offer or a refusal to the broker. Based on these proposals the broker chooses the best offers and calculates a price for each request from the room agents. This information is sent to the room agents which confirm their request. Based on these confirmed requests, the broker then decides which offer is best-suited for the corresponding request and sends out "accept offer" or "reject offer" messages to the producer agents. Producer agents that received "accept offer" messages adjust their capacity accordingly. The communication procedure between the broker and the producer agents follows the FIPA Contract Net Interaction Protocol specification (FIPA, 2002c).

The intermediate agent is important for the flexible use of the agent system. It allows the cascading of the agents and the interconnection of different heating circuits. In the case of Figure 1, the Intermediate agent communicates as a producer to the lower broker and as a consumer to the upper broker. Thereby the two producers in the upper circle can be addressed although they are not part of the same heating circuit. Two examples of the use of the intermediate agent are shown in Figure 2. On the left-hand side of the figure an HVAC system is described in which each room has a heating and a cooling register. The broker mediates between the room agent and the two registers. As the registers need to be supplied with cold or heat themselves, they are each represented by an intermediate agent, that ensures their supply by buying heat or cold from superior markets. On the right-hand side an HVAC system is described in which each room has only one heating device and no cooling device (common European residential building). There are, however, two temperature circuits with different temperature levels. These circuits surround a heat exchanger represented by an intermediate agent. In case of capacity shortage in the warm temperature circuit, the hot temperature circuit can act as a heat producer by transferring heat via the heat exchanger.

3.4 Cost functions

The cost functions are used by the producer agents to calculate a corresponding price for a requested capacity adjustment. In the Modelica Library, producer agents and cost functions are separate components, which means that the cost functions are easily exchangeable, depending on the optimization goal of the MAS. In the library cost functions depending on fuel cost, exergy loss and primary exergy loss are available. For demonstration purposes fuel cost functions are used in the following as these depict

the simplest form of available cost functions. The functionality of the exergy cost functions has been proven in (Bünning, 2015).

In the case of fuel cost functions, the cost per hour of operation of a heat/cold producing entity is calculated as following:

$$\frac{C(cap)}{h} = \frac{p}{\eta} * cap \quad (1)$$

In which cap represents the capacity, $\frac{C(cap)}{h}$ denotes the cost per hour as a function of the capacity, p stands for the price of the fuel per kWh and η for a representative efficiency factor of the device. The costs for a capacity adjustment follows:

$$\begin{aligned} \frac{\Delta C}{h} &= \frac{C(cap_{new})}{h} - \frac{C(cap_{current})}{h} \\ &= \frac{p}{\eta} * (cap_{new} - cap_{current}) \end{aligned} \quad (2)$$

With the help of the calculated $\frac{\Delta C}{h}$, producer agents determine the costs of a capacity adjustment and make an offer to the broker.

4 Modelica HVAC Agent-based Control Library

In the previous sections the development of communication and logic concepts, agent behaviour and cost functions have been illustrated. These aspects are combined and implemented in the Modelica programming language resulting in the Modelica HVAC Agent Library.

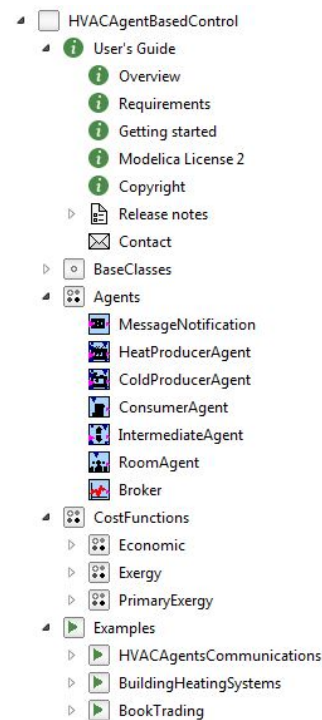


Figure 3. Structure of the HVAC Agent-based Control Library

Figure 3 shows the structure of the library. It features six different agents for the creation of MAS to control building energy systems. For the producer agents, cost functions with different calculation methods based on fuel costs, exergy and primary exergy are available. The fuel cost functions are applied within this study. The other cost functions have been defined in order to be used in an ongoing research project which aims at developing an exergy-based control strategy for building energy systems. Furthermore, the library offers an example agent system with two room agents, one broker and two producer agents, which will be discussed in the following section. Each model of the library is documented regarding its use and function and has its own icon.

Besides the HVAC related agents, agents implementing a book trading example in reference to (Caire, 2009) are included as well as examples to demonstrate agent communication between two different machines.

5 Case study

A scheme of the system under investigation is shown in Figure 4. The system consists of two rooms, each equipped with a convective radiator. The rooms are connected to weather data from the 2012 Test Reference Year. Furthermore, the system features two heat supplies, a gas boiler and an electric heating rod. The heating rod can be run on electricity from the grid or on electricity provided by an additional PV panel. The gas boiler has a maximum capacity of 3 kW and the heating rod of 2 kW.

Both rooms are equipped with a room agent. Both heat sources are equipped with producer agents. To complete the MAS, a broker is used. As cost functions, the fuel cost functions are used with parameters shown in Table 1. Additionally, the cost function of the heating rod is connected to a sensor which measures solar radiation. When solar radiation reaches a value of $310 \frac{W}{m^2}$, it is assumed that sufficient electricity is provided by the PV panel and therefore considered free of charge. Consequently the fuel price for the heating rod becomes zero.

Table 1. Cost function parameters

	$p \left[\frac{\text{Euro}}{\text{kWh}} \right]$	$\eta [-]$
boiler	0.08	0.80
heating rod (grid)	0.35	1.00
heating rod (PV)	0.00	1.00

Besides the control via the agent system, each thermal zone is equipped with a PID controller and a valve, which allow the control of the room temperature to a limited degree.

All physical components of the system are taken from the AixLib library (RWTH-EBC, 2015). The simulation is executed on Dymola 2016 (Dassault Systemes, 2016).

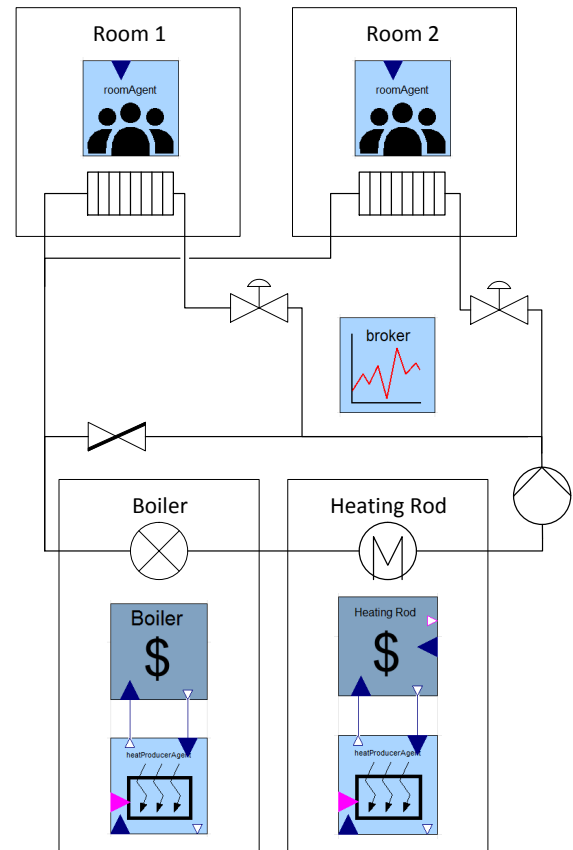


Figure 4. Example heating system with agents

6 Results and discussion

The system was simulated for 28 days with the weather data of the month of February of the 2012 Test Reference Year. Figure 5 shows the behaviour of the outside air temperature during the simulated time period. It can be seen that the temperature ranges from $+15^{\circ}\text{C}$ to -8°C . The data therefore offers situations of both high and low heating requirement for the simulation.

Figure 6 shows the trend of the air temperature in both rooms. The set point for the room temperature is 20°C . It can be seen that the temperatures are kept at a satisfactory level between 19.5°C and 20.5°C during the vast majority of time. Between these temperatures, the control of the room temperature is governed by the PID controllers and valves.

As soon as the room temperature reaches 19.5°C or 20.5°C , the system leaves the control range of the PID controllers and the room agents become active. In case of 19.5°C further heat is requested by the agents, in case of 20.5°C a reduction of heat supply is requested. It can be ascertained that the reaction time of the agent system is sufficient as the system rarely crosses the threshold temperatures.

Figure 8 shows the capacities of the boiler and the heating rod, which are solely controlled by the MAS. It can

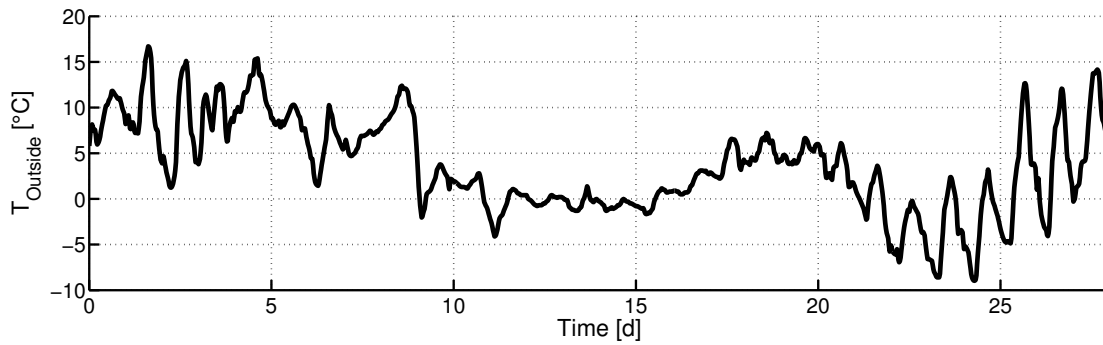


Figure 5. Outside air temperatures

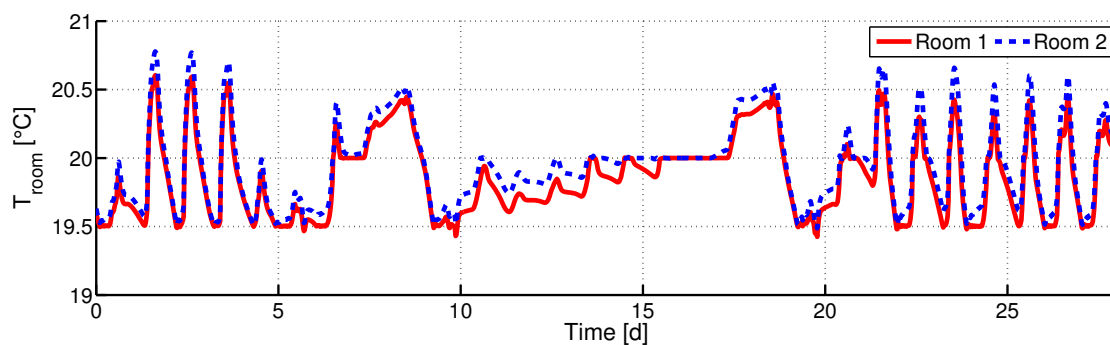


Figure 6. Room temperatures

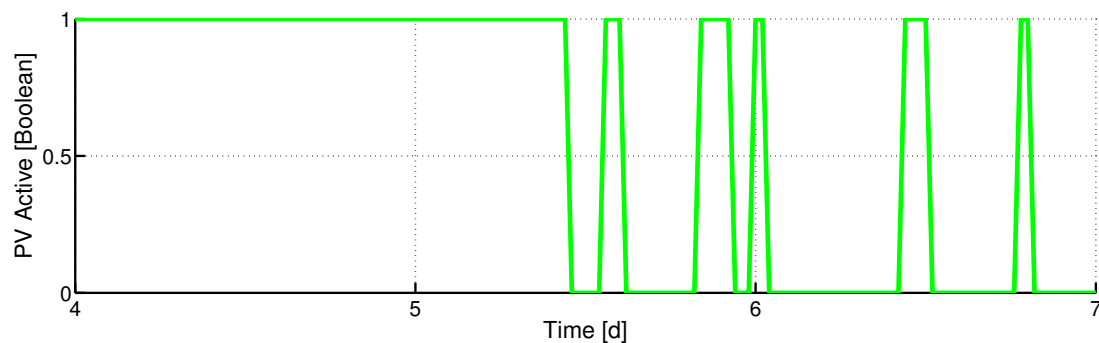


Figure 7. Activeness of PV panel

be seen that the capacities vary based on the current heat demand of the system. The figure also shows that the heat supplies are used less during the first third of the month, when the outside air temperature is generally higher. It can further be seen that the boiler holds the greater share of both heat supplies as it is mostly more cost effective.

Figure 9 shows a detailed segment of the heat supply capacities between day 4 and 7. Figure 7 shows the corresponding activeness of the PV panel which determines whether the electricity for the heating rod is considered free of charge. A comparison of the figures shows the effect of the different electricity prices on the MAS behaviour. When the heat supplies first become active between day 4 and 5, it can be seen that the heating rod is only used once the PV panel is active. In the middle of day 5 the heating rod is switched off as the electricity is not free of charge any more. Shortly afterwards, the heat

supply is increased by the boiler as the PV is no longer active. When the boiler reaches its maximum capacity in the first third of day six, the heating rod is switched on, although the PV is inactive, as the boiler is not able to supply more heat. It can further be seen that the other active PV times, apart from the long period on days 4 and 5, are not used to the fullest extent as no heat requests are made during these times. Agents used to specifically survey PV activeness could be introduced here.

The functionality of the system was validated in the simulation. However, the MAS can also be used to control real life applications because the agents communicate to each other beyond the border of the simulation software framework as they use a UDP/IP protocol to communicate via an Ethernet network. An example for this functionality is provided in the library. It shows the user how to run an MAS on distributed machines. Such a distributed

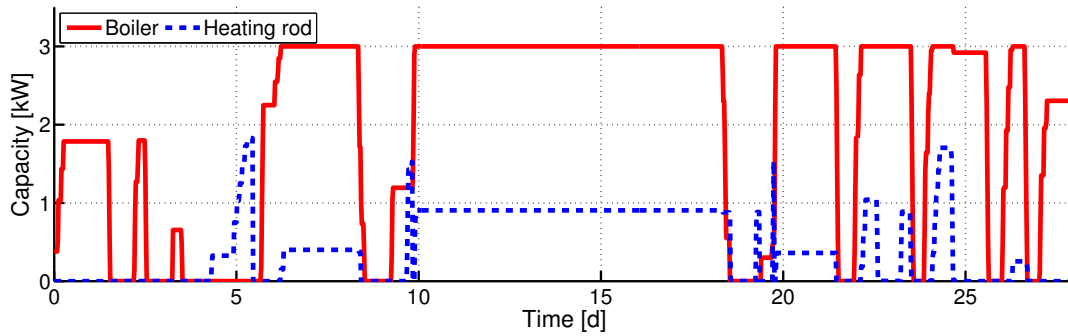


Figure 8. Heat supply capacities

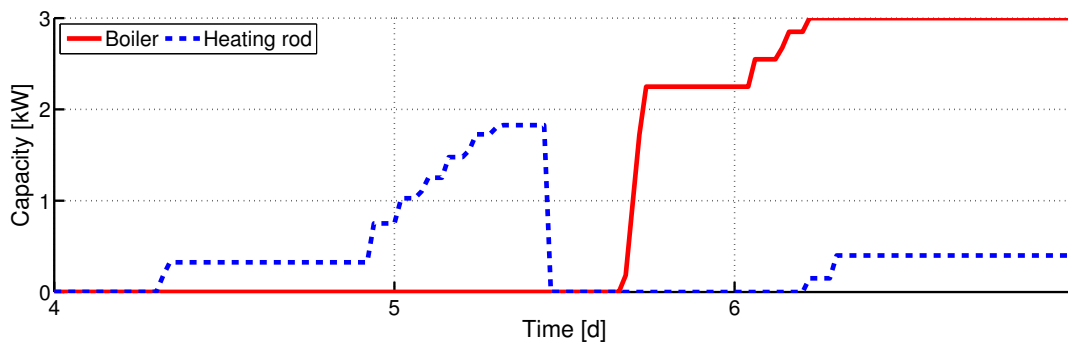


Figure 9. Detailed heat supply capacities

MAS offers much higher reliability compared to a central control as the system is still functional in case an individual producer or consumer agent fails. This means that each entity in the energy system (e.g. room, boiler, etc.) is connected to one computer on which the corresponding agent is running. Communication between agent and device needs to be developed individually as each building energy system device has a different software interface. If another entity is added to the system, it can be integrated into the building energy system control with minimal effort by setting it up with its own agent and introducing the agent to the broker. The concept is therefore highly flexible and convenient.

The results show that the MAS is able to maintain a system variable within margin while reducing the effort in accordance to an optimization goal (in this case fuel price). Although the system is developed and tested for building energy simulation, it can be used and extended to optimize any system in which forms of energy are distributed between components, such as chemical processes, district heating systems and electricity smart grids. The interchangeability of the cost function supports this diverse application as any developer can set up own cost functions for their needed domain.

7 Conclusion

A Modelica library for the agent based control of building energy systems was introduced. The structure of the Multi-Agent System was explained and the roles of different types of agents were discussed. Furthermore,

the trading procedures of capacity adjustments and an example of a type of cost function were introduced. The MAS was tested with an example of a building energy system in the simulation environment of Dymola. The results have shown that the system is capable of keeping a satisfactory room temperature while selecting the heat supply with the most cost effective way of generating heat. The library provides a flexible MAS that can be applied to multiple domains in the energy field due to exchangeable cost functions and that requires minimal implementation effort. Moreover, UDP/IP communication between agents software environment allows real life hardware application in the form of a distributed agent system. Future research will be dedicated to the development of model-predictive cost functions and the application of the MAS to smart district heating grids.

Acknowledgement

The authors gratefully acknowledge the financial support provided by the BMWi (Federal Ministry for Economic Affairs and Energy), Germany, promotional references 03ET1218A.

References

- Afram, A. and Janabi-Sharifi, F. (2014). Theory and applications of hvac control systems – a review of model predictive control (mpc). *Building and Environment*, 72:343–355.
- Ali, M., Vukovic, V., Sahir, M. H., and Fontanella, G. (2013). Energy analysis of chilled water system configurations using simulation-based optimization. *Energy and Buildings*, 59:111–122.

- Allan, R. (2010). Survey of agent based modelling and simulation tools. Technical report, Science and Technology Facilities Council.
- Ansari, J., Gholami, A., and Kazemi, A. (2016). Multi-agent systems for reactive power control in smart grids. *International Journal of Electrical Power & Energy Systems*, 83:411–425.
- Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. Wiley Series in Agent Technology. Wiley.
- Bünning, F. (2015). Development of a modelica-library for the agent-based control of hvac systems. Bachelorthesis, RWTH Aachen University.
- Caire, G. (2009). Jade tutorial - jade programming for beginners. TILAB.
- Dassault Systemes (2016). Dymola. <http://www.3ds.com/products-services/catia/products/dymola>.
- Davidsson, P. and Boman, M. (2000). Saving energy and providing value added services in intelligent buildings: A mas approach. In *Agent Systems, Mobile Agents, and Applications*, pages 166–177. Springer.
- Divenyi, D. (2013). Agent-based modeling of distributed generation in power system control. *IEEE Transactions on Sustainable Energy*, 4:886–889.
- FIPA (2002a). Fipa acl message structure specification.
- FIPA (2002b). Fipa communicative act library specification.
- FIPA (2002c). Fipa contract net interaction protocol specification.
- Fuchs, M., Teichmann, J., Lauster, M., Remmen, P., Streblow, R., and Müller, D. (2016). Workflow automation for combined modeling of buildings and district energy systems. *Energy*.
- Huber, M., Brust, S., Schütz, T., Constantin, A., Streblow, R., and Müller, D. (2015). Purely agent based control of building energy supply systems. In *ECOS - International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems*.
- Huberman, B. A. and Clearwater, S. H. (1995). A multi-agent system for controlling building environments. In *ICMAS*, pages 171–176.
- Hurtado, L., Nguyen, P., and Kling, W. (2015). Smart grid and smart building inter-operation using agent-based particle swarm optimization. *Sustainable Energy, Grids and Networks*, 2:32–40.
- Jiang, Z. (2006). Agent-based control framework for distributed energy resources microgrids. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*.
- Karavas, C.-S., Kyriakarakos, G., Arvanitis, K. G., and Papadakis, G. (2015). A multi-agent decentralized energy management system based on distributed intelligence for the design and control of autonomous polygeneration microgrids. *Energy Conversion and Management*, 103:166–179.
- Khan, M. R. B., Jidin, R., and Pasupuleti, J. (2016). Multi-agent based distributed control architecture for microgrid energy management and optimization. *Energy Conversion and Management*, 112:288–307.
- Kirn, S. (2002). Kooperierende intelligente softwareagenten. *Wirtschaftsinformatik*, 44(1):53–63.
- Kok, K., Roossien, B., MacDougall, P., van Pruissen, O., Venekamp, G., Kamphuis, R., Laarakkers, J., and Warmer, C. (2012). Dynamic pricing by scalable energy management systems—field experiences and simulation results using powermatcher. In *Power and Energy Society General Meeting, 2012 IEEE*, pages 1–8. IEEE.
- Kuznetsova, E., Li, Y.-F., Ruiz, C., and Zio, E. (2014). An integrated framework of agent-based modelling and robust optimization for microgrid energy management. *Applied Energy*, 129:70–88.
- Mokhtar, M., Liu, X., and Howe, J. (2014). Multi-agent gaussian adaptive resonance theory map for building energy control and thermal comfort management of uclan’s westlakes samuel lindow building. *Energy and Buildings*, 80:504–516.
- Mokhtar, M., Stables, M., Liu, X., and Howe, J. (2013). Intelligent multi-agent system for building heat distribution control with combined gas boilers and ground source heat pump. *Energy and Buildings*, 62:615–626.
- Otter, M., Arzen, K., and Dressler, I. (2005). Stategraph — a modelica library for hierarchical state machines. In *In Proceedings of the 4th International Modelica Conference*, pages 569–578.
- Perera, D., Winkler, D., and Skeie, N.-O. (2016). Multi-floor building heating models in matlab and modelica environments. *Applied Energy*, 171:46–57.
- Qiao, B., Liu, K., and Guy, C. (2006). A multi-agent system for building control. In *Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology*, pages 653–659. IEEE Computer Society.
- Radhakrishnan, B. M. and Srinivasan, D. (2016). A multi-agent based distributed energy management scheme for smart grid applications. *Energy*, 103:192–204.
- Rahman, M., Mahmud, M., Oo, A., Pota, H., and Hossain, M. (2016). Agent-based reactive power management of power distribution networks with distributed energy generation. *Energy Conversion and Management*, 120:120–134.
- Rosenschein, J. (1985). *Rational Interaction: Cooperation among Intelligent Agents*. PhD thesis, Stanford University.
- RWTH-EBC (2015). Aixlib - a modelica model library for building performance simulations. <https://github.com/rwth-ebc/aixlib>.
- Sangi, R., Baranski, M., Oltmanns, J., Streblow, R., and Müller, D. (2016). Modeling and simulation of the heating circuit of a multi-functional building. *Energy and Buildings*, 110:13–22.
- Sangi, R., Streblow, R., and Müller, D. (2014). Approaches for a fair exergetic comparison of renewable and non-renewable building energy systems. In *The 27th international conference on efficiency, cost, optimization, simulation and environmental impact of energy systems*. Turku, Finland.
- Thiele, B. and Bellmann, T. (2015). Modelica DeviceDrivers. https://github.com/modelica/Modelica_DeviceDrivers.
- van Pruissen, O., van der Togt, A., and Werkman, E. (2014). Energy efficiency comparison of a centralized and a multi-agent market based heating system in a field test. *Energy Procedia*, 62:170–179.
- Verein Deutscher Ingenieure (2010). Vdi 2653 blatt 1.
- Wang, Z., Wang, L., Dounis, A. I., and Yang, R. (2012). Multi-

- agent control system with information fusion based comfort model for smart buildings. *Applied Energy*, 99:247–254.
- Wang, Z., Yang, R., and Wang, L. (2011). Intelligent multi-agent control for integrated building and micro-grid systems. In *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES*, pages 1–7. IEEE.
- Wernstedt, F. (2005). *Multi-Agent Systems for Distributed Control of District Heating Systems*. PhD thesis, Blekinge Institute of Technology, Department of Systems and Software Engineering.
- Wetter, M., Zuo, W., Nouidui, T. S., and Pang, X. (2014). Modelica buildings library. *Journal of Building Performance Simulation*, 7(4):253–270.
- Xydas, E., Marmaras, C., and Cipcigan, L. M. (2016). A multi-agent based scheduling algorithm for adaptive electric vehicles charging. *Applied Energy*, 177:354–365.
- Yang, R. and Wang, L. (2013). Development of multi-agent system for building energy and comfort management based on occupant behaviors. *Energy and Buildings*, 56:1–7.
- Ye, D., Zhang, M., and Sutanto, D. (2015). Decentralised dispatch of distributed energy resources in smart grids via multi-agent coalition formation. *Journal of Parallel and Distributed Computing*, 83:30–43.

MoVE – A Standalone Modelica Vector Graphics Editor

Nicola Justus¹ Christopher Schölzel¹ Andreas Dominik¹

¹KITE, Technische Hochschule Mittelhessen, Giessen, Germany, {nicola.justus, christopher.schoelzel, andreas.dominik}@mni.thm.de

Abstract

Modelica models can have a graphical icon defined as a bitmap or vector graphics. Vector graphics have several benefits, the most obvious being free scaling of images from icon to poster size. With OpenModelica there already exists one open source tool that can be used for editing these vector graphics icon annotations, but it does not reach the usability comfort of professional vector graphics editing tools.

In this paper we present the Modelica Vector graphics Editor (MoVE), a standalone open source editor for Modelica’s vector graphics syntax that provides many convenience features inspired by the vector graphics editor Inkscape. These features include grouping, snap to grid, move to foreground/background, rotation handles, and drawing perfect circles and squares as well as horizontal and vertical lines when holding *Shift*.

We hope that MoVE, as a part of the Modelica Tool Ensemble (MoTE), can enrich the open source ecosystem of Modelica by simplifying the creation of more elaborate vector graphics icons for Modelica models.

Keywords: JavaFX, vector graphics, open source, SVG, Inkscape, MVC, MoTE, OpenModelica

1 Introduction

Modelica is a language for modeling complex physical systems that also incorporates a graphical representation of model components into the language itself. These graphical representations come in the form of annotation statements that can either contain a link to a bitmap image or define an image using a vector graphics syntax (Modelica Assoc., 2012). Vector graphics have the advantage that they are freely scalable. This is interesting in any context where a model might not only be displayed as a small icon on a screen but also has to be presented to a larger audience on a slide or a poster.

Unfortunately, creating vector graphic icons for Modelica models is not as easy as it could be. Standard vector graphics tools such as Inkscape (Inkscape, 2016) provide a rich set of features for precise and fast interaction, such as grouping, rotation handles, sending elements to the front or back, snap to grid, or drawing straight horizontal lines and perfect circles when holding a modifier key. It would be ideal, if we could use such a tool and save the resulting image in Modelica notation. However, the Modelica annotations are not compatible with vector graphics formats

such as Scalable Vector Graphics (SVG) (Dahlström et al., 2011), since there are both features in SVG that have no equivalent in the Modelica syntax and vice versa.

There are many commercial tools for Modelica but OpenModelica is the only open source choice for graphical editing of Modelica models (Fritzson et al., 2005). This graphical editor of OpenModelica is called OpenModelica Connection Editor (OMEdit) (Asghar et al., 2011). It has all features that are required to create vector graphic annotations, but does not provide the same level of user-friendliness as Inkscape or related tools. For example, non-standard rotation angles, fill patterns and line patterns can only be changed via a properties dialog that has to be opened separately for each component; the order of Elements cannot be changed (although respective entries exist in the context menu); drawing of straight horizontal lines and perfect circles is not supported; and when we began this project, OpenModelica did not even support undo-redo operations for graphical manipulations. Furthermore OMEdit generates the icon annotation as one big line of code. This is especially uncomfortable when the source code is managed through a version control system.

In this Paper we therefore present the Modelica Vector graphics Editor (MoVE), a new standalone open source Modelica vector graphic editor with a streamlined interface similar to Inkscape. In the following, we will first give a bit more detail of the context in which MoVE was created. In section two, we will then present an overview of the technologies used to create the editor followed by a discussion of the major design aspects in section three. Section three presents the major features of MoVE and section four explains current limitations leading to the conclusion in section six.

1.1 Background and Related Work

Modelica Tool Ensemble

MoVE is part of the Modelica Tool Ensemble (MoTE) (Justus, Hoppe, and Schölzel, 2017). MoTE aims to provide small user-friendly standalone applications for editing and simulating Modelica models. With this toolset we follow the Unix philosophy to “make each program do one thing well” (McIlroy, Pinson, and Tague, 1978). Separating the different tasks needed for editing and simulating Modelica models leads to smaller applications that are easier to maintain than a full-featured development environment like OpenModelica. Additionally, users may choose to use the tools that they like and substitute the tools they do not like

with other alternatives, leading to a more flexible ecosystem that can accommodate different user needs. MoVE only touches the main annotation statement of a model. To edit other parts of the model, one can, for example, use the text editor Atom (GitHub, 2016) which can provide type checking, auto-completion and error highlighting when coupled with Modelica | Editor (Mo|E), another tool in the MoTE family. Instead one could also chose to use OMEdit or the Eclipse plugin Modelica Development Tooling (MDT) (Pop et al., 2006).

Inkscape

The main source of inspiration for MoVE was the aforementioned vector graphics editor Inkscape (Inkscape, 2016). Inkscape is an open source application that can be used to create professional and complicated vector graphic images. It supports a rich set of features including alignment of elements or individual nodes, combination and cutting of multiple paths, drawing with Bezier curves, bold and italic text, importing shapes from a PDF-file, and many many more. Features that are not already included can be made available with a language-independent scripting API. These features are presented to the user mainly through toolbars and hotkeys that make the interaction fast and seamless. The native format of Inkscape is SVG (Dahlström et al., 2011), which is an XML-based format that can easily be interpreted by other tools.

MoVE does not nearly provide as many features as Inkscape, but it tries to follow the same principles for usability and precision.

2 Technologies

This chapter is a short overview over the technologies that are used for implementing MoVE. Basically MoVE is written in the programming language Scala using the graphical user interface toolkit JavaFX.

2.1 Scala

Scala is a programming language for the Java Virtual Machine (JVM). This means that it is platform-independent and that it is possible to use any Java library. Especially the library JavaFX is useful for creating a modern Graphical User Interfaces (GUIs). Scala merges object orientation with functional programming, which allows to write code faster and more flexible than in plain Java. It also brings its own set of libraries such as a parser combinator library (EPFL and Typesafe, Inc., 2016) that proved very useful for this project.

2.2 JavaFX

JavaFX is a GUI toolkit for the Java programming language. Because it is written for Java it runs on the JVM and is also usable from Scala. JavaFX is the latest toolkit for GUIs running on the JVM and incorporates many ideas of modern GUI design. JavaFX provides a special format for describing the structure of a UI. This format is called FXML and based on XML. To develop GUIs using the

FXML format, Oracle provides a graphical editor for building the user interface by dragging and dropping interface components, namely the *SceneBuilder*. This makes GUI design much faster and leads to a clean separation of the layout and the actual code.

3 Design

3.1 Parser

To load existing Modelica models we have created a simple parser for Modelica source code. This parser is built using the *scala-parser-combinators* library (EPFL and Typesafe, Inc., 2016). This library allows combination of simple parsers to create more complex ones. An external parser generator is not necessary. MoVE ignores everything in the source file, except the icon annotations of all models defined in the file. This makes the parser (and MoVE) mostly independent of future language modifications, thus MoVE should work with future versions of Modelica. If the icon annotations are modified, the parser has to be modified. This should be a small effort. Additionally this assures that MoVE interacts nicely with version control systems. Since we only parse annotations, we can *guarantee* that we will not change any other part of the model.

During the parsing process, the parser generates a MoVE-specific abstract syntax tree. This tree is then transformed into shapes, that are derived from the standard JavaFX shapes. Finally this shapes are displayed in the user interface.

3.2 Model-View-Controller

JavaFX is built around the Model-View-Controller (MVC) software design pattern (Reenskaug, 1979). MVC splits the application in three parts. The first part is the model¹, which represents the business data. The model updates the view if someone changes the model. The second part is the view, which displays the data and listens on updates to the model. The third part is the controller, which connects a model with the respective view. The controller is also responsible for user interactions and transforms them into commands for the model or the view. The typical MVC workflow is depicted in Figure 1.

Because JavaFX already provides views, which contain the data representation for shapes, MoVE is designed with controllers and views. There are no explicit models. They are *hidden inside* of the JavaFX shapes.

3.3 JavaFX Elements

To display Modelica's graphical primitives (Modelica Assoc., 2012), we have created a small set of JavaFX elements. These elements are all derived from the standard JavaFX shape elements and add additional properties and behavior such as fill and stroke patterns. The JavaFX shapes

¹Here, in section 3.2, the word "model" refers to source code of a software project that is structured with the MVC-Pattern and not to a Modelica model.

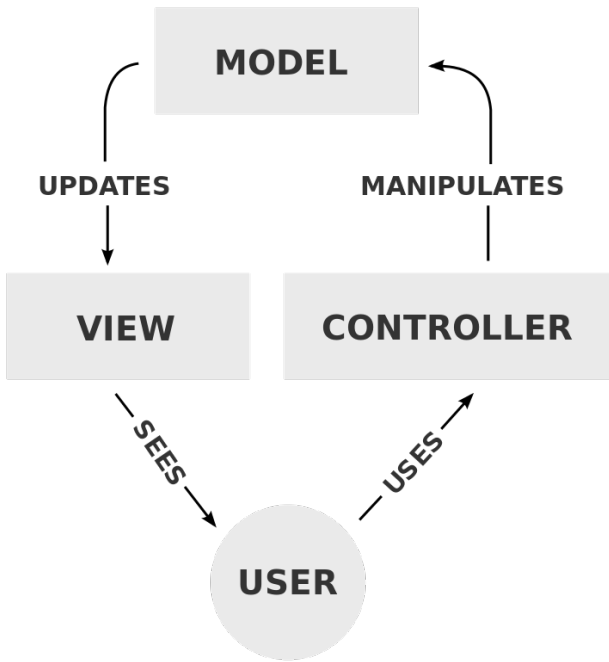


Figure 1. The Model-View-Controller (MVC) software design pattern splits an application into three parts in order to increase maintainability and extensibility (Frey, 2016).

provides the basic properties for rectangles, ellipses, lines, paths, polygons, and images.

Furthermore, for abstracting the common behavior of the shapes, they are all derived from a specific trait, which provides the common behavior. For example, all shapes that *behave like a rectangle* are derived from the trait *RectangleLike*. A similar trait is defined for shapes that *behave like a path*.

3.4 UI Overview

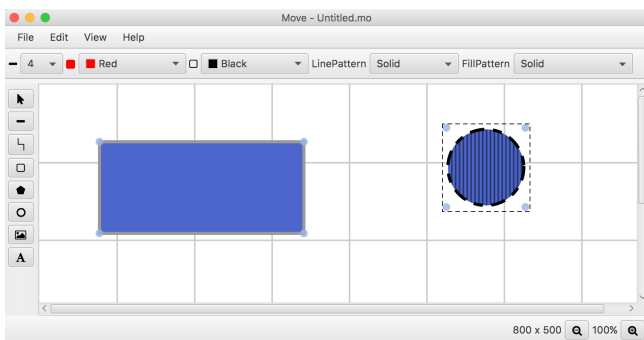


Figure 2. The user interface of MoVE is built with the JavaFX-framework and consists of three main toolbars: tool selection (left), tool properties (top) and zoom and size indicator (bottom).

The user interface contains 3 toolbars for interacting with MoVE (Figure 2).

The top toolbar contains controls for specifying the color of selected or newly drawn shapes. Going left to right this toolbar starts with a selector for the stroke size, followed

by the color pickers for the fill color and stroke color. The color pickers are followed by a selector for the *LinePattern* and *FillPattern*. For these last two elements, all patterns defined in chapter 18.6 from (Modelica Assoc., 2012) can be selected.

The left toolbar contains the tools for selecting and moving as well as drawing the icon primitives. Going top to bottom it starts with the arrow, which is used for selecting and moving shapes. The arrow is followed by the tools for drawing lines, paths, rectangles, polygons and ellipses, and for inserting images, and text.

The bottom toolbar currently only contains two items: an indicator for the size of the draw pane and buttons for controlling the zoom. The magnifying glass with the minus zooms out and the magnifying glass with the plus zooms in.

4 Features

4.1 Code Generation

MoVE provides two code generators for the icon annotation. The first generates the annotation as one big line and writes it into the model. This is similar to OMEdit. The second code generator generates *pretty-printed* code with line breaks and indentations, which is more readable than a big line (Listing 1). This style is also better supported by version control systems as they can recognize which lines or properties have changed instead of reporting only a change of the whole annotation.

Listing 1. MoVE generates a well formatted icon annotation with line breaks and indentation.

```

model Test
  annotation (
    Icon (
      coordinateSystem(
        extent = {{0,0},{200,125}}
      ),
      graphics = {
        Rectangle(
          origin = {34,96},
          lineColor = {0,0,0},
          fillColor = {128,186,36},
          lineThickness = 4.0,
          pattern = LinePattern.Solid,
          fillPattern = FillPattern.Solid,
          extent = {{-14,8}, {14,-8}}
        ),
        Ellipse(
          origin = {75,91},
          lineColor = {0,0,0},
          fillColor = {128,186,36},
          lineThickness = 4.0,
          pattern = LinePattern.Solid,
          fillPattern = FillPattern.Solid,
          extent = {{-13,10}, {13,-10}},
          endAngle = 360
        )
      }
    )
  );
end Test;
  
```

4.2 Grouping

MoVE supports grouping of multiple shapes through *Edit* → *Group* or by pressing *Ctrl+G*. Moving a shape which is part of a group moves all shapes that are part of this group (Figure 3). Ungrouping is also supported through *Edit* → *Ungroup* or by pressing *Ctrl+Shift+G*. Note that the groups are only used in MoVE and are discarded when the model is saved, because this is not supported by the icon annotation syntax of Modelica.

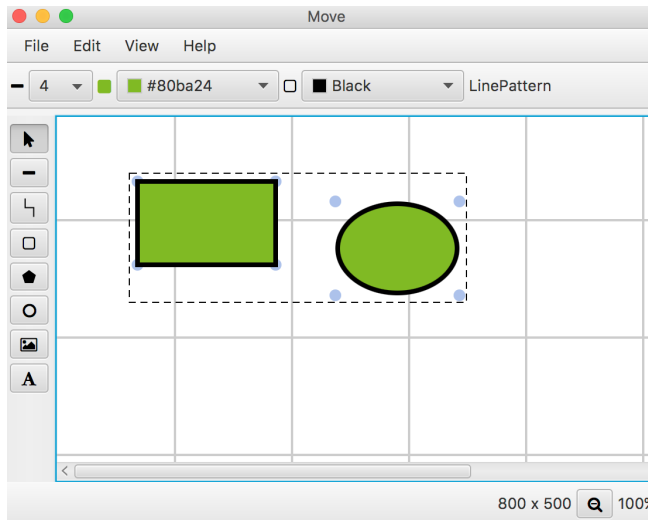


Figure 3. MoVE allows to group shapes together in the user interface, so that they can be easily moved together. These groups are lost when the annotation is saved.

4.3 Stacked Shapes

MoVE allows to move shapes into the background using *ContextMenu* → *In Background* and to move shapes into the foreground using *ContextMenu* → *In Foreground* (Figure 4). This allows easy modifying of the order of stacked shapes.

Furthermore, MoVE also handles shapes with the fill pattern *FillPattern.None* in an intuitive way. Shapes that are behind the transparent filling can still be selected. The transparent shape itself is only selected when the user clicks on the visible border.

4.4 Export as Images

MoVE enables exporting of Modelica icons either as *PNG* or as *SVG* (Figure 5). *SVG* is especially interesting because *SVG* images can be further modified in *Inkscape*. This is useful if the user likes to create a poster which contains a graphic from a Modelica model.

4.5 Rotation

After a double click on a shape, four red anchors appear at the corners of the shape (Figure 6). Moving the anchors rotates the shape around its center. This is more intuitive than rotating a shape by defining a specific degree value through a separate property dialog.

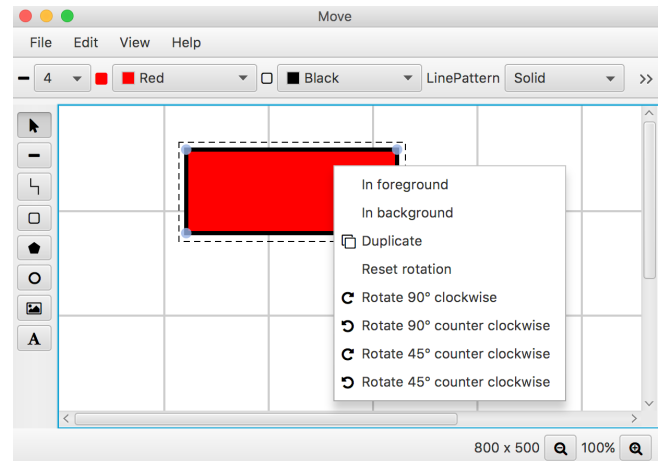


Figure 4. The context menu for a shape contains controls for rotation and stacking order.

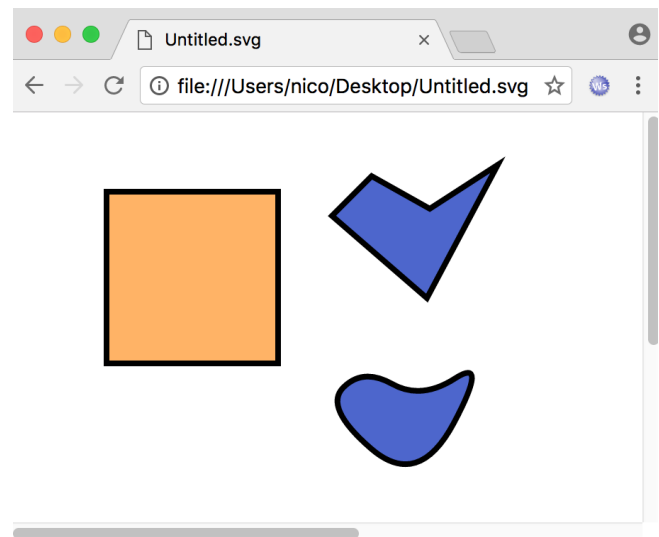


Figure 5. SVG image exported from MoVE displayed in Google Chrome.

Additionally to rotation by moving the anchors, it is possible to rotate an element using the context menu:

- *ContextMenu* → *Rotate 90° clockwise*
- *ContextMenu* → *Rotate 90° counter clockwise*
- *ContextMenu* → *Rotate 45° clockwise*
- *ContextMenu* → *Rotate 45° counter clockwise*

4.6 Snap to Grid

MoVE operates on a customizable grid. The size of the grid can be modified to fit the needs of the user. Via the menu entry *View* → *Enable snapping* or by pressing *Ctrl+A* the *snap to grid* function can be toggled. If activated, elements will snap to the precise location of the grid lines when they are moved close to such a line. This allows for a precise alignment of individual elements.

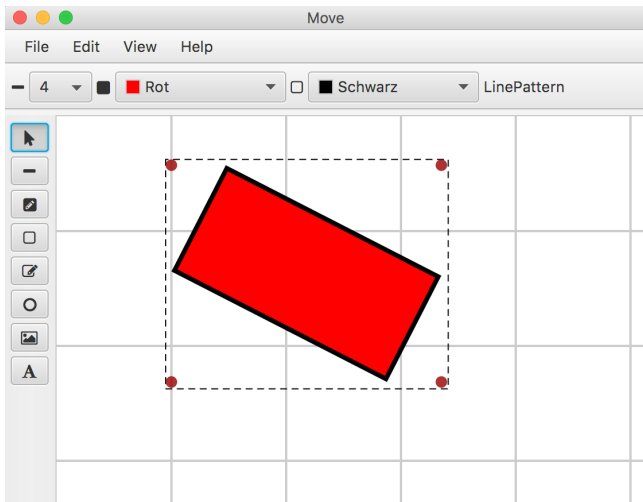


Figure 6. Arbitrary rotations can be realized in MoVE by rotation handles (red dots).

4.7 Config Files

MoVE uses simple text files as configuration files that are placed inside of `~/ .move`. Application settings are placed in the file `~/ .move/move.conf` and keyboard shortcuts are read from `~/ .move/shortcuts.conf`. Both files can be customized with any text editor.

4.8 Additional features

When holding down *Shift* while drawing a shape, it is possible to create straight horizontal or vertical lines, perfect squares, and perfect circles (Figure 7).

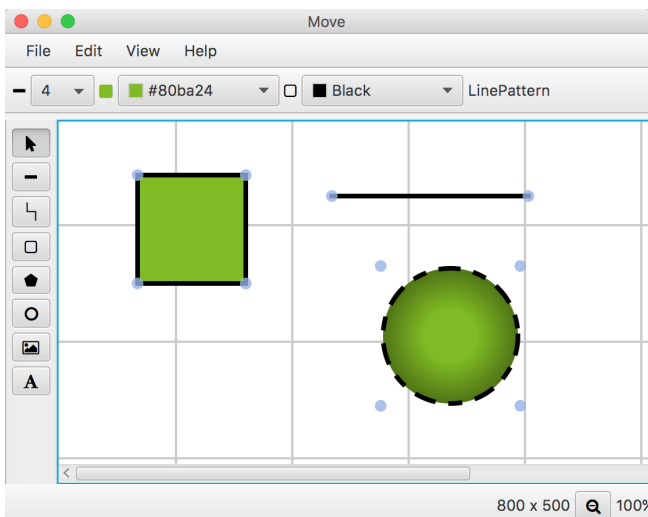


Figure 7. When holding *Shift*, MoVE will create perfect squares and circles and straight horizontal or vertical lines.

MoVE supports *undo* and *redo* using *Edit* → *Undo* / *Edit* → *Redo* or through the shortcuts *Ctrl+Z* and *Ctrl+Shift+Z*. It is also possible to *copy*, *paste* and *duplicate* selected shapes. Holding down *Shift* and selecting a shape selects multiple shapes.

5 Limitations

5.1 Annotations

MoVE supports every icon annotation except properties which are defined using a *if*-clause or a *DynamicSelect* statement, because the result of both statements is a dynamic value, which is only defined at runtime. These dynamic definitions do not fit into the scope of an editor for static vector graphic images. If MoVE finds properties, which are defined using this two statements, it warns the user that this properties will be overwritten by a static value after a save call (see Figure 8).

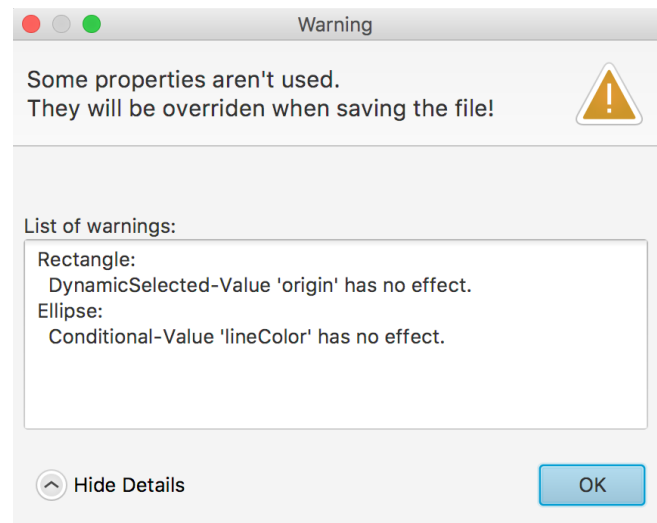


Figure 8. A Warning is displayed when opening a Model whose icon annotation contains *DynamicSelect* and *if*-clause elements in MoVE.

5.2 Line Scaling

The Modelica language specification does not define the meaning of the *thickness* property of a line (Modelica Assoc., 2012). The most intuitive definition would be to assume that the thickness of a line is given in units of the coordinate system of the icon. Both Dymola and OMEdit, however, define the line thickness in terms of the coordinate system of the users screen, so that lines scale automatically when zoomed. At the moment, MoVE does not follow this behavior, because it is unintuitive and cannot be reproduced when the image is exported to SVG or Portable Network Graphics (PNG).

5.3 Placing Connectors

MoVE currently does not support placing connectors in the icon, because this would require parsing and altering connector definitions in the model. Loading and saving models with MoVE does not affect existing connector placements. MoVE is only a graphical editor for the main annotation statement of Modelica models and leaves the rest of the code untouched. Connector placement would add another layer of complexity to the tool that goes beyond its intended scope.

We are currently working on another tool in the MoTE family named Modelica Diagram Editor (MoDE), which will be used for the graphical composition of Modelica models and could also be used handle the connector placement (Hoppe, 2016).

5.4 Inherited Annotations

Modelica allows inheritance of icon annotations. The inherited annotations are currently not displayed in MoVE. This feature was postponed to future versions, because it would require parsing of several files and inspection of inheritance hierarchies.

6 Conclusions

In this paper we presented a new graphical editor for Modelica icon annotations. In contrast to other open source alternatives, the user interface of MoVE is specifically designed to make editing and creating vector graphic icons for Modelica models as easy and fast as creating a vector graphic image with tools such as Inkscape. MoVE builds on the modern platform-independent framework JavaFX. It has many convenience features such as grouping, snap to grid, move to foreground/background, rotation handles, and drawing perfect circles and squares as well as horizontal and vertical lines when holding *Shift*. It is also designed to work well with version control systems so that changes to individual elements can be captured. Except for dynamic elements, it supports every part of the icon definition in the Modelica language specification.

There are many possibilities for future improvement which can be drawn from the feature set of Inkscape such as component and node alignment or the combination and cutting of multiple paths. Ideally, these features could be brought to MoVE by a (partial) import of SVG graphics. This would allow to create icons in Inkscape and convert them into Modelica code so that they are used directly in Modelica models. For this, one would need to define a subset of SVG that is translatable to Modelica and somehow restrict the user in Inkscape to only use this subset. Furthermore, if MoVE should be able to place and display connectors of the model, the parser needs to be extended and additional parts of the model have to be altered.

We hope that this tool can enrich the open source ecosystem of Modelica and will enable more elaborate vector graphic icons for Modelica libraries. MoVE is part of a larger ensemble of tools called MoTE, which also features an integration of Modelica compiler features into a structured text editor.

The projects are open source and hosted on GitHub:
<https://github.com/thm-mote/>

References

Asghar, Syed Adeel et al. (2011). “An Open Source Modelica Graphic Editor Integrated with Electronic Notebooks and Interactive Simulation”. In: *Proceedings of the 8th*

International Modelica Conference. Dresden, Germany, pp. 739–747.

Dahlström, Erik et al. (2011). *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*. W3C Recommendation REC-SVG11-20110816. W3C. URL: <https://www.w3.org/TR/SVG/>.

EPFL and Typesafe, Inc. (2016). *scala-parser-combinators*. GitHub Repository. URL: <https://github.com/scala/scala-parser-combinators> (visited on 12/09/2016).

Frey, Regis (2016). *The model, view, and controller (MVC) pattern relative to the user*. URL: <https://en.wikipedia.org/wiki/File:MVC-Process.svg> (visited on 12/07/2016).

Fritzson, Peter et al. (2005). “The OpenModelica Modeling, Simulation, and Development Environment”. In: *Proceedings of the 46th Scandinavian Conference on Simulation and Modeling (SIMS)*. Trondheim, Norway.

GitHub (2016). *Atom*. URL: <https://atom.io> (visited on 11/01/2016).

Hoppe, Marcel (2016). *Modelica Diagram Editor*. URL: <https://github.com/THM-MoTE/MoDE> (visited on 12/20/2016).

Inkscape (2016). *Inkscape — Draw Freely*. URL: <https://inkscape.org> (visited on 12/09/2016).

Justus, Nicola, Marcel Hoppe, and Christopher Schölzel (2017). *Modelica Tool Ensemble (MoTE)*. URL: <https://github.com/thm-mote> (visited on 03/28/2017).

McIlroy, M. D., E. N. Pinson, and B. A. Tague (1978). “Unix Time-Sharing System: Foreword”. In: *The Bell System Technical Journal* 57.6, pp. 1899–1904.

Modelica Association (2012). *Modelica - A Unified Object-Oriented Language for Systems Modeling*. Language Specification. Version 3.3.

Pop, Adrian Dan Iosif et al. (2006). “OpenModelica Development Environment with Eclipse Integration for Browsing, Modeling, and Debugging”. In: *Proceedings of the 5th International Modelica Conference*. Vienna, Austria, pp. 459–465.

Reenskaug, Trygve (1979). *Thing-Model-View-Editor — An Example from a planning system*. technical note. Xerox PARC.

MolE – A Communication Service Between Modelica Compilers and Text Editors

Nicola Justus¹ Christopher Schölzel¹ Andreas Dominik¹ Thomas Letschert¹

¹KITE, Technische Hochschule Mittelhessen, Giessen, Germany, {nicola.justus, christopher.schoelzel, andreas.dominik, thomas.letschert}@mni.thm.de

Abstract

The Modelica language is becoming increasingly popular among scientists and engineers as platform for modelling physical or biological systems. Although Modelica is maintained as non-proprietary language by the Modelica Association, a considerable number of commercial implementations and development environments is complemented by a surprisingly small number of open source tools.

In this paper, we present the communication service MolE that connects any text editor as front-end with a Modelica compiler as back-end. Based on the simple HTTP communication protocol, editor plugins for a software developer's favourite text editor can be developed easily, hence turning any editor into a lightweight Modelica development tool.

We also present a first implementation of a plugin for the text editor Atom that exhibits features necessary for efficient software development, such as display of compile errors, code completion, go to declaration or view of context-sensitive documentation. In addition, Modelica-specific checking of the number of equations in a model is supported.

Keywords: *Modelica, open source, integrated development environment, distributed systems, structured editor, ENSIME, OpenModelica, JModelica, MoTE*

1 Introduction

Modelica is a powerful object-oriented programming language that facilitates acausal description of physical systems. Although many commercial and open source tools for developing or working with Modelica are available, the OpenModelica suite (Fritzson et al., 2005) is the only comprehensive set of tools for Modelica. OpenModelica provides a standalone Modelica compiler, an Eclipse plugin for developing Modelica inside of Eclipse (MDT), a graphical model editor for connecting components (OMedit), and a Modelica debugger. The primary tools for developing Modelica are MDT and OMedit. Both are full-fledged integrated development environments (IDEs).

IDEs are well suited for working with big projects but may have some disadvantages. They often are slow, difficult to use and may be even scary for novice users. For Modelica additional challenges arise from the differences

between Modelica compilers, such as JModelica or OpenModelica which slightly differ in their understanding of Modelica. In order to develop code compatible with different compilers, the IDE should be able to compile models using different compilers.

Today, when writing source code or any other type of structured text, it is common to use a structured editor which is aware of the document's structure. Structured editors are an essential part of most IDEs. Experienced developers usually prefer them to other – graphical – means of input. A structure aware editor must be able to analyze the text given to it. Thus structure awareness means awareness of the syntax and to some extent also of the semantics of the texts it deals with. The structured editor is deeply integrated with the IDE, rather than being just a mere component.

In this paper we present Modelica | Editor (MolE), a development environment for Modelica, centered on editing and checking complex models, refraining from all issues of model execution. A structured editor is its main component and user interface.

A key concept of MolE is that the user may use a text editor of her own choice, attach it to a service process that provides syntactic and semantic analysis and transforms the plain text editor to a structured editor.

Thus users may edit texts using the editor they are used to and still benefit from automatic recompilation, code completion, semantic highlighting, go to declaration, refactoring, and so on.

A central part of our solution is a server process that mediates between the text editor and Modelica aware analytical services. These services are provided by existing Modelica compilers, and/or further existing or future tools that may be plugged into this infrastructure (Figure 1). We have enhanced one text editor to a Modelica editor, but other text editors may be integrated with little effort. These editors only have to provide a plugin that implements the service API. This API provides a unique interface to different Modelica compilers and eases the communication with compilers and related tools, protecting users from complex and differing command-line interfaces.

The design of MolE was inspired by the ENSIME project (ENSIME Contrib., 2016) with its server process that mediates between text editors and Scala compilers.

MolE is an environment for developing Modelica models

using editors that are enhanced to be Modelica aware. It was realized as part of the first author's bachelor's thesis (Justus, 2016).

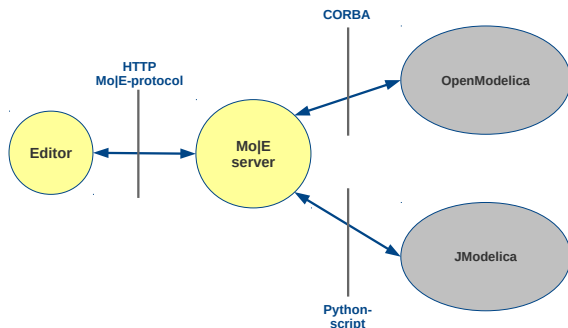


Figure 1. Survey of the communication between a text editor with a Mo|E-plugin, the Mo|E server and OpenModelica or JModelica.

1.1 Structure of the Paper

Section 2 describes which technologies and standards were used to implement Mo|E. Section 3 describes the protocol between client and service process, the communication with OpenModelica as well as with JModelica. Section 4 presents the key features of Modelica | Editor (Mo|E) and their use in the text editor Atom. Finally, section 5 gives a summary and a short outlook on future extensions.

1.2 Naming

The name Modelica | Editor (Mo|E) alludes to the use of the pipe character (|) in UNIX-like operating systems, which establishes a pipeline between two programs. Mo|E can be seen as such a pipeline between the Modelica compiler and a structured editor. In contexts where special characters like the pipe may cause problems, we chose the alternative spelling Modelica–Pipe–Editor (MoPE).

1.3 Goals

Our goals for Mo|E are:

- Provide an extendable client server application which makes it possible to develop Modelica inside existing text editors.
- Provide a client implementation for the text editor Atom as reference for other clients.
- Highlight syntax and type errors, perhaps while typing, inside the text editor.
- Provide code completion for models, data types, and variables.

- Provide jump to the source of a model. This is better known as “go to declaration”.
- Provide a view of the documentation of a model.

1.4 Background and Related Work

OneModelica (Samlaus, 2015) is an Eclipse-based IDE for Modelica models tailored to the domain of fluid dynamics. It was realized using tools and techniques of Model Driven Software Development. It may be compared to our approach in that it restricts itself to syntax and static semantics of the language and refrains from simulation issues. It differs considerably in its technological base, which in the years since its development has lost a lot of its attraction and support, not without reason as we think.

Mo|E is the first tool in a more ambitious project called Modelica Tool Ensemble (MoTE). MoTE aims at the provision of a collection of small user-friendly standalone applications for developing and executing Modelica models, i.e. a lightweight development environment for Modelica.

Modelica does not differ in principle from other languages when it comes to development environments. However, due to its complex static and dynamic semantics, it poses special challenges, mainly for the support of incremental development (see e.g. (Höger, Lorenzen, and Pepper, 2010) or (Broman, Fritzon, and Furic, 2006)).

We are well aware of these problems. Thus, at least for the time being, MoTE and Mo|E do not include a Modelica compiler or tools incorporating compiler features much beyond parsing. Instead we rely on mature compilers like OpenModelica and JModelica.

2 Technologies

2.1 Scala and Akka

Scala (EPFL, 2016) is a hybrid programming language that combines object orientation with functional programming. Because the Scala compiler generates bytecode for the Java Virtual Machine (JVM), it integrates with many available Java libraries. In addition, resulting compiled programs are platform independent. The service process of Mo|E is implemented in Scala.

Akka is a library for concurrent and distributed systems, based on the actor model that facilitates concurrency by providing a high level of abstraction (Allen, 2013). We use Akka as a provider of communication services, such as an implementation of the HTTP-protocol and for structuring the system according to the actor model.

2.2 OMC and CORBA

OpenModelica provides the *Advanced Interactive Open-Modelica Compiler* (OMC), a server that provides an API to query loaded Modelica code (Asghar et al., 2011).

The Common Object Request Broker Architecture (CORBA) is used by the OpenModelica compiler server OMC as interface to other applications and other programming languages.

CORBA developed by the Object Management Group (OMG) defines a standard for inter-process communication modeled as interaction of distributed objects. Because the public API of remote objects is defined in an Interface Definition Language (IDL), processes may be implemented in different programming languages (OMG, 2012).

2.3 Atom and the Electron Engine

Atom (GitHub, 2016a) is a text editor created by GitHub in the style of Sublime Text (Sublime HQ Pty Ltd, 2016). Basic design concepts of Atom include customization and extensibility through plugins (called packages in the context of Atom). Extending Atom is possible with JavaScript, HTML and CSS by using the Electron Engine (GitHub, 2016c). This allows to *rapidly develop* extensions and to implement communication protocols using *AJAX requests*. Furthermore, Atom already includes a package for syntax highlighting for Modelica (Chenouard et al., 2016), a simple API for completion suggestions (GitHub, 2016b) and a plugin for clicking on text (Facebook, 2016), which is used to implement *go to declaration* functionality.

We have created an Atom plugin as first reference implementation of a MoE client.

3 Design

3.1 MoE – Editor Protocol

Clients are connected to the service process, by means of Hypertext Transfer Protocol (HTTP)-based communication and JavaScript Object Notation (JSON) data representation. HTTP provides status codes, Uniform Resource Identifiers (URIs) and content negotiation (Fielding and Reschke, 2014). JSON is a compact text format, based on the JavaScript Object Notation (Bray, 2014).

The communication flow follows several steps: Firstly, the client connects to the service process using a *connect request* that communicates the current project. In this context a project is a directory containing Modelica source files.

Secondly, after initialization the service process answers with the respective *project id*. The unique project id identifies the project in the client server communication.

Henceforth, the client uses this project id to *request further IDE functionality* for this project from the service process.

To finally *finish a session*, the client sends a disconnect request that triggers the service process to delete all project-related information and cached data.

The following sections describe each supported IDE functionality in more detail.

3.1.1 Connecting to the server

As introduced in the preceding section, each client needs to connect initially with the server. A *connect* request is initiated through a POST request containing the respective JSON object with the project description. The JSON object contains the full path into the project directory and the

relative path to a directory that is used to store compiled files:

```
POST /mope/connect

{
  "path": <String>,
  "outputDirectory": <String>
}
```

This project information is stored in the `mope-project.json` file that is placed in the project directory.

If the request was successful, the server answers with a project id. If not, the server answers with 400 `BadRequest` and a detailed error message.

3.1.2 Compiling Modelica source files & Modelica script files

Compiling a Modelica source file is initiated through a *compile* request. The request body contains the path to the currently opened file. As a result of the request a model is instantiated and type errors are retrieved:

```
POST /mope/project/0/compile

{ "path": <String> }
```

If the request was successful, the server answers with a JSON array containing compiler errors:

```
{
  "type": "Error" | "Warning", //type of
    message
  "file": <String>, //path to the file
    which contains the error
  "start": { //start of error
    "line": <Number>,
    "column": <Number>
  },
  "end": { //end of error
    "line": <Number>,
    "column": <Number>
  },
  "message": <String> //compiler error
}
```

Compiling a Modelica script file is initiated by sending an analogous *compileScript* request:

```
POST /mope/project/0/compileScript

{ "path": <String> }
```

Although the request is called “compiling a Script file”, the service process actually executes the script. This action is intended for debugging purposes of smaller scripts and not for scripts that simulate a model, since simulating a model is time-consuming and may freeze or possibly even kill the service process.

3.1.3 Checking a model

To check a model for its number of equations the client sends a *checkModel* request with the model path. The server calls the OpenModelica compiler to run *checkModel* and answers with a string containing the results:

POST /mope/project/0/checkModel

```
{ "path": <String> }
```

This functionality is only available, if the OpenModelica compiler is used.

3.1.4 Go to declaration

To retrieve the declaration of a model, the client sends a declaration request. This request contains the model/-class name as query string¹:

GET /mope/project/0/declaration?class=[Modelname]

The server answers with a JSON object containing the file path and line number of the declaration:

```
{
  "path": <String>, //absolute path to the
    file
  "line": <Number> //line number
}
```

If the project id is unknown or the query string is missing, the server will answer with a 404 Not Found error.

3.1.5 Go to documentation

A model documentation can be retrieved using a doc request with the model name encoded as query string:

GET /mope/project/0/doc?class=[Modelname]

The server embeds the documentation in a template and returns a HTML document that can be viewed in a web browser.

If the project id is unknown or the query string is missing, the server answers with a 404 Not Found error.

3.1.6 Code completion

For code completion the client sends a completion request with a JSON object that describes the position of the cursor as *file* (name of current file), *line* and *column number* (position of the cursor) and *word* (part of the expression to be completed):

POST /mope/project/0/completion

```
{
  "file": <String>, //absolute path to the
    file
  "position": { //position inside the file
    "line": <Number>,
    "column": <Number>,
  },
  "word": <String>
}
```

The server responds by sending an JSON array of possible completions for the expression:

```
{
  //type of completion; 1 of the listed
    strings
}
```

¹A query string is a component of a URI, that starts with a ? (Berners-Lee, Fielding, and Masinter, 2005).

```
"kind": "Type" | "Variable" | "Function"
  | "Keyword" | "Package" | "Model" | "
    Class" | "Property",
"name": <String>, //the completion
//OPTIONAL: list containing names of
  parameters if kind=function
"parameters": [
  <String>,
  <String>,
  ...
],
//OPTIONAL: the class comment describing
  the name attribute
"classComment": <String>,
//OPTIONAL: the type of name
"type": <String>
}
```

kind defines the type of the completion (such as package, class, function, variable, etc.). *name* is the suggestion for the subexpression.

The optional return values for *parameters*, *classComment* and *type* report the list of argument names if the suggestion is a function, the documentation string if the suggestion is a class and the data type of the expression (usually the data type of a variable), respectively.

If the given project id is unknown, the server answers with 404 Not Found.

3.1.7 Display data type of a variable

To retrieve data type and documentation string of a variable, the client sends a *typeof* request with a body identical to the body of the *completion* request. If the request was successful, the server answers with a JSON object containing the name, type, and documentation string of the variable. Otherwise the server answers with 404 Not Found:

POST /mope/project/0/typeof

```
{
  "name": <String>, //name of property
  "type": <String>, //type of property
  //OPTIONAL: property comment
  "comment": <String>
}
```

3.1.8 Disconnecting from the server

A session is terminated by a disconnect request, which initiates the shutdown sequence for this project on the server:

POST /mope/project/0/disconnect

The server returns 204 NoContent if the project id is known or 404 Not Found otherwise.

3.1.9 Stopping the server

The client can stop the whole service process by sending a *stopServer* request. The server answer is 202 Accepted.

POST /mope/stop-server

3.2 Communication with OpenModelica

The modeling and development environment OpenModelica (OSMC, 2016) consists of a Modelica compiler (omc), a graphical connection editor (OMEdit), an Eclipse plugin (MDT) and a Modelica debugger (Fritzson et al., 2005). As described in Chapter 2.2 the compiler enables querying for model information via its CORBA interface that provides several types of information:

- list of all models/classes by sending `getClassNames`,
- source file of a model by sending `getSourceFile`,
- documentation annotation of a model by sending `getDocumentationAnnotation`,
- result of a model check for equations by sending `checkModel`,
- documentation string of a model by sending `getClassComment`,
- arguments of a function by sending `getParameterNames`,
- specialization of a class by sending `getClassRestriction`.

An additional difficulty arises from the fact that OpenModelica uses Modelica expressions as arguments for its CORBA interface. As a result, the functions listed above are not implemented explicitly in the CORBA interface. Instead, OpenModelica only provides a single method in its CORBA interface, namely `sendExpression` and sends Modelica source code strings and API function calls as arguments. Therefore, we create the function calls as strings and interpolate them into the function argument, as shown in Listing 1.

Listing 1. API function call through OpenModelica's CORBA interface.

```
val omc:OmcCommunication = ...
val fileName = "/tmp/model.mo"
val errors:String =omc.sendExpression(s"""
  parseFile("$fileName") """)
```

3.3 Communication with JModelica

JModelica (Modelon AB, 2016) is a Modelica compiler developed by Modelon AB (Åkesson et al., 2010). To allow dynamic adjustments during execution, JModelica offers a Python interface which enables code modification at run time. In addition it enables compilation of Modelica code. We are using this Python interface for compilation of the models by delivering the Modelica source files to a custom Python script, which calls the JModelica compiler, parses JModelica's output and encodes the output into JSON. The resulting JSON is printed to `stdout` which is afterwards parsed by the service process and finally decoded as Scala

Command	Description
Mope: Disconnect	Disconnect Atom from the service process
Mope: Compile Project	Compile the project
Mope: Run Script	Execute the Modelica script
Mope: Check Model	Check the model for its number of equations
Mope: Show Type	Display the data type of the variable below the cursor
Mope: Open Documentation	Open the documentation of the type below the cursor
Mope: Open Server Log	Open the log file of the service process
Mope: Open Server Config	Open the configuration file of the service process
Mope: Stop Server	Stop the server

Table 1. List of commands implemented in the Atom plugin.

objects. The communication scheme is depicted in Figure 2.

Unfortunately JModelica does not offer access to the parsed model or its abstract syntax tree. That is the reason why code completion is restricted to local variables and go to documentation is not yet supported in the presented MolE Atom plugin.

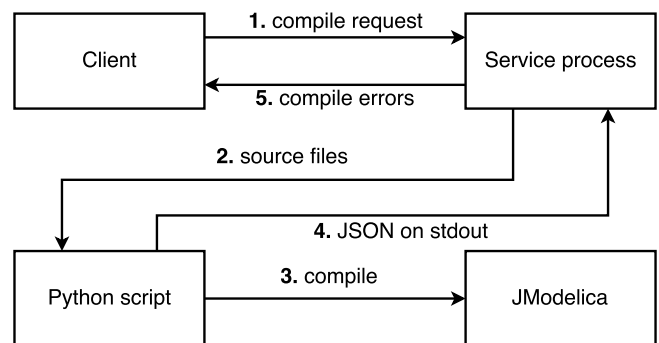


Figure 2. Diagram of the communication between a text editor (client) and JModelica.

4 Features

4.1 Client commands

Table 1 gives a full list of the commands available in the Atom plugin.

4.2 Compiler Feedback

Modelica | Editor (MolE) provides *instant compiler feedback* for syntax errors and *type errors*. Background compili-

lation is automatically triggered when a file is saved and the errors are highlighted in the editor with a red indicator at the left side of the editor tab. Error messages are displayed at the bottom of the tab (Figure 3). Alternatively automatic compilation can be disabled and triggered manually.

As Mo|E supports JModelica and OpenModelica it is possible to use either JModelica or OpenModelica or both compilers for one project.

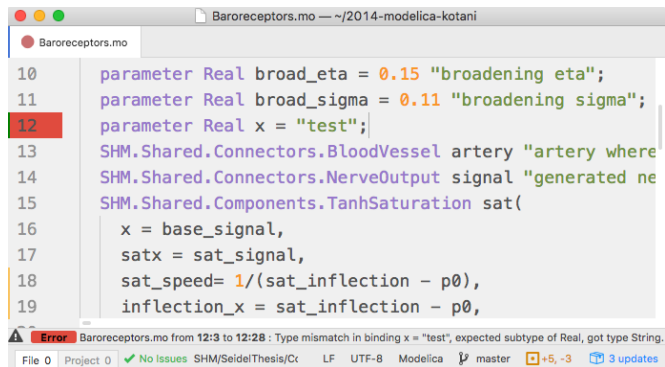


Figure 3. Compile errors are retrieved from the back-end (OpenModelica or JModelica) by the Mo|E server and highlighted in the source code by the editor plugin.

4.3 Code Completion

Modelica | Editor (Mo|E) features *enhanced code completion* on keystrokes or by pressing **Ctrl + Space**. Suggestions include classes, models, functions, model parameters and variables, keywords, built-in types as well as local variables. The suggestions contain a type indicator, documentation string and a link to the model's documentation (Figure 4). The type indicator displays the type of the suggestion (package, model, function or variable).

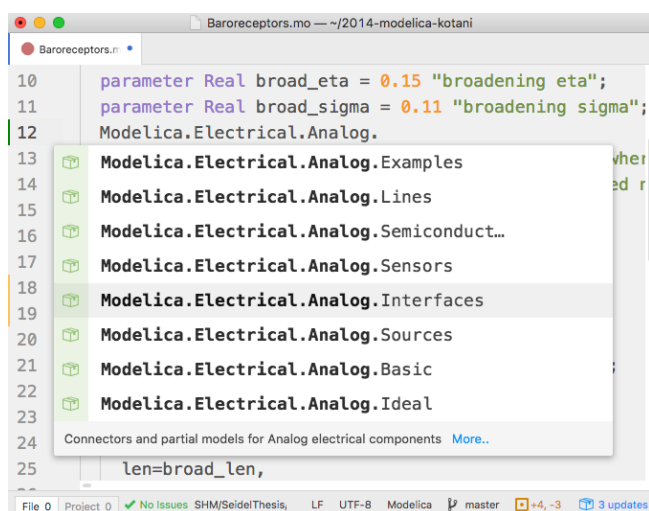


Figure 4. Code completion allows for selecting classes, models, functions, model parameters, variables, keywords or built-in types from a list of suggestions retrieved by the Mo|E server.

4.4 Go to Declaration

Mo|E provides *go to declaration* by clicking on the model/class name while holding down **Ctrl**. The source file of the model/class is opened in a separate tab. Go to declaration is mostly used for discovering source code or when editing multiple models that are linked to each other.

4.5 Documentation View

Mo|E embeds the queried documentation of a model in a predefined template and provides the documentation as HTML document. The implementation in the Atom plugin opens the requested documentation in the default browser. Furthermore it is possible to browse the model's child components using the links in the subcomponents section of the documentation (Figure 5).

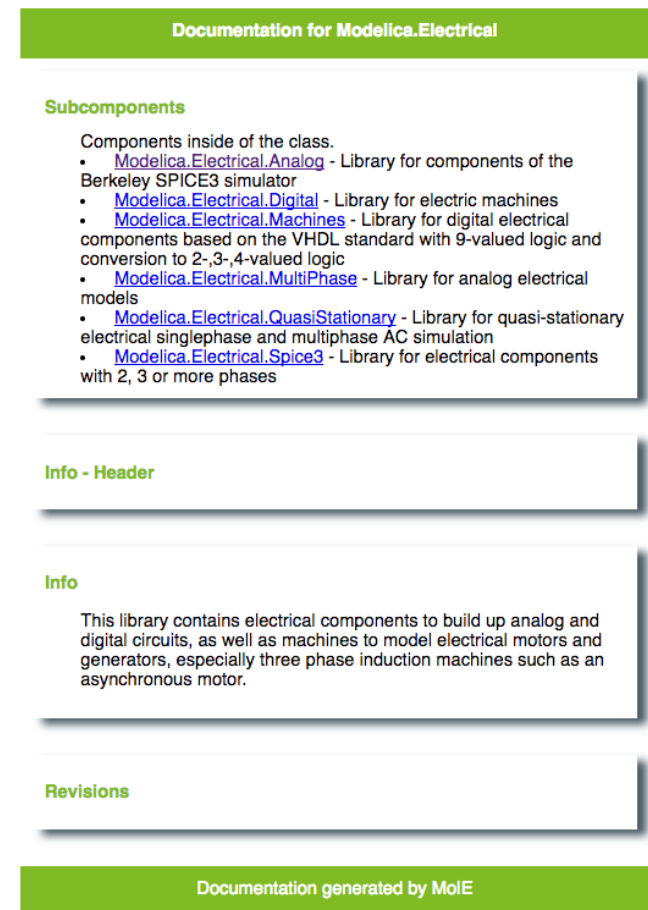


Figure 5. Example of a documentation display generated as HTML page by Mo|E by embedding the retrieved documentation string with a template page.

4.6 Type & Documentation String Display

Mo|E provides a command for displaying the type and documentation string of the variable at the cursor position. Type and documentation are displayed at the bottom of the editor tab (Figure 6).

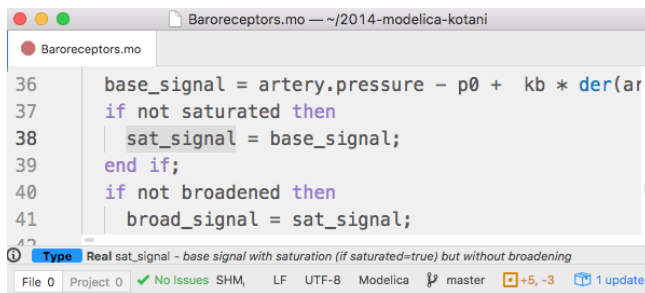


Figure 6. Data type and documentation string are displayed in the editor window by the Atom plugin.

4.7 Execution of Modelica Scripts

If the OpenModelica compiler is used, MoIE allows manually triggered execution of Modelica scripts and displays error messages in the editor.

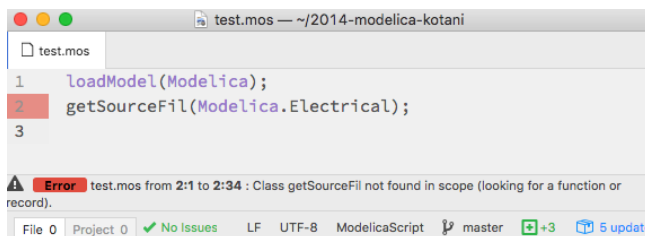


Figure 7. Compile errors of Modelica scripts are displayed in the editor window by the Atom plugin.

4.8 Model Check

MoIE supports checking of a model for the number of equations (Figure 8), if the OpenModelica compiler is used.

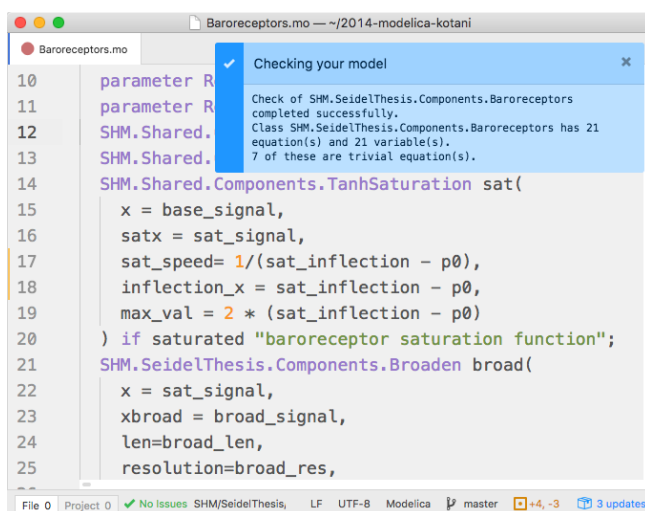


Figure 8. Result of a model check, performed by the OpenModelica back-end, is displayed as pop-up in the editor window by the Atom plugin.

5 Conclusions

This paper presented a extendable client/server application for developing Modelica in enhanced text editors like Atom. It shows how a service process is used to simplify communication with multiple Modelica Compilers and provide IDE features to various text editors through a simple interface. Text editors have to implement a small number of basic HTTP calls, which should be a minimal effort. A minimal setup with compilation and code completion would only require four HTTP calls. Installation instructions for MoIE can be found at <https://github.com/THM-MoTE/mope-server>.

MoIE is a base for further extensions. E.g. we intend to implement plugins for different editors, such as Sublime Text (Sublime HQ Pty Ltd, 2016), Visual Studio Code (Microsoft Corporation, 2016) or vim (Moolenaar, 2016). Including Visual Studio Code should not be a problem because it uses TypeScript for its plugins, which is a superset of Atom's JavaScript.

MoIE is part of a larger ensemble of tools called MoTE (Schölzel et al., 2016). MoTE will also include a vector graphic editor called Modelica Vector Graphics Editor (MoVE) (Justus et al., 2017) and a diagram editor called Modelica Diagram Editor (MoDE) (Hoppe et al., n.d.). Together with MoIE these tools provide alternative user interfaces for the interaction with existing Modelica compilers, which allow a simpler interaction than full-fledged IDEs like OpenModelica.

The projects are open source and hosted on GitHub:

<https://github.com/thm-mote/>

References

- Åkesson, J. et al. (2010). "Modeling and Optimization with Optimica and JModelica.org — Languages and Tools for Solving Large-Scale Dynamic Optimization Problems". In: *Computers & Chemical Engineering* 34 (11), pp. 1737–1749.
- Allen, Jamie (2013). *Effective Akka*. Sebastopol, USA: O'Reilly Media.
- Asghar, Syed Adeel et al. (2011). "An Open Source Modelica Graphic Editor Integrated with Electronic Notebooks and Interactive Simulation". In: *Proceedings of the 8th International Modelica Conference*. Dresden, Germany, pp. 739–747.
- Berners-Lee, T., R. Fielding, and L. Masinter (2005). *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986. IETF.
- Bray, T. (2014). *The JavaScript Object Notation (JSON) Data Interchange Format*. RFC 7159. IETF.
- Broman, D., Peter Fritzson, and S. Furic (2006). "Types in the Modelica Language". In: *In Proceedings of the 5th International Modelica Conference*. Ed. by Ch. and Haumer A. Kral. Vienna, Austria: The Modelica Association, pp. 303–317.

- Chenouard, Raphael et al. (2016). *Modelica language support in Atom*. GitHub Repository. URL: <https://github.com/modelica-tools/atom-language-modelica> (visited on 11/03/2016).
- École Polytechnique Fédérale de Lausanne (2016). *The Scala Programming Language*. URL: <http://www.scala-lang.org/> (visited on 11/01/2016).
- ENSIME Contributors (2016). *ENSIME*. URL: <http://ensime.github.io/> (visited on 11/01/2016).
- Facebook (2016). *Hyperclick*. GitHub Repository. URL: <https://github.com/facebooknuclide/hyperclick> (visited on 11/03/2016).
- Fielding, R. and J. Reschke (2014). *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. RFC 7230. IETF.
- Fritzson, Peter et al. (2005). “The OpenModelica Modeling, Simulation, and Development Environment”. In: *Proceedings of the 46th Scandinavian Conference on Simulation and Modeling (SIMS)*. Trondheim, Norway.
- GitHub (2016a). *Atom*. URL: <https://atom.io> (visited on 11/01/2016).
- (2016b). *Autocomplete+ Package*. GitHub Repository. URL: <https://github.com/atom/autocomplete-plus> (visited on 11/03/2016).
 - (2016c). *Electron — Build cross platform desktop apps with JavaScript, HTML, and CSS*. URL: <http://electron.atom.io/> (visited on 09/14/2016).
- Höger, Christoph, Florian Lorenzen, and Peter Pepper (2010). “Notes on the Separate Compilation of Modelica”. In: *3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*. Ed. by P. Fritzson et al. Oslo, Norway: Linköping Electronic Conference Proceedings, pp. 43–53.
- Hoppe, Marcel, Christopher Schölzel, and Andreas Dominik. *MoDE – A Standalone Modelica Diagram Editor*. unpublished.
- Justus, Nicola (2016). “Design and Implementation of a Client/Server Application for Editing Modelica Inside Various Text Editors”. BA thesis. Giessen, Germany: Technische Hochschule Mittelhessen.
- Justus, Nicola, Christopher Schölzel, and Andreas Dominik (2017). “MoVE – A Standalone Modelica Vector Graphics Editor”. In: *12th International Modelica Conference*. Prague, Czech Republic. to be published.
- Microsoft Corporation (2016). *Visual Studio Code - Code Editing. Redefined*. URL: <https://code.visualstudio.com/> (visited on 22/12/2016).
- Modelon AB (2016). *JModelica.org*. URL: www.jmodelica.org (visited on 11/01/2016).
- Moolenaar, Bram (2016). *welcome home : vim online*. URL: <http://www.vim.org/> (visited on 12/21/2016).
- Object Management Group (2012). *Common Object Request Broker Architecture (CORBA). Part 1: CORBA Interfaces*. OMG document formal/2012-11-12. Version 3.3.
- Open Source Modelica Consortium (2016). *OpenModelica*. URL: <https://openmodelica.org> (visited on 11/01/2016).
- Samlaus, Roland (2015). “An Integrated Development Environment with Enhanced Domain-Specific Interactive Model Validation”. PhD thesis. Linköping University, The Institute of Technology.
- Schölzel, Christopher et al. (2016). *Modelica Tool Ensemble*. GitHub Repository. URL: <https://github.com/orgs/THM-MoTE/> (visited on 12/22/2016).
- Sublime HQ Pty Ltd (2016). *Sublime Text: The text editor you'll fall in love with*. URL: <https://www.sublimetext.com/> (visited on 11/01/2016).

Traceability Support in OpenModelica using Open Services for Lifecycle Collaboration (OSLC)

Alachew Mengist Adrian Pop Adeel Asghar Peter Fritzson

PELAB – Programming Environment Lab, Department of Computer Science, Linköping University, Sweden,
{alachew.mengist, adrian.pop, adeel.asghar, peter.fritzson}@liu.se

Abstract

A common situation in industry is that a system model is composed of several sub-models which may have been developed using different tools. The quality and effectiveness of large scale system modeling heavily depends on the underlying tools used for different phases of the development lifecycle. Available modeling and simulation tools support different operations on models, such as model creation, model simulation, FMU export, model checking, and code generation. Seamless tracing of the requirements and associating them with the models and the simulation results in the context of different modeling tools is becoming increasingly important. This can be used to support several activities such as impact analysis, component reuse, verification, and validation. However, due to the lack of interoperability between tools it is often difficult to use such tools in combination. Recently, the OSLC specification has emerged for integrating different lifecycle tools using linked data. In this paper we present new work on traceability support in OpenModelica where the traceability information is exchanged with other lifecycle tools through a standardized interface and format using OSLC. In particular, OpenModelica supports automatic recording and tracing of modeling activities such as creation, modification, and destruction of models, import model description XML, export of FMUs, and creation of simulation results.

Keywords: *OpenModelica, traceability, OSLC, tool interoperability, tool integration, model management, Modelica*

1 Introduction

Modeling and simulation tools have become increasingly used for industrial applications. Such tools support different activities in the modeling and simulation lifecycle, like specifying requirements, model creation, model simulation, Functional Mock-up Unit (FMU) export (Blochwitz et al, 2011; FMI-Standard.org, 2014), model checking, and code generation. However, the heterogeneity and complexity of modern industrial products often require special purpose modeling and simulation tools for different

phases of the development life cycle. Seamless exchange of models between different modeling tools is needed in order to integrate all the parts of a complex product model throughout the development life cycle.

During the past decade, the Open Services for Lifecycle Collaboration (OSLC) specifications (Open-services.net, 2008) have emerged for integrating development lifecycle tools using Linked Data (Heath and Bizer, 2011). For traceability purposes, in particular the OSLC Change Management specification is relevant. In earlier work (Elaasar and Neal, 2013) OSLC has successfully been demonstrated for integration of modeling tools in general, and traceability in particular.

OpenModelica (Fritzson *et al*, 2006) is an open source modeling, simulation, and optimization tool for Modelica (Modelica Association, 2012; Fritzson, 2014) language. The OpenModelica Connection Editor OMEdit (Asghar *et al*, 2010) is a graphical Modelica model editing and simulation tool. It supports model creation, deletion, FMU export/import, textual and graphical model editing including connections drawing, simulation, plotting, and documentation presentation. In the previous version of OpenModelica (Pop *et al*, 2014) the compiler supports traceability in terms of tracing generated C code back to the originating Modelica source code, but not in the OSLC sense, and mostly used for debugging.

In this paper we present new traceability support in OpenModelica where the traceability information is exchanged with other lifecycle tools through a standardized interface and format using OSLC. In particular, OpenModelica supports automatic recording and tracing of modeling activities such as creation, modification, and destruction of models, import of model description XML, export of FMUs, and creation of simulation results to link models from various tools. OpenModelica supports simple queries (traces to and traces from) to present the traceability information to the user.

The rest of this paper is structured as follows: In Section 2 an overview of OSLC is given. The traceability design and architecture is presented in Section 3. An Example of integrated tools to trace

artifacts created during the system development process is presented in Section 4. Section 5 describes the traceability and model management workflow in OpenModelica. The prototype implementation is described in Section 6. Conclusions and future work are presented in Section 7.

2 Open Services for Lifecycle Collaboration (OSLC)

Open Services for Lifecycle Collaboration (OSLC) (Open-services.net, 2008) is an open source initiative for creating a set of specifications that enables integration of development life cycle tools (e.g., modeling tools, change management tools, requirements management tools, quality management tools, configuration management tools). The goal of OSLC is to make it easier for tools to work together by specifying a minimum amount of protocol without standardizing the behavior of a specific tool.

The OSLC specifications use the Linked Data model to enable integration at the data level via links between tool artifacts defined as Resource Description Framework (RDF) (Manola and Miller, 2004) resources (beside other possible representations such as XML, JavaScript Object Notation (JSON) (json.org, 2016), Atom, and Turtle). The resources are identified by HTTP URIs. A common protocol to perform creation (HTTP POST) and retrieval (HTTP GET), update (HTTP PUT) and delete (HTTP DELETE) operations on resources is also specified.

3 Traceability Design and Architecture

The traceability design and architecture is mainly being developed in the INTO-CPS project (into-cps.au.dk, 2015) which contains a set of tasks. One of these is the design of traceability and model management with the following goals (Laudahl *et al.*, 2016):

- Checking the realization of requirements in models
- Enabling collaborative work by connecting artifacts and knowledge from different users
- Decreasing redundancy by connecting different tools to a single requirements source and allowing a system-wide view that is not only limited to single tools

The Provenance (PROV) (Moreau *et al.*, 2013) and OSLC standards presented in (Fitzgerald *et al.*, 2015) are used to support traceability activities. PROV is a set of documents built on the notation and relation of entities, activities, and agents.

The design and architecture of the traceability-related tools has recently been developed in (Laudahl *et al.*, 2016) and is shown in Figure 1. Any modeling tool written in any programming language can use these traceability standards to support the traceability

of activities performed within the tool and interact with other tools.

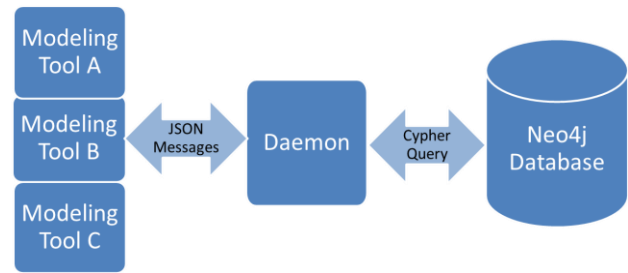


Figure 1. Schematic architecture of the traceability-related tools.

As depicted in Figure 1, the architecture is divided into three parts:

- **Modeling Tools** – The modeling tools send traceability information from activities that are performed within the tools (e.g., model creation, modification, import model description in XML) to the daemon.
- **Daemon** – The daemon provides an OSLC interface compliant with RESTful (Richardson and Ruby, 2007) to store the traceability information into the database and retrieve the traceability data from the database. It is launched and terminated by modeling tools.
- **Neo4j Graph Database** – The Neo4j database (Neo Technology, Inc, 2007) is a graph database to store the OSLC triples that make up the traceability data.

4 An Example of Integrated Tools for Cyber-Physical Model Development

OpenModelica has been successfully integrated with the INTO-CPS tool chain to trace artifacts created during the system development process from high level requirements to simulation results. The tools involved are Overture (Larsen *et al.*, 2010), 20-sim (Controllab Products B.V, 2013), Modelio (Favre, 2005) and RT-Tester (Verified Systems International GmbH, 2012). The tool chain as shown in Figure 2 is defined by the connections between the system architecture and the simulation via the model description XML file and the FMU.

The SysML Connection diagram defines the components of the system and their connections. The internals of these block instances are created in the various modeling tools and exported as FMUs. The modeling tools support importing the interface definition (ports) of the blocks in the Connection diagram by importing a modelDescription.xml file containing the block name and its interface definition linked with requirements. All tools are storing information in Git and sending information about existing and created artifacts to the global database.

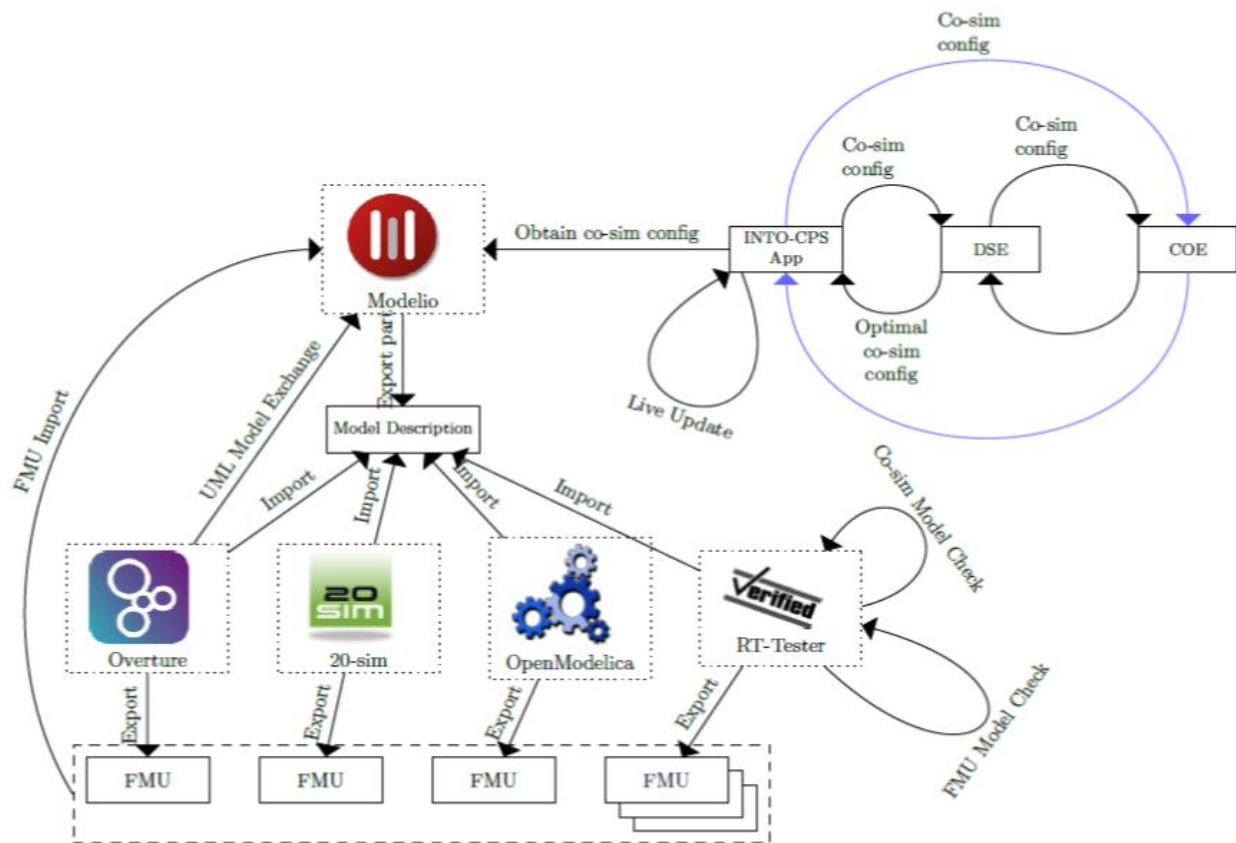


Figure 2. An Example of integrated tools to trace artifacts created during the system development process (Bandur *et al*, 2016).

5 Traceability and Model Management in OpenModelica

In the new work reported in this paper, OpenModelica has been extended with support of traceability in the OSLC sense, where traceability information is exchanged with external tools through a standardized interface and format. The implementation is based on an architecture and a common interface defined in (Lausdahl *et al*, 2016) for exchanging traceability information.

The modeling activities that can be recorded automatically and traced within OpenModelica are:

- Model description XML import (linked with requirements)
- Model creation
- Model modification
- Model destruction
- FMU export
- Simulation result creation

The complete workflow for traceability artefacts within OpenModelica and the different components that rely on are shown in Figure 3.

The following summarizes the main workflow that could be used to create and record traceability information in OpenModelica during cyber-physical model development process.

1. Commit model file entity to Git repository and record the Git-hash
2. Create URIs of the activity based on the Git-hash
3. OSLC triples describing the activity are generated using the URIs
4. OSLC triples are sent to the traceability Daemon
5. Retrieve the traceability information (traces to and traces from)

The traceability information is represented in JSON format. The modeling activities described by OSLC triples represented in JSON format are sent from OpenModelica to the daemon. These traces are then sent through the daemon to the Neo4j database, where they are stored. In order to view and analyze traceability data, this is later retrieved (traces to and traces from) from OpenModelica, through the appropriate queries from the daemon to the database.

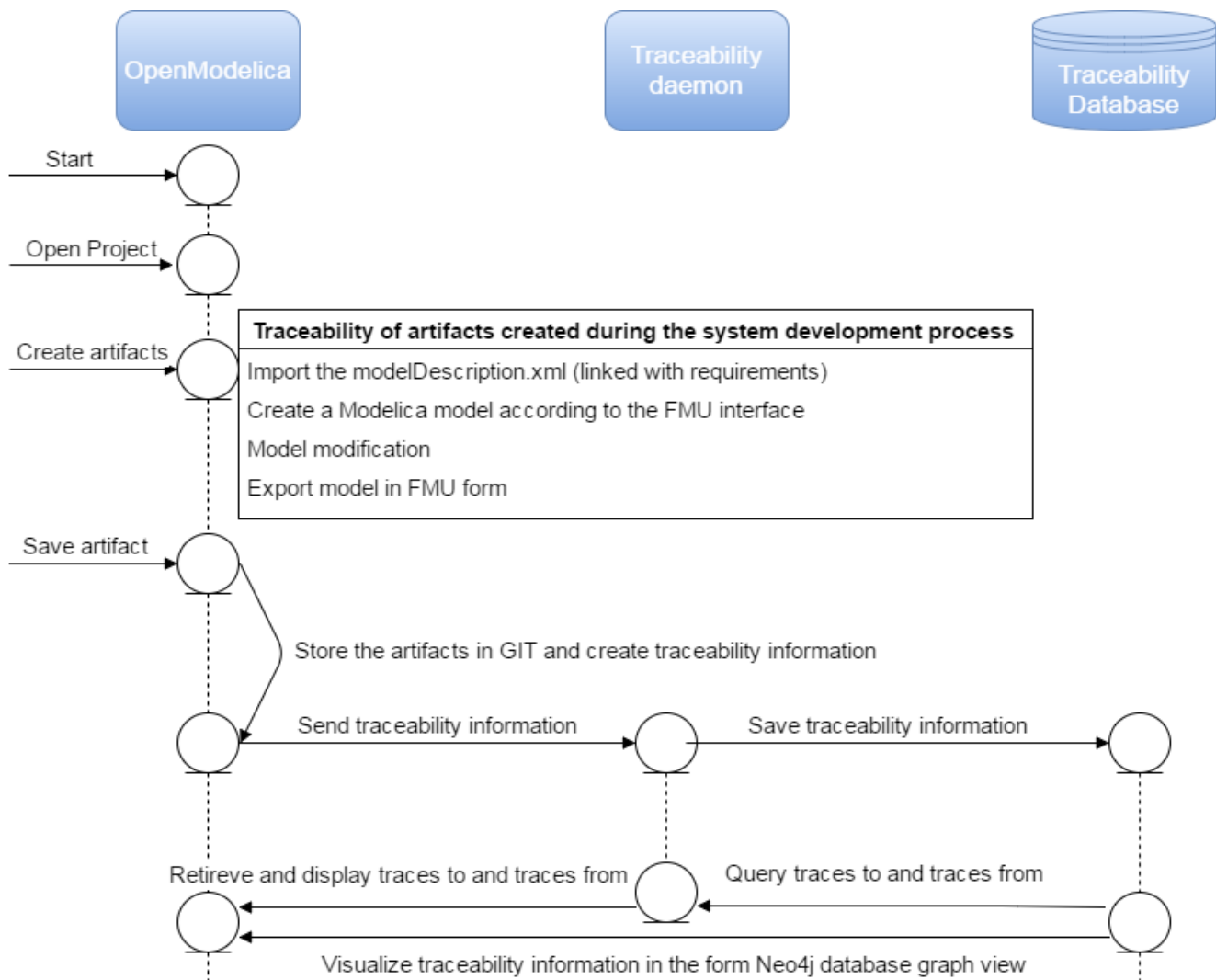


Figure 3. Workflow of traceability of artifacts during the system development process in OpenModelica.

6 Prototype Implementation

We have implemented a prototype to demonstrate the idea of exchanging traceability information for integrating lifecycle modeling tools using OSLC. The prototype is implemented based upon the design and architecture presented in Section 3.

As mentioned, the implementation of this prototype is an extension of OMEdit (Asghar *et al.*, 2010) which is implemented in C++ using the Qt Framework (Nokia Corporation, 2011) graphical user interface library. For presentation reasons, we have grouped the prototype functionality into three categories: importing model description XML, model management with Git integration, and traceability support using OSLC, which are described in the following subsections.

6.1 Import Model Description in XML

As a preparation for the extension to support tracing for importing modelDescription.xml interface files, we

extended OpenModelica to support importing modelDescription.xml (See Figure 4).

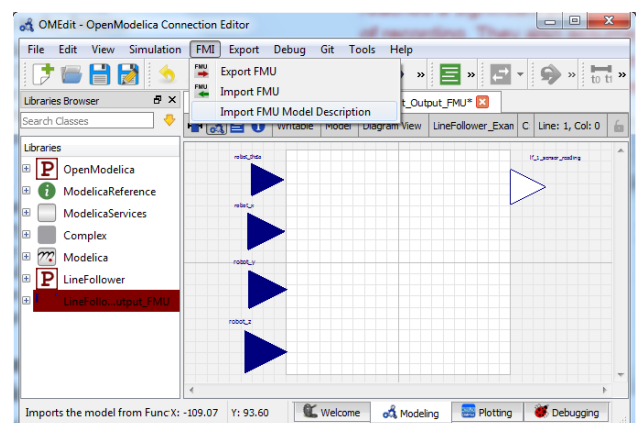


Figure 4. A screen shot of the model description XML import operation.

OpenModelica can import model description XML interface files (linked with requirements) created using other system architectural modeling tools and create

Modelica models from this information. The result is a generated file with a Modelica model stub containing the inputs and outputs specified in the model Description.xml file. Then the user can create a complete model using the GUI via drag and drop in the editor. Hence, the traceability chain within OpenModelica traces models linked with requirements through model description xml import, model creation, model modification, FMU export and simulation results.

6.2 Model Management with Git Integration

One of the objectives of the traceability tooling is to manage the development process in terms of modeling activities within the modeling tools. In order to achieve this objective access to the version control system is required in OpenModelica. Therefore the OpenModelica Connection Editor OMEdit has been enhanced to support Git version control as shown in Figure 5.

The OMEdit Git integration is currently in an early stage of development but already supports some basic functionality (See Figure 5) such as staging modified tracing operations on files for commit, committing, and reverting changes. It is useful to provide viewing of status and version history which can be used for creating the resource URIs for the modeling activities on each new commit.

The implemented prototype also allows to create a local Git repository by selecting Git -> Create New Repository from the menu bar. Since the URI, as presented in (Fitzgerald et al, 2015) is the combination of the Git-hash and the unique path for every file in the project, creating a Git repository for traceability purposes automatically adds a structure (See the left part of Figure 5) for models, simulation results, FMUs, and model description XML files to the Git repository.

6.3 Traceability Support in OpenModelica

The traceability support in OpenModelica provides a graphical user interface to interact with other lifecycle modeling tools.

As already mentioned in Section 4, OpenModelica supports traceability in the OSLC sense, where traceability information is exchanged with external tools through a standardized interface and format. The implementation is based on the architecture and a common interface defined in (Laudahl et al, 2016) for exchanging traceability information.

OpenModelica imports the modelDescription.xml and creates a Modelica model according to the FMU interface. The generated Modelica model is completed with behavior for the SysML block and the final model is exported in the FMU form. The generated FMU is then used in a whole system simulation connected according to the SysML connection diagram. The

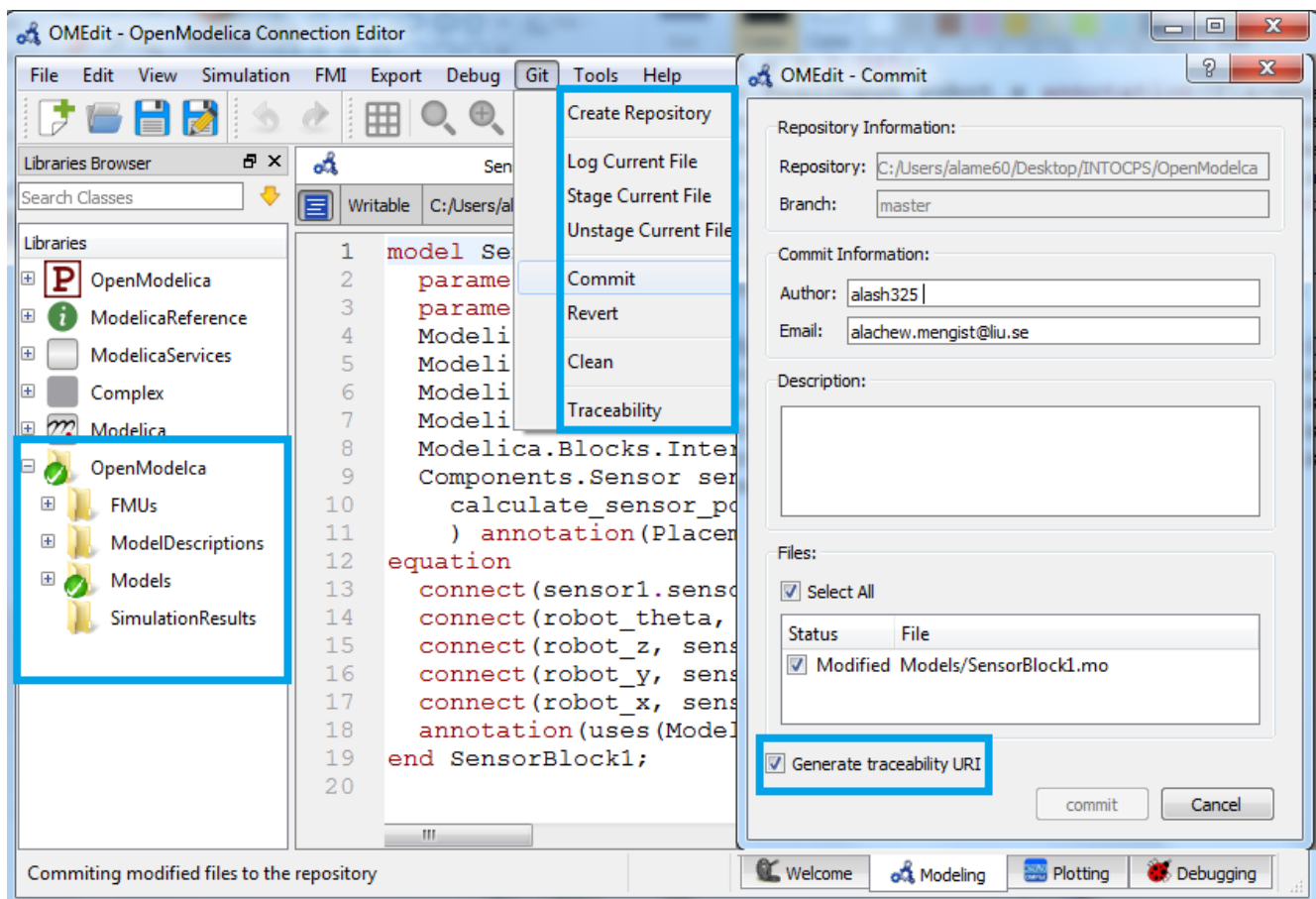


Figure 5. GUI of Git Integration in OpenModelica and functions available to create traceability URI.

FMU master simulation algorithm component performs the simulation via the INTO-CPS App. This whole chain is traced using OSLC.

We have designed a graphical user interface shown in Figure 6 which allows the user to record the traceability information and send to the Daemon (OSLC triples in JSON format), describing the activity using the URIs generated in the GUI shown in Figure 5. The PROV and OSLC relations that are mainly used in this work can be found in (Fitzgerald *et al*, 2015).

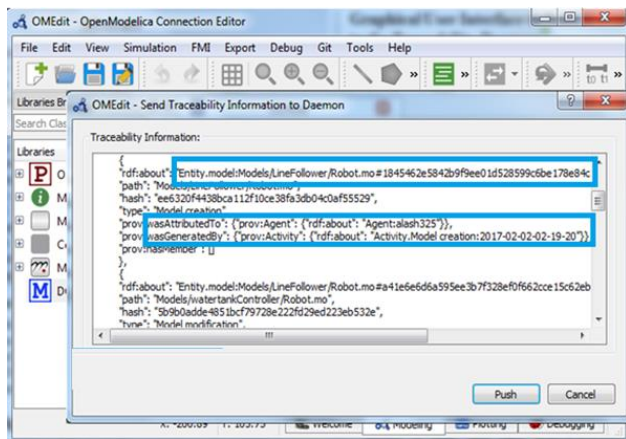


Figure 6. GUI to send traceability information to daemon.

These traces are then sent through the daemon to the database via HTTP POST <http://localhost:8080/traces/push/json>, where they are stored. Figure 7 shows an example of traceability information sent from

OpenModelica to the daemon and visualized in the Neo4j database.

Entities (e.g. *Modelica files*, *FMUs*, *modelDescription XML file*) are shown in green, actions (e.g. *model creation*, *FMU export*, *modelDescription XML import*) are shown in yellow, agents (e.g. users with the names "Alachew", "Adrian", "Peter", and "Adeel") are shown in blue, and their relationships "what come from what" and "what used what" (e.g. "wasGeneratedBy", "wasDerivedFrom", "usedTool") are shown with red arrows.

In order to view and analyze traceability data, we have also designed a graphical user interface shown in Figure 8 which allows the user to query traceability information (traces to and traces from) from the daemon to the database (via HTTP GET):

- <http://localhost:8080/traces/from/<URI>/json> and
- <http://localhost:8080/traces/to/<URI>/json>

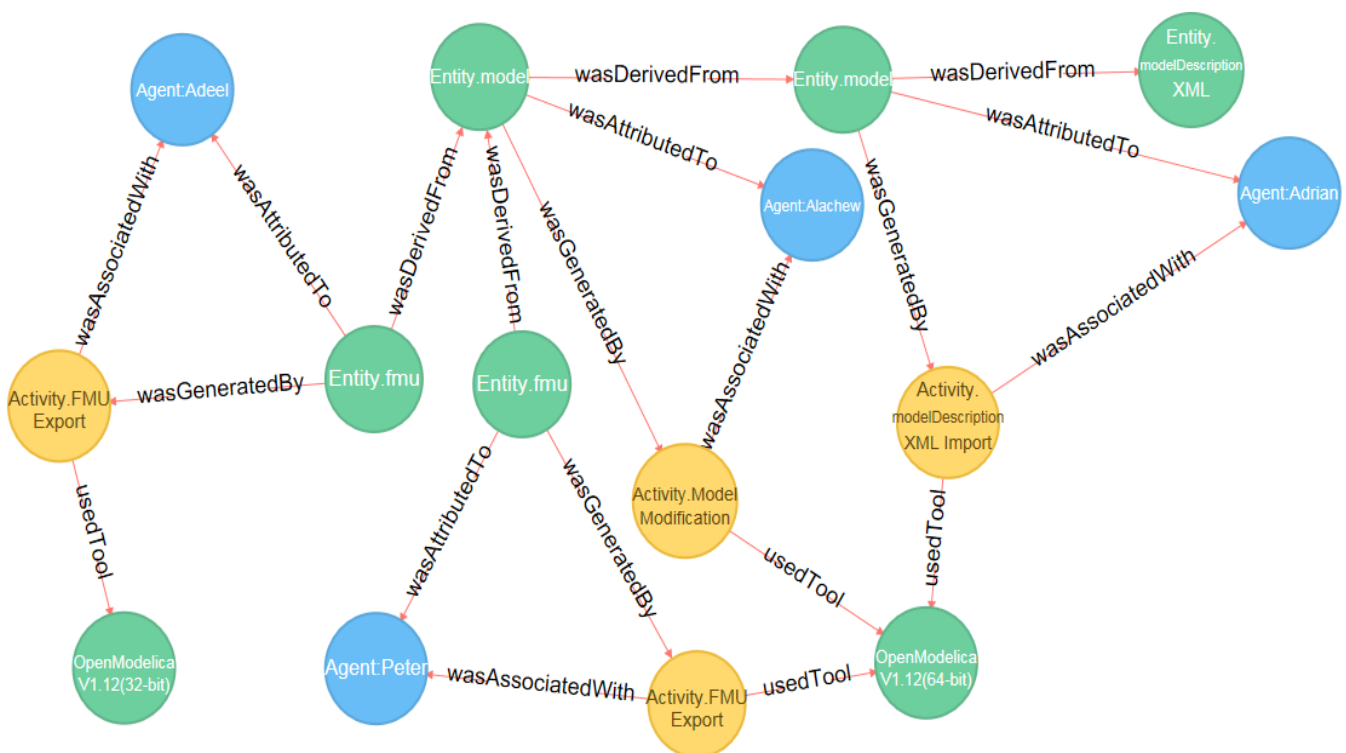


Figure 7. An example of traceability information sent from OpenModelica to the daemon and visualized in the Neo4j database.

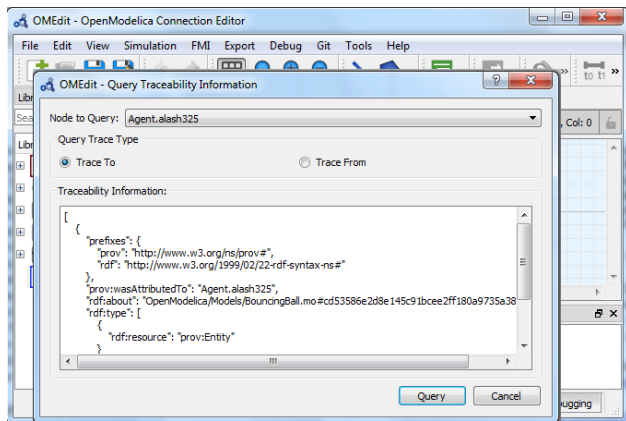


Figure 8. GUI to query traceability information (traces to and traces from) from Neo4j database.

7 Conclusions and Future Work

This paper has presented a framework for traceability and model management in OpenModelica, and its integration with the Git version control system.

The new version of OpenModelica supports traceability in the OSLC sense, where traceability information is exchanged with external tools through a standardized interface and format. The Modeling activities that can be recorded automatically within OpenModelica and traced are import of model description XML linked with requirements, creation of models, modification of models, destruction of Models, export of FMUs, and creation of simulation results.

A first prototype to query traceability information (traces to and traces from models or simulation results) from the database and display to end-users in JSON format is also complete. As future work, we also intend to extend the OpenModelica tool to support visualization and presentation of the traceability data viewed both in the form of graphs and trees.

The OpenModelica model management with Git integration is currently in an early stage of development but is already being able to support end-users to trace back all steps of the modeling process and to revert each step in the development history, and also model collaboration between end-users. Ongoing work is focused on having fully functional Git integration including showing two versions of the same model in parallel.

Future work also involves computing the impact of two different versions of the same model on simulation results and merging the models in way that the resulting model can be valid without modification.

Acknowledgments

This work has been supported by the European Union in the H2020 INTO-CPS project. Support from Vinnova in the ITEA3 OPENCPS project has been received. The OpenModelica development is supported by the Open Source Modelica Consortium. Special

thanks to Kenneth Lausdahl, Peter Niermann, Jos Höll, Carl Gamble, Oliver Möller, Etienne Brosse, Tom Bokhove, and Luis Diogo Couto for collaboration and valuable input to traceability related tools design.

References

- Adeel Asghar, Sonia Tariq, Mohsen Torabzadeh-Tari, Peter Fritzson, Adrian Pop, Martin Sjölund, Parham Vasaieli, and Wladimir Schamai. An Open Source Modelica Graphic Editor Integrated with Electronic Notebooks and Interactive Simulation. In *Proc. of the 8th International Modelica Conference 2011*, pp. 739–747. Modelica Association, March 2011. Linköping University, Sweden, 2010.
- Victor Bandur, Peter Gorm Larsen, Kenneth Lausdahl, Casper Thule, Anders Franz Terkelsen, Carl Gamble, Adrian Pop, Etienne Brosse, Jrg Brauer, Florian Lapschies, Marcel Groothuis, Christian Kleijn, and Luis Diogo Couto. INTO-CPS Tool Chain User Manual. Technical report, INTO-CPS Deliverable, D4.2a, December 2016.
- Torsten Blochwitz et al. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In *Proceedings of the 8th International Modelica Conference*, Dresden, Mar. 2011. doi: 10.3384/ecp11063105.
- Controllab Products B.V. Modelling and simulation software package for mechatronic systems <http://www.20sim.com/>, January 2013.
- Maged Elaasar and Adam Neal. Integrating Modeling Tools in the Development Lifecycle with OSLC: A Case Study, pages 154-169. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- Jean-Marie Favre. Foundations of Model (Driven) (Reverse) Engineering: Models – Episode I: Stories of The Fidus Papyrus and of The Solarus. In *Language Engineering for Model-Driven Software Development*, March 2005.
- John Fitzgerald, Carl Gamble, Richard Payne, and Ken Pierce. Methods Progress Report 1. Technical report, INTO-CPS Deliverable, D3.1b, December 2015.
- FMI-Standard.org (2014). Functional Mock-up Interface for ModelExchange and Co-Simulation Version 2.0. <https://www.fmi-standard.org/> (accessed: 10th of December 2016).
- Peter Fritzson. Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach. 1250 pages. ISBN 9781-118-859124, Wiley IEEE Press, 2014.
- Peter Fritzson, Peter Aronsson, Adrian Pop, Hakan Lundvall, Kaj Nyström, Levon Saldamli, David Broman, Anders Sandholm. OpenModelica – A Free Open-Source Environment for System Modeling, Simulation, and Teaching. *Proceedings of the 2006 IEEE Conference on Computer Aided Control System Design*, Munich, Germany, October 4–6, 2006.
- Tom Heath and Christian Bizer (2011) *Linked Data: Evolving the Web into a Global Data Space* (1st edition). Synthesis Lectures on the Semantic Web: Theory and Technology, 1:1, 1-136. Morgan & Claypool, 2011. doi: 10.2200/S00334ED1V01Y201102WBE001.

- into-cps.au.dk (2015). Integrated Tool Chain for Model-based Design of Cyber-Physical Systems. <http://into-cps.au.dk/> (accessed: 10th of December 2016).
- json.org. JavaScript Object Notation. <http://www.json.org/> (accessed: 10th of December 2016).
- Peter Gorm Larsen, Nick Battle, Miguel Ferreira, John Fitzgerald, Kenneth Lausdahl, and Marcel Verhoef. The Overture Initiative – Integrating Tools for VDM. SIGSOFT Softw. Eng. Notes, 35(1):1–6, January 2010.
- Kenneth Lausdahl, Peter Niermann, Jos Höll, Carl Gamble, Oliver Möller, Etienne Brosse, Tom Bokhove, Luis Diogo Couto, Adrian Pop, and Christian König. INTO-CPS Traceability Design. Technical report, INTO-CPS Deliverable, D4.2d, December 2016.
- Frank Manola and Eric Miller, editors (2004). RDF Primer. W3C Recommendation. World Wide Web Consortium. <https://www.w3.org/TR/2004/REC-rdf-primer-20040210/> (accessed: 10th of December 2016).
- Modelica Association (2012). Modelica: A Unified Object Oriented Language for Physical Systems Modeling, Language Specification version 3.3. <https://modelica.org/> (accessed: 10th of December 2016).
- Luc Moreau, Paolo Missier, James Cheney and Stian Soiland-Reyes, editors and contributors (2013): An Overview of the PROV Family of Documents. <https://www.w3.org/TR/prov-n/> (accessed: 10th of December 2016).
- Neo Technology, Inc (2007). Neo4j Database. <https://neo4j.com/> (accessed: 10th of December 2016).
- Nokia Corporation (2011). Qt Project. <https://www.qt.io/> (accessed: 10th of December 2016).
- Open-services.net (2008): Open Services for Lifecycle Collaboration – Lifecycle Integration Inspired by the Web. <http://open-services.net/> (accessed: 10th of December 2016).
- Adrian Pop, Martin Sjölund, Adeel Ashgar, Peter Fritzson, and Francesco Casella. Integrated Debugging of Modelica Models. Modeling, Identification and Control, 35(2):93–107, 2014.
- Leonard Richardson and Sam Ruby. RESTful Web Services (First ed.), O'Reilly, 2007.
- Verified Systems International GmbH, Bremen, Germany. RTTester Model-Based Test Case and Test Data Generator – RTTMBT: User Manual, 2015. <https://www.verified.de/products/model-based-testing/>, Doc. Id. Verified-INT-003-2012.

A Simulation Environment for Efficiently Mixing Signal Blocks and Modelica Components

Ramine Nikoukhah Masoud Najafi Fady Nassif

ALTAIR ENGINEERING, FRANCE, ramin@altair.com

Abstract

There exist several specialized tools that provide environments for the development and simulation of either pure Modelica models or pure signal based models. These environments have each their own advantages and flaws. *solidThinking Activate*TM has been developed to mix these domains and take advantage of both of these approaches to system modeling. This paper presents this mixed Signal-Modelica environment, and in particular the efforts and challenges faced in its development.

Keywords: *Modelica tool, Signal based tool, FMI*

1 Introduction

The Modelica[®] language¹ and tools are successfully used for modeling physical systems in industrial applications. This success is primarily due to the ability of Modelica to express mathematical equations corresponding to physical phenomena in a natural way (Modelica Association; Peter Fritzson).

For modeling complete systems, for example systems including controllers, Modelica provides other features that makes it go beyond a declarative language for expressing equations. Data types other than reals, algorithm sections, Matlab-like matrix operations are introduced to dispense of the use of other tools, in particular Matlab[®] and Simulink[®] for handling models with control components. Yet, still in many applications, the design process requires using Modelica to model the physical plant and exporting the model in the Matlab/Simulink environment for controller design. The reason for this is in part the limitations of the Modelica language, which is not well suited for creating block diagrams, such as the ones used in control applications, for which specialized tools such as Simulink, Scicos (Campbell et al., 2010), and *solidThinking Activate*TM have been developed.

In an attempt to provide an environment for modeling efficiently both blocks and physical components, in 2002 Modelica was introduced in the Scicos environment in the framework of the publicly funded project RNTL (Réseau National des Technologies Logicielles) **Simpa** (Simulation pour le Process et l'Automatique). This Scicos extension (Najafi et al., 2004, 2005a,b; Nikoukhah, 2006; Nikoukhah and Furic, 2009) allowed Scicos users to

mix both standard Scicos blocks and Modelica components in the same environment. A similar extension was later introduced in Simulink with the introduction of the *Simscape*TM language (Simscape).

Scicos/Modelica environment based on the Modelicac compiler (Furic, 2007) provides a versatile modeling environment, especially thanks to the Coselica library². Even though this extension allows Scicos users to use some Modelica components in the construction of their models, it has many limitations. For example Modelica libraries cannot be automatically imported and used in Scicos.

Activate is a professional simulation tool developed by Altair Engineering based on the open source academic simulation software Scicos. As such, it inherits many of Scicos features including the close integration with a matrix-based scripting and programming language. In Activate, the HyperMath Language (HML) has replaced Scilab³ and NSP⁴. And for the Modelica extension, Scicos Modelicac has been replaced with the MapleSimTM compiler developed by Maplesoft⁵ in Activate.

Activate and Scicos both use the same mechanism to integrate Modelica: at compile time, they aggregate Modelica components and create a Modelica program which is then processed by the Modelica compiler providing the C code corresponding to the simulating function of a block replacing these Modelica components in the original model⁶. The Activate environment however provides specific features that has allowed taking the Modelica integration beyond what is available today in Scicos. This paper presents this new modeling environment.

2 Motivations

It is widely agreed upon that for many applications Modelica today does not provide a viable alternative to block-based modeling tools such as Simulink, Scicos and Activate. The limitations imposed by the language make it difficult to provide the types of blocks that are needed

²<http://www.kybrdr.de/software>.

³<http://www.scilab.org>

⁴<https://cermics.enpc.fr/~jpc/nsp-tiddly>

⁵<http://www.maplesoft.com>

⁶A noteworthy difference is that in Scicos this simulation function represents a DAE (Differential Algebraic Equations) forcing Scicos to use a DAE solver, whereas in Activate the simulation function is provided as a model-exchange FMU representing ODEs. This difference however is not relevant to the presentation here.

¹<http://www.modelica.org>.

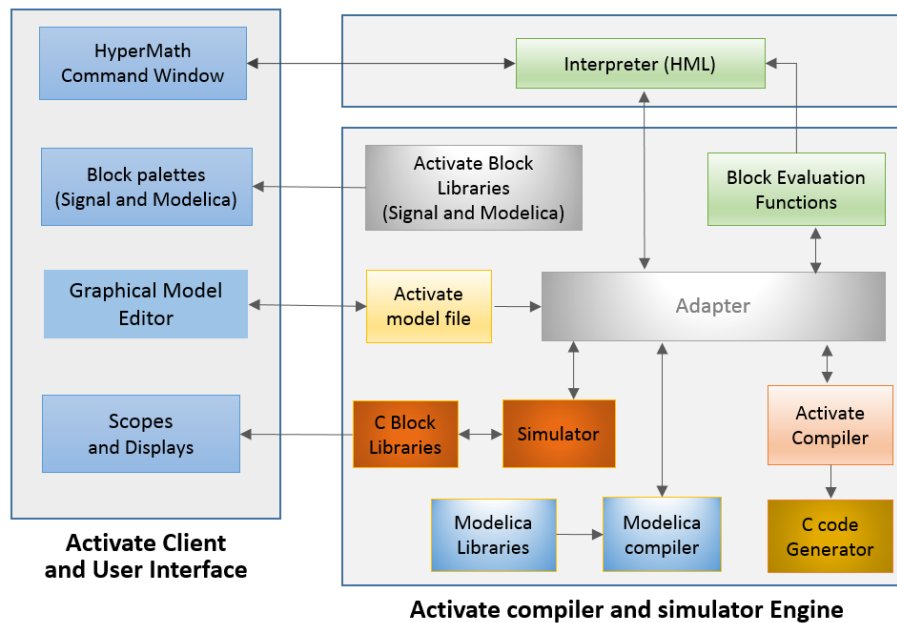


Figure 1. Different modules in Activate and their interactions.

to model control systems. For example creating a simple multiplexer block capable of concatenating a variable number of vectors and scalars of different data types is complicated in Modelica. Same is true for the summation block and many other basic mathematical operations (Elmqvist et al., 2016).

Other limitations come from the lack of a powerful supporting math environment. The computation of model parameters, post processing of the simulation results, etc., require access to math and engineering libraries, which could in theory be developed or interfaced in Modelica, but would require an enormous and lasting effort. In short it would amount to developing alternatives to Matlab, Scilab, HML, or Nsp, including their specialized toolboxes in control, signal processing, communication, optimization, etc. Some Modelica tools already use other languages, for example Maple and Python, for such support.

A reasonable solution to this problem is to base the simulation environment on a “User Language”, preferably a matrix-based mathematical language such as Scilab, Matlab, Nsp, Octave, HML, or even on non-matrix based languages such as Python and LUA. The key point is to give users the ability to interact with the simulation model through this language for anything from block/component creation, model construction, parameterization, compilation, code generation and simulation to data collection, post processing, optimization, and more. The Scicos environment was developed in this spirit with Scilab as User Language. Matlab is the User Language for Simulink and Simscape.

A very interesting effort in this direction is undertaken in (Elmqvist et al., 2016), where a complete re-

implementation of Modelica is considered with Julia⁷ as User Language. This undertaking is very ambitious in that the Underlying Language is also used for defining the dynamics of blocks and components. The Activate/Modelica environment presented here is developed with this consideration in mind and follows the spirit of Scicos but uses HML as the Underlying Language. It does not go as far as defining dynamics of blocks in HML (except for embedded code generation purposes (Chancelier and Nikoukhah, 2015)); but rather it makes a clear distinction between the block/model creation and compilation, and runtime simulation. Model creation, evaluation and compilation, and in general anything that can be done before the start and after the end of runtime simulation are based strongly on the User Language. On the other hand the block dynamics need not be based on the User Language. The “standard” (Signal) Activate blocks have in general their runtime simulation functions expressed in C, and the equations of Activate physical components are expressed in Modelica.

This approach allows the Activate/Modelica environment to take advantage of existing technologies: Activate (synchronous semantics, block libraries, compiler, Simulink import (Weis, 2015) facility) and Modelica (existing Modelica compilers, in particular the MapleSim compiler, and existing Modelica libraries such as MSL).

3 Activate/Modelica environment features

Activate is not a Modelica tool per se; it cannot be used conveniently to build Modelica libraries. Its objective is to propose a unique harmonious environment to allow mixing regular Activate blocks and Modelica components in a

⁷<http://julialang.org>.

same model. The user interface and behavior of Modelica blocks and regular Activate blocks are designed to be as similar as possible without being too different from user interface of other Modelica tools. The Modelica components are seen as regular Activate block in this environment.

3.1 Modularity

A key architectural element in Activate is modularity. The diagram in Figure 1 shows different modules that constitute Activate. There are three main modules, the graphical user interface, the interpreter language, and the Activate engine. The graphical user interface and the interpreter can be replaced with similar modules fairly easily. For example the HML interpreter can be replaced with another interpreter, and alternative user graphical interfaces (for example javascript based tools) can be considered. The other module that can easily be replaced is the Modelica compiler. Currently the MapleSim compiler is used. In Scicos, Modelicac was used. Other compilers may be considered in the future.

The modularity between the engine and the graphical user interface is enforced by the usage of file based exchanges. The model, once edited is saved in an XML format and the engine uses this file to proceed with the compilation and simulation. The modularity of the interpreter is guaranteed through the specification of a set of APIs for the exchange with the graphical user interface and the engine.

3.2 Double layer implementation

In the Activate environment, a model is constructed using blocks. The compiler however does not operate on these blocks; it interacts with Atomic Units (AU). In many cases a block is associated with a single AU, but not always: a block may produce a network of AUs. The AU or AUs produced by a block may depend on the values of the block parameters. Specifically, the choice of the AU(s), their parameters, and the topology of the network is specified by an HML function associated with the block based on the values of the block parameters.

The ability to programmatically instantiate an AU or a network of AU(s) is an elementary feature in Activate but provides a particularly useful functionality in the context of Modelica components, as it will be described later.

Atomic unit (AU)

An AU may be presented as a "basic" block, but this would be misleading. An AU has ports that are connected to links, just like a block. It has parameters, like a block, but these parameters are not in general the block parameters. Consider for example the Activate block that implements a transfer function. The block parameters are the numerator and the denominator coefficients of the transfer function. The AU associated with this block operates in time domain and implements the dynamics based on the state-space realization of the transfer function. The parameters

of the AU in this case are the A , B , C , D matrices, which are computed by the HML function associated with the block.

In general an AU is a computational unit providing APIs to be used by the simulator. The APIs are C functions that are called by the simulator at different stages of the simulation: computation of the output, of the state derivative, of the next discrete state, etc. But the AUs can also be Modelica components. An AU may also be virtual.

The creation of AUs from Activates blocks based on a User Language script is a process that does not have an equivalent in standard Modelica or in Simulink (S-Functions). This process, which provides a clear separation between the model at the graphical layer and at the compiler layer, has been first implemented in Scicos.

3.3 Modelica components

In Activate, Modelica components are Activate blocks and treated as such in the graphical editor. They are also treated similarly at the evaluation phase, prior to compilation. This means that certain properties of Modelica components that are coded as annotations are handled by the corresponding Activate XML file and HML evaluation script. These properties include in particular the graphical properties and the parameter descriptions. When a Modelica library is imported into Activate, these component annotations are used to create the Activate blocks. These annotations are never directly used in Activate.

So, having the Modelica component as an Activate block means that all graphical features, parameter definitions, code instantiations, ..., are done in the usual Activate way. The use of Activate block to instantiate the Modelica components provides facilities that allows for example the creation of components with variable number of ports or different data types based on block parameters. The Activate block is thus a lot more versatile than a standard Modelica component; even the internal Modelica code of the block/component can be customized. At the extreme case, the Modelica code itself could become a block parameter.

On the graphical editor, the visible difference between a regular Activate block and a Modelica Activate block is that the latter has special (implicit) ports. No connections can be made between these ports and other Activate port types. Two special interface blocks are used to interface the Modelica world with the regular Activate world. One has an implicit input port and a regular output port and the other, the opposite (see Figure 2). Such connections are meaningful only if the the connection on the Modelica side is of type Modelica Signal.



Figure 2. Special blocks for Modelica-Activate world interface

Importing Modelica libraries

The import of a Modelica library is done by the MapleSim compiler, which creates HML scripts, the execution of which create the corresponding Activate library. An Activate library is a collection of XML files, HML functions, image icons and palettes. The MapleSim compiler also uses the definition of component icons described in Modelica language to generate image files (svg format) to be used by Activate as block icons. Certain features such as dynamical icons (icons changing during simulation) are not supported.

Currently, most but not all MSL (Modelica Standard Library) blocks are imported and integrated in Activate palettes.

3.4 Model compilation

Compiling a model consists of producing a structure to be used by the simulator. This structure contains all the information needed by the simulator that can be computed before the start of the simulation. It contains in particular type and size information, and scheduling tables specifying the condition and the order in which AU computational functions are to be called during simulation. The same structure is used for code generation.

Model evaluation

The evaluation is the first phase of model compilation. In this phase, the model parameters are evaluated and the HML function associated with the blocks are executed producing the network of AUs associated with the model. Note that this network of AUs, which retains a hierarchical structure, does not in general present a one to one correspondence with the original block diagram model.

At the end of model evaluation phase, all model and block scripts and parameters are removed. They are used in this phase to construct the AUs and evaluate the numerical values of their parameters. They are not available or needed for the rest of the compilation process, which deals exclusively with the network of AUs.

Model flattening

Model flattening is the second phase of the compilation. The hierarchical network of AUs produced by the model evaluation phase is converted into a flat network of computational units. All virtual AUs are removed and all Modelica AUs have been replaced with computational AUs (in particular derived from an FMU produced by the Modelica compiler).

A simple example is provided in Figure 3. This model contains an electrical circuit, modeled for the most part using Modelica components. The regular Activate blocks are the sine wave generator and the Scope. There are three interfacing blocks connecting the Activate environment to the Modelica environment.

The Modelica part is aggregated into a single block as shown in Figure 4. This step is of course fully transparent to the user and is presented here as an illustration of the

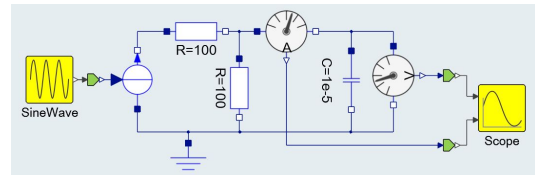


Figure 3. Simple Activate diagram containing Modelica components.

way the mechanism operates. The newly created block has one input and two outputs, as expected.



Figure 4. Equivalent Activate model after aggregation of Modelica components.

The Modelica code corresponding to the Modelica part is generated automatically by Activate and sent to the Modelica compiler for compilation. The Modelica compiler then generates a corresponding FMU, which replaces the Modelica part as shown in Figure 5. This step is of course again transparent to the user and is presented here as an illustration

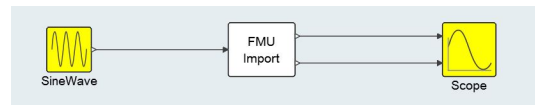


Figure 5. Resulting regular Activate model with no Modelica components.

Back-end compiler

In this phase, which consists of computing the scheduling tables for the simulator, the structure contains no trace of the Modelica components; they have been replaced with computational AUs in the previous phase. So the introduction of the Modelica extension does not affect this phase.

4 Modelica integration through FMI

The way Activate handles the Modelica components is by grouping them into a single Modelica model with inputs and outputs that are clearly specified by special interfacing blocks, as presented in the previous section. In the Modelica code generated by the Activate compiler, the interfacing blocks (shown in Figure 2) are instantiated as

```
Modelica.Blocks.Interfaces.RealInput
Modelica.Blocks.Interfaces.RealOutput.
```

The Modelica model is then compiled by the Modelica compiler, which in turn generates a code executable in Activate. This code is then imported in the Activate model as an FMU to replace the Modelica part. The FMI has been

chosen as the exchange format because it is a standard already supported both by Activate and MapleSim.

The FMI format is a rich interface format and quite compatible with Activate and Modelica. There are however a few shortcomings that need to be considered. Some challenges encountered in the usage of FMI standard in this context is discussed in this section.

4.1 Choice of the FMI type: Model-Exchange or Co-Simulation

The Modelica part of the Activate model is converted into an FMU and imported as a regular Activate block. In the exported FMU, both Model-Exchange and CoSimulation implementations are available. Using the model-exchange implementation allows taking advantage of different numerical solvers of Activate. The co-simulation implementation is useful for complex models where different parts of the model are needed to be simulated separately or even in parallel. Currently only the model-exchange implementation is used in Activate.

4.2 FMI import preserving full output/input dependency property

A challenge in importing the FMI generated from the Modelica code (or more generally any FMI) in Activate is the treatment of output/input dependencies. In the Activate block (or more specifically its AU) output/input dependencies are expressed as a vector of dependencies specifying which inputs affect any of the outputs. So the dependency is solely a property of an input port. The reason is that an AU computes all of its outputs in the same call, so all its dependent inputs must be up to date when the call is made. An FMU on the other hand specifies output/input dependencies as a matrix specifying which output depends on which known variables including individual inputs. The FMU provides routines that allow the computation of output ports separately and take advantage of variable caching.

A way to deal with this situation, which is the way the Modelica extension is implemented in Scicos, is to simply project the matrix of dependencies into a vector. This conservative approach properly assigns dependencies in Activate but "loses" information along the way. This may lead in particular to detection of algebraic loops by the Activate compiler that are not true algebraic loops (artificial algebraic loops). Even though there are ways to break algebraic loops in an Activate model, it is not the best way to deal with this situation. A very simple example that illustrates this problem is shown in Figure 6.

After compiling the Modelica part, a model similar to what is shown in Figure 7 is obtained in Activate. In the generated FMU, there is a direct dependency between the `SignalCurrent` input port (`in`) and the `CurrentSensor` output signal (`A`). The dependency is depicted by a red dashed line in Figure 7. If the dependency matrix is projected into a vector, both the output ports `A` and `V` are considered depend on the input port `in`, which results in

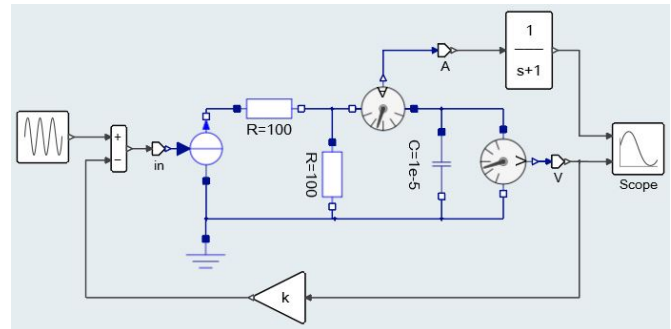


Figure 6. A simple model mixing Modelica and Activate blocks

an artificial algebraic loop.

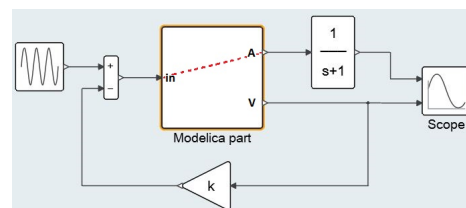


Figure 7. The model in Figure 6, after converting the Modelica part into an FMU block.

There is no solution to this problem as long as the FMU block implements a single AU. But as it was stated previously, Activate blocks can implement a network of AUs, the topology of which can depend on block parameters. It turns out that the matrix output/input dependency can be properly implemented by a properly constructed network of AUs to implement the FMU.

In this case the block parameters are provided by the FMU XML file. By reading and parsing the XML inside the FMU, the block generates a network of AUs, as shown for example in Figure 8 in the case of a 2 input 4 output FMU block. The network contains a central AU, always present, and an AU associated with each output port. The input dependency associated with an output is specified in the AU associated with that output. In this particular example it can be seen that the first output depends on both inputs whereas the second output has no input dependency, the third output depends only on the first input and the last output depends on the second input.

The central AU includes the simulation APIs for state derivative computation and discrete state updates and does not have any input dependency. All the AUs in the network use the same internal structure, which is instantiated by the central AU. The central AU provides a pointer to this structure to the other AUs through its output port.

In the case of the model in Figure 6, the network of AUs is generated as in Figure 9. By using this network to replace the FMU block, the resulting Activate model contains no algebraic loop.

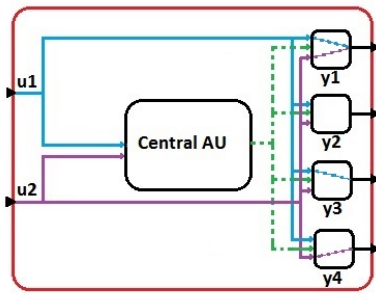


Figure 8. Automatically generated network of AUs from FMU import for an FMU with two input and four output ports.

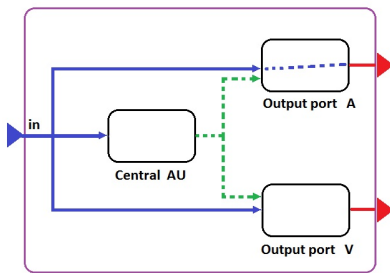


Figure 9. The network of AUs corresponding to the example in Figure 6.

4.3 DAE support and constraints on states

The current FMI standard is powerful enough to be used for implementing the Modelica extension in Activate for many situations but some extensions would be particularly useful.

Compiling complex Modelica models, in particular mechanical models, very often results in high index DAEs or sometime ODEs and DAEs with constraints. Keeping the constraints valid is important to avoid drift in the solution. In the current FMI specification, only ODEs are supported. Activate currently supports both DAEs, and ODEs with constraints. But these solvers cannot be used for the Modelica extension since the FMI does not support DAEs and ODEs with constraints.

The DAE support is currently being considered for FMI. ODEs with constraints, should also be considered. If it is known that an ODE $\dot{x} = f(x)$ satisfies a constraint $C(x) = 0$, information that could be available in various scenarios, then the solver should take advantage of this information to reduce drift in the solution. The constraint information may be provided as a residual function returning the constraint value, i.e., $C(x)$, or as a projection function such as $J^T(JJ^T)^{-1}$ where $J = \frac{\partial C}{\partial x}$.

This FMI extension can be done in several ways. One way would be to add one of these APIs to FMI interface:

```
fmi2Projection(fmiComponent c, double *J)
fmi2Constraint(fmiComponent c, double *C)
```

If the second API is used, then the number of constraints should also be declared as an attribute in the XML

file inside the FMU.

Another way is to add a new function to the set of FMI APIs in order to bring back the solution on the constraint after each completed integration step.

```
fmi2ApplyProjection(fmiComponent c)
```

This function would apply a near-minimal projection to the continuous states in the model. This is often done via a Newton-based method, and terminates when it achieves the desired precision. This method can be applied on single-step solvers where memory of the past solution is not used. It will be necessary to call `fmi2GetContinuousStates` after the projection to obtain the continuous states satisfying the solution. Having this as a separate function allows the simulator to choose when it is applied (e.g. at the end of an integration step, internal to the step, after events, etc.).

A third way, which does not require adding a new API, a projection is implicitly applied when `fmi2CompletedIntegratorStep` is called by the simulator. This solution would work only with single-step solvers. No error tolerance control can be used on the constraints in that case.

4.4 Handling input derivatives

Consider the simple example shown in Figure 10. In this model the derivative of the input is required.

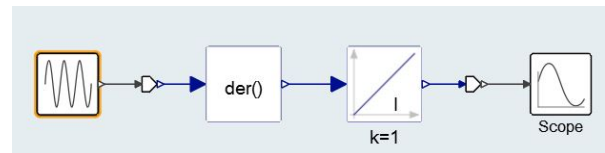


Figure 10. model requiring the derivative of inputs

When the time derivative of an input is required, the derivative can be computed numerically inside the FMU, but this does not always work for variable-step size solvers since the derivative value is not necessarily stable as the integrator step-size changes. Furthermore, at initial step or just after an event that changes the internal model configuration, no derivative can be computed. If there are constraints that depend on these derivatives, the integration step rapidly reduces to zero, stalling the simulation. In FMI for CoSimulation, the derivative of inputs can be provided via the API `fmi2SetRealInputDerivatives`, but nothing is available for ModelExchange FMI. The only robust alternative currently is to add an extra input port to provide the derivative of input from the environment, if available.

4.5 Using the Jacobian of the FMU

The numerical solvers often need the Jacobian of the model for numerical integration. The Jacobian can either be provided analytically or computed numerically. In

complex models providing the analytical Jacobian is crucial for obtaining reliable results. The FMU block⁸ may provide directional derivatives of its state derivatives and outputs with respect to its states and inputs. These directional derivatives can be used to compute the equivalent linear model of the block. If an FMU block with nonlinear dynamics defined as (1) provides its directional derivatives

$$\begin{cases} \dot{x} &= f(x, u) \\ y &= g(x, u), \end{cases} \quad (1)$$

The matrices (A, B, C, D) as defined in (2) can be obtained by repeated calls to `fmi2GetDirectionalDerivative` function in FMI.

$$\begin{aligned} A &= \frac{\partial f}{\partial x} & B &= \frac{\partial f}{\partial u} \\ C &= \frac{\partial g}{\partial x} & D &= \frac{\partial g}{\partial u} \end{aligned} \quad (2)$$

The (A, B, C, D) matrices are equivalent linear system of the FMU block. The numerical solver, on the other hand, needs the complete Jacobian of the entire model which may be composed of other FMU blocks and other regular Activate blocks. In order to obtain the complete Jacobian of the model, Activate offers the following solutions.

- Computing a pure numerical Jacobian, i.e., ignoring the local analytical linear system of blocks and compute the complete Jacobian of the model using the numerical differentiation method. This method usually works fine and it is fairly fast, but may fail for complex stiff models.
- Mixing numerical and analytical Jacobian. In many cases, the highly nonlinear part of the Jacobian of the model is present in matrix A of the block. The analytically obtained matrix A of blocks may be used to populate the Jacobian matrix of the model, then the rest of the Jacobian matrix can be filled numerically. This method works fine, and is the default method in Activate.
- Fully analytical method. This method which is more complex than other two methods is useful if all blocks provide their analytical equivalent linear system matrices (A, B, C, D) . Since this method does not require calling the $f(x, u)$ and $g(x, u)$ function in (1), it is useful when calling these functions is expensive.

5 Challenges

Activate is not a Modelica tool and cannot provide the same Modelica functionalities as do pure Modelica tools such as Dymola or OpenModelica. Modelica is an extension for the modeling and simulation environment Activate. Efforts have been made to provide a user-friendly interface both for native Activate users as well as Modelica component users in this environment. There are currently a number of limitations in this extension.

⁸Only FMI-2.0 blocks provide directional derivative.

Modelica expressions, records and functions

The parameters of Modelica components present in an Activate models follow the scoping rules of Activate. So the records and functions used in the definition of parameters in Modelica are not always consistent with the way Activate handles parameters. This creates a complex problem for importing Modelica components. A translator of expressions is being developed to deal with this issue. For importing models, the records should be converted into HML scripts to be placed in Activate diagram contexts. This is a complex task, in general, but solutions have been found in special cases.

Initial equations

Initial equations in Modelica are global information that are not related to a specific component. Adding such information, even in specialized Modelica tools, cannot be easily done in the user interface and must be added textually. Since Activate does not provide a textual interface, the addition of initial equations currently is not possible. Various solutions are being considered but for the moment Activate does not allow the definition of initial equations in models. Initial equations in library components are of course handled by the compiler as usual.

6 Conclusion

Activate provides a complete environment for modeling systems with both physical components and signal based control parts where the physical components are modeled in Modelica. The integration of Activate and Modelica is done by respecting the semantics of the two languages. But there remain issues for going towards full Modelica support. This paper has presented the Modelica extension in Activate and the issues that remain open.

References

- Stephen L. Campbell, Jean-Philippe Chancelier, and Ramine Nikoukhah. *Modeling and Simulation in Scilab/Scicos with ScicosLab 4.4*. Springer-Verlag New York, 2010. ISBN 978-1-4419-5526-5.
- Jean-Philippe Chancelier and Ramine Nikoukhah. A novel code generation methodology for block diagram modeler and simulators scicos and VSS. *CoRR*, abs/1510.02789, 2015. URL <http://arxiv.org/abs/1510.02789>.
- Hilding Elmqvist, Toivo Henningson, and Martin Otter. Systems modeling and programming in a unified environment based on julia. In *Proceedings of the ISO/FA 2016 - 7TH International Symposium On Leveraging Applications of formal methods, verification and validation; 2016*, pages 198–217, 2016.
- Sébastien Furic. Using modelica under scilab/scicos, 2007. URL <http://www.scicos.org/ScicosModelica/Formation/Documentation/IntroductiontoModelica.pdf>.

Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Wiley, 2014. ISBN 9781-118-859124.

Modelica Association. The Modelica Language Specification, Version 3.3 Revision 1, 2014. URL <https://www.modelica.org/documents/ModelicaSpec33Revision1.pdf>.

Masoud Najafi, Azzedine Azil, and Ramine Nikoukhah. Extending scicos from system to component level simulation. In *Proceedings of the ESMc2004 international Conference*, Paris; France; October, 2004, 2004.

Masoud Najafi, Sébastien Furic, and Ramine Nikoukhah. Scicos: a general purpose modeling and simulation environment. In *Proceedings of the 4th International Modelica Conference*, Hamburg; 2005, 2005a.

Masoud Najafi, Ramine Nikoukhah, Serge Steer, and Sébastien Furic. New features and new challenges in modeling and simulation in scicos. In *Proceedings of the IEEE conference on control application*, Toronto; Canada; August, 2005, 2005b.

Ramine Nikoukhah. Challenges in integrating modelica in the hybrid system formalism scicos. In Claude Gomez Shi Li, Long-Hua Ma, editor, *The Oxford Handbook of Innovation*. Tsinghua University Press, Beijing, 2006.

Ramine Nikoukhah and Sébastien Furic. Towards a full integration of modelica models in the scicos environment. In *Proceedings of the 7th International Modelica Conference*, Como; Italy; 20-22 September 2009, pages 641–645, 2009.

Simscape. Physical systems simulation. URL <https://www.mathworks.com/products/simscape.html>.

Pierre Weis. Simport: A simulink model importer for scicos. In *Proceedings of The 3rd International Workshop on Simulation at the System Level for Industrial Applications*, Ecole Normale Supérieure de Cachan, France, October, 2015.

Component Development for Nuclear Hybrid Energy Systems

M. Scott Greenwood¹

¹Oak Ridge National Laboratory, USA, greenwoodms@ornl.gov

Abstract

A Nuclear Hybrid Energy System (NHES) uses a nuclear reactor as the basic power generation unit. The power generated is then used by multiple customers as either thermal power, electrical power, or both. The definition and architecture of an NHES can be adapted based on the needs and opportunities of a given local market. For example, locations in need of potable water may be best served by coupling a desalination plant to the NHES. Similarly, a location near an oil refinery may have a need for emission-free hydrogen production. Using the flexible, multi-domain capabilities of Modelica, Argonne National Laboratory, Idaho National Laboratory (INL), and Oak Ridge National Laboratory (ORNL) are investigating the dynamics (e.g., thermal hydraulics and electrical generation/consumption) and cost of such a hybrid system. This paper examines ongoing NHES work including the modeling organizational layout, highlighting a few subsystems, describing some of the component development and providing results from a study of multi-dimensional conduction model development.

Keywords: *thermal hydraulic, nuclear, economics, hybrid systems*

1 Introduction

Electricity markets in the United States are undergoing significant shifts in the traditional market structure. Factors such as mandates for renewable energy, overall carbon reduction, and the emergence of cheap natural gas have strained the profitability of primary baseload electricity suppliers, including nuclear power plants.

As the typical nuclear power generating station traditionally has only one customer—the grid—diversification of the customer portfolio in an integrated or hybrid manner may be advantageous. A representative NHES is depicted in Figure 1.

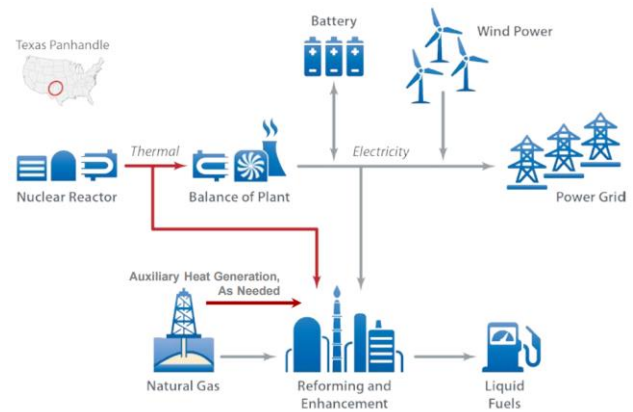


Figure 1. A representative NHES demonstrating a possible coupling scenario of both thermal and electrical energy with additional systems (e.g., an industrial process and energy storage system) (Bragg-Sitton et al. 2015).

A hybrid energy system approach, coupling base load energy suppliers and energy customers (thermal and/or electric), may be profitable and preferred in future energy markets. Possible scenarios include producing products that are more profitable than electricity or mitigating the possible load-following need—and subsequent cost increases—that significant renewable penetration may impose on nuclear power plants. For example, Figure 2 is a representative summary of the Electric Power Research Institute's (EPRI) recent study on the impact of renewable energy generation on grid variability (EPRI, 2015). Given current economic and political trends, future electrical grids will require highly variable operations that impose significant technical and economic challenges for power producers. Introducing hybrid energy systems may help create a path to achieving highly variable markets that are economically sound and do not compromise grid reliability.

This paper presents background information on the methodology being developed to evaluate the economic merit of an NHES, with a focus on the development of dynamic multiphysics models in Modelica that play a key role in the economic evaluation. Additional information beyond the scope of this paper can be found in ORNL, 2016a, ORNL, 2016b, and ORNL, 2017.

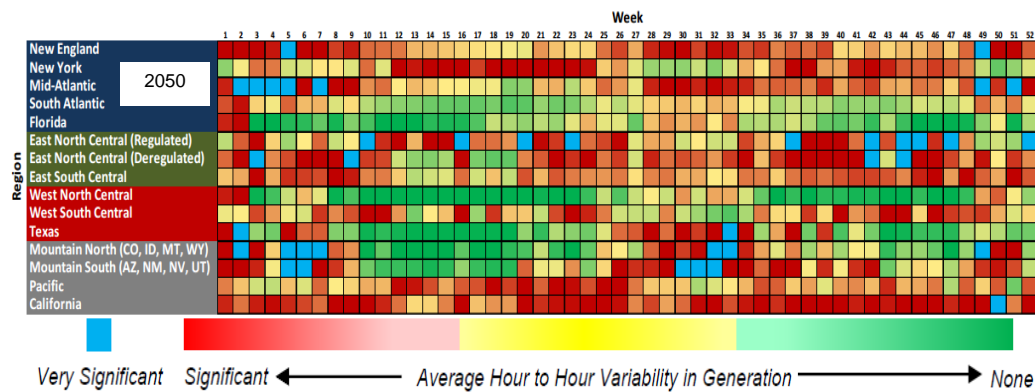


Figure 2. Prediction of electrical grid variability for regions of the United States in 2050. The color of the cells represents the variability. Regions approaching red and blue have demands that will be difficult and expensive for the electrical grid to meet—especially power producers operating under traditional market paradigms (EPRI, 2015).

2 The Tightly Coupled NHES

The reference hybrid energy system is referred to as a “tightly coupled” system. This coupling indicates that both the thermal and the electrical energy from the base load power supplier are integrated with one or more systems (e.g., industrial plant). The Modelica-based system under development is presented in Figure 3. The numbers in the figure correspond to the brief descriptions in Table 1.

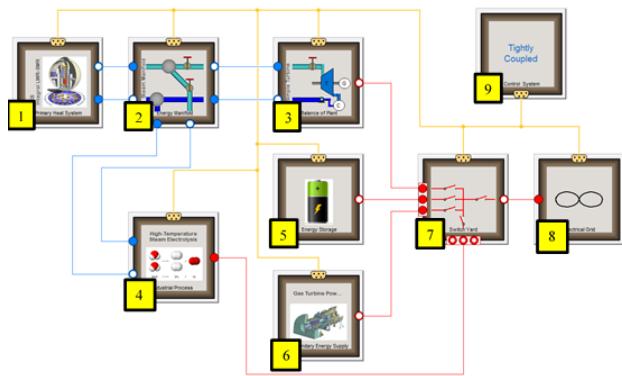


Figure 3. The tightly coupled NHES under development. The blue lines indicate fluid, the red lines indicate electricity, and the yellow lines indicate sensor/control signals.

The dynamic model is used to provide non-economic figures of merit—such as the ability to meet specified energy demands and overall system stability and reliability—to supplement the economic cost evaluation.

Table 1. Description of the various subsystems comprising a tightly coupled hybrid energy system.

Identifier	Component	Description	Example
1	Primary Heat System	Baseload heat and power	Nuclear reactor
2	Energy Manifold	Diverts energy to subsystems	Steam distribution
3	Balance of Plant	Primary electricity producer	Turbine and condenser
4	Industrial Process	Non-electric commodity revenue stream	Steam electrolysis or desalination
5	Energy Storage	Energy buffer to increase overall system robustness	Batteries and firebrick
6	Secondary Energy	Energy makeup	Gas turbine make-up
7	Switchyard	Electrical load distributor	Electricity distribution
8	Electrical Grid	Electrical customer	Large/small markets
9	Control Center	Hub for sub-system controls	Control/supervisory systems

2.1 Economic Evaluation: Cost

An economic evaluation of NHESs will be performed to investigate the minimum cost a hybrid system. This information informs decision makers on the planning and development of business/government agendas. To evaluate the economic cost of a given hybrid system, the Modelica model is coupled to the Reactor Analysis and Virtual control ENvironment (RAVEN), a multi-purpose software framework developed by INL that allows for dispatching different software functionalities, including surrogate model generation and optimization routines (Rabiti et al., 2012). As outlined in Figure 4, RAVEN supplies the dynamic model demand time histories for specific subsystems along with subsystem capacities (e.g., industrial process production capacity). The system control logic then operates the overall system to meet the supplied demand. At the end of the simulation, various figures of merit (e.g., ability to meet demand, reliability based on operation of components) are passed to RAVEN. RAVEN then creates simplified surrogate models of the dynamic system and performs a cost-based optimization. This optimization generates

new capacity parameters, and the process repeats until convergence to an optimized system is achieved. Using a high-performance computing cluster, this process is applied for many different cases in parallel.

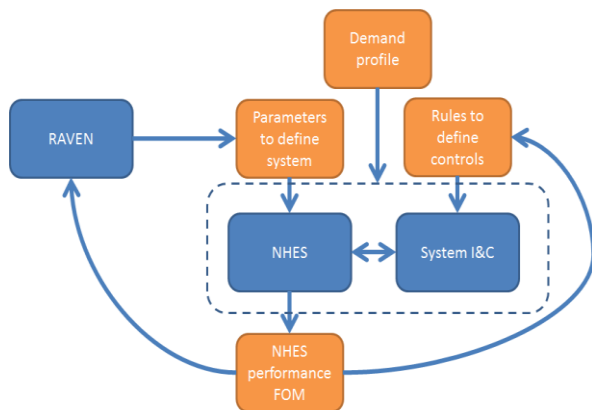


Figure 4. Modelica NHES dynamic model and RAVEN cost optimization process diagram (ORNL, 2016a).

2.2 Electricity Demand Profile

The specific energy demand profiles that provide set points to the dynamic model capture the variability of a specific energy market. For example, in a region with large solar power installations, the net electricity demand—consumer demand minus renewable supply—profile would show significant reductions in demand in the middle of the day—the period with greatest insolation. Figure 5 demonstrates a characteristic demand profile and the associated contributions of each of an example set of power producers over the course of a year. The demand profile is fed to the Modelica model using the `combiTimeTable` component in the Modelica Standard Library (MSL), which uses a relative path to an external text file to enable operation on the cluster.

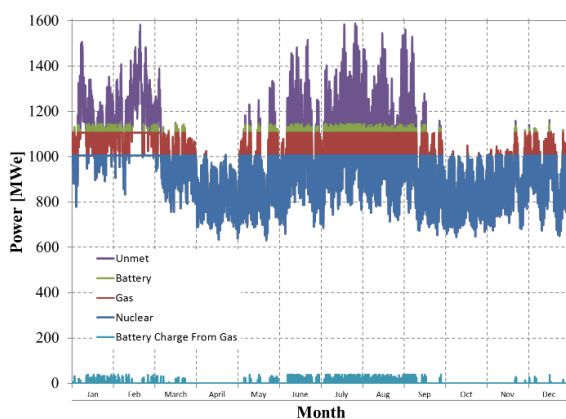


Figure 5. A one year electrical power demand profile characteristic taken from the north-east region of the United States (PJM, 2016). Each color represents the respective contribution of a subsystem energy supplier (ORNL, 2016b).

3 Dynamic Subsystem Models

Each of the subsystem models is built from a template, which allows for replaceable classes, improved interchangeability of control system approaches, and quick introductions of alternative subsystem models (e.g., replacing a steam electrolysis plant with a desalination plant). Figure 6 shows the template used when generating new subsystems and an example use case of the primary heat system. The subsystem models utilize the expandable connector signal bus for all control and sensor signals. Data records are also used to facilitate common reference values between subsystems, their control schemes, and the overall system.

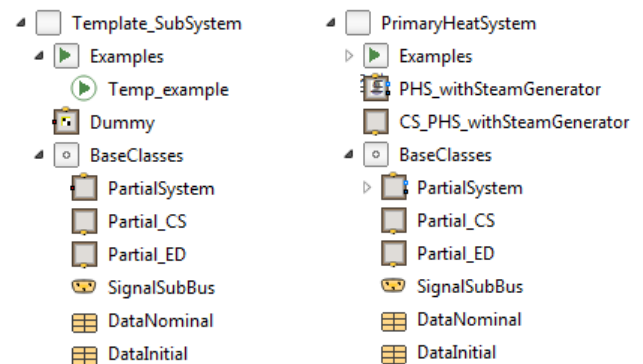


Figure 6. Subsystem template (left) and example use of the template (right).

3.1 Example Subsystems

In this section, the primary heat system, energy manifold, and balance of plant are briefly presented to better illustrate the physics-based modeling approach.

3.1.1 Primary Heat System

Figure 7 demonstrates the implementation of a primary heat source option, which—in this case—is an integral pressurized water nuclear reactor based on the International Reactor Innovative and Secure (IRIS) (Westinghouse, 2007). A few important physical phenomena captured in the model include the two phase dynamic interactions of the pressurizer, the generation of steam in a helical coil steam generator, and the behavior of a nuclear core. The nuclear core model is shown in Figure 8, and this model integrates the coolant flow geometry and behavior, fuel behavior, and point kinetics neutronics behavior, with feedback from the fuel and coolant temperature.

Models in the various subsystems use custom models, models from the MSL, and ThermoPower models. See Section 5 for more discussion on specific component modeling efforts.

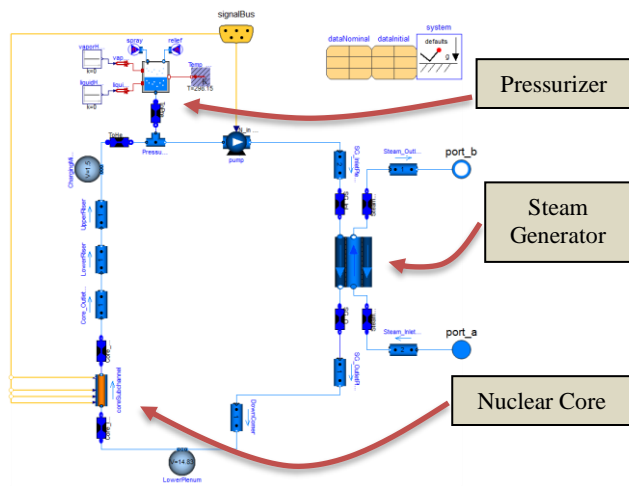


Figure 7. The Modelica model of the IRIS integral pressurized water nuclear reactor being used as the primary heat source subsystem.

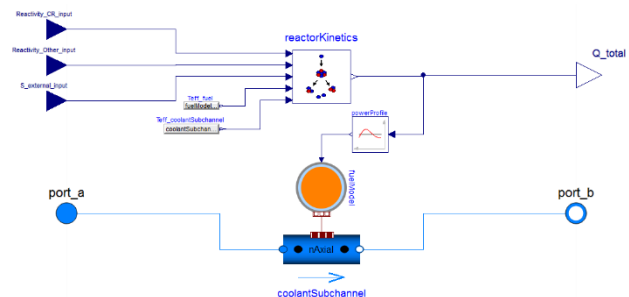


Figure 8. Model of a nuclear sub channel incorporating the neutronic behavior, non-uniform power generation, fuel conduction model, and coolant sub channel flow model.

3.1.2 Energy Manifold

The current distribution system under consideration is a purely thermal (i.e., steam/water) manifold (Figure 9). The energy manifold relies on controller logic to actuate distribution valves to handle large and slow power-set point changes to other subsystems, as specified by the demand profile. This actuation diverts hot steam coming from the primary heat system to the desired destination. The manifold also gathers return streams and directs the flow back the primary heat system steam generator at the proper temperature and pressure. Mixing and splitting volumes then add thermal mass to the system, dampening transient behaviors.

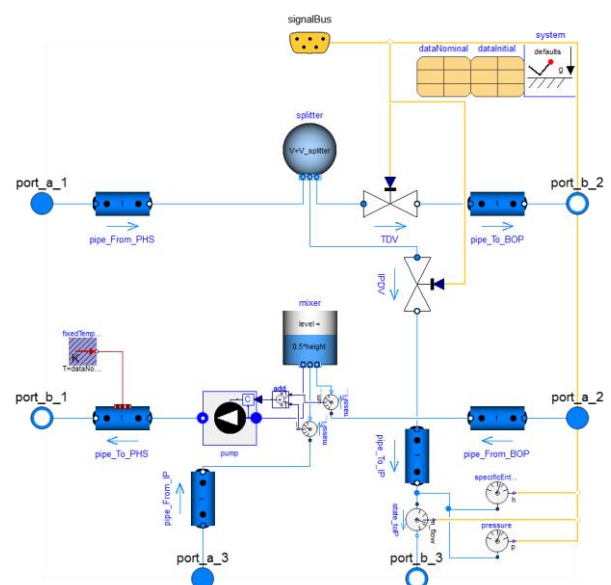


Figure 9. “Steam” energy manifold responsible for directing thermal energy to connected subsystem models.

3.1.3 Balance of Plant

One of the connections to the energy manifold is the balance of plant. The balance of plant is responsible for generating the primary share of electrical energy in the hybrid system. The current, simple model contains a steam turbine for electrical power generation, a condenser, and a control valve (Figure 10). The turbine control valve is responsible for small, fast control modulations.

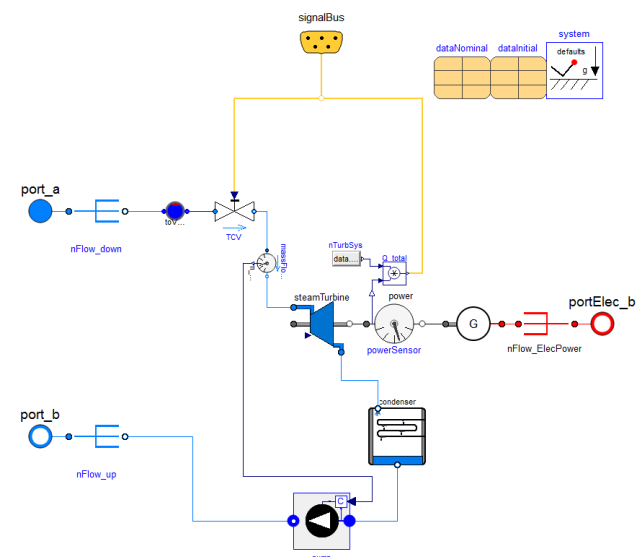


Figure 10. Simple balance of plant model, which converts steam thermal energy to electrical energy and returns subcooled water back to the energy manifold.

4 Preliminary Model Performance

A preliminary testing of the model reads the external data that contains the time series electrical demand profile and then feeds this data to the balance of plant control system. The control system actuates the control valve to match demand as much as the physical process allows. Figure 11 demonstrates 10 hours of dynamic turbine power operation based on a demand profile. The power changes are accomplished via the manipulation of actuators such as the turbine control valve position. At approximately hour three, the power set point is above the deliverable power. Situations like this period of unmet demand are tracked to inform the economic evaluation.

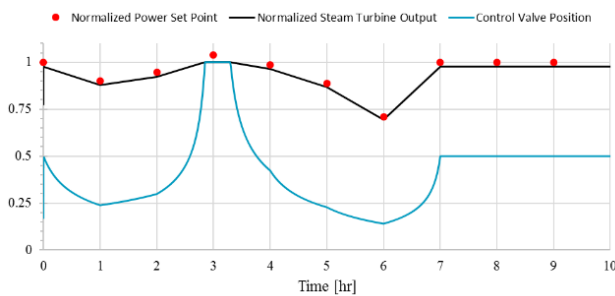


Figure 11. Preliminary test case demonstrating the ability of the coupled hybrid system to track a variable electrical demand profile by diverting flow to/from the steam turbine. Note the unmet demand at hour three.

The current NHES model consists of 14,581 equations and simulates a one-week period in approximately 2 hours using Dymola 2017 FD01 on a desktop computer (16 GB ram, Intel Xeon CPU ES-1607 v3 3.10GHz). Figure 12 presents the set point and measured electrical production values of a week-long simulation.

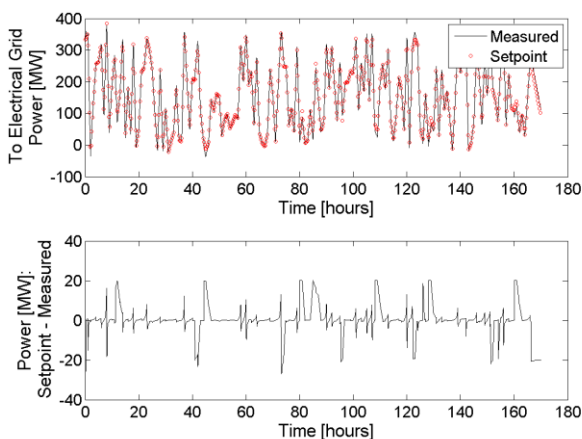


Figure 12. Load following electrical power production from the NHES model over a period of one week.

5 Component Development

The modeling activity uses components and connectors from the MSL along with a few components from the

ThermoPower library. However, user experience has identified various limitations to some components. Therefore, several components have been improved or remade for the needs of this project. A brief discussion of two major components are presented in this section.

5.1 MSL: Dynamic Pipe to GenericPipe

There are many positive aspects of the current version of the MSL *DynamicPipe* model. For example, the flexibility of specifying the model structure and the ability to easily change the number of discretized volumes, flow, and heat transfer models is incredibly useful. However, some significant limitations were discovered when attempting to couple the dynamic pipe with fuel and reactor neutronics models.

One primary issue was the inability to specify temperatures or enthalpy distribution for the start values of each control volume. The current *DynamicPipe* assumes a linear distribution between the ports. Since the neutronics model is highly sensitive to the temperature of the coolant and fuel, simulations often failed during the initial transient phase due to extreme power fluctuations in the reactor core.

To more generalize the capabilities of a pipe model, a new *GenericPipe* model was created. This model is similar in structure to the *DynamicPipe* model, but it removes some of its restrictions (e.g., added control of initialization and geometry) and works towards a more standard, organized approach to model development. Figure 13–Figure 15 show a few parameter windows displaying the new controls of *GenericPipe* along with the modified structure for various closure models, including heat transfer, pressure loss, and geometry.

This generic pipe can also be used to create simpler versions with more refined parameters—including *DynamicPipe*—to ease user interaction. For comparison, a few examples provided in the MSL fluid package (e.g., *BranchingDynamicPipes*) were recreated using the *GenericPipe* model and then benchmarked. Current tests using *GenericPipe* yield the same solutions as the *DynamicPipe* model but with computational speeds up to 30% faster.

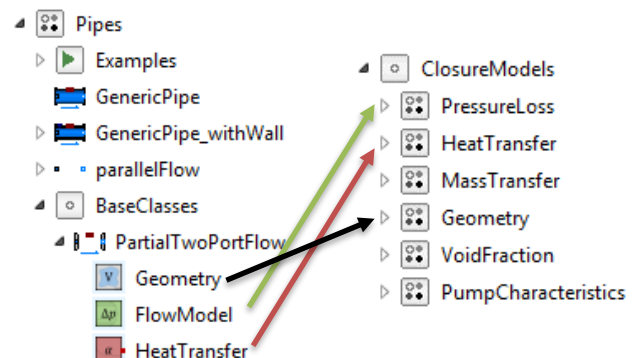


Figure 13. Structure of the generic pipe model demonstrating the expanded flexibility of the model.

Figure 14. “General” parameter tab of the generic pipe. Note the “Geometry” parameter allows for replaceable geometries (e.g., straight pipe and shell and tube or plate heat exchanger).

Figure 15. Improved initialization control for the pipe model permits simple initialization schemes based on port values or more precise schemes based on discretized volume states.

5.2 Custom: Thermal Library

The temperature response of a system is very important, particularly in nuclear reactors. The nuclear fuel temperature impacts not only the coolant flow behavior but also the power of the reactor itself by altering the behavior of the neutronics. To produce reasonably accurate models of nuclear fuel, a generic multi-dimensional discretized conduction model was created.

As part of this effort, the MSL Thermal package was completely redone to create a standalone library, which also includes a package of thermal resistances for steady-state evaluations, fin efficiency calculations, etc. (Figure 16). The created models are generic and can be

incorporated into cases that require thermal inertia or dynamics of conduction in solids. An important aspect of the library is the limited application of parameters to only those variables which require the variable type (e.g., initialization variables). All other parameters are specified as type `input` to ensure the user has maximum flexibility in model development. Additional features such as radiation models will be added to the Thermal package in the future.

5.2.1 Multi-Dimensional Conduction Models

Given the complex nature of multi-dimensional models, additional discussion on a conduction model is presented. Three different approaches were evaluated in

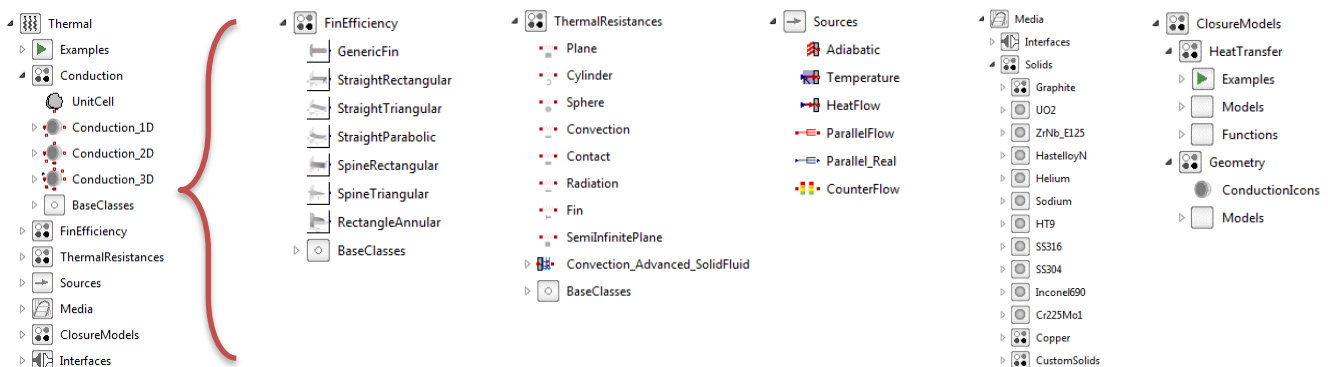


Figure 16. Thermal library with multi-dimensional conduction models, thermal resistance models, etc.

developing the conduction models: the “classical”, “modelica”, and “mixed” approach.

The classical programming approach relied on a replaceable “solution method” that defined the connections between cells. This approach has limited flexibility as equations (e.g., spatial differentiation of the energy equation) are hard-coded to initial assumptions such as geometry.

The modelica approach relied on independent, single-node, models to specify the behavior of unit volumes and the energy flow between cells; these models were then connected using `connect()` statements. Figure 17 shows the diagram layer of this modelica method and depicts the use of simpler models to build up more complex models.

The mixed approach limits use of `connect()` and instead applies models that have built-in nodalization which allows direct assignment of the variables that must be shared between models. In other words, the mixed method attempts to hard-code all generally applicable features of the model and only rely on the Modelica generated equations/connections when necessary while avoiding the embedded assumptions of the classical approach.

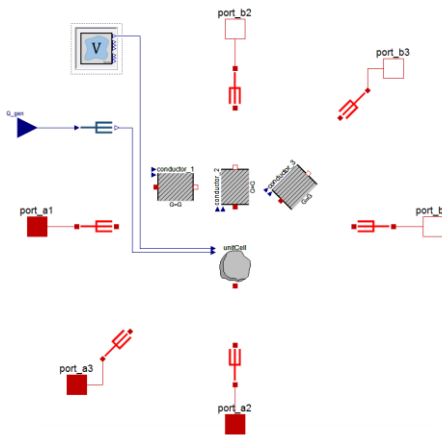


Figure 17. Diagram layer of `Conduction_123D` using the modelica approach. This method creates multi-dimensional conduction models by using independent models to build more complex, standardized models.

Each of the approaches have successfully modeled the needs of the hybrid energy system (e.g., fuel element modeling and heat exchanger walls). Figure 18 shows a surface plot of a fuel model with a fuel, gas gap, and cladding region created using the conduction models. Each of the regions have temperature-dependent properties specified by the solid media package.

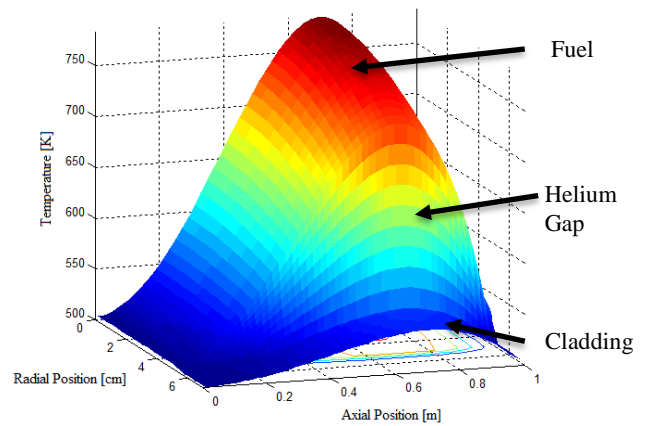


Figure 18. Surface plot of a non-uniformly heated fuel element (fuel, gap, and clad) with external convection created using the discretized conduction models.

Comparisons have shown that all three approaches produce results comparable within a small and reasonable margin of error (fractions of a degree Kelvin), however, the computational resources of the three approaches vary significantly. The classical approach passes the translation process quickly—even for a large number of discretizations—and then simulates quickly. The modelica approach can complete a simulation in similar or less time than the classical approach; however, the time it takes for the modelica approach to translate the model becomes more significant as the number of nodes being used increases (Figure 19).

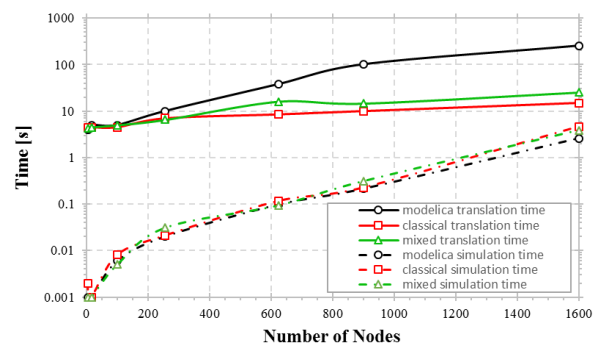


Figure 19. Demonstration of the translation and simulation times required for each of the methods of a discretized conduction model. For a given number of nodes, the modelica method requires far more translation time than the classical or mixed method, whereas the simulation times for each method remain similar.

The issue of translation time stems is primarily a result of relying on `connect()`, and therefore the translator, to generate the necessary equations for the solver. Figure 19 demonstrates that for a fixed number of many equations (e.g., 90,000), the classical approach—compared to the modelica method—can achieve a much finer discretization scheme (6,100 vs. 1,500 nodes) without compromising the translation time (50 s vs. 240 s). However, the mixed approach also generates ~90,000

equations with 1,500 nodes but passes through the translation phase in 25 s. This demonstrates that the method in which the equations are generated, rather than just the number of equations, is the primary controller of translational time. The mixed approach resolves the translational time penalty while also preserving efficient simulations.

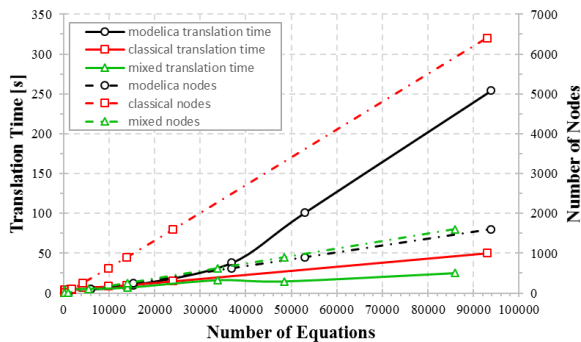


Figure 20. Demonstration of the relationship between the number of equations generated, the discretization scheme, and the required translation time for each of the discretized conduction model methods. The method of equation generation is the primary controller of translational time.

Although the modelica method adopts the best practices of Modelica programming by not repeating code, the ability to address the time for translation required the use of an alternative mixed approach. The findings of this study are important for the development of any discretized model and will be applied to additional physics of interest such as fluid flows and neutron behavior.

6 Conclusion

As energy markets shift to a highly variable demand profile, traditional base load power suppliers will be required to modify their business models. A hybrid energy system approach, coupling base load energy suppliers and energy customers (thermal and/or electric), may be profitable and preferred in future energy markets. The detailed dynamic multi-physics models discussed in this paper are being coupled to an economic cost optimization study that will inform the potential benefits and limitations of these hybrid systems by providing critical dynamic physical data of a potential hybrid system's operation.

As part of NHESs development, various components models are required to capture the important physical responses of the system. Two key models are the pipe model and thermal conduction models. This paper discussed adaptations and improvements to a Generic Pipe model and the creation of a new Thermal library. The thermal library includes multi-dimensional conduction models. Using these conduction models, an investigation of proper model formulation has been performed demonstrating a methodology to maximize

model flexibility while retaining computational efficiency.

Acknowledgments

This project was funded by the US Department of Energy's Office of Nuclear Energy under the Office of Advanced Reactor Deployment.

References

- Bragg-Sitton, S.M., R. Boardman, M. Ruth, O. Zinaman, C. Forsberg. 2015. *Rethinking the Future Grid: Integrated Nuclear Renewable Energy Systems*. Report no. NREL/CP-6A20-63207.
- EPRI (Electric Power Research Institute). 2015. *Program on Technology Innovation: Fossil Fleet Transition with Fuel Changes and Large Scale Variable Renewable Integration*. Technical report no. 3002006517.
- ORNL (Oak Ridge National Laboratory). 2016a. *Nuclear Hybrid Energy System FY16 Modeling Efforts at ORNL*. Report no. ORNL/TM-2016/418. Oak Ridge, TN.
- ORNL (Oak Ridge National Laboratory). 2016b. *Nuclear Hybrid Energy System Initial Integrated Case Study Development and Analysis*. Report no. ORNL/TM-2016/707. Oak Ridge, TN.
- ORNL (Oak Ridge National Laboratory). 2017. *Nuclear Hybrid Energy System Model Stability Testing*. Report no. ORNL/TM-2017/153. Oak Ridge, TN.
- PJM. 2016. *Estimated Hourly Load*. Accessed November 4. <http://www.pjm.com/markets-and-operations/energy/real-time/loadhryr.aspx>.
- Rabiti, C., A. Alfonsi, J. Cogliati, D. Mandelli, and R. Kinoshita. 2012. *Reactor Analysis and Virtual control ENVIRONMENT (RAVEN), FY12 report*. Technical report no. INL/EXT-12-27351. Idaho Falls, ID: Idaho National Laboratory (INL).
- Westinghouse. 2007. *Computer Models for IRIS Control System Transient Analysis*. Report no. STD-AR-06-04.

Modeling and simulation of fixed bed regenerators for a multi-tower decoupled advanced solar combined cycle

Iván Mesonero Jesús Febres Susana López

IK4-TEKNIKER, Spain, {ivan.mesonero, jesus.febres, susana.lopez}@tekniker.es

Abstract

Two dynamic models of fixed bed regenerators for metallic and ceramic configurations have been developed in Modelica. These models have been both worked out within CAPTURE European project and will serve as design tool for a fixed bed regenerative heat exchange system. The present article describes in detail both models and presents a case study that compares experimental and simulation results for the testing of a ceramic honeycomb regenerative matrix.

Keywords: fixed bed regenerator, ceramic honeycomb, stacked wire cloths, solar Brayton cycle

1 Introduction

The recently granted EU R&D project CAPTURE (<http://www.capture-solar-energy.eu>) pursues a new concept of central receiver system based on the Decoupled Solar Combined Cycle (DSCC) plant concept (see Figure 1). In such a plant, a multi-tower approach is employed with a solar Brayton cycle turbine on the top of each tower.

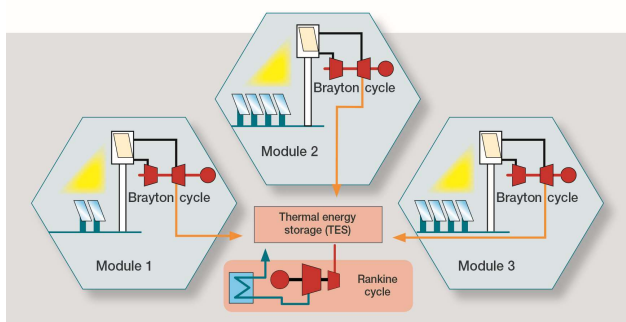


Figure 1. CAPTURE plant configuration based on DSCC concept

In CAPTURE project, a non-pressurized volumetric receiver will be employed to feed the solar turbine using a fixed bed regenerative heat exchange system for connecting both pressurized and non-pressurized air loops (see Figure 2). The fixed bed regenerative heat exchangers are alternatively connected to the two different air loops through a group of two-way on-off valves. Thus, the system allows the continuous

operation of the receiver and the turbine through the charging and discharging of a certain number of fixed bed regenerators.

Two approaches have been defined and modelled for the configuration of the fixed bed regenerator matrix material, a metallic approach based on stacked wire cloths, and a ceramic approach based on ceramic honeycombs monoliths. For the analysis of both options, one-dimension dynamic Modelica model of each approach have been developed within CAPTURE project and will be included in a free Modelica library. These models had to be parametric and flexible enough to allow the analysis of the effect of the design variables, such as the material characteristics and the bed geometry, in the behavior of the regenerative heat exchange.

Besides the two presented models, Modelica models of the receiver and the turbine shall be developed within CAPTURE in order to completely simulate the plant shown in Figure 2. This development will be carried out during incoming phase of the project.

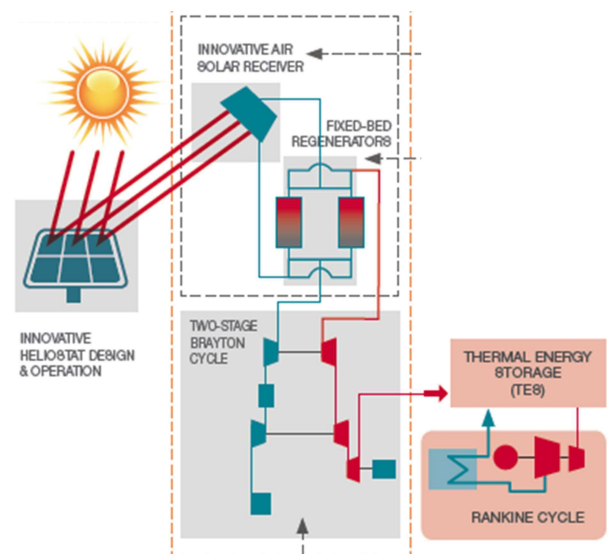


Figure 2. Main subsystems of a single module of CAPTURE plant

2 Metallic regenerative bed model

A cylindrical regenerative bed has been modelled (MetallicRegenerativeBed1D) considering the following main assumptions:

- Regenerative beds are made by a randomly stacked woven-screen matrix with plain square (see Figure 3).

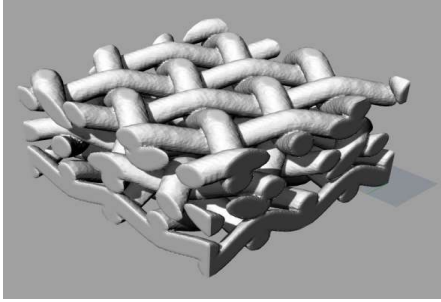


Figure 3. 3D view of three randomly stacked screens (plain square weave)

- The matrix is made of metallic materials and the model takes into account the dependency of their conductivity and specific heat capacity with the temperature.
- One-dimensional fluid flow is assumed, including only as heat transfer phenomenon the heat convection between the fluid and the matrix, i.e. radiative heat transfer is disregarded.
- One-dimensional heat conduction along the matrix (parallel to the fluid flow) is assumed. Perfect insulation is considered at the lateral area, thus heat losses can only be taken into account through upper and lower end of the matrix

2.1 Model structure

The model is mainly composed of two components that represent the solid and the fluid phases of the regenerative bed.

The model also includes a replaceable porosity function for the calculation of the volumetric porosity¹ of the matrix (ϵ), which is required for further calculation in equations (5) – (10). The following options are available to be chosen from a drop down menu: ESDUPorosity, NASAPorosity, ArmourCannonPorosity and XuWirtzPorosity. All of them were described by (Li & Peterson, 2006). However, some discrepancies were found for the porosity calculation defined by Xu & Wirtz, in consequence the original reference (Xu & Wirtz, 2002) was chosen for this case in order to define the necessary equations implemented in the code.

¹ In here the volumetric porosity is defined as the ratio between the void volume and the total volume of a porous body.

The following equations (1, 2, 3 and 4) describe how to calculate the porosity in each option.

- ESDU:

$$\epsilon = 1 - 0.25 * \pi * \frac{d_w}{pitch} \quad (1)$$

- NASA:

$$\epsilon = 1 - 0.25 * \pi * f_{crimping} * \frac{d_w}{pitch} \quad (2)$$

- Armour & Cannon:

$$\epsilon = 1 - \pi * \left(\frac{A*B}{2*(A+B)} \right) * \sqrt{1 + \left(\frac{A}{1+A} \right)^2} \quad (3)$$

$$A = \frac{d_w}{pitch_w}$$

$$B = \frac{d_w}{t}$$

- Xu & Wirtz:

$$\epsilon = 1 + \frac{3.906E^{-4} * \pi * C * \left(\frac{dw}{pitch} \right)}{C_f}$$

$$C_f = \frac{L}{n_{layer} * 2 * d_w} \quad (4)$$

$$C = 123 * \left(\frac{d_w}{pitch} \right)^4 - 384 * \left(\frac{d_w}{pitch} \right)^2 - 640$$

Where:

- d_w is the wire diameter;
- $pitch$ is the distance between two wires or the aperture;
- $f_{crimping}$ is a factor that describes how compressed are the meshes in the matrix;
- t is the thickness of the mesh;
- L is the regenerative bed length;
- n_{layer} is the number of layers in the matrix.

It must be noted that for the last two options the user must provide the value of some more parameters, the thickness of the wire mesh or the number of layers in the matrix, as shown in Figure 4.

General

Component: _____

Name: _____

Comment: _____

Model: _____

Path: CAPTure.Obsolete.RegenerativeBedDynamicModels.MetallicRegenerativeBed1D

Comment: 1-D Model for a cylindrical regenerative bed made by a randomly stacked woven-screen matrix (only valid for a plain square weave)

Parameters

d_w	m	Wire diameter
pitch	m	Distance between two wires, aperture
n_layer	1	Number of layers in the matrix, only for Xu & Wirtz porosity calculation
porosity	CAPTure.Obsolete.RegenerativeBedDynamicModels.Bz	Formulation of porosity calculation
t	m	Thickness of the mesh, only for Armour & Cannon porosity calculation
L	m	Regenerative bed length
D	m	Diameter of regenerative matrix
nNodes	2	Number of discrete flow volumes
Medium	Modelica.Media.Interfaces.PartialMedium	
material	Randomly stacked woven-screen matrix of Inconel625	

Figure 4. Parameter dialog for the metallic regenerator model

2.1.1 Solid phase

Solid phase component is an instance of the class `HollowCylinder` that is a lumped parameter thermal system in the radial direction (there is no radial variation of the solid temperature). It is composed of an array of nodes which are instances of the class `HollowCylinder_Lumped` that represents one section of the solid material along its axis. Each node consists of two classes. One that describes the conduction along the material, and other that represents the thermal inertia of the material section.

In order to calculate the thermal characteristics of the solid phase model, the material properties have to be entered in the model. `HollowCylinder` class includes a replaceable instance of a class that contains the characteristics of the material of the regenerative bed. This material can be selected in the “Solid” tab of the main model. In addition, the regenerative bed total mass can be entered as a parameter. If no value of the total mass is used, it is calculated multiplying the material density by the total volume of the solid.

If a new material is required, the material class has to be declared as a Modelica package extended from the base class `PartialMaterial`. This package must contain the thermal and physical properties of the chosen material. The minimal set of properties required consists of the density, the thermal conductivity and the specific heat capacity. All of them may be defined either as a constant or as function of the temperature.

Since the model was meant to describe the behavior of porous materials and more in concrete woven-screen matrixes, this class takes into account the porosity of the matrix and the conductivities of the solid and the fluid in order to calculate the real thermal conductivity of the matrix using the following equation (Martini, 2004):

$$k_{matrix} = k_{gas} * \frac{\left(\frac{1 + \left(\frac{k_{material}}{k_{gas}} \right)}{1 - \left(\frac{k_{material}}{k_{gas}} \right)} - (1 - \eta) \right)}{\left(\frac{1 + \left(\frac{k_{material}}{k_{gas}} \right)}{1 - \left(\frac{k_{material}}{k_{gas}} \right)} + (1 - \eta) \right)} \quad (5)$$

Where:

- k_{matrix} is the thermal conductivity of the matrix;
- k_{gas} is the thermal conductivity of the gas in the matrix;
- $k_{material}$ is the thermal conductivity of the matrix material (solid).
-

The class defining the solid phase has an array of inputs named `thermalConductivity_medium` in order to have access to the instantaneous value of the internal variable of the fluid phase `fluidPhase.heatTransfer.lambdas` that is exactly the instantaneous value of the thermal conductivity of the gas in the different nodes along the matrix.

2.1.2 Matrix materials

Apart from the basic partial models described in the previous sections, four specific materials have been added to the `Materials` library: DIN EN 10095, DIN 17742, DIN 17470 and DIN EN 10302. These materials were chosen taking into account the expected operating temperatures, their availability as meshed material, thermal and mechanical properties, and sintering possibility. In all cases, the specific heat capacity and thermal conductivity are temperature dependent values while density is assumed constant. In

addition, whether the material is sintered or not is not taken into account in this version of the model. However, the effect of sintering the material shall be included in further versions as it influences the thermal conductivity (Li & Peterson, 2006) and the mechanical stability during the cycling regime, e.g. sintering stabilizes an in-line stacked bed (unstable before sintering) as a permanent link between wire meshes is guaranteed.

Regarding this two last properties, the available information from datasheets was fitted to polynomial expressions (linear or quadratic) in most cases and logarithmic expressions in others.

2.1.3 Fluid phase

Fluid phase component is an instance of the class `DynamicRegenerativeBedFluidPhase` that is based on the `DynamicPipe` class from the Modelica Standard Library (Casella, 2009) that is the model of a straight pipe with distributed mass, energy and momentum balances providing the complete balance equations for one-dimensional fluid flow. It treats the partial differential equations with the finite volume method and a staggered grid scheme for momentum balances.

The main differences between the original `DynamicPipe` and the `DynamicRegenerativeBedFluidPhase` are the following:

- Specific equations have been implemented under the `Detailed` option of `FlowModel`. When the `Detailed` option is selected, the relationship between the mass flow rate and the pressure loss is determined with experimental correlation for a flow through an infinite randomly stacked woven-screen matrix.
- Flow friction characteristics were originally defined by Kays & London (Kays & London, 1998). They determined experimentally the relationship between the friction factor and the Reynolds number for different porosity values of the matrix. But the equations implemented within this model correspond to the following approximation determined by Martini (Martini 2004):

$$dp = \frac{m_{flow}^2 * L * C_w}{2 * A^2 * \rho * \left(\frac{D_h}{4}\right)} \quad (6)$$

$$\log C_w = \begin{cases} 1.73 - 0.93 * \log Re & \text{if } Re < 60 \\ 0.714 - 0.365 * \log Re & \text{if } 60 \leq Re < 1000 \\ 0.015 - 0.125 * \log Re & \text{if } 1000 \leq Re \end{cases}$$

Where:

- dp is the pressure drop along the matrix;
 - m_{flow} is the fluid mass flow rate;
 - L is the length of the matrix;
 - C_w is the factor of friction for matrix;
 - A is the area of flow;
 - ρ is the density of the fluid at regenerator;
 - D_h is the hydraulic diameter of the matrix;
 - Re is the Reynolds number.
- A new option was added to the list of classes that describe the convective heat transfer within this model with equation (7). It is especially suited for gas flow through an infinite randomly stacked woven-screen matrix being a correlation from Organ (Organ, 2010) of experimental data from wire screens and crossed rods simulating wire screens from Kays & London (Kays & London, 1998).
 - Main assumptions of the correlation are: perfect stacking, i.e. screens touching is assumed, and volumetric porosity between 0.602 and 0.832.

$$St * \sqrt{Pr^3} = \frac{1.25}{\sqrt{Re}} \quad (7)$$

Where:

- St is the Stanton number;
- Pr is the Prandtl number;
- Re is the Reynolds number.

Regarding the parameterization of the fluid phase model, the hydraulic radius, r_h , of the individual wire screen and matrixes is determined by equations (8) and (9) when porosity is calculated by the expression defined by ESDU (Organ, 1997) (Kays & London, 1998):

$$r_h = \frac{pitch}{\pi} - \frac{d_w}{4} \quad (8)$$

$$r_h = \frac{d_w * \eta}{4 * (1 - \eta)} \quad (9)$$

It is worth to mention also that for both correlations (flow friction and convective heat transfer) the Reynolds

number in equation (10) is calculated with a velocity (v_s) that is not the real velocity of the fluid along the matrix.

$$\begin{aligned}
 Re &= \frac{\rho * v_s * D_h}{\mu} \\
 &= \frac{\frac{m_{flow}}{A_c} * D_h}{\mu} \\
 &= \frac{\frac{m_{flow}}{A_{fr} * \P} * D_h}{\mu}
 \end{aligned} \quad (10)$$

Where:

- ρ is the density of the fluid at the matrix;
- D_h is the hydraulic diameter of the matrix;
- μ is the cinematic viscosity of the fluid;
- m_{flow} is the fluid mass flow rate;
- A_c is the free flow area of the matrix;
- A_{fr} is the frontal area of the matrix;
- \P is the volumetric porosity of the matrix.

The reason for that is that the free flow area is calculated as the product of the frontal area of the matrix and its volumetric porosity. Usually, the volumetric porosity and the screen porosity have different values, being the second one bigger than the first one. Accordingly, the computed values for the fluid velocity within the fluid phase model will be bigger than the real ones.

3 Ceramic regenerative bed model

During the specification definition phase of CAPTURE, the partners decided that the type of ceramic regenerator to be modelled was to be a honeycomb with straight channels (see Figure 5).

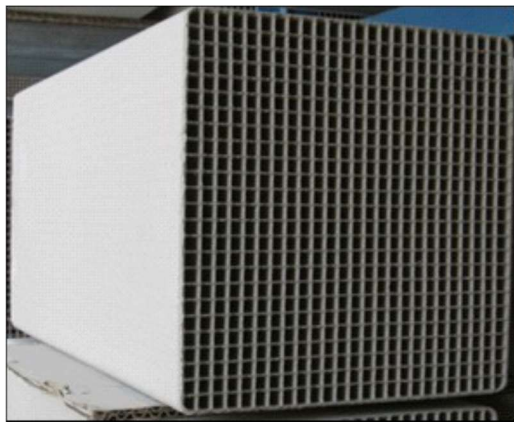


Figure 5. Cordierite honeycomb brick with high density of straight channels.

This model (CeramicRegenerativeBed1D) is based on the modelling approach presented by Muske et al (Muske, 2000). Even though it was originally meant for checkerwork regenerators, the model may be used for other regenerators with the same type of geometry, i.e.

parallel straight channel geometry (two dimensions honeycomb). Each channel of the regenerator is modelled as a hollow cylinder tube, whose external wall is assumed perfectly isolated. The radius of the fluid channel, the internal radius r_i , is one-half of the average hydraulic diameter of the real fluid channels and the outside radius is given by equation (11):

$$r_o = \sqrt{\frac{m}{\pi * \rho * N_c * L} + \frac{D_h^2}{4}} \quad (11)$$

Where:

- N_c is the total number of gas channels;
- D_h is the hydraulic diameter of the gas channel;
- m is the total mass of the bed;
- ρ is the density of the bed material;
- L is the length of the bed.

The following considerations were taken into account when modelling the ceramic regenerative bed:

- The fluid velocity in the tubes is determined assuming a uniform distributed fluid flow through all channels.
- As in the metallic bed, the ceramic regenerative bed model is constituted by two namely solid and fluid phases.
- Regarding the solid phase, there is no radial variation of the temperature (lumped parameter model in the radial direction is assumed).

Note that for simulations where cycling regimes are required, the last assumption is expected to be valid only when the cycle time of the system is, at least, an order of magnitude bigger than the characteristic time for radial heat conduction in the material (Muske 2010) which is defined by:

$$\tau = \frac{(r_o - r_i)^2}{\alpha} \quad (12)$$

Where:

- τ is the characteristic time for radial heat conduction;
- α is the thermal diffusivity of the bed material;
- r_i is the radius of the gas channel in the tube;
- r_o is the outside radius of the tube.

For the case of highly channeled honeycombs, the reduced thickness of walls assures a good agreement with the last assumption.

3.1 Model structure

Figure 6 shows the icon of the Modelica model of the ceramic regenerator bed. Both heat and fluid ports are taken directly from the Modelica Standard Library, which means the model is compatible with any element found in `Modelica.Fluid` and `Modelica.Thermal`.

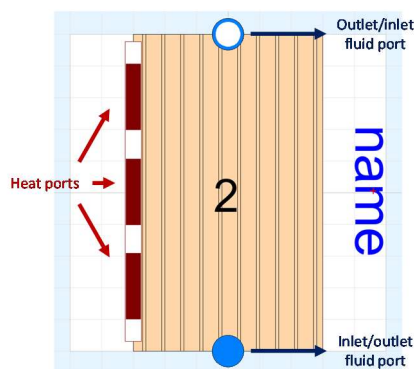


Figure 6. Modelica ceramic regenerative bed model icon.

The model allows users to initialize the temperature of the fluid and the solid elements. Initial values can be input in the “Initialization” tab. If no value is passed to the model, 24 °C is used as default value for both temperatures.

In order to define the geometry of the regenerative bed, six parameters can be input in the “General” tab:

- The bed length: `length` in [m];
- The bed cross-section area: `area_s` in [m²];
- Channels cross-section area: `area_c` in [m²];
- Channels cross-section perimeter: `perimeter_c` in [m];
- Number of channels: `N_c`;
- Segmentation perpendicular to heat conduction: `nNodes`.

Note that the model is discretised in finite volumes (solid and fluid volumes) and the degree of discretisation is defined by the `nNodes` parameter.

As mentioned previously, the model is composed, as can be appreciated in Figure 7, by two components that represent the solid and the fluid phases of the regenerative bed.

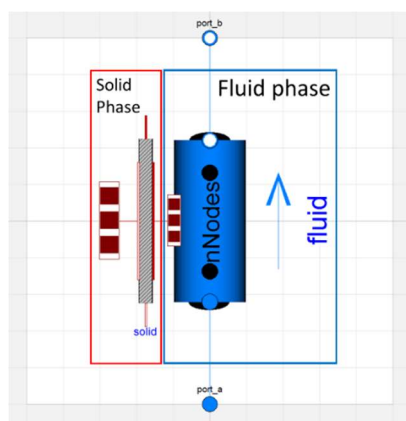


Figure 7. Diagram of the ceramic regenerative bed model.

3.1.1 Solid phase

The model presented in section 2.1.1 was used to model the solid phase of the ceramic regenerator as this model allows the user to work with non-porous material. There is a Boolean parameter in the “Porosity” tab that disables the use of the equations mentioned in section 2.1.1., making the model suitable to represent ceramic materials.

3.1.2 Fluid phase

The fluid phase component is an instance of the `DynamicPipe` class from the Modelica Standard Library which is the model of a straight pipe with distributed mass, energy and momentum balances providing the complete balance equations for one-dimensional fluid flow. It treats the partial differential equations with the finite volume method and a staggered grid scheme for momentum balances.

Most of the parameters that define the `DynamicPipe` have been fixed and only three of them (the fluid medium, the heat transfer model and the flow model) are accessible from the GUI.

A new option was added to the list of classes for the heat transfer that describes the convective heat transfer with a correlation for rough pipes by Bhatti and Shah (Muske 2000).

4 Ceramic regenerative bed case study

The case study proposed in this paper is centered on the ceramic honeycomb approach for regenerative matrix. The proposed model was verified against the experimental results presented in a technical report elaborated by SANDERS Associates in 1980 (Sanders, 1980). This report describes the application of the regenerator in a solar prototype small plant as well as the experimental set-up and test results of a ceramic honeycomb regenerator manufactured for real demonstration at laboratory level. Next paragraphs describe the manufactured regenerator and the tests performed that were employed for the case study.

4.1 Ceramic regenerator description

The ceramic regenerator is a cylindrical matrix installed inside an internally insulated cylindrical pressure vessel, the complete system is called Thermal Storage Module or TSM. Dimensions of the complete matrix are 914 mm (36 inches) in diameter and 787mm (31 inches) in length. Base material for the regenerator matrix are cylindrical ceramic honeycombs logs from CORNING with the same long as the complete matrix and a diameter of 114 mm (4.5 inches). The shape of the logs

are modified in order to get the sectors that finally conforms the complete regenerative cylinder. **Figure 8**, Figure 9 and Figure 10 clarify above description.

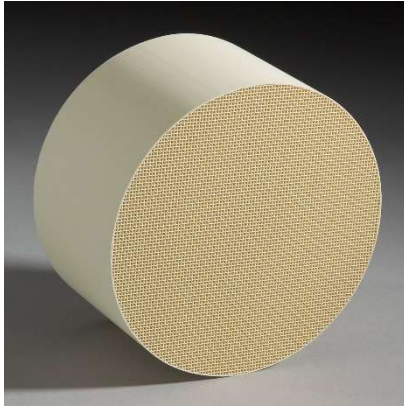


Figure 8. Typical Cordierite honeycomb cylinder for catalytic converters in automotive application from CORNING

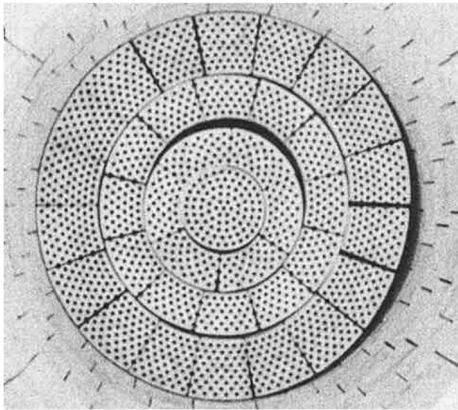


Figure 9. Picture of the internal cross section of a ceramic heat exchanger with the same manufacturing approach as SANDERS's regenerator (Sheindlin, 1986)

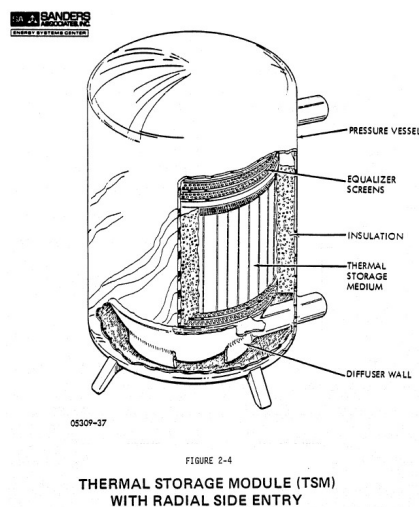


Figure 10. Diagram of the Thermal Storage Module developed by SANDERS (Sanders, 1980)

The Cordierite log employed for manufacturing the TSM is a square cell based honeycomb designated by 300/12, which corresponds to 300 cpsi (cells per square inch) and 12x10⁻³ inches of wall thickness. The thermal properties of the Cordierite material in the honeycomb are summarized in Table 1

Table 1. Thermal properties of the Cordierite material in the honeycomb

Temperature °C (°F)	260 (500)	399 (750)	538 (1000)	815 (1500)
Specific heat J/kg K (BTU/lb °F)	1005 (0.24)	1118 (0.267)	1193 (0.285)	1289 (0.308)
Thermal conductivity W/mK (BTU in/h ft ² °F)	1.44 (10) constant			
Bulk density g/cm ³ (lb/ft ³)	0.589 (36.4) constant			

4.2 Test set-up and instrumentation

The test set-up is mainly composed of a ceramic regenerator, a four-way valve, a gas burner, a compressor, control valves and piping. The test schematic can be seen in Figure 11.

The test set-up is equipped with sensors allowing the analysis of the regenerator performance under different test conditions. Principal sensors of interest for the analysis are: air mass flow meter, air temperature sensors, air pressure sensors and temperature sensors for the measurement of the ceramic material in different positions along the bed.

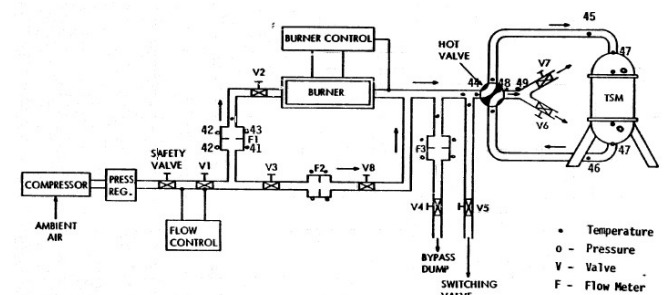


Figure 11. Test set-up schematic developed by SANDERS (Sanders, 1980)

4.3 Test selection for the case study

For the model validation two different types of tests were chosen, a single shot test and a cycling test. Both tests are described in the next paragraphs.

4.3.1 Single shot test

This test is based on a sudden heating up (or cooling down) of the regenerator with an air stream that flows from top to bottom (or from bottom to top). The initial temperature of the entire matrix is constant and homogeneous.

The objective of the test is to analyze the thermal performance of the bed in terms of thermocline propagation along the matrix (transient response).

Table 2. Selected single shot test data

Air flow direction	Upward
Air mass flow kg/s (lb/s)	0.19 (0.43)
Air temperature °C (°F)	702 (1295)
Initial homogeneous temperature in the matrix °C (°F)	146 (295)

Figure 12 shows the original graph with the experimental results of the selected test (Sanders 1980) and the simulation results obtained overlapped. The simulation results were obtained with the model described on section 3 under the general test conditions of Table 2, but it has not been possible to accurately reproduce the variable inlet temperature of the experimental data. Moreover, the initial temperature of the entire regenerative bed was assumed homogeneous but the report points out the existence of an initial temperature profile along the bed due to experimental difficulties in setting up initial test conditions. Consequently, further analysis of the system transient shall be performed in order to understand the quantitative deviations. Nevertheless, a good agreement, from the qualitative point of view, has been achieved between the experimental and the simulation results.

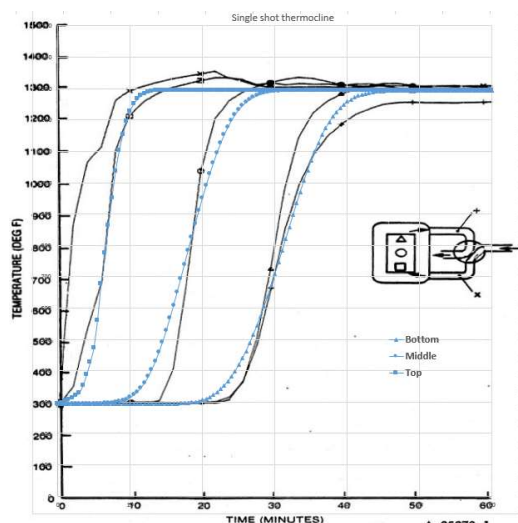


Figure 12. Experimental (black) and simulation (blue) results of single shot test of the ceramic regenerative bed model.

4.3.2 Cycling test

This test is based on a continuous cyclic operation (charging and discharging) of the regenerator starting from an initial steady state (constant temperature within the entire matrix). During the charging phase, hot air flows at atmospheric pressure, which simulates the heat input of the solar receiver. In the discharging phase cold air is blown in the opposite direction to simulate the inlet from the process return (a pressurized air would simulate compressor outlet of an air turbine).

The objective of the test is to analyze the performance of the bed working in cycling conditions in terms of thermocline evolution until the system becomes stable (cyclic state).

Table 3. Selected cycling test data

Air flow direction	Charging (Downward) Discharging (Upward)
Air mass flow kg/s (lb/s)	0.19 (0.43)
Air temperature °C (°F)	Charging 702 (1295) Discharging 146 (295)
Air pressure	Atmospheric pressure for charging and discharging
Initial homogeneous temperature in the matrix °C (°F)	146 (295)

In the same way as for the single shot test, Figure 13 shows the experimental and the simulation results for this test. The simulation results were obtained with the model described on section 3 under the general test conditions of Table 3. For this test the experimental data for the inlet temperature of the air was not available so it was assumed constant during each phase, charge and discharge, of the cycle.

It can be appreciated in Figure 13 the very good agreement between the experimental and the simulation results for this test. The results are very similar especially in the last cycles where probably the effect of the different initial conditions applied was disappeared.

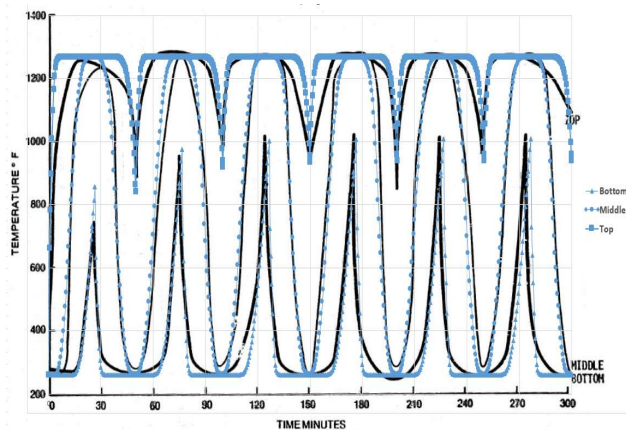


Figure 13. Experimental (black) and simulation (blue) results of cycling test of the ceramic regenerative bed model.

5 Conclusions

Two Modelica models have been described for the dynamic simulation of two types of regenerative beds, a metallic one based on stacked wire meshes, and a ceramic one based on straight-channelled honeycomb. These models will be used as design tool in CAPTURE project and will support the evaluation of the performance of fixed bed regenerative heat exchangers.

These models are part of a public deliverable of CAPTURE project and will be included in a free Modelica library. In addition, prototypes designed using the presented models will be tested along 2018 in the Plataforma Solar de Almería (PSA). This gives the chance for validating the model with real data coming from operation of the regenerative beds.

A case study has been presented where experimental results from the testing of a ceramic honeycomb regenerator were compared with simulation results obtained with the model developed for this regenerator typology. The model provides a suitable representation of the regenerative bed behaviour and constitutes a useful tool for the design of these components.

Finally, the authors are currently working on the metallic bed models in order to study the effect of different configurations of the metallic meshes to assess the feasibility of using this type of regenerators. In addition, the metallic bed model is being extended to include other stacking configuration (in-line and staggered). These results will be presented in future publications.

Acknowledgements

The authors would like to thank the European Commission for partial funding of this work related to CAPTURE project (H2020 research and innovation programme, grant agreement No 640905).

References

- R.B. Bird et al, "Transport phenomena", Wiley, New York, 1960
- F. R. Casella et al. "Standardization of Thermo-Fluid Modeling in Modelica.Fluid", Proceedings of 7th International Modelica Conference, 2009, Como, Italy
- S. Kakac et al, "Handbook of single-phase convective heat transfer", Wiley, New York, 1987
- W.M. Kays and A.L. London, "Compact Heat Exchangers", Krieger Publishing Company, Malabar, Florida, 1998
- C. Li and G.P. Peterson, "The effective thermal conductivity of wire screen", International Journal of Heat and Mass Transfer 49 (2006) 4095 -4105
- W.R. Martini, "Stirling Engine Design Manual", University Press of the Pacific, Honolulu, Hawaii, 2004
- Modelica Association, "A Unified Object-Oriented Language for Physical System Modeling", 2012
- K.R. Muske et al, "Model-based control of a thermal regenerator. Part 1: dynamic model", Computers and Chemical Engineering 24 (2000) 2519-2531
- A. J. Organ, "The Regenerator and the Stirling Engine", Mechanical Engineering Publications Limited, London and Bury St Edmunds, UK, 1997
- A. J. Organ, "Thermodynamics and Gas Dynamics of the Stirling Cycle Machine", Cambridge University Press, Cambridge, 2010
- Sanders Associates Inc, Small solar electric system components demonstration final report, JPL contract 955279, Nashua, New Hampshire, August 20, 1980
- A.E. Sheindlin, High temperature equipment, Hemisphere publishing Corp., Washington, 1986
- J. Xu and R.A. Wirtz, "In-plane effective thermal conductivity of plain-weave screen laminates", IEEE TCPT 25 (4) (2002) 615-620

Annual Performance of a Solar-Thermochemical Hydrogen Production Plant Based on CeO₂ Redox Cycle

Alberto de la Calle¹ Alicia Bayon¹

¹CSIRO Energy, 10 Murray Dwyer Ct., Mayfield West, NSW 2304, Australia,
{alberto.delacallealonso, alicia.bayonsandoval}@csiro.au

Abstract

For the first time, a dynamic model of a 1-MW_{th} thermochemical hydrogen production plant is developed and implemented for CeO₂ redox cycle. The work explores how the variables of the process like the direct normal irradiation (DNI), temperature, pressure and degree of oxidation affect the annual production of hydrogen. The model reveals that the thermal inertia of CeO₂ is significantly high to accomplish the oxidation without refrigerate the oxidizer. The operation is optimized to obtain the maximum amount of hydrogen in a year by only modifying the mass flow rates at the inlet of the reactors. The flexibility and adaptability of the model allows to test different system configurations and optimize the hydrogen production.

Keywords: Solar fuels, Central receiver, High temperature, Dynamic modelling

1 Introduction

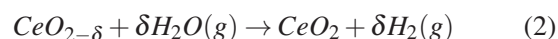
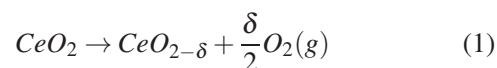
Solar energy is, by far, able to be massively exploited for delivering all of the world energy needs utilizing only a few percent of the deserted areas (IRENA and IEA-ETSAP, 2013; Lewis and Nocera, 2007). Nevertheless, the storage of the thermal energy for its use during the non-solar periods is required to couple production/demand rate in the energy market. In this context, the conversion of the solar concentrated source into storable and transportable fuels is a remarkable alternative to extent the commercialization of solar power technologies.

One attractive pathway is the solar thermal production of hydrogen. Within all possible solar driven routes, solar thermochemical H₂O splitting offers a path to produce carbon-free hydrogen. Hydrogen is an energy carrier in addition to a commodity used for the several industrial processes (Ramachandran and Menon, 1998). Nevertheless, direct thermolysis of H₂O requires temperatures well above 2000 K to obtain significant H₂ concentrations (Fletcher, 2001). In addition, to avoid the recombination of the product gas H₂ and O₂ upon cooling, they need to be separated at the dissociation temperature, which is technically challenging (Fletcher, 2001). In this respect, H₂O-splitting thermochemical cycles have been investigated to reduce the process operating temperature compared to direct thermolysis. In addition, the need for

high-temperature product gas separation is eliminated, because H₂ and O₂ are produced in separate process steps. Compared to multi-step cycles, two-step cycles promise to reach higher process efficiencies due to higher operating temperatures and less irreversibilities (Abanades et al., 2006).

Besides the environmental benefits of the thermochemical cycles, several impediments must be confronted to the economic realization which concerns the design of reactor to reduce the radiation and conduction losses and materials development revealing satisfactory durability, reactivity and efficiencies (D'Souza, 2013; Roeb et al., 2012). Likewise, heat and mass transfer play a crucial role in the building components and for the technological implementation of thermochemical reactors.

Up to date, 300 redox systems have been proposed although only few tens of them have been performed experimentally mainly due to temperature and thermodynamic limitations (Muhich et al., 2015). In terms of economic assessment, a recent report has indicated that solar fuels produced with 20% efficiency are likely to be cost competitive (Kim et al., 2012). Upon all the possible metal oxide candidates, Ceria (CeO₂) is the most promising material so far studied during the last 50 years because it demonstrates faster hydrogen production kinetics and high selectivity (Ackermann and Steinfeld, 2014; Chueh et al., 2010; Furler et al., 2012; Gao et al., 2016; Scheffe and Steinfeld, 2012). In this cycle, the nonstoichiometric ceria, with fluorite-type structure, retain the oxygen vacancies maintaining its cyclability. The reactions involved in this process are:



The thermal reduction (Equation 1) occurs at temperatures not lower than 1500 °C accompanied by a low O₂ partial pressure about 1 Pa (Chueh et al., 2010). This condition requires a large amount of inert gas flowing into the reaction media and, consequently, an enormous economic penalty influenced by three factors: cost of inert gas, separation of O₂ produced downstream and energy losses transferred to the gas (Furler et al., 2012). In the low temperature step, the exothermic water splitting (Equation 2)

takes place at lower temperatures commonly between 600 and 1000 °C.

The thermochemical efficiency was largely explored in previous works (Bader et al., 2013; Ermanoski, 2015; Ermanoski et al., 2013; Bulfin et al., 2015). In this context, a maximum of 68% could be obtained if all the CeO₂ is reduced to Ce₂O₃ at 2200 °C with the sun as only heat source. However, at 1500 °C only 2% conversion is obtained in real conditions, lowering the efficiency up to 1.72% from solar to fuel without heat recovery (Furler et al., 2012). This value could be enhanced to 20% if ideal heat recovery is applied (Ermanoski et al., 2013). However, all the previous thermodynamic analysis are based on steady-state simulations with the aim of maximizing the reactor efficiency of the process giving an single value of DNI without considering the variability of the solar resource, heliostat field design and receiver performance. The goal of this work is to provide insights on the effect of the variability of the solar resource over the annual performance emulating a solar production plant based on a Ceria thermochemical water splitting cycle.

In the present paper, a new dynamic model of a solar hydrogen production plant is developed for annual simulations. The model is based on an object-oriented modelling methodology following a modular and hierarchical structure. The final model has been graphically implemented by connecting different components which encapsulate the main thermodynamic processes that take place in the plant. Modelica and Dymola 2017 were the language and the simulation environment used in this work.

2 System description

A 1-MW_{th} solar hydrogen plant is designed to be placed in Geraldton (WA), Australia. Table 1 shows the system design specifications. It consists of two rotatory reactors (for reduction and oxidation), where a flow of particles of CeO₂ is recirculated in order to efficiently use the thermal inertia of the reactors.

The thermal reduction (Equation 1) is endothermic and takes place in a windowed reactor-receiver where the concentrated solar radiation directly heats the moving bed of particles. Bader et al. (2013) suggests a concentration ratio of 3000 to get a high efficiency ratio according the following equation:

$$\dot{Q}_{sol,0} = A_{rea} C_0 G_0, \quad (3)$$

The design parameters are defined as follows: $\dot{Q}_{sol,0} = 10^6$ W which is the design power at the receiver, $G_0 = 950$ W/m² that is the DNI and $C_0 = 3000$ which is the concentration ratio. This expression allows to obtain the diameter of the aperture (considered circular) of the receiver at the design point (0.67 m).

In order to get a suitable concentration ratio at the receiver aperture, a secondary concentrator is placed to increase the flux density of the radiation. A compound parabolic concentrator (CPC) has demonstrated high performance in this kind of processes (Pitz-Paal et al., 2011).

Table 1. System design specification.

<i>Solar resource</i>	
Location:	Geraldton (WA)
Longitude:	114.7°
Latitude:	-28.8°
Local time zone:	UCT+8
<i>Heliostat field</i>	
Heliostat size:	2.44 x 1.84 m
Number of heliostats:	604
Mirror reflectivity:	0.95
Soiling factor:	0.95
Heliostat availability:	0.99
<i>Solar tower</i>	
Design thermal power:	1 MW
DNI design value:	950 W
Tower height:	19.45 m
Receiver elevation:	-10°
Receiver acceptance angle:	70°
CPC aperture diameter:	1.16 m
Reactor aperture diameter:	0.67 m
Flux shape factor:	0.87
Solar concentration ratio:	3000

These devices, based on non-imaging optics, collect radiation entering the entrance aperture diameter (D_{CPC}) within angle of θ_{CPC} and direct it to the reactor aperture diameter (D_{rea}) with negligible losses (O'Gallagher and Winston, 1983). The relationship between the aperture angle and the concentration ratio is:

$$C_{CPC} = \frac{1}{\sin^2(\theta_{CPC})}, \quad (4)$$

and the relationship between both (CPC and reactor) aperture diameters is:

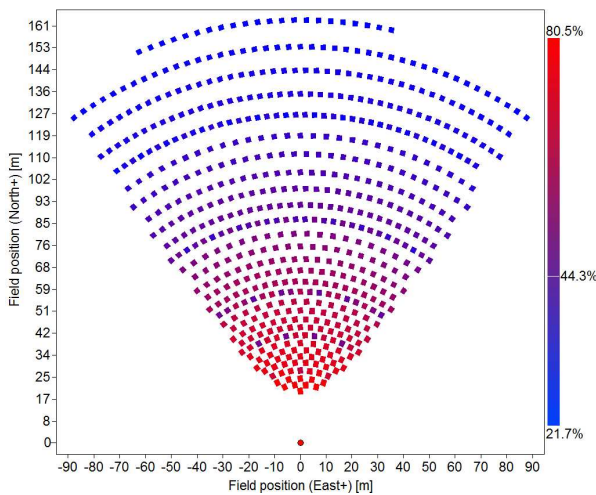
$$D_{CPC} = \frac{D_{rea}}{\sin(\theta_{CPC})}. \quad (5)$$

The typical values of the heliostat field concentrating ratio rounds 1000. In this respect, a value of 3 for the CPC concentrating ratio is required to provide the required design parameters. According this value, the acceptance angle (i.e. $2\theta_{CPC}$) is 70° and the CPC diameter 1.16 m.

SolarPILOTTM (NREL, 2016) was used to design and optimize the heliostat field. It allows fast generation and optimization of solar fields according a series of design parameters. Figure 1 shows the optimized solar field layout for this study. In addition, SolarPILOTTM provides the total optical efficiency of the solar field which includes cosine error, reflectivity and soiling, blocking and shading, atmospheric attenuation and scattering and spillage of a whole year as function of the zenith and azimuth solar angles (Table 2). The efficiency factor is calculated for a specific receiver, in this case, a 1.16 m side-squared receiver. In order to compensate the difference between

Table 2. Reference solar field optical efficiency as a function of Zenith and Azimuth angles.

$\theta_{zen} \setminus \theta_{azi}$	-150°	-120°	-90°	-60°	-30°	0°	30°	60°	90°	120°	150°	180°
0.5°	0.42947	0.42832	0.41992	0.40721	0.39708	0.39435	0.39443	0.39397	0.39521	0.40576	0.41858	0.42729
7°	0.37594	0.38236	0.39179	0.40135	0.40839	0.41095	0.40835	0.40135	0.39176	0.38262	0.37587	0.37355
15°	0.35315	0.36675	0.38640	0.40662	0.42085	0.42596	0.42087	0.40617	0.38656	0.36641	0.35311	0.34833
30°	0.30117	0.33196	0.37561	0.41417	0.43945	0.44756	0.43918	0.41437	0.37532	0.33190	0.30110	0.29084
45°	0.25081	0.30291	0.36867	0.41824	0.44617	0.45552	0.44604	0.41737	0.36826	0.30301	0.25063	0.23260
60°	0.20337	0.26952	0.34444	0.39892	0.43285	0.44481	0.43247	0.39877	0.34452	0.26885	0.20281	0.16978
75°	0.15311	0.20778	0.27510	0.33189	0.37052	0.37133	0.37056	0.33101	0.27444	0.20749	0.15253	0.11308
85°	0.09236	0.11869	0.14569	0.14285	0.15079	0.14289	0.15139	0.14214	0.14519	0.11818	0.09174	0.06352

**Figure 1.** Heliostat field layout.

both shapes, square and circle, a correction factor is applied. This shape factor, that is the fraction of the total concentrated power in both shapes, has a value of 0.87.

The oxygen generated during the reduction of CeO_2 should be removed in order to get an optimum reduction performance. O_2 is pushed out by a purge flow of high purity N_2 allowing reach a very low oxygen partial pressure inside the reactor.

The hydrogen production is accomplished at the oxidizer and depends on the temperature, the reduction degree of the moving particles of CeO_2 and the amount of water entering the reactor. In order to obtain a high production of hydrogen, this plant considers $\text{CeO}_{2-\delta}$ as the limiting reagent (Equation 2). It is expected that the residence time of the $\text{CeO}_{2-\delta}$ inside the oxidizer is sufficient to achieve the complete oxidation. A small tank of CeO_2 after the oxidizer allows a better management of the CeO_2 particles in the cycle.

In order to achieve a higher system efficiency, several heat recovery strategies were implemented. Two shell-and-tubes heat exchangers placed at the input of both reactors to recover the sensible heat of the gases. Furthermore, it is assumed that steam lines are pre-heated up to 200 °C in order to prevent condensation. Finally, a solid-solid heat exchanger is placed between both reactors (receiver and oxidizer) to recover the sensible heat of CeO_2 particles exiting the receiver as proposed in previous works (Bader et al., 2013; Ermanoski, 2015; Ermanoski et al.,

2013; Bulfin et al., 2015).

3 Object-oriented modelling

The model described in this section follows an object-oriented methodology based on equations. The main physical and chemical phenomena were identified and encapsulated into independent and reusable modules. These modules are connected creating hierarchical structures. This approach allows to study different plant configuration to improve the annual performance.

The model was implemented in Modelica language (Modelica Association, 2016) and is fully compatible with Modelica Standard Library (MSL). Modelica Fluid and Modelica Thermal connectors were used to define relationships between components. The thermodynamic properties of fluids are obtained from medium models that extend from Modelica Media Interface (MMI). All the sub-models are locally balanced ensuring robust modelling and debugging (Olsson et al., 2008).

3.1 Subsystem modelling

The system model that reproduces the plant described in §2, is presented in Figure 2. It consist on the following sub-models: data source, sun, heliostat field, receiver, oxidizer, tank, heat exchangers, pumps and valve. This model also includes: fluid source, fluid boundary, thermal source, real expression and medium sub-models.

General assumptions are summarized as follows:

- CeO_2 particle properties are assumed to be quasi-fluid.
- One-dimensional consideration within the direction of heat and mass flows.
- Heat conduction and radiation are negligible in fluid components. Axial heat flow is also negligible in both fluids.
- Lumped thermodynamic properties are assumed in fluid components.
- Chemical reactions only take place in receiver and oxidizer.

Receiver and oxidizer sub-models are fully described with complete set of equations in this work. The remaining models were obtained and adapted from exist-

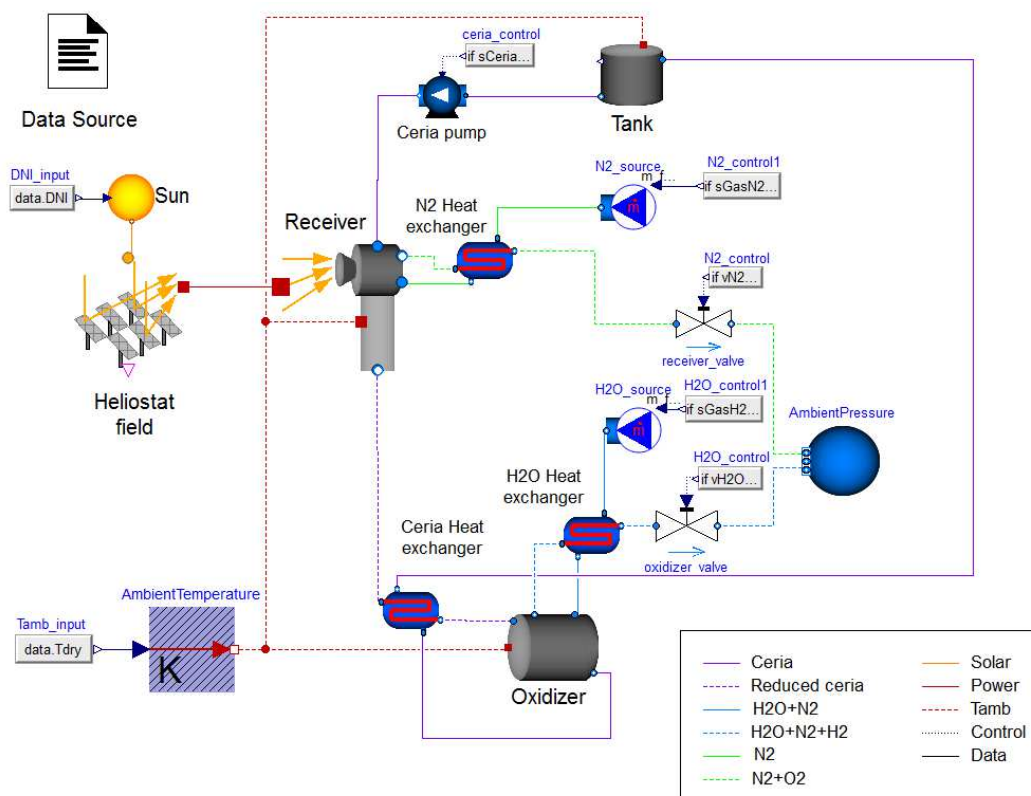


Figure 2. Modelica component diagram of the system model.

ing libraries. In fact, this demonstrates the high re-usability, extensibility and customizability of the modelling methodology used. The models of data source, sun, heliostat field and tank are been re-used from the open-source SolarTherm library (de la Calle et al., 2016a) with some adaptations and extensions. This library is available at <https://github.com/solartherm/solartherm> and consists on concentrating solar thermal (CST) components that are used to perform the annual simulations and the economic assessments of solar thermal plants. The models of heat exchangers and pumps are utilized and adapted from previous works of de la Calle et al. (2016b). The models of fluid source, fluid boundary, linear pressure drop valve, thermal source and real expression are included on the MSL and medium models extends from MMI.

A brief description of each one of the sub-models is provided below:

3.1.1 Medium models

Two medium models were implemented to describe gas mixture and Ceria properties. The gas medium is used in both reactors and extends from the `Modelica.Media.IdealGases.Common.MixtureGasNasa`. This medium is composed of water, oxygen, hydrogen and nitrogen at its gaseous state and assumes ideal gas properties provided by McBride et al. (2002).

The ceria medium model includes a function of the degree of reduction (δ) explicit in temperature and oxygen

partial pressure (Ermanoski et al., 2013). In addition, the medium includes a function for knowing the minimum required amount of water to achieve the oxidation based on the water equilibrium and heat of reaction explicit in δ (Bulfin et al., 2015).

3.1.2 Data source

This model encapsulates the extraction of weather data. It uses a `MSL's CombiTimeTable` with spline interpolation such that derivatives are continuous. The raw file is a typical meteorological year data set in the TMY3-file format (Wilcox and Marion, 2008). In order to be readable, the file is modified being compatible with Modelica specifications (Modelica Association, 2016).

3.1.3 Sun

This model provides the sun position relative to the plant location and the DNI in every time step. Users can choose between different correlations such as Duffie and Beckman (2013) or Blanco-Muriel et al. (2001) for calculating the declination and solar hour angles. The time variable matches with the local time where 0 s is 00:00 of 1st of January in this time zone. The DNI is provided by a RealInput connector.

3.1.4 Heliostat field

This model calculates the total concentrated solar power of the heliostat field (\dot{Q}_{sol}) as:

$$\dot{Q}_{sol} = N_{hel} A_{hel} \eta_{av} \eta_{op} G, \quad (6)$$

where the number of heliostats (N_{hel}), the heliostat area (A_{hel}) and the heliostat availability (η_{av}) are design parameters (Table 1). Solar angles and DNI (G) are provided by the SolarPort connector and the total optical efficiency (η_{op}) is calculated using the MSL's CombiTable2D with spline interpolation and the Table 2 as input. The start-up and the shutdown of the plant is automatically controlled according the minimum starting power, minimum operating power and the deploy angle, which are model parameters (Appendix I).

3.1.5 Receiver

This model provides the dynamic amount of CeO_2 reduced at the reactor. It is designed to perform annual simulations, therefore it is able to deal with zero-mass flow rates and zero mass. It is a lumped parameter model which assumes a single control volume (CV). The main particular assumptions are the following:

- Infinite thermal conductivity inside the reactor: same temperature at shell, CeO_2 and gas.
- Black body receiver approach: while radiative thermal losses are considered only at the reactor aperture, convective thermal losses are considered at all the external reactor surface.
- Perfect mixer approach: both inner CeO_2 particles and gas are perfectly mixed with their respectively accumulated masses.
- Pressure drop is neglected inside the reactor. The same pressure is assumed in all the CV.
- Constant inner molar flow rate of CeO_2 is assumed. The residence time of CeO_2 inside the reactor is constant.

The gas mass inside of the reactor (m_g) is determined by the inner gas ($\dot{m}_{g,in}$), the outer gas ($\dot{m}_{g,out}$) and the gas produced during the reduction (\dot{m}_{gen,O_2}):

$$\dot{m}_g = \dot{m}_{g,in} - \dot{m}_{g,out} + \dot{m}_{gen,O_2}. \quad (7)$$

The gas pressure (p) is determined by means of the ideal gas law:

$$pV_g = m_g k_g T, \quad (8)$$

where the specific gas constant (k_g) depends on the mass fraction of gases. The volume of the reactor (V) is constant and filled with CeO_2 (V_{ce}) and gas (V_g):

$$V = V_g + V_{ce}. \quad (9)$$

The oxygen mass balance is calculated according the mass fractions:

$$\dot{m}_{g,O_2} = \dot{m}_{g,in} X_{O_2,in} - \dot{m}_{g,out} X_{O_2} + \dot{m}_{gen,O_2}. \quad (10)$$

The outer mass fraction is the same as the CV mass fraction:

$$X_{O_2} = \max\left(0, \frac{m_{g,O_2}}{m_g}\right), \quad (11)$$

$$X_{H_2O} = 1 - X_{O_2}, \quad (12)$$

where the maximum function is used to avoid numerical problems. The oxygen generated during the reduction is calculated as function of the oxygen molecular mass (M_{O_2}) and the generated molar flow (\dot{n}_{gen,O_2}):

$$\dot{m}_{gen,O_2} = M_{O_2} \dot{n}_{gen,O_2}. \quad (13)$$

The generated molar flow depends on the degree of reduction (δ) and the inner CeO_2 molar flow ($\dot{n}_{ce,in}$):

$$\dot{n}_{gen,O_2} = \frac{\dot{n}_{ce,in} \delta}{2}. \quad (14)$$

The amount of CeO_2 (n_{ce}) inside the reactor is calculated by the molar balance:

$$\dot{n}_{ce} = \dot{n}_{ce,in} - \dot{n}_{ce,out}, \quad (15)$$

where the molar flow at outlet ($\dot{n}_{ce,out}$) is calculated according the following when-clause:

$$\dot{n}_{ce,out} = \begin{cases} \dot{n}_{ce,in} & \text{when } n_{ce} \geq n_{ce,max}, \\ 0 & \text{elsewhen } n_{ce} \leq 0. \end{cases} \quad (16)$$

The maximum number of moles ($n_{ce,max}$) is calculated according the maximum volume of CeO_2 which is a model parameter (Appendix I).

The mass of CeO_2 (m_{ce}) is determined by its molar mass (M_{ce}) which depends on delta:

$$m_{ce} = M_{ce} n_{ce}, \quad (17)$$

and the volume by:

$$V_{ce} = \frac{m_{ce}}{\rho_{ce}}. \quad (18)$$

δ is calculated as function of the temperature and the partial pressure of oxygen (p_{O_2}). The maximum value of δ is limited to 0.25 which is the maximum value possible to maintain the fluorite structure of CeO_2 . The partial pressure is calculated as:

$$p_{O_2} = \frac{m_{O_2}}{M_{O_2}} p. \quad (19)$$

The temperature of the reactor is calculated according to the global energy balance:

$$\eta_{sh} \dot{Q}_{sol} - \dot{Q}_{loss} = \Delta \dot{Q}_{re} + \Delta \dot{Q}_g + \Delta \dot{Q}_{ce}, \quad (20)$$

where the concentrated solar power (\dot{Q}_{sol}) coming from the heliostat field is attenuated by the shape factor (η_{sh}).

The heat loss (\dot{Q}_{loss}) is the sum of the radiative ($\dot{Q}_{loss,rd}$) and convective losses ($\dot{Q}_{loss,cv}$):

$$\dot{Q}_{loss,rd} = A_{ap} \sigma \xi (T^4 - T_{amb}^4), \quad (21)$$

$$\dot{Q}_{loss,cv} = A_{re} \alpha (T - T_{amb}). \quad (22)$$

The radiative losses are only applicable to the aperture area of the reactor due to the lack of thermal insulation, and the convective losses to the whole reactor. Emissivity (ξ) and heat transfer coefficient (α) are model parameters and the ambient temperature (T_{amb}) is an input of the model.

The receiver mass contribution into the energy balance ($\Delta\dot{Q}_{re}$) is mostly due to the thermal inertia of the metal cover:

$$\Delta\dot{Q}_{re} = m_{re} C_{p,re} \dot{T}, \quad (23)$$

where the mass (m_{re}) and the specific heat capacity ($C_{p,re}$) are model parameters. The gas contribution into the energy balance is determined by:

$$\Delta\dot{Q}_g = m_g C_{p,g} \dot{T} + \dot{m}_{g,in} (h_g - h_{g,in}). \quad (24)$$

where it is assumed that the outlet temperature is the same as the temperature inside the reactor. The following equation calculates the CeO₂ contribution into the energy balance:

$$\Delta\dot{Q}_{ce} = m_{ce} C_{p,ce} \dot{T} + \dot{m}_{ce,in} (h_{ce} - h_{ce,in}) - n_{ce,in} \delta Q_{red}. \quad (25)$$

where it is also assumed that the outlet temperature is the same as the temperature inside the reactor and heat of reduction (Q_{red}) depends on δ .

3.1.6 Oxidizer

This model dynamically provides the amount of hydrogen produced. It is a lumped parameter model (1 CV) similar to the receiver. The particular assumptions are the same as for the receiver but in this case, due to the lack of an aperture, only convection losses are been considered. The main assumption is the complete oxidation of the CeO₂.

The gas mass inside of the reactor is determined by the inflowing gas, the outflowing gas, the gas produced (\dot{m}_{gen,H_2}) and the consumed at the oxidation (\dot{m}_{con,H_2O}):

$$\dot{m}_g = \dot{m}_{g,in} - \dot{m}_{g,out} + \dot{m}_{gen,H_2} - \dot{m}_{con,H_2O}. \quad (26)$$

The oxidizer pressure and the volume calculation are determined by Equations 8 and 9.

The mass balances of nitrogen, hydrogen and water are calculated according the mass fractions:

$$\dot{m}_{g,N_2} = \dot{m}_{g,in} X_{N_2,in} - \dot{m}_{g,out} X_{N_2}, \quad (27)$$

$$\dot{m}_{g,H_2} = \dot{m}_{g,in} X_{H_2,in} - \dot{m}_{g,out} X_{H_2} + \dot{m}_{gen,H_2}, \quad (28)$$

$$\dot{m}_{g,H_2O} = \dot{m}_{g,in} X_{H_2O,in} - \dot{m}_{g,out} X_{H_2O} - \dot{m}_{con,H_2O}. \quad (29)$$

The outflowing mass fractions are determined by:

$$X_{N_2} = \min \left(1, \frac{m_{g,N_2}}{m_g} \right), \quad (30)$$

$$X_{H_2} = \max \left(0, \frac{m_{g,H_2}}{m_g} \right), \quad (31)$$

$$X_{H_2O} = \min \left(1, \frac{m_{g,H_2O}}{m_g} \right), \quad (32)$$

where maximum and minimum functions are used for preventing numerical problems.

The mass flows due to the oxidation are:

$$\dot{m}_{gen,H_2} = M_{H_2} \dot{n}_{gen,H_2}, \dot{m}_{con,H_2O} = M_{H_2O} \dot{n}_{con,H_2O}, \quad (33)$$

where the molar flows depends on the degree of reduction of the inflowing CeO₂:

$$\dot{n}_{con,H_2O} = \frac{\dot{n}_{ce,in} \delta}{2}, \quad (34)$$

$$\dot{n}_{gen,H_2} = \dot{n}_{con,H_2O}. \quad (35)$$

Equations 15-18 are used in this model to calculate the CeO₂ mass and volume dynamics. The temperature is calculated according to the global energy balance:

$$-\dot{Q}_{loss,cv} = \Delta\dot{Q}_{re} + \Delta\dot{Q}_g + \Delta\dot{Q}_{ce}, \quad (36)$$

where Equations 22-24 provide the heat losses, the receiver mass contribution and the gas contribution to the energy balance. The CeO₂ contribution is determined by:

$$\Delta\dot{Q}_{ce} = m_{ce} C_{p,ce} \dot{T} + \dot{m}_{ce,in} (h_{ce} - h_{ce,in}) - n_{ce,in} \delta Q_{ox,ce}. \quad (37)$$

where heat of oxidation is assumed as $Q_{ox} = -Q_{red}$.

3.1.7 Tank

This model introduces the dynamics of a small storage element which pressure is fixed parametrically. It is a lumped parameter model which assumes a cylinder volume and an ideally mixed fluid. The mass balance is:

$$\dot{m} = \dot{m}_{in} - \dot{m}_{out}, \quad (38)$$

and the energy balance is:

$$\dot{m} \dot{h} = \dot{m}_{in} (h_{in} - h) - \dot{Q}_{loss}, \quad (39)$$

where shell capacitance is neglected. The convective heat losses to the environment are only applied to the metal surface that is in contact with the fluid.

3.1.8 Heat exchanger

This quasi-steady-state heat exchanger model allows to calculate sensible heat transfer between two fluids based on the mathematical development of Spakovszky (2008) and whose implementation was performed by de la Calle et al. (2016b). It is a lumped parameter model which assumes that every state at the heat exchanger is locally steady. The model is able to manage zero-mass flow rates.

3.1.9 Pump

This ideal pump model provides a controlled mass flow rate between two points in the same streamline.

3.1.10 Linear valve

This model calculates the mass flow rate that crosses through the valve opening with a linear approximation of the pressure drop.

3.1.11 Fluid source

This model provides an input boundary condition where the mass flow rate, the specific enthalpy and mass fraction are defined. It is used to simulate the inflowing gas at the reactors.

3.1.12 Fluid boundary

This model provides an output boundary condition where the pressure, the specific enthalpy and mass fraction are defined. It is used to simulate the environment (fluid models).

3.1.13 Thermal source

This model provides a thermal boundary condition where the temperature is defined. It is used to simulate the environment (thermal models).

3.1.14 Real expression

It is a model used to connect experimental data as inputs of the models in a graphical way.

3.2 Automatic control system

The automatic control system (ACS) is designed to guarantee the stability of the plant in annual simulations. This system is made up by a series of on-off controllers which control the circulation of the fluids inside the plant.

The ACS must prevent the reverse flow of gases at valves. For this reason, the valves are only opened when the pressure drop is higher than half of the nominal pressure drop.

The gas source is opened since the heliostat field reaches the start-up power and it is closed when the heliostat field is shut down and the reactor temperatures are below a certain shutdown temperature.

The ceria pump starts when the heliostat field is started, both gas valves are open and the receiver temperature is higher than a minimum operating temperature. In order to take advantage of the thermal inertia of the reactors, the pump shut down when the heliostat field is shut down and the receiver temperature is below the minimum operating temperature.

4 Simulation

Dymola 2017 (Dassault Systemes, 2016) was the tool used for the Modelica implementations and simulations. The numerical solver used for the dynamic simulations has been DASSL (Petzold, 1983) whose absolute and relative tolerances were set to 10^{-4} .

The model is a set of high-index differential and algebraic equations (DAEs) of 745 scalar variables. After the translation, the model has 255 time-varying variables and 14 continuous-time states.

The numerical value of model parameters can be reviewed in Appendix I. The annual performance of the plant is very sensible to the operating parameters. Few operating parameters have been optimised: CeO_2 mass flow rate, inlet receiver gas flow rate and inlet oxidizer mass flow rate and its composition. The optimization method was the Simplex algorithm and the objective function was the final hydrogen production.

The annual simulation was performed using weather data for the Geraldton location provided by AUSTELA (2016). The CPU-time for integration was 161 s for the whole year simulated with 9997 state events mostly related with the ACS.

The amount of hydrogen produced during the simulated year is 46.57 t. The variation in time of this production is depicted in Figure 3, where the different seasonal rate (winter-summer) can be observed. The solar to hydrogen efficiency of the plant, defined as:

$$\eta = \frac{m_{\text{H}_2} \text{HHV}}{Q_{\text{sol}}}, \quad (40)$$

where m_{H_2} is the annual amount of hydrogen, HHV is the hydrogen heating value and Q_{sol} is the annual amount of energy that reach the heliostat field, has a value of 25.27%. This result is in line with previous works (Bader et al., 2013; Ermanoski, 2015; Ermanoski et al., 2013; Bulfin et al., 2015).

Figure 4 shows the simulation details of 5 days (from 28 August to 1 September). This week has one sunny day (240), three partially cloudy days (241, 243 and 244) and one completely cloudy day (242) (Figure 4(a)). The concentrated solar power by the heliostat field is shown in Figure 4(b). The plant does not use all the available energy (\dot{Q}_{raw}) because a minimum start-up power and minimum operating power are applied. Although 1.2 MW of peak power is reached 2 days, only few hour per day the power is higher than 1 MW. At the day 242, the system did not achieve the start-up power in the whole day and at

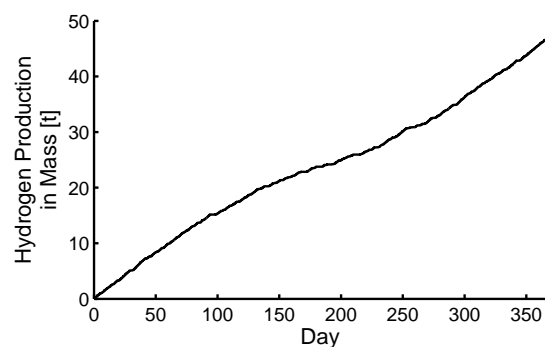


Figure 3. Annual hydrogen production.

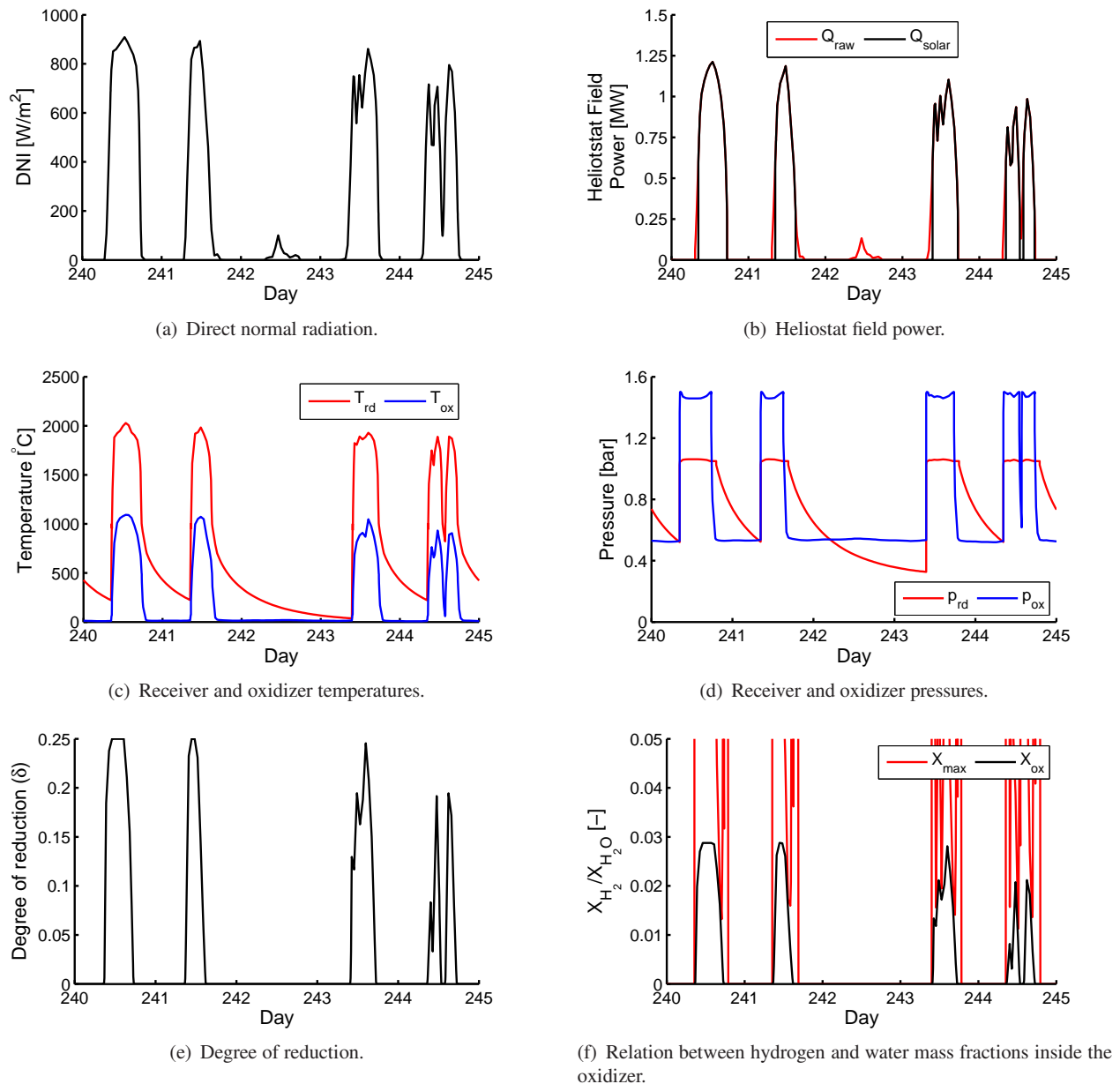


Figure 4. Simulation details of 5 days.

days 243 and 244 the peak power barely reached 1 MW. Figure 4(c) shows the CeO₂ temperatures inside the receiver and the oxidizer. The operating reduction temperature is ranged between 1850 – 1950 °C and the operating oxidation temperature is around 1000 – 1100 °C. When the plant is shut down, the temperature decreased quickly and 32 hours after is close to the ambient temperature. The pressure inside both reactors is shown in Figure 4(d). While the receiver works at ambient pressure, the oxidizer nominal pressure has been set to 1.5 bar in order to assure higher pressure than ambient when the hydrogen production is large. The degree of reduction is depicted in Figure 4(e), where peaks of 0.25 can be observed. For achieving the total oxidation of the CeO₂, a minimum amount of water per hydrogen released is required inside the ox-

idizer (Bulfin et al., 2015). In Figure 4(f) is depicted as x_{max} (the mass fraction of hydrogen into water) and it limits the amount of hydrogen produced at the oxidizer with the water used. The figure shows that in the whole simulation, the hydrogen released (x_{ox}) is lower than the maximum amount expected at the equilibrium.

5 Conclusions

In this work, a dynamic model of a solar hydrogen plant based on the CeO₂ redox cycle has been presented. The model has been developed with an object-oriented modelling methodology that it allows the re-used of several work previously developed. The system is design to study the transient behaviour of the plant in annual simulations. It was implemented with the Modelica language and sim-

ulated with Dymola 2017. A basic automatic control system based on on-off controllers to guarantee the system stability was included.

The model predictions are reasonable and some usual simulation problems like zero-mass flows were solved with effectiveness. The computational effort of the model is low, therefore it can be used in optimization and control studies.

Increasing the model accuracy should be the objective of next works. The model reveals that thermal inertia of the CeO_2 is too much high to accomplish the oxidation without extract heat flow from the reactor. The results suggest to review the assumptions related with heat losses and design a cooling systems at the oxidizer. The optimization of the plant through few operating parameters has demonstrated the flexibility of the system to be improved. Future studies should include operating cost and advanced operating strategies.

Acknowledgement

This work was performed as part of the ASTRI, a project supported by the Australian Government, through the Australian Renewable Energy Agency (ARENA).

Appendix I: Model parameters

References

- Stéphane Abanades, Patrice Charvin, Gilles Flamant, and Pierre Neveu. Screening of water-splitting thermochemical cycles potentially attractive for hydrogen production by concentrated solar energy. *Energy*, 31:2469–2486, 2006. doi:10.1016/j.energy.2005.11.002.
- Simon Ackermann and Aldo Steinfeld. Diffusion of oxygen in Ceria at elevated temperatures and its application to $\text{H}_2\text{O}/\text{CO}_2$ splitting thermochemical redox cycles. *The Journal of Physical Chemistry Journal of*, 118, 2014.
- AUSTELA. The NREL System Advisor Model for Australian CSP Stakeholders (SAM), 2016. URL <http://www.austela.org.au/projects>.
- Roman Bader, Luke J. Venstrom, Jane H. Davidson, and Wojciech Lipiński. Thermodynamic analysis of isothermal redox cycling of ceria for solar fuel production. *Energy and Fuels*, 27(9):5533–5544, 2013. doi:10.1021/ef400132d.
- Manuel Blanco-Muriel, Diego C. Alarcón-Padilla, Teodoro López-Moratalla, and Martín Lara-Coira. Computing the solar vector. *Solar Energy*, 70(5):431–441, 2001. doi:10.1016/S0038-092X(00)00156-0.
- B. Bulfin, F. Call, M. Lange, O. Lübben, C. Sattler, R. Pitz-Paal, and I. V. Shvets. Thermodynamics of CeO_2 thermochemical fuel production. *Energy and Fuels*, 29(2):1001–1009, 2015. doi:10.1021/ef5019912.
- William C Chueh, Christoph Falter, Mandy Abbott, Danien Scipio, Philipp Furler, Sossina M Haile, and Aldo Steinfeld. High-flux solar-driven thermochemical dissociation of CO_2 and H_2O using nonstoichiometric ceria. *Sci-*

<i>Heliostat field</i>	
Start-up power:	0.6 MW
Shutdown power:	0.3 MW
Deploy angle:	8°
<i>Ceria pump</i>	
Mass flow rate:	2.5 kg/s
Shutdown Temperature:	1000 °C
<i>Ceria Heat exchanger</i>	
Heat transfer coefficient:	500 W/(m ² K)
Area of exchange:	5 m ²
<i>Receiver</i>	
Reactor mass:	500 kg
Diameter:	0.67 m
Lenght:	0.34 m
Maximum CeO_2 volume:	25%
Emitance:	0.88
Convective coefficient:	10 W/(m ² K)
<i>Oxidizer</i>	
Reactor mass:	500 kg
Diameter:	0.67 m
Lenght:	0.34 m
Maximum CeO_2 volume:	25%
Convective coefficient:	500 W/(m ² K)
<i>Tank</i>	
Diameter:	0.5 m
Height:	0.5 m
Convective coefficient:	10 W/(m ² K)
<i>N₂ Source</i>	
Gas Mass flow rate:	0.25 kg/s
Shutdown temperature:	700 °C
<i>N₂ valve</i>	
Design pressure drop:	0.05 bar
Closing pressure drop:	0.025 bar
<i>N₂ Heat exchanger</i>	
Heat transfer coefficient:	500 W/(m ² K)
Area of exchange:	5 m ²
<i>H₂O Source</i>	
Gas Mass flow rate:	0.71 kg/s
H ₂ O Mass fraction:	0.45
Shutdown temperature:	400 °C
<i>H₂O valve</i>	
Design pressure drop:	0.5 bar
Closing pressure drop:	0.25 bar
<i>H₂ Heat exchanger</i>	
Heat transfer coefficient:	500 W/(m ² K)
Area of exchange:	5 m ²

- ence (New York, N.Y.), 330(6012):1797–801, dec 2010. doi:10.1126/science.1197834.
- Dassault Systemes. Dymola 2017 - Dynamic Modeling Laboratory, 2016. URL www.3ds.com.
- Alberto de la Calle, Jim Hinkley, Paul Scott, and John Pye. SolarTherm : A New Modelica Library and Simulation Platform for Concentrating Solar Thermal Power Systems. *Proc. 9th EUROSIM Congress on Modelling and Simulation*, pages 1–2, 2016a. doi:10.1109/EUROSIM.2016.162.
- Alberto de la Calle, Lidia Roca, Javier Bonilla, and Patricia Palenzuela. Dynamic modeling and simulation of a double-effect absorption heat pump. *International Journal of Refrigeration*, 72:171–191, 2016b. doi:10.1016/j.ijrefrig.2016.07.018.
- Lawrence D'Souza. Thermochemical hydrogen production from water using reducible oxide materials: a critical review. *Materials for Renewable and Sustainable Energy*, 2(1):7, feb 2013. doi:10.1007/s40243-013-0007-0.
- John A. Duffie and William A. Beckman. *Solar Engineering of Thermal Processes*. Wiley, New York, USA, 4th edition, 2013. ISBN 9780470873663. doi:10.1002/9781118671603.
- Ivan Ermanoski. Maximizing Efficiency in Two-step Solar-thermochemical Fuel Production. *Energy Procedia*, 69:1731–1740, 2015. doi:10.1016/j.egypro.2015.03.141.
- Ivan Ermanoski, Nathan P. Siegel, and Ellen B. Stechel. A New Reactor Concept for Efficient Solar-Thermochemical Fuel Production. *Journal of Solar Energy Engineering*, 135(3):031002, 2013. doi:10.1115/1.4023356.
- Edward A. Fletcher. Solarthermal Processing: A Review. *Journal of Solar Energy Engineering*, 123(May 2001):63, 2001. doi:10.1115/1.1349552.
- Philipp Furler, Jonathan R. Scheffe, and Aldo Steinfeld. Syn-gas production by simultaneous splitting of H₂O and CO₂ via ceria redox reactions in a high-temperature solar reactor. *Energy & Environmental Science*, 5(3):6098, 2012. doi:10.1039/c1ee02620h.
- Xiang Gao, Alejandro Vidal, Alicia Bayon, Roman Bader, Jim Hinkley, Wojciech Lipiski, and Antonio Tricoli. Efficient ceria nanostructures for enhanced solar fuel production: Via high-temperature thermochemical redox cycles. *Journal of Materials Chemistry A*, 4(24):9614–9624, 2016. doi:10.1039/c6ta02187e.
- IRENA and IEA-ETSAP. Technology Brief 4: Thermal Storage. Technical Report January, 2013.
- Jiyong Kim, Terry a. Johnson, James E. Miller, Ellen B. Stechel, and Christos T. Maravelias. Fuel production from CO₂ using solar-thermal energy: system level analysis. *Energy & Environmental Science*, 5(9):8417, 2012. doi:10.1039/c2ee21798h.
- Nathan S Lewis and Daniel G Nocera. Powering the planet: Chemical challenges in solar energy utilization. *PNAS*, 104(42):15729–15735, 2007.
- Bonnie J. McBride, Michael J. Zehe, and Sanford Gordon. NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species. Technical Report NASA/TP-2002-211556, National Aeronautics and Space Administration (NASA), Cleveland OH, USA, 2002.
- Modelica Association. Modelica Specification 3.3, 2016. URL www.modelica.org/documents.
- Christopher L. Muhich, Brian D. Ehrhart, Ibraheam Al-Shankiti, Barbara J. Ward, Charles B. Musgrave, and Alan W. Weimer. A review and perspective of efficient hydrogen generation via solar thermal water splitting. *Wiley Interdisciplinary Reviews: Energy and Environment*, pages n/a–n/a, 2015. doi:10.1002/wene.174.
- NREL. The Solar Power Tower Integrated Layout and Optimization Tool (SolarPILOT), 2016. URL <http://www.nrel.gov/csp/solarpilot.html>.
- J. O'Gallagher and R. Winston. Development of compound parabolic concentrators for solar energy. *International Journal of Ambient Energy*, 4(4):171–186, oct 1983. doi:10.1080/01430750.1983.9675885.
- Hans Olsson, Martin Otter, Sven Erik Mattsson, and Hilding Elmqvist. Balanced Models in Modelica 3.0 for Increased Model Quality. In *Proc. 6th International Modelica Conference*, pages 21–33, Bielefeld, Germany, 2008.
- Linda R. Petzold. A description of DASSL: a Differential/Algebraic System Solver. *Scientific Computing*, pages 65–68, 1983.
- Robert Pitz-Paal, Nicolas Bayer Botero, and Aldo Steinfeld. Helio-stat field layout optimization for high-temperature solar thermochemical processing. *Solar Energy*, 85(2):334–343, 2011. doi:10.1016/j.solener.2010.11.018.
- R. Ramachandran and R. K. Menon. An overview of industrial uses of hydrogen. *International Journal of Hydrogen Energy*, 23(7):593–598, 1998. doi:10.1016/S0360-3199(97)00112-2.
- Martin Roeb, Martina Neises, Nathalie Monnerie, Friedemann Call, Heike Simon, Christian Sattler, Martin Schmücker, and Robert Pitz-Paal. Materials-Related Aspects of Thermochemical Water and Carbon Dioxide Splitting: A Review. *Materials*, 5(12):2015–2054, oct 2012. doi:10.3390/ma5112015.
- Jonathan R. Scheffe and Aldo Steinfeld. Thermodynamic Analysis of Cerium-Based Oxides for Solar Thermochemical Fuel Production. *Energy & Fuels*, 26(3):1928–1936, mar 2012. doi:10.1021/ef201875v.
- Z. S. Spakovszky. Unified Engineering: Thermodynamics and Propulsion, 2008. URL web.mit.edu/16.unified.
- S. Wilcox and W. Marion. Users manual for TMY3 data sets. Technical Report NREL/TP-581-43156, The National Renewable Energy Laboratory (NREL), Golden CO, USA, 2008.

Applying the Power Plant Library ClaRa for Control Optimisation

Friedrich Gottelt¹ Timm Hoppe¹ Lasse Nielsen²

¹XRG Simulation GmbH, Germany, {gottelt, hoppe}@xrg-simulation.de

²TLK Thermo GmbH, Germany, l.nielsen@tlk-thermo.com

Abstract

This paper presents the current state of the open-source Modelica library *ClaRa* which enables the user to solve complex power plant simulation tasks. The library can be used to help control experts to develop and test controllers without disturbing the daily operation of the applying power plant. This reduces project risks and costs significantly. As a use case the analysis and optimisation of a Benson boiler power control is presented. The presented solution reduces the impact of soot blowing on the power output by 57 %.

Keywords: Power plant, Clausius-Rankine cycle, open-source library, control optimisation, soot blowing, simulation

1 Introduction

The global energy markets are in a phase of significant changes due to increasing power production from renewable sources like solar and wind power, see (International Energy Agency, 2015). These renewables are usually fluctuating energy inputs as they depend on local weather. In order to ensure a stable grid operation conventional power suppliers have to outbalance these fluctuations. This introduces new modes of operation to new and existing power plants. The questions and challenges that arise from these new operation modes can be solved at low costs with the help of system simulation. Therefore, it is very likely that activities in this field will grow in both, university-based and industry research and development.

The most recent Modelica library in this field is the library *ClaRa* which was developed from 2011 in a German collaboration¹ of Hamburg University of Technology, TLK-Thermo GmbH and XRG Simulation GmbH. Its first official release of version 1.0.0 dates from March 2015 (see (Brunnemann et al., 2012; Gottelt et al., 2012) for an introduction to *ClaRa* and a control-related application). The aim of this development was to provide a library that is both, suitable for beginning and advanced researchers in the field of simulation with Modelica. This leads to special requirements with respect to usability and flexibility which are described in section 2. Currently the library is under ongoing development within the follow up project Dynstart². New features, for example a parameter based

initialisation concept, have been added and the library is going to be enhanced to handle extra-low-loads, start-up and shut-down processes. To illustrate the potential of the library, the challenging task of a soot blowing simulation is presented as use case in this paper.

2 Overview of the Library ClaRa

2.1 Scope of Library

The library *ClaRa* was created to enable simulation of large steam power plants with coal dust firing. At the heating side the library comprises models for the complete fuel handling process from the grinding via the fuel combustion to the flue gas cleaning. At the water steam side the library features models for the cooling of the combustion chamber to the electrification of the steam in the turbo generator.

The library is intended to be the centre of a whole family of Modelica libraries in the field of electricity production and consumption. Any extension of *ClaRa* will be handled in a separate, so-called *ClaRa_AddOn*. This avoids the constant growing of the library, limits its scope and reduces the effort of library maintenance since not all add-ons must be fully compatible to each other. Especially the last mentioned aspect eases the library development by external developers. Two *ClaRa_AddOns* are currently under development in close collaboration with the *ClaRa* developers: *ClaRa_Control* supplies blocks for an efficient implementation of state of the art power plant process control including the start-up and shut-down processing. *Transient* widens the scope of *ClaRa* by the energy distribution and allows for the simulation of strongly coupled electric grids, gas grids and district heating grids as well as its economic evaluation (Andresen et al., 2015).

Consequently, also the *ClaRa* library makes use of two external libraries, namely the open source library *Fluid-Dissipation* (Vahlenkamp and Wischhusen, 2008, 2009) providing functional descriptions of pressure losses and heat transfer and the free-of-charge version of the *TILMedia* (Schulze, 2014) providing media data for water, CO₂ and gaseous flue gas and air mixtures.

2.2 Structure of Library

The structure of the library follows a functional approach rather than structuring according to the used media. For

¹Funded by the German Ministry for Economic Affairs and Energy under reference number FKZ 03ET2009








²Funded by the German Ministry for Economic Affairs and Energy

under reference number FKZ 03ET7060E

example, a steam-heated shell-and-tube heat exchanger for the preheating of feedwater will be found in the same package `HeatExchanger` as a flue gas to fresh air heat exchanger.

Table 1 gives an overview of the top level package structure. Herein, the `UsersGuide` provides a brief introduction to the library as well as developer's contact data and information on the license model and the revisions. The package `Examples` provides a number of introductory examples making new users familiar with the capabilities of the library. The package `Basics` contains basic models and other internally used codes like functions, records and interfaces. Most users will not get in touch often with this package. In contrast, the package `Components` contains all the component models required to build up a power plant model.

Table 1. ClaRa library structure

	UsersGuide	Information on basic modeling concepts, contact and license
	Examples	Introducing examples
	Basics	Basic models and informatics
	Components	Models for turbo machines and electrical machines, connection pipings, heat exchanger, mass storage and steam separation, valves, coal grinding, furnace, flue gas cleaning, and sensors, i.e. "the core of the library"
	SubSystems	Conceptual package aiming at supporting team work
	Visualisation	Auxiliaries for the visualisation of results
	StaticCycles	Static models for the calculation of consistent initial guess values

The package `SubSystems` contains some examples for the definition of subsystems. This package is still somewhat conceptual since it does not feature a complete set of reasonable sub systems but rather aims at introducing ideas for an efficient team work. The package `Visualisation` provides means to visualise the results using both, Modelica-based annotations and third party post processing tools. The section 2.5 gives a brief overview of the options. Finally, the package `StaticCycles` contains simplified, static and parameter-based models of most of the power plant components. Details on this package and the idea for its application within *ClaRa's* dynamic modelling approach are discussed in section 2.6

2.3 Overview of Component Modelling

One aspect of the *ClaRa* modelling concept is to provide as much physics as applicable to achieve a close to reality model behaviour of single components as well as whole power plant models. Different modelling approaches are used throughout the library, for example a finite volume approach for pipes, flow models for valves, zonal models for reservoirs and characteristic maps for compressors and pumps while almost all components are using balance equations.

The different models are reasonably modular in structure avoiding doubled code and providing high code transparency. Another aspect in model structuring is to gain flexibility in the model application. The structuring is done according to the following concepts:

Models at different levels of detail are provided as separate classes, they can be exchanged in many cases (e.g. via Dymola's context menu "change class...") since they have equal interfaces. However, its internal structure is significantly different providing a differently detailed view into the component.

For instance, a pipe can be modelled applying an integrated, slim set of balance equations (as described in (Velut and Tummescheit, 2011)) or its balance equations can be discretised applying a finite volume approach. The first mentioned approach will be very efficient for the simulation of water hammer effects but loosing information about the internal, local fluid temperature and pressure distribution. The latter mentioned approach is very robust for reverse flow and off-design conditions and gives detailed information about local and temporal effects like steaming but will be less efficient for very long pipes.

Physical effects at different levels of insight are provided to allow to apply a single class to be instantiated in different contexts. For instance, the pressure loss of a valve might be calculated either considering supercritical or subcritical flow conditions. In other cases, if the valve's specific behaviour is not of particular interest it might be sufficient to simply consider a linear hydraulic correlation. All this is handled in different *replaceable* models.

Basic models are reused by instantiation to form new component models. For instance, a shell-and-tube heat exchanger model is a compound of the shell's volume, the tube's volumes and the separating wall. *ClaRa* provides basic models for these sub-compounds which makes the models very easy to understand and to maintain and allows users to easily create new component models.

The component list is complete for most tasks around evaluation of the normal power plant operation of both once-through boilers and circulation boilers. Heating can be realised either by coal dust firing or heat recovery (other heat sources like solar energy or biomass firing will be subject to *ClaRa_AddOns*). However, future releases will increase this content further, e.g. there will be a cooling tower enabling studies on the performance of the cold side

of the process.

2.4 Media Supported

ClaRa is shipped with a non-profit version of the *TILMedia* featuring four different media types³. For pure mediums like water/steam there are table based and spline interpolated data available which are very encouraging concerning simulation speed and simulation stability, see (Schulze, 2013). The flue gas is described by a gas-vapour mixture with eight substances similar to humid air. A mixture of real fluids for application in CO₂ separation processes is supported as well as pure CO₂ for ORC applications.

The calculation of substance properties within *TILMedia* is applied by external C code for faster simulation speed and reusability. The robust media formulation has also a positive effect on the initialisation of *ClaRa* models because media evaluation outside of the range of validity is handled by extrapolation or limitation to numerical range of validity to prevent division by zero or infinity. Significant less crashes were experienced compared to work with other media libraries.

2.5 Visualisation and Usability

The usability is a key point to ease new users the introduction to the library. Therefore, *ClaRa* component models feature well-structured parameter dialogues that make extensive use of Modelica's dialogue structuring annotations like "tab" and "group". This helps the user to distinguish between expert settings and fundamental, geometric settings of a component. Where applicable, descriptive figures help to understand the technical context.

Modelica libraries often provide only brief documentation thus implying that the source code is self-declarative. Although this might be true for certain libraries available it is often an obstacle for those who are less familiar either with commonly used modelling techniques or the programming language used. *ClaRa* provides a comprehensive documentation for these users aiming at deepening the understanding of the work and improving the confidence in the library by granting a maximum of transparency. The documentation gives detailed insight into the underlying theory and explains expert user settings and spent validation efforts.

In order to help users to keep track of very complex power plant process designs a number of visualisers can be included into the models, see Figure 1. These items help to better understand the current state of the process by visualising important process variables (pressure, temperature, spec. enthalpy and mass flow and levels) and highlighting critical values like negative or zero flows.

2.6 Global Initialisation

The initialisation of a large differential-algebraic system of equations is a challenge that has been discussed in

³There are more media available at costs for a commercial *TILMedia* license

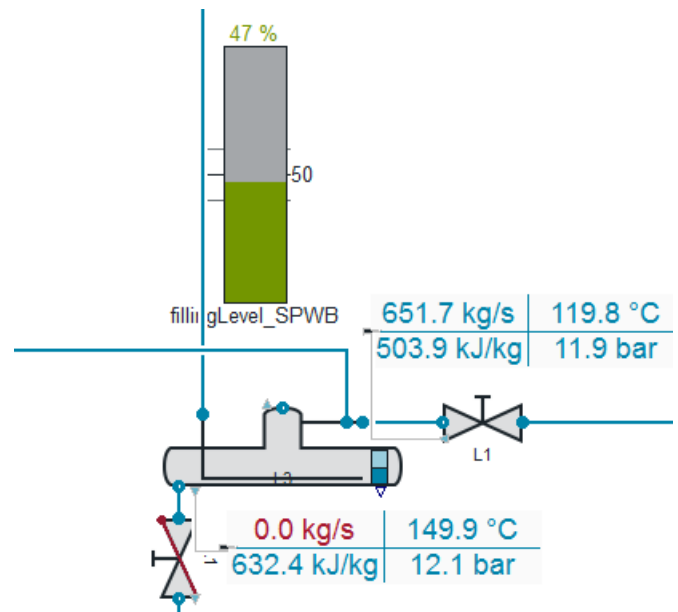


Figure 1. Screenshot of *ClaRa* visualisation

the Modelica community from the very beginning, e.g. (Mattsson et al., 2002; Bachmann et al., 2006; Najafi, 2008). Since initialisation can often be a time-consuming and frustrating task, especially for beginners, a library's quality can also be measured in its features that support the user in getting robustly and reproducibly initialising models. For *ClaRa*'s implementation five aspects are considered to ease the initialisation process: First, a new user will expect to get some kind of guidance in the task of initialisation, i.e. it should be clear for which variables the user should provide estimation values and for which not. Second, the library should support the homotopy concept that was proved to be advantageous according to (Casella et al., 2011). Third, the available initialisation options shall provide a flexibility to initialise models in arbitrary combinations. Fourth, the initialisation should be reliable, i.e. it should not be sensitive to smaller model changes. This point also refers to the accuracy that estimated values must have. Finally, taking the system topology into account is a valuable feature that would significantly improve the initialisation process. The last point is not straightforward since it counteracts the modular modelling principles Modelica is based on.

If we take a look on a dynamic, 0D T-join for example. The volume normally is initialised by applying estimation values for the state variables, i.e. pressure and specific enthalpy provided by the user as parameters. In principle, these parameters are defined locally and they are completely independent from its neighbouring components. However, from a technical point of view it is clear that it would be useful to take the neighbours into account since the mixer's specific enthalpy will be the weighted mixing enthalpy of the two inlet enthalpy flows coming from the upstream components.

The *ClaRa* approach to take this kind of informa-

tion about the system topology into account is the *StaticCycle* package. This package contains a set of stationary models which can be used to create a simplified, static and parameter-based mimic of the dynamic cycle. The result is a consistent, load depending set of parameters for mass flow, pressure and specific enthalpy or temperature for the complete cycle. Linking the results of this static cycle to the respective initial guess values in the dynamic cycle allows to give flexible and consistent initial values at all dynamic components considering system topology and the possibility to use a cascaded initialisation with values of upstream neighbouring components. This feature allows for a very robust initialisation with automated adaptation on varying design points.

Overview of the *StaticCycle* package features:

- parameter based
- signal connectors
- different input/output combinations for different assembly yield four differently coloured connectors. Equally coloured connects match
- connectors are error proof with respect to wrong connections (e.g. outlet-outlet connection, blue-red connection)
- functionality for load dependent initialisation (table based)

Figure 2 shows an exemplary cut-out of a *StaticCycle* circuit with visualised signal flow directions and corresponding parameters. Blue arrows indicate the local flow direction. In this example the heat exchanger calculates for the outlet at position 1 the enthalpy according to the energy balance, passes through the mass flow from the inlet and receives a pressure value at the blue outlet connector. In contrast, at the red steam inlet connector (position 2) mass flow and pressure are defined via user input and passed over to the upstream valve. The red inlet connector receives a value for the enthalpy. In the red-connected steam valve a nominal pressure drop is assigned so that the component passes pressure and mass flow to the red outlet connector of the T-split which itself provides a value for the enthalpy. The green outlet connector of the T-split provides values for mass flow, enthalpy and pressure for the top valve component which serves as a pressure break because this components also receives a pressure signal at its blue inlet connector while sending parameters for mass flow and enthalpy for the downstream component.

In the following an example of a simple tube is used to illustrate the *StaticCycle*'s work with `fixed=false` parameters to determine parameters that are passed over from neighbouring components.

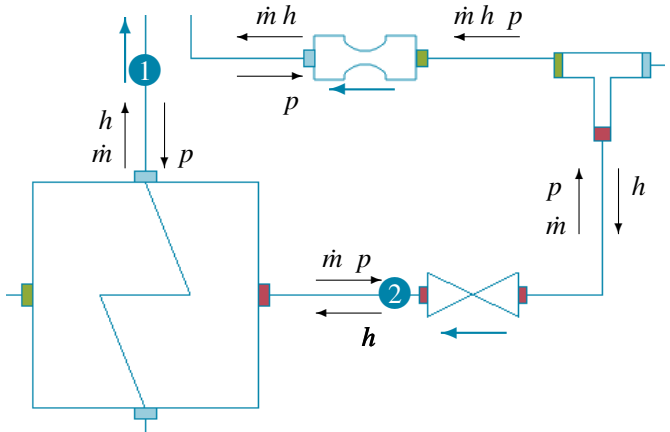


Figure 2. *StaticCycle* component example cut-out. Blue arrows indicate mass flow direction, black arrows indicate signal flow directions

```

model tube
  blueConnector_inlet inlet(p=p_in); //
    send pressure to upstream component
  blueConnector_outlet outlet(m_flow=
    m_flow, h=h_in); // send mass flow
    and enthalpy to downstream component
  parameter Real Delta_p_nom; // nominal
    pressure loss

  final parameter Real m_flow(fixed=false);
  final parameter Real h_in(fixed=false);
  final parameter Real p_in = p_out+
    Delta_p_fric;
  final parameter Real Delta_p_fric =
    m_flow / m_flow_nom * Delta_p_nom; //
    pressure loss calculation

  initial equation
    outlet.p=p_out; // get pressure from
      downstream neighbour
    inlet.m_flow=m_flow;
    inlet.h=h_in; //get the enthalpy and
      mass flow from the upstream
      neighbour
end tube;

```

As can be seen, parameters calculated or set by the component are set via the connector's modifier, e.g. `inlet.p`. In contrast, parameters that are set by neighbouring components are made available by parameters that apply the `fixed = false` feature. The corresponding internal values are set in an initial equation environment, ensuring that the results can be used as initial values for the dynamical simulation of the main model.

3 Use Case for Control Analysis and Improvement

3.1 Scope of Work

The idea of this generic use case is to illustrate *ClaRa*'s capabilities to model power plants at the state of the art including the full complexity of common condensation power plants due to extensive feedwater preheating and

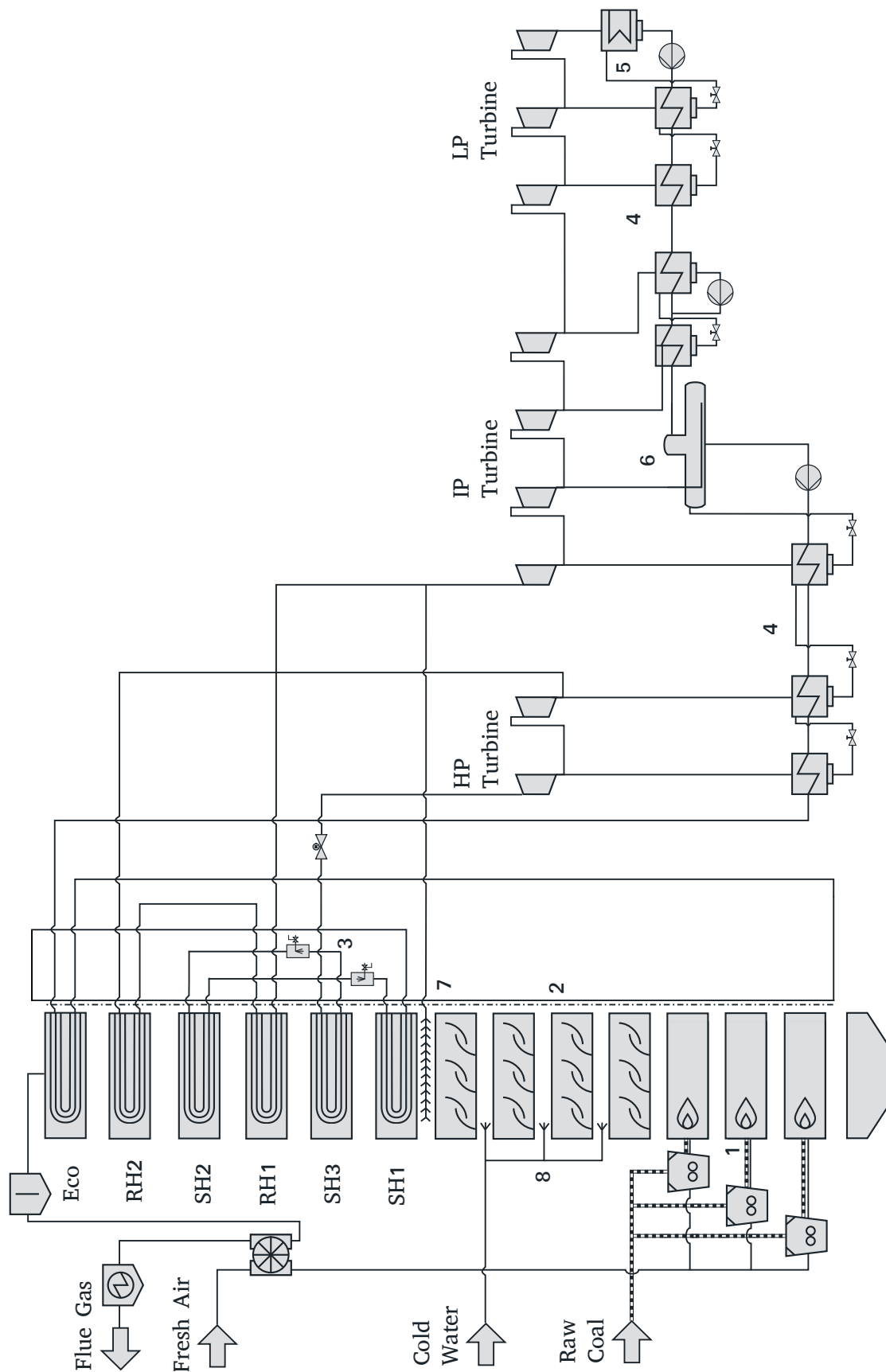


Figure 3. P&ID of simulation model

exhaust gas cleaning and cooling technologies. The use case also aims at proving Modelica's power to analyse complex processes at a physical level and its interaction with control structures. System simulation tools are used to analyse and improve a power plant's unit control to better handle disturbances due to soot blowing.

3.2 Model Description

The dynamic power plant model analysed in this paper is a generic once-through steam generator and has a nominal electrical power output of 600 MW, see Figure 3. The design life steam pressure is about 270 bar at a temperature of 600 °C. The combustion chamber has an overall height of approx. 80 m and is heated with three coal dust-fired burners (1) at three different burner levels. The chamber's walls are cooled with evaporating water which is superheated in three convective heating surfaces (SH1-SH3) to the live steam conditions. The layout comprises one reheating (RH1-RH2) to 520 °C at 52 bar after the high pressure turbine stage. The model discretises the boiler in height by using an arrangement of 14 different components including the hopper, the burners, the radiation zone (2), the convective heat exchangers and the spray injectors (3) for steam temperature control. Each of these models is parametrised with detailed geometry information and connected to neighbouring components according to the piping and instrument diagram. The heat exchange between combustion chamber and walls is implemented with detailed correlations for radiative and/or convective heat transfer. The flue gas is post processed with respect to NO_x , SO_x and ash and is used to preheat the fresh air carrying the coal dust from the mills. In order to reproduce the impact of soot blowing to the process at several positions steam and cool water can be introduced to the combustion chamber.

The steam is expanded in nine turbine stages with bleeds for several preheaters (4). The models for condenser (5) and feed water tank (6) are taking non-ideal phase separation into account and the resulting water levels are controlled by valves and pumps applying PI controllers.

The controlling system is build up applying the upcoming ClaRa_AddOn *ClaRa_Control* and based on a German guideline for unit control of conventional and nuclear steam power plants, (VDI/VDE, 2003). This guideline, defining a baseline for the power plant control, is the starting point for numerous implementation in German power plants.

The purpose of the model discussed in this paper is to analyse soot blowing of the evaporator furnace and its impact on the controlling system. The soot blowers (7,8) are modelled as a dynamical gas volume with an additional connector for water/steam inlet. The inflowing water is mixed ideally with the gas mass flow and the enthalpy of evaporation is considered by the dynamical energy balance. An effect of the soot blowing on the fouling coefficient, which is a parameter that reduces the heat transfer

to the heating surfaces, is not considered. An approach to model variable fouling factors which are affected by soot blowing is presented in (Gierow et al., 2015). This simplification is acceptable since the focus of the investigation is on the short-term energy and mass transfer of steam and water from the water steam cycle to the furnace and its impact on the control performance of the power plant. The medium-term improvement of the local efficiency of the heat transfer is of minor interest here.

The resulting power plant model consists of 9212 components with 28264 equations and 1606 differentiated variables.

3.3 Scenario Description

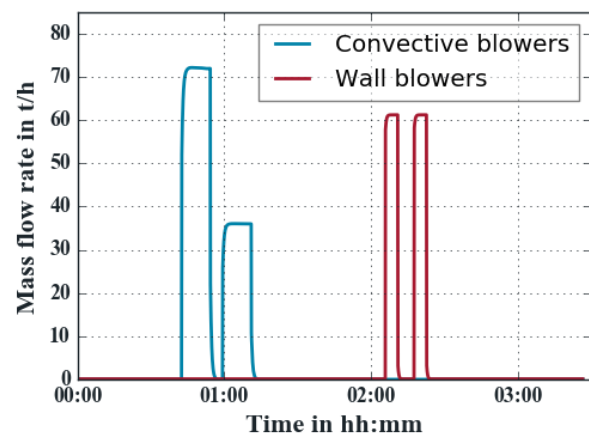


Figure 4. Soot blower injection mass flows

The underlying scenario to analyse the improved control strategy comprises two soot blowing events occurring during normal power plant operation. Heating surfaces in coal fired power plants tend to foul due to the high ash and slag content of the combustion air. Particles stick to the heating surfaces and cause a rising heat transmission resistance over operation time. Thus, the heat transfer to the water steam cycle is reduced. Soot blowing is a measure to clean the tube bundles and evaporator walls during operation by spraying steam or water through special lances onto the heating surfaces. This measure uses the combined effect of a thermal shock and the kinetic energy of the water/steam jet to reduce fouling and improve the heat transfer.

In general, two different soot blower types can be distinguished: The soot blowers for the tube bundles are fed with steam which is extracted ahead the intermediate pressure turbine. This rededication of steam has a direct impact on the electric power production and its control. In contrast, the soot blowers for the evaporator walls are fed with external water at 20° C which has a more indirect effect on the power production as the water injection cools down the furnace and thus reduces the steam production.

In figure 4 the soot blower injection mass flows are

shown. The convective blowers are active two times over a timespan of 15 min with a mass flow of $20 \frac{\text{kg}}{\text{s}}$ and $10 \frac{\text{kg}}{\text{s}}$, respectively while the wall blowers are active for time spans of 5 min with a mass flow of $17 \frac{\text{kg}}{\text{s}}$.

3.4 Analysis and Improvement of the Control Strategy

The plant is run in "steam generator control" mode and natural sliding pressure which means that the power output controller acts on the fuel mass flow and the turbine valve is fully opened. Therefore, deviations in fuel mass flow or disturbances on the steam generation have direct impact on pressure and generator output. Compensation of the generator power output by the fuel mass flow only takes effect with a delay.

A basic model based unit control strategy according to the German guideline for power plant unit control VDI 3508 (VDI/VDE, 2003) is implemented. The control is sketched in Figure 5, the basic control in black, the additions of the improved control in green. In the figure three sub-figures may be distinguished, starting from top to bottom we find the unit feed forward control and the process predictor in sub-figure a), the feedback controllers in sub-figure b) and the process itself in sub-figure c). Furthermore, three different line types are used: solid lines for internal control signals, dashed lines for measurement signals and dot-dashed lines for process input signals. For the sake of simplicity only the turbo generator power output control and the soot blowing control are sketched in the figure.

In the following the basic control set up is explained in detail. The turbine valve is fully opened by setting the turbine valve opening set value $y_{T,set}$ to 100 %. In sub-figure a) the target power output is fed into the output limiter which applies a limiting according to static and dynamic limits. From that value the fuel feedforward block generates a load dependent value of the firing power $\dot{Q}_{F,FF}$. The firing power is input to the simplified process model, referred to as the predictor in the further course, and to the process itself via a recalculation of the firing power into a corresponding fuel mass flow \dot{m}_F . The predictor calculates a corresponding expectation value of the generator power \hat{P}_G . In detail the predictor works as follows. From the fuel forward signal $\dot{Q}_{F,FF}$ a transfer function calculates the expected steam generation of the boiler $\hat{m}_{St,G}$. The expected steam mass flow to the turbine \hat{m}_T is subtracted from that value and the result serves as input to an integrator block from which the expected live steam pressure \hat{p}_{LS} is obtained. By multiplication with the set value of the turbine valve opening $y_{T,set}$ the steam mass flow to the turbine is calculated. The turbine is modelled with a first order transfer function block, thus receiving the expected power output \hat{P}_G .

However, the real process will be disturbed e.g. by fouling of heating surfaces which leads to a deviation between the ideal and the real process. In subfigure b) it can be seen that this deviation in power generation is calculated

from the difference of the expected value \hat{P}_G and the measured value P_G . The difference is input to the feedback controller of the power output which is a conventional PI-controller. It corrects the feed forward value $\dot{Q}_{F,FF}$ by the value deviation $\Delta\dot{Q}_F$.

Furthermore, in subfigure b) the feedback controller of the convective tube bundles soot blowing mass flow can be found. From the required soot blowing mass flow $\dot{m}_{SB,set}$ the measured soot blowing mass flow \dot{m}_{SB} is subtracted. The difference is input to the soot blowing mass flow controller which is a conventional PI controller.

Applying the concept of a model based unit control the process will be controlled in open loop as long as the process reacts as predicted. Thus, the control effort of the feedback control is significantly reduced compared to a conventional control system in which load changes are acting on the set value of a feedback controller. However, a disturbance like soot blowing has to be outbalanced by the power feedback controller. In case of the tube bundle soot blowing, intermediate pressure steam is consumed. Thus, the steam mass flow to the turbine is reduced and the generated electric power drops. The power feedback controller raises the fuel mass flow to compensate the power generation drop, but it takes effect with a delay. Poor control accuracy is the result.

To improve the behaviour of the model based unit control during soot blowing of the convective tube bundles some additions have been made, marked green in Figure 5. The set value of the required soot blowing mass flow $\dot{m}_{SB,set}$ is introduced as positive disturbance signal to the predictor's expected steam mass flow to turbine \hat{m}_T . To overcome the delayed reaction of the boiler on changes of the fuel mass flow, the disturbance signal to the predictor is activated prior to the starting time of the soot blowing process. This is done by delaying the soot blowing set point signal to the soot blowing mass flow controller with a delay block. The predictor reacts on the mass flow disturbance signal and raises the expected power output. In consequence, the feedback control outbalances the disturbance and raises the control signal. Thus, the coal mass flow rises and at the start time of the soot blowing additional steam is produced, which then can be consumed for the soot blowing. A drop of the produced power can be compensated.

The presented modifications acting on the feedback control are very efficient and easy to implement. A physical signal, the soot blowing steam mass flow, can be used directly as a disturbance signal without complicated recalculation. It is also independent of the load point as the signal is added to the boiler predictor of the unit control. The predictor's output is a load dependent rise of the generator power. The approach to derive an improvement of an existing control system from a profound analysis of the system of question rather than applying purely control-theoretical approaches has the advantage that the ideas can be easily understood and discussed with the power plant's operating personnel. This eases the acceptance of

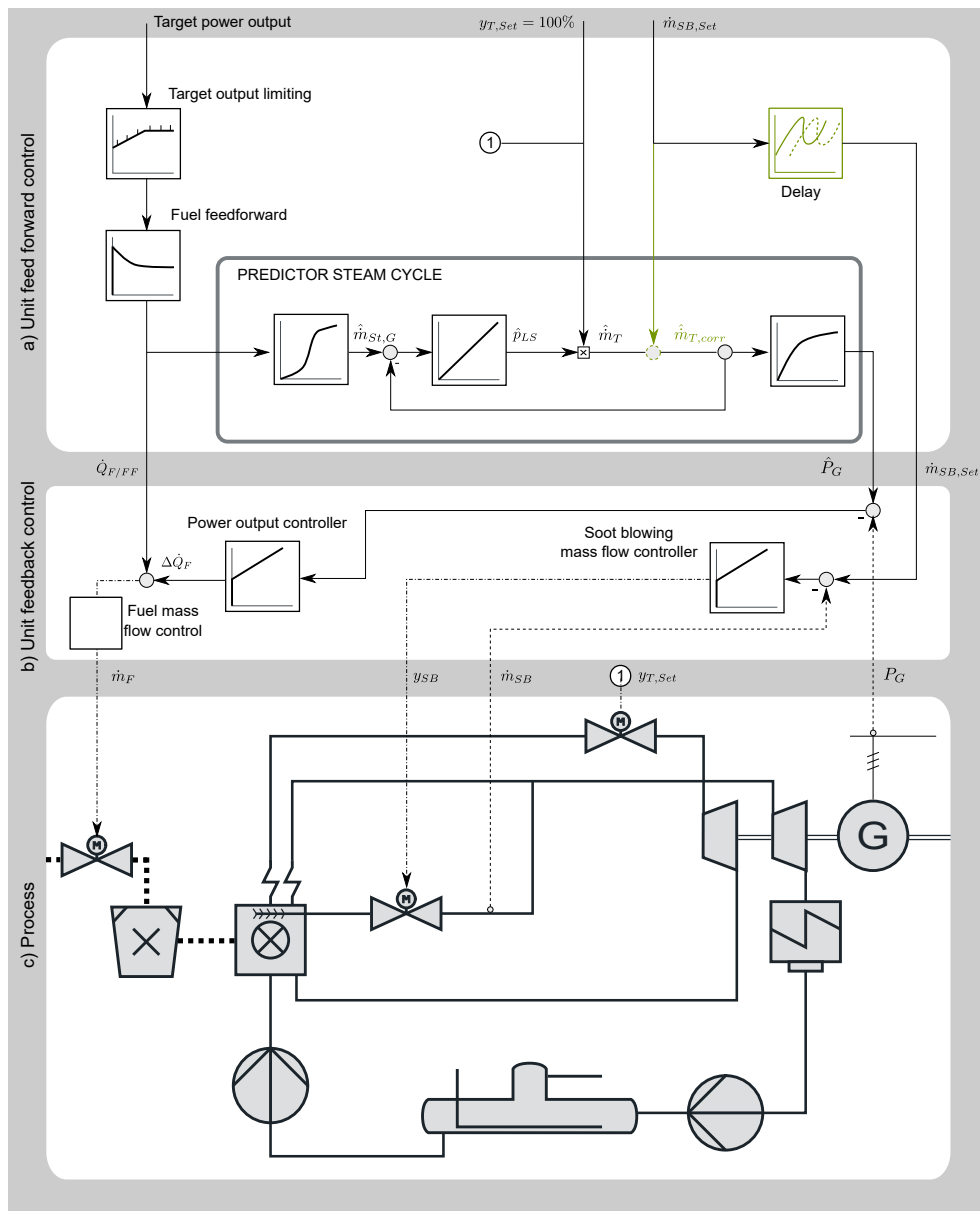


Figure 5. Model-based unit control and convective sootblowing control

such optimisation projects. System simulation prior to the commissioning can help to reduce technical and economic risks.

For the sake of completeness it shall be mentioned that an implementation of the disturbance signal in the feedforward control would require a load dependent recalculation of the physical signal, the steam mass flow rate, to a corresponding firing power signal.

Analogous additions also improve the control behaviour during steam generator wall soot blowing. As cold water is injected by the wall blowers the mass flow of cold injection water has to be expressed as a corresponding reduction of the steam mass flow to turbine. This is done by multiplication of the cold water mass flow with a load independent factor. The result is input as a disturbance signal to the predictor in the previously described manner.

In the following the simulation results are discussed. Figure 6 shows the flue gas outlet temperatures of the first superheater bundle during the soot blowing events for both control strategies. As can be seen, the impact of the soot blowing on the flue gas temperatures is higher for the wall blowers compared to the convective blowers. The reason behind this is that the convective blowers are fed with hot steam taken from the hot reheating pipe and the wall blowers are fed with cold water from an external reservoir. Due to the huge amounts of evaporating water the flue gas temperature is being reduced which results in less steam production. In comparison to the original control strategy, the improved one shows a significant reduction of flue gas temperature oscillations during wall soot blowing.

Figure 7 shows the intermediate pressure turbine mass flow during the soot blowing process with and without the

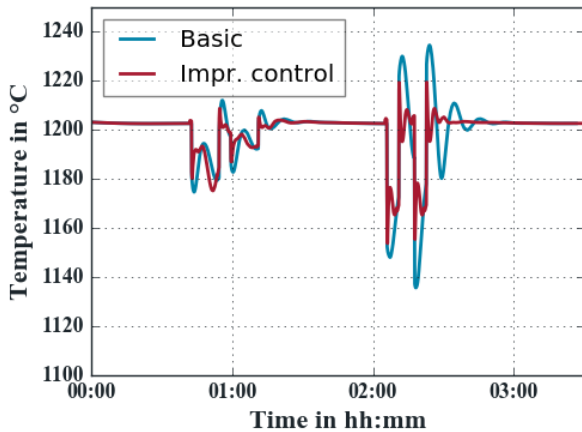


Figure 6. Impact of soot blowing on flue gas temperatures

improvement of the controller system. The steam for soot blowing of convective heating surfaces is taken directly from before the intermediate pressure turbine. The control improvement raises the steam generation such that we see less reduction of steam mass flow through the turbine. The soot blowing induced mass flow oscillations are of lower amplitude and the system reaches a steady state faster with the modified control strategy especially during the soot blowing of the furnace walls with cold water, because a swing up of the turbine mass flow is prevented.

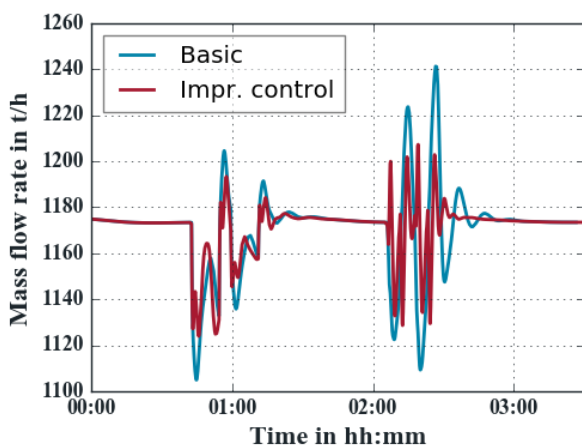


Figure 7. Impact of improved control on IP turbine flow during soot blowing

In Figure 8 the generator output for both control strategies are shown. The improvement in quality of control by the additional disturbance value is visible here too and results in lower amplitudes and a faster reaching of a stable state. The greater impact of the wall soot blowing with cold water in comparison to the convective soot blowing with hot steam manifests in higher amplitudes.

In the following, simulation results of the wall soot blowing process with cold water are discussed in detail,

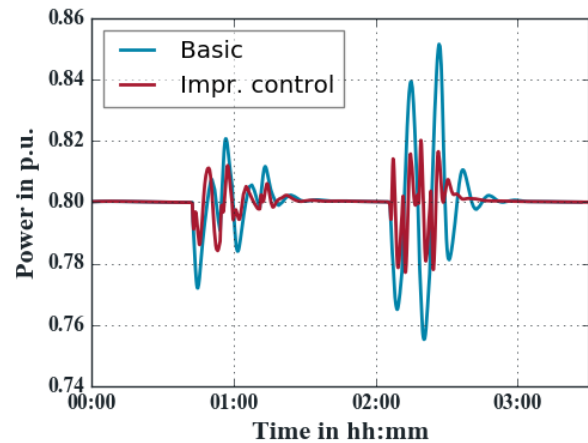


Figure 8. Impact of improved control on generator output during soot blowing. The results corresponds to a reduction of primary control from 10.7 MWh to 4.6 MWh.

which is significantly improved by the alternative control strategy. Figure 9 shows the fuel mass flows of mills and burners, soot blowing mass flow and first superheater flue gas outlet temperature during the period of wall soot blowing without control improvement. When the soot blowing process is started, the flue gas temperature inside the furnace reacts with an initial drop due to the fed in cold soot blowing water and its evaporation, which can also be seen in the superheater outlet temperature. This results in a reduced mass flow inside the turbine forcing the power controller to increase the fuel mass flow at mill inlet. The mass flow at burner inlet reacts time shifted due to mass storage effects in the mill. When the increased but delayed fuel mass flow is burned, the flue gas temperature is rising again until the first soot blowing process is immediately stopped, which causes a temperature step up above the initial temperature. The power controller then reduces the coal mass flows until the second soot blowing, causing a comparable temperature characteristic which transitions into a swinging state when the soot blowing process has ended.

Figure 10 shows the same variables during the same time span for the soot blowing process with the improved control strategy. In comparison to Figure 9 it can be seen that the power controller raises the fuel mass flow already before the soot blowing process is started. This is caused by the disturbance signal of the pressure steam mass flow being activated 50 s prior to the soot blowing process. By the time the soot blowing process starts, the increased fuel mass flow already enters the burner levels resulting in a shorter and not so big initial flue gas temperature drop. In addition, the flue gas temperature only swings with a lower amplitude compared to the case without improvement and reaches a stable state in a significantly shorter time span which results in a higher quality of control. Because of the higher flue gas mass flows and a change of the heat transfer coefficients due to the fed in water, the super-

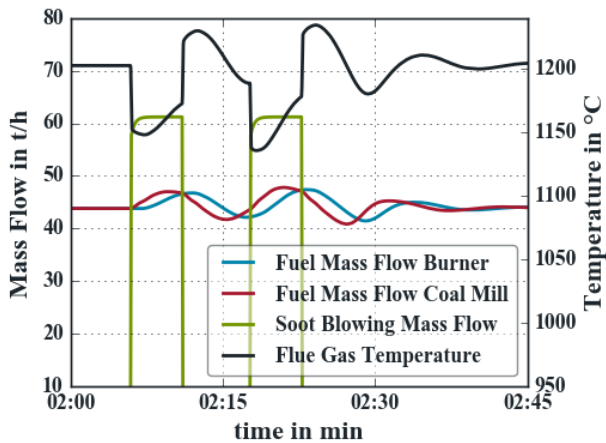


Figure 9. Fuel mass flows of mills and burners, soot blowing mass flow and superheater outlet temperature during wall soot blowing without control improvement

heater outlet temperature tunes in to a lower temperature during soot blowing.

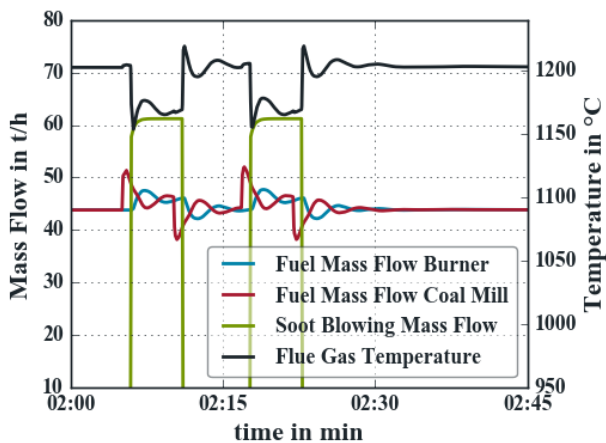


Figure 10. Fuel mass flows of mills and burners, soot blowing mass flow and superheater outlet temperature during wall soot blowing with control improvement

Figure 11 shows the impact of the alternative controller system on generator output at different power plant loads. In order to ensure, that the enhancement of control quality can be achieved at varying loads and not only the one discussed previously, additional simulations have been carried out for 100% and 65% load. As can be seen, the same improvements are obtained at all load points shown in the diagram where the best results are obtained for nominal load. The *ClaRa* library is an appropriate tool to carry out such additional comparisons very comfortably.

Power plants which participate at the secondary control power market, must guarantee the offered power at any time during operation. While control power is being called, the plant operators would like to avoid fluctuations

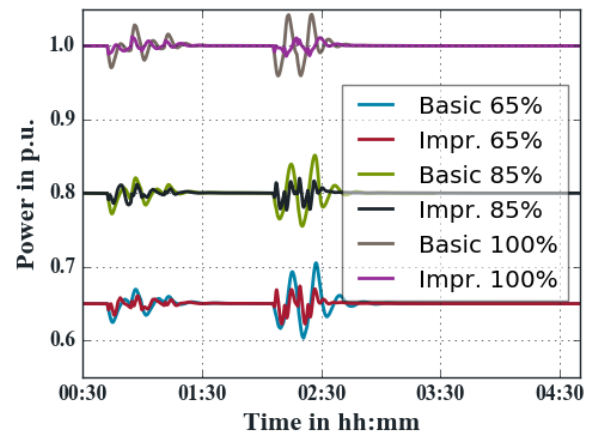


Figure 11. Impact of control optimisation on generator output during soot blowing at different loads

in power output, occurring for example during soot blowing processes which are a common routine, because underfulfillment would be charged with penalties and overfulfillment will not be compensated. This makes a higher quality of control beneficial. When the operators offers positive control power during full load operation, the plant only could be run with a certain margin to its maximum load under consideration of load fluctuations. If the deviations in power output during soot blowing cause the highest amplitudes in the current power plant operation schedule it can be seen, that an improved control strategy, like the one proposed in this paper, enables the operator to run the plant at a higher load and making more profit while maintaining the needed margin to maximum load for secondary control power.

4 Conclusion

The current state of the library *ClaRa* for the simulation of power plants has been presented to be one of the most complete and complex open source library for the simulation of Clausius Rankine cycles. Due to its deep insight at equally high transparency it addresses both new and experienced users. The library is open to be extended in the future, by both new component and physics models within the *ClaRa* (which is work in progress by the authors) as well as in terms of new libraries, the so-called *ClaRa_AddOns*. The latter mentioned path allows the scope to be widened to new fields like biomass or solar heated applications.

As a use case the optimisation of a power plant unit control has been outlined. The results prove the outstanding opportunities that are introduced by system simulation allowing to understand complex processes better by evaluating unmeasurable process variables and to test innovative control concepts without risks.

5 Outlook

The further development of the library will, amongst others, introduce so-called six-equation-models for extra precise calculation of two phase flow conditions in pipes. Furthermore, all models are under permanent review with respect to zero flow and other non-design conditions. The library is planned to be extended by special header components enabling stress evaluation.

References

- L. Andresen, P. Dubucq, R. Peniche, G. Ackermann, A. Kather, and G. Schmitz. Status of the transient library: Transient simulation of coupled energy networks with high share of renewable energy. In *Proceedings of the 11th International Modelica Conference, Versailles, France*, 2015.
- B. Bachmann, P. Aronsson, and P. Fritzson. Robust initialization of differential algebraic equations. In *Proceedings of the 4th International Modelica Conference, Vienna, Austria*, 2006.
- J. Brunnemann, F. Gottelt, K. Wellner, A. Renz, A. Thüring, V. Roeder, C. Hasenbein, C. Schulze, G. Schmitz, and J. Eiden. Status of ClaRaCCS: Modelling and Simulation of Coal-Fired Power Plants with CO₂ Capture. *Proceedings of the 9th International Modelica Conference, Munich, Germany*, pages 609 – 618, 2012.
- F. Casella, M. Sielemann, and L. Savoldelli. Steady-state initialization of object-oriented thermo-fluid models by homotopy methods. In *Proceedings of the Modelica Conference 2011, Dresden, Germany*, 2011.
- C. Gierow, M. Hübel, J. Nocke, and E. Hassel. Mathematical model of soot blowing influences in dynamic power plant modelling. In *Proceedings of the 11th Modelica Conference*, 2015.
- F. Gottelt, K. Wellner, V. Roeder, J. Brunnemann, G. Schmitz, and A. Kather. A Unified Control Scheme for Coal-Fired Power Plants with Integrated Post Combustion CO₂ Capture. In *Proceedings of the 8th IFAC Conference on Power Plant & Power System Control, Toulouse, France*, 2012.
- International Energy Agency. Tracking Clean Energy Progress 2015. Technical report, 2015.
- S.E. Mattsson, H. Elmqvist, M. Otter, and H. Olsson. Initialization of Hybrid Differential-Algebraic Equations in Modelica 2.0. In *Proceeding of the 2nd International Modelica Conference, Oberpfaffenhofen, Germany*, 2002.
- M. Najafi. Selection of variables in initialization of modelica models. In *Proceedings of the 2nd International Workshop on Equation-Based Object-Oriented Languages and Tools, Paphos, Cyprus*, 2008.
- C. Schulze. Table based calculation of thermophysical properties for simulation of thermodynamic systems. In *Proceedings of the ITI Symposium*, 2013.
- C. Schulze. *A Contribution to Numerically Efficient Modelling of Thermodynamic Systems*. PhD thesis, Technische Universität Braunschweig, 2014.
- T. Vahlenkamp and S. Wischhusen. FluidDissipation - A Centralised Library for Modelling of Heat Transfer and Pressure Loss. In *International Modelica Conference, Bielefeld, Germany*, 2008.
- T. Vahlenkamp and S. Wischhusen. FluidDissipation for Applications - A Library for Modelling of Heat Transfer and Pressure Loss in Energy Systems. In *Proceedings 7th Modelica Conference, Como, Italy*, September 2009.
- VDI/VDE. VDI/VDE Guideline 3508: Unit control of thermal power stations. *Association of German Engineers (VDI) / German Association of Electrical Engineering and Information Technology (VDE)*, 2003.
- S. Velut and H. Tummescheit. Implementation of a Transmission Line Model for Fast Simulation of Fluid Flow Dynamics. In Christoph Clauß, editor, *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical University; Dresden; Germany*, Linköping Electronic Conference Proceedings, 2011. doi:<http://dx.doi.org/10.3384/ecp11063446>.

Interactive FMU-based Visualization for an Early Design Experience

Volker Waurich¹ Jürgen Weber²

¹Chair of Construction Machines, TU Dresden, Germany, volker.waurich@tu-dresden.de

²Chair of Fluid-Mechatronic Systems, TU Dresden, Germany, weber@ifd.tu-dresden.de

Abstract

User experience is an eminent part of holistic product design. Especially in the field of mobile machinery, the driver's impression of the machine handling is crucial for successful design. To get an early understanding of the ergonomic aspects of a new concept of operation, functional prototypes can be applied. This paper presents the tools to develop a functional prototype using free software and low-cost hardware. This includes prototyping of control devices, interfaces to the Modelica-based simulation models and a generic visualization using a game engine. In order to speed up the process of functional prototyping, an approach to automatically visualizing FMUs based on a scene description file is presented. The application of interactive simulation was used to support the development of a novel control device for excavators in a student project at TU Dresden.

Keywords: *visualization, OpenModelica, engineering education, construction machines, rapid prototyping*

1 Introduction

The operation of mobile machinery, e.g. excavators, puts ambitious requirements on the driver. Therefore, the ergonomic aspects of the control environment are an important selling point. The innovation of new operating concepts should be supported by an early design experience. In a student project at Technische Universität Dresden, a collaboration of students from the fields of Technical Design, Mechanical Engineering and Media Computer Science developed an innovative control concept for mobile excavators. The project was initiated by an OEM of mobile machinery. Although the actual project results are confidential, the applied methods and tools shall be presented and serve as a motivation for similar projects.

To support the design process, a prototypic control device was engineered to get a haptic experience. With the help of novel rapid prototyping technologies, as *3d-printing* or *lasercutting*, complex designs can be realized quickly and cheaply. Since the required machines became affordable, public workspaces, so-called *makerspaces* spread out more and more. Due to that, even with a small budget, realistic prototyping is possible.

Another innovation is the availability of easy-to-use, low-cost microcontrollers. Using different sensors, e.g. potentiometers, motion concepts of the prototypes can be tested. Utilizing functional prototypes during an early design phase, facilitates more profound impressions of the product than using CAD-models or plastic prototypes. Machine tools and electronics are available in Makerspaces and easy to apply for students. With an easy-to-use connection to virtual environments based on simulation models, the design process can be enhanced further. With the help of the OpenModelica tool chain, an FMU-visualization has been developed which allows an automated generation of appealing 3d environments.

This paper covers the different aspects of developing functional prototypes with a high level of automation and tool support. For the presented use-case of the machine control development, only freely available software (i.e. open-source Modelica-tool *OpenModelica* and the free gaming engine *unity*), low-price hardware and cheap prototyping technologies that are becoming widely accessible, are applied. This paper is meant to be a motivation for combining physical prototyping and virtual mockups within the training of engineers. As experience has shown, the development of functional prototypes creates a high level of self-motivation and perfectionism among participants.

In chapter 2, the applied methods of physical rapid prototyping are presented. Chapter 3 discusses the means of developing interactive Modelica models. Afterwards, the basic idea behind a generic FMU-visualization is presented and the tools for visualizing the simulation models are introduced in chapter 4. The presented approach is compared to existing visualization workflows. Chapter 5 describes the manufacturing of a control device. Finally, chapter 6 concludes the paper and gives an outlook on future work.

2 Physical Prototype Manufacturing

2.1 Makerspaces for Higher Education

In recent years, affordable technologies for rapid prototyping have spread widely. Various libraries and higher educational institutions like Saxon State and University

Library Dresden (slu) offer public access to rapid prototyping machines in so called makerspaces. Makerspaces are collaborative work spaces which provide rapid prototyping tools and the knowledge to utilize them. Typically, projects in the field of model making and electronics can be realized using the facilities of a makerspace. The interest in makerspaces emerges and despite the lack of long-term investigation, the impact on engineering education has been promising among many universities (ISA, 2016). Ongoing studies will give an overview of how the overall impact of academic makerspaces has to be assessed. At least with regard of the presented student project, a high level of motivation to realize the projects and to acquire the necessary knowledge has been observed.

In order to develop a control device for an excavator, 3D printers and foam cutters have been used to produce haptic prototypes. The production costs are very low and therefore are best suited to use them in student projects. Project participants from the field of technical design developed design drafts which have been modelled in CAD-software. The printed 3D prototypes give a spatial impression and provide enough stability to integrate joints and sensors.

2.2 Application of Sensors and Microcontrollers

Besides machine tools, makerspaces offer a range of electronic components and easy-to-use microcontrollers such as Arduino (Ard). With these low-cost controllers, sensor concepts can be set up easily and data can be processed and transferred to a computer. In the presented project, buttons, rotary and translational potentiometers have been set up to map the functionalities of a conventional excavator control. The sensors have been attached in the joints of 3D-printed control devices in order to access the control device condition. The Arduino reads the sensors and transfers the signals to a computer, either via USB-connection or with an additional Bluetooth module. The messages can be processed by *SerialPortReceive* of the *Modelica_DeviceDrivers* library. When using the Arduino IDE, users write C-like code, compile and transfer it directly to the board and are able to monitor serial connection communication. There is a vast amount of documentation and tutorials available that simplifies microcontroller programming for students outside this subjects area. The following Arduino code can be applied to transfer signal data of the Arduino's analog pin 1 via USB-connection with a sample time of 0.1 s.

```
byte buf[2];
unsigned long lastSignal = 0;
unsigned long interval = 100; //ms
int value = 0;

void setup() {
  Serial.begin(9600);
}
```

```
void loop() {
  while(millis() - lastSignal > interval)
  {
    lastSignal += interval;
    value = analogRead(1);
    buf[0] = lowByte(value);
    buf[1] = highByte(value);
    Serial.write(buf,2);
  }
}
```

3 Modelling of Interactive Simulation Environments

3.1 Model Interaction

The use-case of models involving external inputs during runtime became much more accessible by *Modelica_DeviceDrivers* library (*M_DD*) (Bellmann, 2009). The library interfaces various input devices and communication protocols. Hence, Modelica models can be enhanced with direct user-inputs or connected to other processes during runtime. A realtime synchronization is provided as well. For the presented demonstrator, the serial port implementation was utilized in order to communicate with an Arduino microcontroller. *M_DD* supports packing and unpacking of byte-messages which allows to access data e.g. sensor signals via a serial port connection. *OpenModelica* supports serial communication and packaging both in simulation mode and in *FMUs*. Figure 1 displays the graphical model view of an excavator model, that is controlled via serial communication. The message protocol is modelled with *unpackInt*-models, that split incoming messages into a sequence of integer variables. These integer variables are converted to real variables and conditioned to fit the excavator interface. The excavator model has been taken from a Modelica-library by the Chair of Construction Machines, TU Dresden.

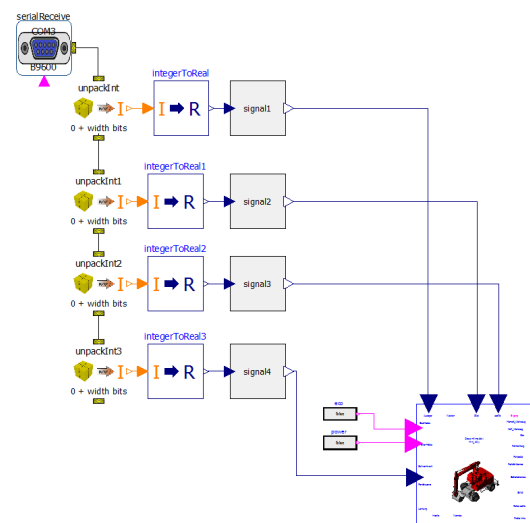


Figure 1. Model of an excavator and a serial port interface using *Modelica_DeviceDrivers* library

The following listing shows the parametrization of a model to read a two-byte message sent by an Arduino, whereas the parameters *baud*, *sampleTime* *userBufferSize* and *Serial_Port* have to be adapted to the sending controller. The *width* parameter of the *UnpackUnsignedInteger* model has to be set to 16 bit in order to deserialize the two byte value, sent from the microcontroller.

```

model arduino
Modelica_DeviceDrivers.Blocks.Communication
.SerialPortReceive
arduinoRead(
baud=Modelica_DeviceDrivers.Utilities
.Types.SerialBaudRate.B9600,
parity=0,
enableExternalTrigger=false,
startTime=0.0,
autoBufferSize=false,
userBufferSize=2,
sampleTime=0.1,
Serial_Port="COM5");

Modelica_DeviceDrivers.Blocks.Packaging
.SerialPackager.UnpackUnsignedInteger
unpackInt(
bitOffset=0,
width=16,
nu=1);
equation
  connect (arduinoRead.pkgOut,
            unpackInt.pkgIn);
end arduino;

```

3.2 Realtime Capabilities

Realtime requirements restrict the model to simulating within a specified interval of time. Hard realtime criteria demand a deterministic execution time whereas soft realtime allows the simulation to exceed the time limit occasionally. In the presented use case, soft realtime criteria are assessed. Nevertheless, for realtime application, it is favourable to reduce the simulation time. Modelica compilers allow different kinds of performance optimization for simulations. The time integration method has a big influence on the execution time, depending on the number of iterations and step sizes. In most realtime applications, explicit, fixed step methods are preferable. The lack of stability and the necessity of small step sizes lead to the development of more sophisticated methods e.g. inline integration (Elmqvist et al., 1995). Besides that, the evaluation of parameters is an effective option to increase simulation speed. There are various optimization techniques to improve calculation of algebraic loops, e.g. structural methods like tearing (Elmqvist and Otter, 1994) or reshuffling (Waurich et al., 2014). Calculations of jacobian matrices can perform differently depending on whether numerical, symbolical or colored jacobians are used. Automatic parallelization is also a feature to speed up simulation. Of course, the operating system, the hardware and the C/C++ compiler influence the simulation time as well.

The excavator model consists of a multi body system and some simple hydraulic components (cylinders, valves, flow sources). In most cases, the BLT-matrix of a mechanical model is dominated by a linear system of equations. Hence, parallelization of BLT-blocks will not improve the simulation speed. The present model benefits mostly from the evaluation of parameters. The dominating system of equations with 437 equations including 9 tearing variables is reduced to a system with 379 equations including 7 tearing variables. The amount of single equations reduces from 1208 to 1162. This results in a simulation speed-up of 1.33. This is sufficient to run the simulation without exceeding the realtime limits on a Windows 7 desktop computer with i7-3930K processor. The *FMU* was compiled using OpenModelica and gcc 5.3.0 as *FMU 2.0 model exchange*.

4 A Generic Visualization of FMUs

4.1 The Functional Mock-Up Unit

In order to exchange simulation models and to use them across various software, the Functional Mock-Up Interface was developed (Blochwitz et al., 2012). The *Modelica* language and its tools are highly involved in the development and application of *FMI*. The *FMI*-standard features two variants, i.e. *model-exchange* without internal time integration and *co-simulation* that includes a time integration solver. The black-box models that provide the *FMI*-API are called *Functional Mock-Up Units* and contain the functional behaviour of a simulation model that can be accessed via interface variables. The model variables are listed in the *modelDescription.xml*. The connections and relations of these model variables are hidden from the user since *FMUs* are compiled as a shared library. This is very useful since it protects intellectual property but it is cumbersome if information of the model structure is of interest. Hence, a generic visualization of *FMUs* is not possible in general.

4.2 Existing Approaches to Visualize Multi-body Models

Commercial *Modelica*-tools offer built-in visualization features for multibody systems based on the *Modelica.Mechanics.MultiBody.Visualization.Advanced* models. Visualization comprises both subsequent and concurrent visualization of simulation. This visualization is possible since the tools have full access to the model information and the variables that are used to visualize the shapes. Another approach would be to add dedicated animation objects to the Modelica model and let them communicate with an external visualization software, e.g. in the commercial *DLR Visualization library* (Hellerer et al., 2014). Also the *Modelica3D* implementation by Höger relied on Client/Server communication (Hoeger et al., 2012). Yamaura et al. (Yamaura et al., 2016) described a comprehensive framework of different tools that exchange model variables via *UDP* communication with

a corresponding *Unity* model. This approach combines the physical model capabilities of engineering tools like *Simulink* and *Dymola* with the highly developed gaming engine *Unity* which offers much more visualization and graphical modeling features than any simulation software. Another promising implementation for discrete time simulations was presented by (Bijl and Boer, 2011), that is designed on a database which feeds the 3D visualization. The use of appealing 3D visualization and the potential of 3D game engines is described as well. An entirely different concept was presented in (Elmqvist et al., 2015) in which even the modeling is performed in a 3D visualization environment that provides direct feedback on the model structure of a multibody system. This visualization uses the web interface of the simulation tool *Dymola*.

Since there was no free Modelica tool that features visualization in an integrated manner, the open-source Modelica Compiler *OpenModelica* and its graphical editor *OMEdit* have been enhanced to visualize results of simulations. Therefore, the *OpenModelica* Compiler has to extract all necessary information about the visualization shapes from its internal model representation. Hence, the animation of *Modelica.Mechanics.MultiBody* models can be provided without adding dedicated visualization objects to the model.

Instead of implementing a new *OpenModelica*-specific API to transfer visualization variables between simulation and animation-software, the authors decided to choose an already existing API, i.e. the *FMI*. By means of a visualization scene description file that is generated by the *OpenModelica* Compiler, the visualization software in *OMEdit* can access relevant variables and maps them to the corresponding animation shape properties. In the following chapters, the details of FMU-based visualization are presented.

4.3 A Specification of Visualization

As described in the previous chapter, *OpenModelica 1.11* is able to create a scene description XML-file that contains the information about the *Modelica.Mechanics.MultiBody.Visualizers.Advanced.Shape* objects within a model. The *shape* model contains the basic visualization information like position, orientation, scale and color. This approach was already mentioned in (Waurich et al., 2016) and a proof of concept implementation was presented. The scene description XML-file simply lists all instances of the *Shape* model and assigns values to their parameters. The following exemplary snippet of a scene description XML-file contains information about the model "shape1" which is of type "cylinder". The position vector *r* is defined by constant expressions and lies in the root "{0,0,0}" whereas the *length* attribute depends on the component reference "shape1.length".

```
<visualization>
```

```
<shape>
  <ident>shape1</ident>
  <type>cylinder</type>
  <r><exp>0.0</exp>
    <exp>0.0</exp>
    <exp>0.0</exp>
  </r>
  <length>
    <cref>shape1.length</cref>
  </length>
</shape>
</visualization>
```

The shape parameters are either defined by an `<exp>` tag which refers to a constant expression of type real or to a `<cref>` tag, which stands for a reference given by a string-type. `<cref>` elements have to be updated during runtime. Shapes can be either geometric primitives or CAD-files, like .stl or .dxf that are referenced by their absolute path names in the scene description file. Besides the shape models, there are more visualization models that could be defined, e.g. *Surface* or *PipeWithScalarField*, but the current implementation covers shape only. A *XML Schema Definition* is available at <https://github.com/vwaurich/visxml>

4.4 The Visualization Architecture

No matter which frontend is used to display the 3D scene, the mechanism to animate the shapes is identical as depicted in Figure 2. The visualization backend needs an *FMU* and a corresponding scene-description file, both generated by the *OpenModelica* Compiler. It has to be ensured, that all variables which are used to visualize the scene, are accessible in the *FMU*. This means that these variables must be retrievable via *fmiGetReal* API. Therefore, *OpenModelica* changes protected variables to public if needed.

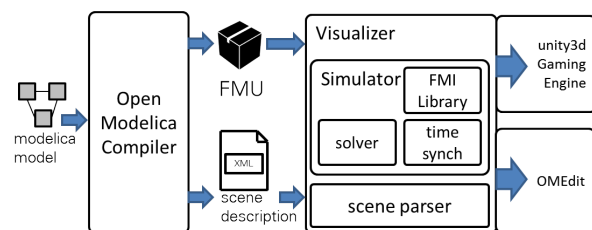


Figure 2. Overview of FMU-based visualization both with *unity* and *OMEdit* frontend.

After the selection of an *FMU*, the visualization backend instantiates all shapes listed in the scene description file. These can be either geometric primitives such as cubes or spheres, or imported CAD-files. Constant shape properties can be set directly during initialization of shapes. In contrast, variable properties cannot be set before the solution of the initial system of the *FMU*.

Unpacking, loading, instantiation, initialization and simulation of the *FMU* is performed by the *FMI*Library

(FMI). For the *Model Exchange FMUs*, a simple Explicit Euler solver with a default step size of 1ms is used. The simulation is synchronized with realtime by the visualizer backend itself. Hence, no synchronization on the model side is necessary (e.g. *from Modelica_DeviceDrivers.Blocks.OperatingSystem.SynchronizedRealtime*).

Compared to other visualization approaches, the generic FMU visualization has the following advantages:

- A specification of the visualization objects allows different tools to create the same scene automatically.
- No model modifications have to be applied in order to generate a visualization. No additional dependencies have to be included. No additional equations are added to the existing multi-body model.
- It is easy for simulation tools to generate scene description files. Based on this visualization formalism, the visualization is independent of the simulation software and does not rely on vendor specific interfaces.
- It enables automatic integration of physical simulation in graphical modelling software (as will be shown for the gaming engine *unity*).
- The simulation and variable access is achieved via shared memory communication and therefore does not need (but can be extended for) simulation via a network connection.
- It is helpful to visualize third party FMUs automatically to get an understanding of their behaviour without having access to the model itself.
- It is more convenient to add and edit advanced visualization features in a proper visualization tool and not in the simulation model by a Modelica-Editor.

4.5 OMEdit FMU-Visualization

The graphical connection editor *OMEdit* features basically textual and graphical modeling views, result plotting and algorithmic debugging. The lack of 3D animation hindered the use for mechanical applications. The novel implementation of a result-file based and FMU-based visualization helps to get a better understanding of mechanical systems.

Figure 3 displays the visualization view of *OMEdit*. The visualization is implemented using OpenSceneGraph and features the animation of mat-result files, csv-result files and *FMUs*. In each case, a scene description file is needed, to map the model variables to the shape properties.

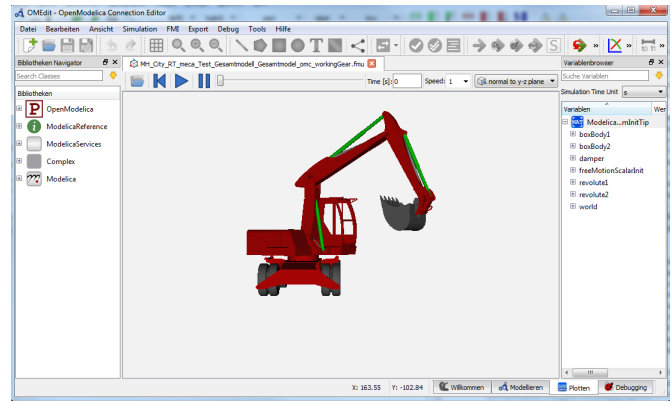


Figure 3. Screenshot of the visualization perspective in *OMEdit*.

4.6 Unity FMU-Visualization

The implementation in *OMEdit* based on OpenSceneGraph is not visually attractive and makes it very cumbersome to enhance the scene with additional graphical objects. A gaming engine with graphical editor and a huge asset store like *unity* (Uni), would allow an easy setup of appealing graphical scenes as in Figure 4. Hence, the mechanism of loading an *FMU* and a scene description file has been implemented in a *unity* plugin. The user simply chooses an *FMU* via a dialog and the plugin creates so called *GameObjects* for the corresponding shapes. Besides that, an *FMU*-simulator *GameObject* is created, which simulates the FMU and accesses the necessary variables. This comprises everything to run the scene either in the unity debugger or from a compiled unity project.



Figure 4. Unity scene with an FMU-based excavator model that is controlled by an Arduino board in realtime.

Next to the shape objects and the FMU-simulator, additional *GameObjects* can be added in order to create an adequate environment. Accessing the FMU-inputs and FMU-outputs from the unity model is possible via interface functions of the FMU-simulator *GameObject*. Hence, the FMU-generating simulation tool is only responsible for the physical simulation. The graphical modelling can be performed by a special purpose tool. The FMU-simulator

plugin supports this separation by generating the basic mapping between simulation and visualization automatically. The unity user interface with FMU-selection dialog and loaded FMU-visualization is depicted in Figure 5. Since the FMU has to be initialized to calculate the position, orientation and color of the bodies, all shapes are still in the root of the coordinate system.

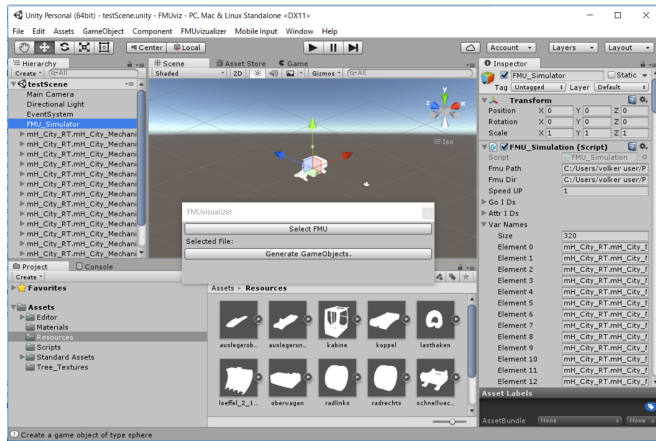


Figure 5. Unity user interface with loaded FMU. The GameObjects for the shapes and the FMU-Simulator are listed in the hierarchy view, the .dae files are copied to the resources and the inspector view displays all variables that are updated during runtime.

When interchanging variables between the unity world and the Modelica-based FMU, it has to be considered, that the coordinate systems are different. Modelica uses a right-handed system whereas unity relies on a left-handed system. Furthermore, the y-axis should be used as vertical since *unity* uses it as vertical by default (which is essential since available skyboxes display a horizon in the x-z-plane).

The FMU-simulator plugin automatically converts the position and orientation of the Modelica-variables to the left-handed system of the unity variables and switches the vertical axis if desired. Another issue is the lack of stl-file support in unity. It needs an stl-importer plugin or the CAD-files have to be converted to a 3D data format e.g. COLLADA. File conversion can be done manually or scripted by tools like blender (Ble).

5 The Development of a Remote Control Device for an Excavator

The previous chapters depicted the necessary tools to set up a functional prototype. To try out novel control concepts, physical prototypes have been equipped with sensors to measure the motion of the joints. The signals are used to control the volume flow in and out of the cylinders. Hence, the velocity of motion for the boom, the arm and the shovel are controlled. Even inverse kinematics can be tried out if the cylinders are controlled to follow cartesian inputs to set the position of the shovel. Furthermore, as-

sistance systems are experienceable without implementing them in fully operable systems. This simplifies the evaluation of acceptance and ergonomics. Even exceptional control mechanisms like handheld controllers for remote control are possible. Figure 6 shows the setup to control a model in OMedit via Bluetooth connection, which is handled as an ordinary serial port.

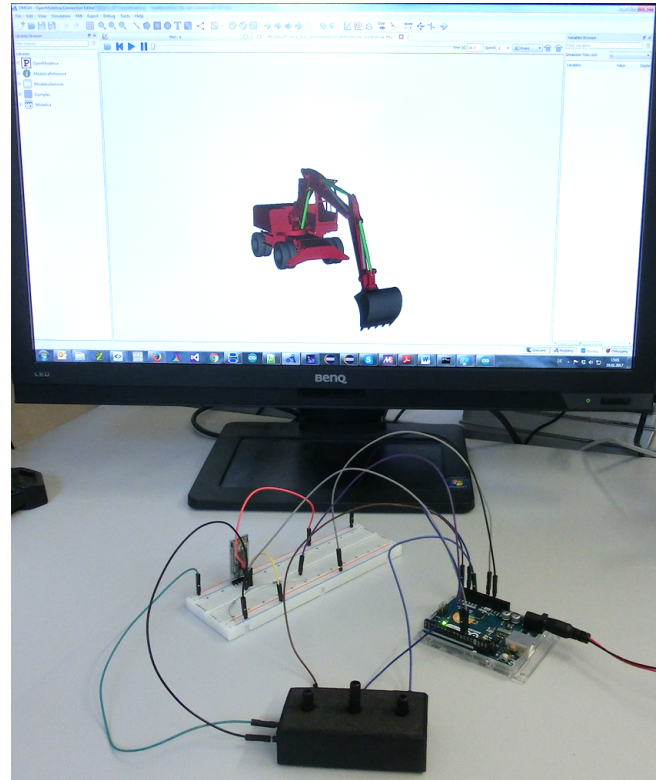


Figure 6. Remote control setup to control an excavator model in OMedit via Bluetooth connection. The control device is a printed box with 3 rotary potentiometers.

The unity editor allows further settings for camera position (first or third person view) as well as lighting and terrain modelling. In the unity asset store, various objects to populate the scene can be downloaded for free or charged.

6 Conclusion and Outlook

This paper comprises a workflow for developing functional prototypes that have been realized within a student project at TU Dresden. The usage of Makerspace facilities, low-budget electronics and free software together in an interdisciplinary design project, was a successful experiment. The motivation of students was huge and both the familiarisation with novel technologies as well as its application are valuable experiences. Besides the individual learning success, the developed prototypes are highly praised by the project initiator, an OEM of excavators.

During the project, improvement opportunities have been revealed. Basically, the development of a virtual environment which can be controlled with external

hardware in realtime and based on the simulation of a Modelica model needed improvement. Hence, an automated approach to setup visualizations of FMU-based multi-body systems was implemented. The integration via FMUs in the game engine *unity* leads to satisfying results. More importantly, the scene description of FMUs enables new generic interfaces to visualization tools and their features. As a future extension, Modelica models could be extended with contact-force-interfaces or collision interfaces that could be generated automatically in a unity project in order to interact with *unity*'s physics engine and feedback the results to the simulation model. In the field of mobile machinery, interaction to soil or particle models in unity would be very useful as well. Since Game Engines feature comprehensive possibilities to model environments, experimental grounds can be set up to test e.g. assistance and automation systems. Through the network protocol interfaces of *M_DD*, even web-based services in mobile machines can be experienceable in early design stages.

The scene description file is currently an OpenModelica specific feature. Further support of this FMU extension would leverage the advantage of the unity-plugin and the development of other FMU-Visualization-Add-Ons in additional tools. A discussion about adding the scene description file as an optional extension to the FMI-Standard would be highly appreciated by the authors.

References

- The arduino webpage. www.arduino.cc. Accessed: 2016-11-18.
- The blender webpage. www.blender.org. Accessed: 2016-12-08.
- The fmilibary webpage. www.jmodelica.org/FMILibrary. Accessed: 2016-11-21.
- The unity3d webpage. www.unity3d.com. Accessed: 2016-11-18.
- The saxon state and university library dresden (slub) webpage. <http://www.slub-dresden.de/en/service/workplaces-workspace/makerspace/>. Accessed: 2016-12-07.
- Proceedings of the 1st International Symposium on Academic Makerspaces ISAM 2016*, 2016. URL www.project-manus.mit.edu/home/conference.
- Tobias Bellmann. Interactive simulations and advanced visualization with modelica. In *Proceedings 7th Modelica Conference*. Linköping University Electronic Press, 2009.
- Jonatan L. Bijl and Csaba A. Boer. Advanced 3d visualization for simulation using game technology. In *Proceedings of the Winter Simulation Conference, WSC '11*, pages 2815–2826. Winter Simulation Conference, 2011. URL <http://dl.acm.org/citation.cfm?id=2431518.2431853>.
- Torsten Blochwitz, Martin Otter, Johan Åkesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. pages 173–184, 2012. doi:10.3384/ecp12076173.
- Hilding Elmqvist and Martin Otter. Methods for tearing systems of equations in object oriented modeling. In *In ESM 94 European Simulation Multiconference*, 1994.
- Hilding Elmqvist, Martin Otter, and François E. Cellier. In-line integration: A new mixed symbolicnumeric approach for solving differential-algebraic equation systems. In *Proceedings of the 1995 European Simulation Multiconference*, pages 23–34. Society for Computer Simulation International, June 1995.
- Hilding Elmqvist, Alexander D. Baldwin, and Simon Dahlberg. 3d schematics of modelica models and gamification. In *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*, number 118, pages 527–536. Linköping University Electronic Press, Linköpings universitet, 2015.
- Matthias Hellerer, Tobias Bellmann, and Florian Schlegel. The dlr visualization library - recent development and applications. In *Proceedings of the 10th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, number 96, pages 899–911. Linköping University Electronic Press; Linköpings universitet, 2014. doi:10.3384/ecp14096899.
- Christoph Hoeger, Alexandra Mehlhase, Christoph Nytsch-Geusen, Karsten Isakovic, and Rick Kubiak. Modelica3d - platform independent simulation visualization. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 76, pages 485–494. Linköping University Electronic Press; Linköpings universitet, 2012. doi:10.3384/ecp12076485.
- Volker Waurich, Ines Gubsch, Christian Schubert, and Marcus Walther. Reshuffling: A symbolic pre-processing algorithm for improved robustness, performance and parallelization for the simulation of differential algebraic equations. In *Proceedings of the 6th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, EOOLT '14*, pages 3–10. New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2953-8. doi:10.1145/2666202.2666203. URL <http://doi.acm.org/10.1145/2666202.2666203>.
- Volker Waurich, Martin Großer, and Sebastian Voigt. Generische visualisierung von fmu-basierten modellen für die interaktive simulation. In *Tagungsband Workshop ASIM STS/GMMS 2016*, pages 230–236. ASIM STS/GMMS, 2016. ISBN 978-3-901608-48-3.
- Masahiro Yamaura, Nikos Arechiga, Shinichi Shiraishi, Scott Eisele, Joseph Hite, Sandeep Neema, Jason Scott, and Theodore Bapty. Adas virtual prototyping using modelica and unity co-simulation via openmeta. In *The First Japanese Modelica Conferences, May 23-24, Tokyo, Japan*, number 124, pages 43–49. Linköping University Electronic Press, Linköpings universitet, 2016.

Using Modelica for advanced Multi-Body modelling in 3D graphical robotic simulators

Gianluca Bardaro¹ Luca Bascetta¹ Francesco Casella¹ Matteo Matteucci¹

¹Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy,
{luca.bascetta,gianluca.bardaro,francesco.casella,matteo.matteucci}@polimi.it

Abstract

This paper describes a framework to extend the 3D robotic simulation environment Gazebo, and similar ones, with enhanced, tailor-made, multi-body dynamics specified in the Modelica language. The body-to-body interaction models are written in Modelica, but they use the sophisticated collision detection capabilities of the Gazebo engine. This contribution is a first step toward the simulation of complex robotics systems integrating detailed physics modelling and realistic sensors such as lidar and cameras. A proof-of-concept implementation is described in the paper integrating Gazebo collider and the Modelica Multi-Body library, and the results obtained when simulating the interaction of an elastic sphere with a rigid plane are shown.

Keywords: Multi-Body Dynamics, 3D Robotic Simulators, Autonomous Robotics, Autonomous Vehicles.

1 Introduction

The popularity of research on autonomous mobile robots, including autonomous vehicles and mobile manipulators, has been recently increasing due to the huge number of potential applications, ranging from self-driving cars and robots for logistics, to planetary explorations, search and rescue missions, surveillance, humanitarian de-mining, as well as precision agriculture activities such as pruning vines and fruit trees (Paden et al., 2016; Roa et al., 2015; Ko et al., 2015; Chitta et al., 2012).

The design and development of such systems, whose main functionalities are perception, planning, and control, is a multidisciplinary and complex work that has to be supported by virtual prototypes, allowing for a preliminary design and testing of the corresponding algorithms in safe operating conditions. However, due to the huge differences among the three mentioned skills a mobile robot should own, the virtual prototype has to satisfy various requirements. Considering, for example, the development of perception algorithms, the most important characteristics of the virtual prototype are a realistic description, mainly from a geometrical and graphical point of view, of the scene, and the availability of realistic models for the most common commercial sensors, i.e., laser range finders and cameras. On the other hand, testing a control algorithm, e.g., an Advanced Driver Assistance System in a critical

situation, requires an accurate physical modelling of the vehicle, including all (and sometimes even only) the phenomena the designer knows to be relevant in the specific application, e.g., cornering stiffness for lateral dynamics control.

Nowadays there are many different, open source and commercial, modelling and simulation environments that are suitable to model vehicles and mobile robots.

A first family is represented by 3D robot simulators, like for example Gazebo¹, V-Rep², Webots³, Morse⁴, that are widespread in the robotics community. These simulators allow for an easy development of complex natural/artificial simulation environments, they are already equipped with models of perception devices, and they can be easily integrated with standard robot control middlewares like ROS⁵. For these reasons, they are particularly suitable for the development and testing of planning and perception algorithms, and for the validation of the whole control software before moving to field tests (Bardaro et al., 2014).

The physical simulation implemented in these tools is targeted at real-time execution and ease of virtual prototype set-up; this is obtained by providing the 3D kinematic models for rotational and translational joints to assemble robots and vehicles, and collision detection primitives with simplified translational and rotational friction models. These building blocks are implemented with low level C++ libraries, such as ODE (Drumwright et al., 2010) or MuJoCo (Erez et al., 2015), and the experimenter is expected to use them in a black box fashion with little, if any, way to alter their physical behaviour. Indeed, the differential equations characterizing the physical behaviour of each building block are hidden in the code, often undocumented, and with no direct tool for altering their behaviour. This makes current 3D robotics physical simulation fidelity and accuracy somehow limited, and requires the coding of external plug-ins, e.g., using C++ custom code, every time the phenomenon we are interested in replicating is more complex than the one which can be obtained assembling the available building blocks.

On the other side of the spectrum, a second family of sim-

¹<http://gazebo-sim.org>

²<http://www.coppeliarobotics.com>

³<http://www.cyberbotics.com>

⁴<http://www.openrobots.org/wiki/morse>

⁵<http://www.ros.org>

ulators is represented by multi-body and/or multi-physics simulators, like for example Modelica tools⁶ such as SimulationX⁷, whose aim is to accurately represent the dynamic behaviour of the system, and that are thus particularly suitable for accurate dynamic analysis, control system development, and validation in repeatable and safe operating conditions (D'Amelio et al., 2015). These simulators allow a general mechanism for physical systems modelling, based on an high-level language for the definition of the differential equations describing the relevant aspects of the simulation, but little, if any, support is available for geometrical and graphical simulation of the environment and thus for the simulation of robot sensors such as lidar and cameras.

In this paper we present an approach, inspired by the idea already introduced in (Bardaro et al., 2016), to extend the multi-body modelling in the 3D Gazebo simulator using Modelica and the MultiBody library (Otter et al., 2003). This allows to introduce ad-hoc physical models which are tailored to the specific needs of a particular application in a convenient, declarative, equation-based framework, leveraging on the basic infrastructure already provided by the MultiBody library. On the other hand, we are able to extend the level of simulation provided by the Modelica framework by the 3D simulation capabilities of the Gazebo simulator. In particular, this paper focuses on adding customized body-to-body interaction models to the standard components of the MultiBody library, combining the advanced capabilities of collision detection provided by the Gazebo framework with the flexibility provided by the Modelica environment to define sophisticated, tailor-made, equation-based physical models. It must be emphasised, however, that this topic is not important per se, instead it represents a proof-of-concept of the possibility of integrating the two simulation environments in order to set up a new one that is able to better address graphical and physical aspects as well. As a consequence, the contribution of this paper is not related to an innovative or improved interaction model, but to the framework that allows to extend Modelica modelling capabilities by the 3D Gazebo simulation.

The paper is structured as follows. Section 2 describes the design of the modelling framework. In the following Section 3, a proof-of-concept implementation is described, and the results obtained with a simple sphere-to-plane interaction simulation are presented. Section 4 concludes the paper with an outlook to further developments.

2 Design of the modelling framework

The rationale behind the design is to let Gazebo and the Modelica tool each perform the tasks at which they excel, for which they already have good built-in support, and which are more conveniently programmed by the end-user.

⁶<http://www.modelica.org>

⁷<http://www.simulationx.com>

Modelica will then be used for the accurate and tailor-made dynamic modelling of the multi-body objects for which the standard modelling approach of the physical engine embedded in Gazebo is not adequate. Modelica could also be used to represent low-level sensing, actuation and control, such as electric motors and drives, pneumatic actuation, low-pass signal filtering, etc., which are not covered by Gazebo, when their accurate modelling is essential to assess the success or failure of higher-level control functions.

All other tasks, such as building and managing the scenes, simulating other objects for which ad-hoc dynamic modelling is not required, simulating vision-based sensing, and providing geometrical information about object collisions, will be managed by Gazebo.

The present paper focuses on the integration between Gazebo and Modelica to provide accurate ad-hoc physical modelling where needed. How the resulting physical model can then be integrated in the Gazebo environment, together with all the other objects and functions simulated by Gazebo, goes beyond the scope of this paper and will be addressed in future works.

The basic framework for the modelling of multi-body objects is provided by the Modelica MultiBody library, which allows to build modular models of multi-body systems by the connection of link and joint models. Since the Gazebo engine also uses corresponding primitives, automatically generating the Modelica code of the model corresponding to any Gazebo multi-body object is a straightforward task. The availability of flexible link models compatible with the MultiBody library, e.g., those described in (Ferretti et al., 2014), allows to easily take into account flexibility in all those cases where this is crucial to replicate the system dynamic behaviour. This is a feature that could be very useful in the case of soft or flexible robots and which is still not present in Gazebo.

A key ingredient of any multi-body model of robots or autonomous vehicles is the modelling of the interaction between different bodies, in particular the tyre-road interaction in vehicles and the interaction between hands or grippers and objects to be manipulated for robots. For this purpose, Gazebo provides so-called collider objects, which take as input the position of the reference frames of any two objects, possibly having a complex shape, and returns information about the presence or absence of contact points, their location, the depth of penetration, and the normal vectors to the object surface at the contact point. Gazebo can also compute the resulting interaction forces and torques, according to some standard embedded model; the idea in the context of this paper is to ignore this information and use Modelica instead to compute them, according to a tailor-made equation-based physical model that is appropriate for the specific simulation scenario.

The Modelica code of the base model for two-body interaction, `PhysicalInteraction`, is listed in the appendix. The model extends the `PartialTwoFrames` model of the MultiBody library. It gets the position and

orientation of the two potentially interacting objects from the two frame connectors and passes them to the `collisionDetectionModelica` function. This in turn converts the rotation objects into quaternions and calls the external function `collisionDetection`, that passes the two object ID strings and their position and orientation to the Gazebo server. The collider in Gazebo responds returning the number of contact points, the arrays of contact points on both bodies, as well as the penetration depths and the normals to the surface for each contact point.

This data is then passed to the replaceable function `computeInteraction`, which uses the kinematic information to compute the forces and torques exerted on body a by body b . As Modelica functions cannot generate events, a conditional equation is then written in the `PhysicalInteraction` model, which applies the forces and torques computed by the external function to the connector if the penetration depth is positive, zero otherwise. This allows to precisely compute the contact event instant and handle the discontinuity properly, if the Modelica solver provides proper event handling. Finally, the corresponding forces and torques applied on body b by body a are computed by Newton's 3rd law.

In this context, the Gazebo tool only acts as a server, providing the service of computing the kinematic information regarding the collisions between any two objects of interest. The physical simulation is carried out by the code generated by the Modelica tool, which is the simulation master. This means that the sequence of calls to the Gazebo server does not correspond to a physical sequence of points in time, but rather to the individual function calls required by the Modelica solver, which might go backward and forward in time to compute a solution, e.g., when locating event instants or when a time step is rejected by an adaptive step-size solver. As the Gazebo tool is not the master of the simulation in this context, this is not a problem. In fact, time is not even part of the data which is communicated to the Gazebo server from the Modelica side.

Specific physical interaction models can then be obtained by extending the `PhysicalInteraction` class and by redeclaring the `computeInteraction` function with the specific algorithm that computes the interaction forces and torques, based on the model of interest for the end user. All the infrastructure provided by the Modelica MultiBody library can be used to carry out this task with ease, in particular the functions to resolve vectors in different reference frames and all the functions implementing vector algebra operations.

3 Proof of concept

In this section, a proof-of-concept implementation that demonstrates the proposed approach is presented.

3.1 Implementation details

In order to avoid all the problems related to memory management, in this implementation the external C function

`computeInteraction` uses Unix IPC sockets to communicate with the Gazebo server. In the future, this mechanism will be substituted by some more efficient, shared-memory based communication, e.g., by embedding the Modelica model into an FMI and using external objects to set up the communication framework.

A simple exemplary test case has been selected for the demonstration, namely the interaction between an elastic ball and a fixed, rigid plane. When the two bodies collide, the force F_a applied on the sphere at the point of contact is the sum of three components:

$$F_a = F_e + F_d + F_f.$$

The elastic force F_e is directed as the normal vector (which points to the sphere's centre) and its magnitude is computed according to (Nassauer and Kuna, 2013)

$$F_e = k_e \sqrt{V} d,$$

where k_e is an elastic constant, d is the penetration depth, and V is the volume of the spherical cap of height d

$$V = \pi d^2 \left(r - \frac{d}{3} \right).$$

The damping force F_d is proportional to the normal component v_n of the relative velocity between the two bodies at the point of contact and opposed to it, thus providing dissipation each time the sphere hits the plane.

The friction force F_f depends on the tangential component of the relative velocity v_t at the point of contact, has the opposite direction and a magnitude

$$-\mu F_e \frac{v_t}{\sqrt{v_t^2 + v_\epsilon^2}};$$

where μ is the dry friction coefficient, v_ϵ is a small velocity threshold, and the fraction is approximately equal to one for $v_t \gg v_\epsilon$ and approaches zero as $v_t \rightarrow 0$. This model is not accurate at low relative velocities, since it leads to a slow sliding at velocities around v_ϵ instead of proper stiction. On the other hand, it has the nice property of not becoming singular at zero relative velocity and is perfectly adequate for the purposes of this demonstration. Other more sophisticated models that include stiction, such as the one described in (Deur et al., 2004) could be employed if needed.

As to the torques, only the friction force exerts a net torque on the sphere's frame connector, located at the centre of the sphere; the torque vector is simply $\tau = r \times F_t$. For simplicity, the torsional torque due to rolling friction has been neglected in this demonstrator.

3.2 Test cases and simulation results

The results of three sphere-to-plane interaction simulations are here presented. The sphere represents a big inflated balloon, modelled as a hollow sphere of mass $m = 1$

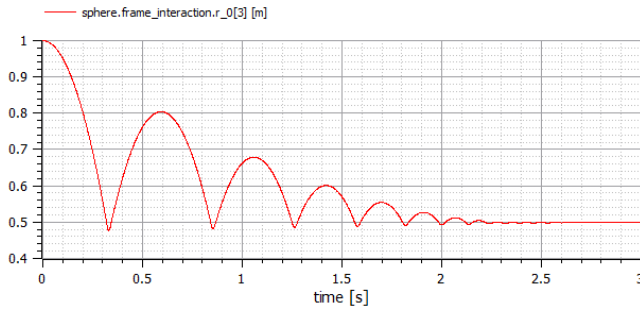


Figure 1. Simulation 1 – Ball height over time.

kg, radius $r = 0.5$ m, moments of inertia $J = \frac{2}{3}mr^2$, elastic constant $k_e = 10^3$ N/m² and with a relatively low friction coefficient $\mu = 0.1$. Air friction is neglected. All the simulations start with the center of the sphere at a height of 1 m above the horizontal xy -plane, the z -axis pointing upwards.

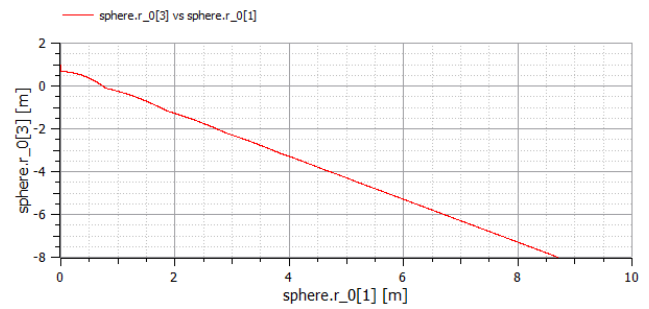
The Modelica code was compiled into executable simulation code with the OpenModelica compiler⁸ version 1.12.0-dev, using a Runge-Kutta fixed time step integration algorithm with a time step of 1 ms, which is short enough to correctly handle the elastic impacts, whose typical duration is about 10 ms.

The simulation were first tested by emulating the response of the Gazebo server by a Modelica function. This required to extend the `Sphere2Plane` physical interaction model, which uses the external function calling Gazebo, and to redeclare the `collisionDetection-Modelica` function so that it directly computes the contact point locations, depths of penetration and normal vectors, rather than calling the external function and getting them from Gazebo. This function is implemented easily in Modelica, as the geometry of the sphere-to-plane interaction is extremely simple. Eventually, the same simulation results were obtained when using the Gazebo server, thus validating the entire proof-of-concept implementation. Also, the qualitative behaviour of the system in the three simulations corresponds to what one would expect from physical intuition.

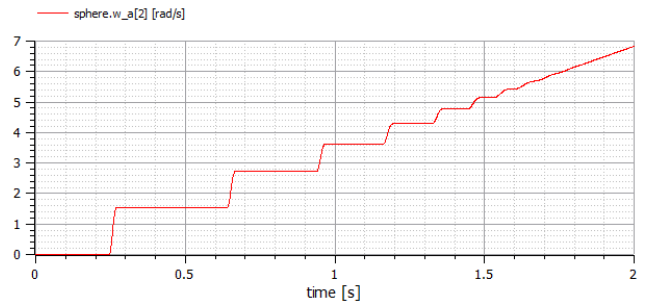
Many different simulations were run, in order to validate each component (elastic, damping, friction) of the interaction forces and torques separately. In this paper, the results of three simulation experiments with realistic choices of the interaction model parameters are reported.

In the first simulation, the plane is horizontal and the sphere has zero initial velocity and angular velocity. As expected, the ball falls onto the plane and bounces a few times before getting to rest, due to the dissipative effect of F_d . Figure 1 shows the vertical position of the sphere centre over time.

The second simulation scenario is similar, save that the plane is tilted by 45° along the y -axis. When the ball hits the plane, it bounces off horizontally. Due to fric-



(a) Trajectory of the sphere centre in the xz -plane



(b) Angular velocity of the sphere in the y -axis direction

Figure 2. Simulation 2 – Ball bouncing on a tilted plane.

tion, it also gets some angular momentum on the y -axis during the bounce, and thus starts spinning slowly. It then bounces a few more times on the tilted plane until dissipation causes it to remain in contact with the tilted plane and to accelerate while rolling downwards. Figure 2(a) shows the trajectory of the sphere's centre in the vertical plane, while Figure 2(b) shows the angular momentum over time, which increases abruptly at each bounce and finally increases with a constant slope once the sphere stops bouncing and rolls down on the plane surface always remaining in contact.

The last simulation considers again a horizontal plane; in this case the sphere starts with a non-zero horizontal velocity in the negative x -axis direction, spinning fast backward around the y -axis. Every time the ball bounces on the plane, the friction force slows down the spinning a bit, and accelerates the sphere in the positive x -axis direction, so that eventually the ball changes its horizontal direction and rolls back to a point on the plane below the initial position. Figure 3(a) shows the position of the sphere's center in the vertical xz -plane, while Figure 3(b) shows the angular momentum along the y -axis over time⁹.

4 Conclusions

In this paper, a proof-of-concept for the integration between the Gazebo 3D robotic simulation tool and Modelica has been presented. The proposed framework al-

⁸<https://openmodelica.org>

⁹The 3D videos generated by Gazebo of the three simulations are available online at this URL: <https://home.deib.polimi.it/casella/gazebo/videos.html>.

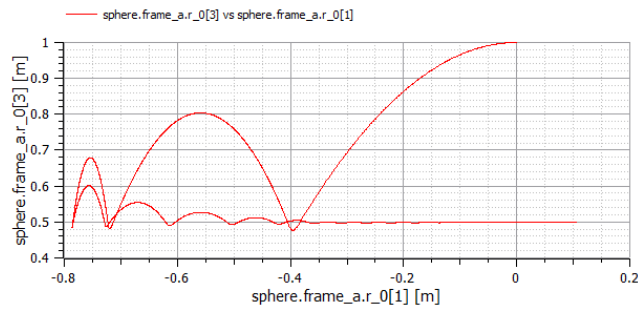
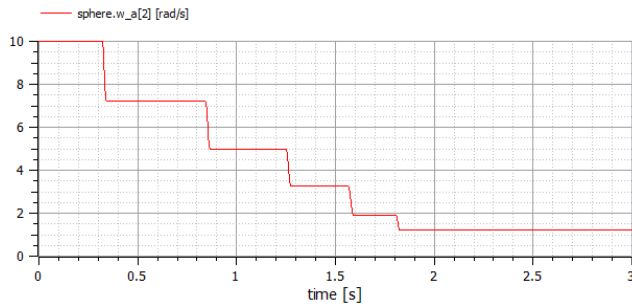
(a) Trajectory of the sphere centre in the xz -plane(b) Angular velocity of the sphere in the y -axis direction

Figure 3. Simulation 3 – Ball starting with a non-zero horizontal velocity in the positive x -axis direction and spinning backward around the y -axis.

lows to extend the basic 3D multi-body engine embedded in Gazebo, by providing equation-based customized 3D multi-body dynamics. The extension is very convenient and easy to implement, as it leverages on the existing sophisticated collision detection functionality of Gazebo, on the Modelica MultiBody library, and on the possibility of describing an ad-hoc physical behaviour in a high level, equation-based modelling environment. It also makes it possible to perform equation-based multi-domain physical modelling, e.g., by adding Modelica models of physical sensors, actuators and low-level controllers to the mechanical model, and in general by modelling any kind of physical behaviour beyond that of multi-body systems.

The framework has been demonstrated with a proof-of-concept implementation, using IPC sockets to enable the communication between the Gazebo tool and Modelica automatically generated simulation code. In particular, the results of the simulations of a simple system with an elastic ball bouncing on a rigid plane with low friction have been presented. The obtained results are very encouraging and suggest that it might indeed be possible to propose these Modelica extensions, implemented with the open-source OpenModelica compiler, as the preferred way to extend the native Gazebo simulation engine.

To reach our final aim, further developments are under investigation. First of all, we would like to validate the concept with scenarios involving multiple object interactions; currently we already generate Modelica simulation code in the presence of multiple object, what has to

be validated is the collision between multiple objects handled by Modelica. To improve on performance and ease of deployment, we are currently encapsulating the Modelica model in an FMU to handle the communication with Gazebo via shared memory and external object interface. Once the FMU will be integrated with the Gazebo plug-in mechanism, it will be possible to integrate the FMU-based simulation into the master simulation loop of the Gazebo tool in a seamless way and transparently to the designer of the simulation.

Finally, we would like to experiment with hybrid simulations with some physical behaviour simulated by the Gazebo physics engine and some physical behaviour with special modelling requirements simulated by the Modelica/FMU code. This set-up could be necessary to handle demanding simulation scenarios with many objects, since we expect the Modelica-based simulation code to be slower than the native and somewhat simplified Gazebo simulation engine, so that using Modelica only where needed could end up in much faster simulations.

References

- G. Bardaro, D.A. Cucci, L. Bascetta, and M. Matteucci. A simulation based architecture for the development of an autonomous All Terrain Vehicle. In *SIMPAR*, pages 74–85, 2014.
- G. Bardaro, L. Bascetta, F. Casella, and M. Matteucci. Advancement in multi-body physics modeling for 3d graphical robot simulators. In *Workshop on Modelling and Simulation for Autonomous Systems*, pages 189–195, 2016.
- S. Chitta, E.G. Jones, M. Ciocarlie, and K. Hsiao. Mobile manipulation in unstructured environments: Perception, planning, and execution. *IEEE Robotics & Automation Magazine*, 19(2):58–71, 2012.
- E.L. D’Amelio, L. Bascetta, D.A. Cucci, M. Matteucci, and G. Bardaro. A modelica simulator to support the development of the control system of an autonomous all-terrain mobile robot. In *International Conference on Mathematical Modelling*, pages 274–279, 2015.
- Joško Deur, Jahan Asgari, and Davor Hrovat. A 3D brush-type dynamic tire friction model. *Vehicle System Dynamics*, 42(3): 133–173, 2004. doi:10.1080/00423110412331282887.
- Evan Drumwright, John Hsu, Nathan Koenig, and Dylan Shell. Extending Open Dynamics Engine for robotics simulation. In *Proceedings of the Second International Conference on Simulation, Modeling, and Programming for Autonomous Robots, SIMPAR’10*, pages 38–50. Springer-Verlag, 2010.
- Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
- Gianni Ferretti, Alberto Leva, and Bruno Scaglioni. Object-oriented modelling of general flexible multi-body systems. *Mathematical and Computer Modelling of Dynamical Systems*, 20(1):1–22, 2014. doi:10.1080/13873954.2013.807433.

- M. Ko, B.-S. Ryuh, K.C. Kim, A. Suprem, and N.P. Mahalik. Autonomous greenhouse mobile robot driving strategies from system integration perspective: Review and application. *IEEE/ASME Transactions on Mechatronics*, 20(4): 1705–1716, 2015.
- Benjamin Nassauer and Meinhard Kuna. Contact forces of polyhedral particles in discrete element method. *Granular Matter*, 15(3):349–355, 2013. doi:10.1007/s10035-013-0417-9.
- M. Otter, H. Elmqvist, and S. E. Mattsson. The new Modelica MultiBody library. In *Proceedings 3rd International Modelica Conference*, pages 311–330, Linköping, Sweden, Nov. 3–4 2003.
- B. Paden, M. Cap, S. Zheng Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.
- M.A. Roa, D. Berenson, and W. Huang. Mobile manipulation: Toward smart manufacturing. *IEEE Robotics & Automation Magazine*, 22(4):14–15, 2015.

A Listing of the PhysicalInteraction model

```

model PhysicalInteraction "Base class for all physical interaction models"
  extends Modelica.Mechanics.MultiBody.Interfaces.PartialTwoFrames;
  import Modelica.Mechanics.MultiBody.Frames;
  parameter Integer maxContacts = 10 "Number of max contact points";
  parameter String id_a = "" "Id of interacting object a";
  parameter String id_b = "" "Id of interacting object b";
  Real numberOfContactPoints "Number of actual contact points";
  Real cp_a[maxContacts, 3] "Array of contact points on body a, resolved in frame_a";
  Real cp_b[maxContacts, 3] "Array of contact points on body b, resolved in frame_b";
  Real depth_a[maxContacts] "Array of penetration depths in body a";
  Real depth_b[maxContacts] "Array of penetration depths in body a";
  Real normals_a[maxContacts, 3] "Array of normals on body a, resolved in world frame";
  Real normals_b[maxContacts, 3] "Array of normals on body b, resolved in world frame";
  Real r[3] "Vector from frame_a to frame_b resolved in frame_a";
  SI.Force f_a[3] "Interaction force applied on body a, resolved in frame_a";
  SI.Torque t_a[3] "Interaction torque applied on body b, resolved in frame_b";

  replaceable function collisionDetectionModelica
    input Integer maxContacts "Maximum number of contact points";
    input Real r_a[3] "Position vector of interaction frame of object a, resolved in world frame";
    input Frames.Orientation R_a "Orientation of interaction frame of object a";
    input String id_a "unique id for object a";
    input Real r_b[3] "Position vector of interaction frame of object b, resolved in world frame";
    input Frames.Orientation R_b "Orientation of interaction frame of object b";
    input String id_b "unique id for object b";
    output Real numberOfContactPoints "Number of actual contact points";
    output Real cp_a[maxContacts, 3] "Array of contact points on body a, resolved in frame_a";
    output Real cp_b[maxContacts, 3] "Array of contact points on body b, resolved in frame_b";
    output Real depth_a[maxContacts] "Array of penetration depths in body a";
    output Real depth_b[maxContacts] "Array of penetration depths in body a";
    output Real normals_a[maxContacts, 3] "Array of normals on body a, resolved in frame_a";
    output Real normals_b[maxContacts, 3] "Array of normals on body b, resolved in frame_b";
  algorithm
    (numberOfContactPoints, cp_a, cp_b, depth_a, depth_b, normals_a, normals_b) :=
      collisionDetection(maxContacts, r_a, Frames.to_Q(R_a), id_a, r_b, Frames.to_Q(R_b), id_b);
  end collisionDetectionModelica;

  function collisionDetection
    input Integer maxContacts "Maximum number of contact points";
    input Real r_a[3] "Position vector of interaction frame of object a, resolved in world frame";
    input Frames.Quaternions.Orientation Q_a "Quaternion of the orientation of interaction frame of object
      a";
    input String id_a "unique id for object a";
    input Real r_b[3] "Position vector of interaction frame of object b, resolved in world frame";
    input Frames.Quaternions.Orientation Q_b "Orientation of interaction frame of object b";
    input String id_b "unique id for object b";
    output Real numberOfContactPoints "Number of actual contact points";
    output Real cp_a[maxContacts, 3] "Array of contact points on body a, resolved in frame_a";
    output Real cp_b[maxContacts, 3] "Array of contact points on body b, resolved in frame_b";
    output Real depth_a[maxContacts] "Array of penetration depths in body a";
    output Real depth_b[maxContacts] "Array of penetration depths in body a";
    output Real normals_a[maxContacts, 3] "Array of normals on body a, resolved in frame_a";
    output Real normals_b[maxContacts, 3] "Array of normals on body b, resolved in frame_b";
  external "C"
  end collisionDetection;

```



```
replaceable partial function computeInteraction "Compute interaction torques and forces on frame_a,
  resolved in frame_a"
input Real numberOfContactPoints "Number of actual contact points";
input Integer maxContacts "Maximum number of contact points";
input Real r_a[3] "Position of frame_a resolved in world frame";
input Real r_b[3] "Position of frame_b resolved in world frame";
input Real v_a[3] "Velocity of frame_a resolved in world frame";
input Real v_b[3] "Velocity of frame_b resolved in world frame";
input Frames.Orientation R_a "Orientation of frame_a";
input Frames.Orientation R_b "Orientation of frame_b";
input Real cp_a[maxContacts, 3] "Array of contact points on body a, resolved in frame_a";
input Real cp_b[maxContacts, 3] "Array of contact points on body b, resolved in frame_b";
input Real depth_a[maxContacts] "Array of penetration depths in body a";
input Real depth_b[maxContacts] "Array of penetration depths in body b";
input Real normals_a[maxContacts, 3] "Array of normals on body a, resolved in frame_a";
input Real normals_b[maxContacts, 3] "Array of normals on body b, resolved in frame_b";
output SI.Force[3] f_a "Equivalent force applied to frame_a, resolved in frame_a";
output SI.Torque[3] t_a "Equivalent torque applied to frame_a, resolved in frame_a";
end computeInteraction;

equation
(numberOfContactPoints, cp_a, cp_b, depth_a, depth_b, normals_a, normals_b) =
  collisionDetectionModelica(maxContacts, frame_a.r_0, frame_a.R, id_a, frame_b.r_0, frame_b.R, id_b);
assert(numberOfContactPoints <= maxContacts, "Too many contact points");
(f_a, t_a) = computeInteraction(numberOfContactPoints, maxContacts,
  frame_a.r_0, frame_b.r_0, der(frame_a.r_0), der(frame_b.r_0), frame_a.R, frame_b.R,
  cp_a, cp_b, depth_a, depth_b, normals_a, normals_b);
if sum(depth_a + depth_b) > 0 then
  frame_a.f = f_a;
  frame_a.t = t_a;
else
  frame_a.f = {0, 0, 0};
  frame_a.t = {0, 0, 0};
end if;
r = Frames.resolve2(frame_a.R, frame_b.r_0 - frame_a.r_0);
zeros(3) = frame_a.f + Frames.resolveRelative(frame_b.f, frame_b.R, frame_a.R);
zeros(3) = frame_a.t + Frames.resolveRelative(frame_b.t, frame_b.R, frame_a.R) - cross(r, frame_a.f);
end PhysicalInteraction;
```

A New Object-Oriented Approach for Integrating Discrete Element Method into Modelica

Christian Richter¹ Jürgen Weber² Florian Ohser³ Thomas Beutlich⁴

¹Chair of Construction Machines, TU Dresden, Germany, christian.richter1@tu-dresden.de

²Chair of Fluid-Mechatronic Systems, TU Dresden, Germany, weber@ifd.tu-dresden.de

³ESI ITI GmbH, Germany, florian.ohser@esi-group.com

⁴ESI ITI GmbH, Germany, thomas.beutlich@esi-group.com

Abstract

In this paper a new library for co-simulation of discrete element method and Modelica models is presented. For this a component-based approach is used that allows closed modeling and visualization of discrete element systems in a modelica tool. Translation into a native DEM description language and co-simulation is done by a separate compiler and backend. Usage and functionality are shown in a simple use case of a bucket excavator digging a hole.

Keywords: *discrete element method, co-simulation, construction machines*

1 Introduction

Working process of construction and conveying machines is characterized by the interaction with granular materials. In order to allow prospective analysis of machine behavior under real operating conditions, coupled simulations are increasingly used. In these cases, particle-mechanical behavior is reproduced by using discrete element method (DEM). Up to now the creation and calculation of coupled simulations between system models and DEM is very expensive and time-consuming. This effort can be significantly reduced by using the new library presented in this work, which uses a new component-oriented modeling approach for discrete element systems.

1.1 Discrete Element Method

The discrete element method (DEM) is a numerical method for simulating the behavior and motion of large numbers of discrete, interacting objects (Cundall, 1971). In most cases, as done here, these objects are referred as particles. Basis of the method is the calculation of forces acting between the particles or between a particle and an adjacent surface. The basic calculation cycle should be explained briefly below.

After insertion every particle has an initial position and velocity. The simulation loop starts by determining all particle-particle and particle-wall contacts. After that the forces and torques acting on every particle have to be calculated. These forces result on the one hand from field forces like gravity and on the other hand from the particle

deformation as a consequence of collision. For that different contact-models and force-deformation laws are used. Figure 1 shows an example of such a contact model. By summing up all single forces and torques, the translational and angular acceleration of each particle can be obtained. The last step is solving the equations of motion. For that the new positions and velocities are resolved by integrating translational and angular acceleration. The whole loop is repeated for a predetermined number of iterations.

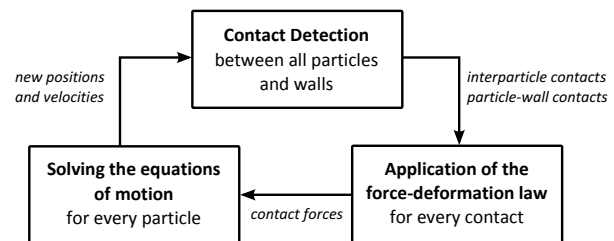


Figure 1. DEM Computation Loop.

1.2 LIGGGHTS®

One of the most used non-proprietary software applications for discrete element simulations is LIGGGHTS® (LAMMPS improved for general granular and granular heat transfer simulations) (Kloss and Goniva, 2011). Main advantages of it are:

- Open source
- Large number of available contact models
- Extensive import and export capabilities for geometry and results
- Various implementations and methods for parallelization of computation (MPI, OpenMP, CUDA)

Besides these points it also has some disadvantages:

- Command-oriented modeling-paradigm
- Complicated syntax
- Elaborate parametrization
- No graphical user interfaces

- No integrated visualization and post-processing capabilities

All these disadvantages will be eliminated with the solution presented here.

1.3 Earlier Solutions and Classification

Coupled simulation of Modelica-based machine models and discrete element method has been a big field of research in recent years. Several solutions have been developed till now. For a better understanding of the differences between them, classification shown in figure 2 (Geimer et al., 2006) should be used.

Core idea and principle of Modelica is to use an equation-based approach for behavioral description and linkage of different models from different physical domains. This is called a *classic simulation*. While this works fine for some domains, like hydraulics or mechanics, this won't work for discrete element systems. To ensure fast contact detection or force computation the specialization of another simulation tool is necessary.

For coupling two different simulation tools special interfaces must be developed. In 2010 we started with the software-framework *SARTURIS* providing a network based coupling of both domains (Kunze et al., 2010). Another coupling technique using functional mock-up units (FMU) was presented in 2012 (Kunze et al., 2012). Based on the functional mock-up interface, a FMU describes a non-proprietary data format containing encapsulated simulation models (Blochwitz et al., 2012). FMU's can be exported and imported by many simulation tools and used for simulation coupling. Referring to figure 2 both solutions are *co-simulations*. The biggest drawback is that distributed modeling, as well as coupling of different input and output values, is very time-consuming and error-prone.

The solution presented in this paper allows a closed modeling and an automatic coupling of DEM and Modelica. A similar approach was used in (Elmqvist et al., 2015). Additionally, the new library uses a component-based modeling paradigm for discrete element models.

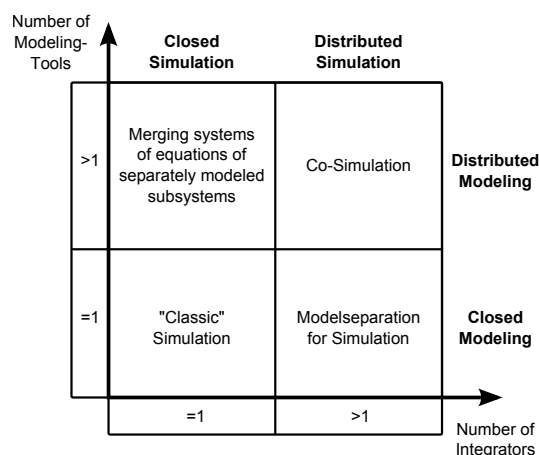


Figure 2. Classification of coupled simulations.

2 Object-oriented design for DEM

As already mentioned LIGGGHTS® follows a command-oriented modeling-paradigm. This is typical for all DEM applications. Usually the user writes an input script containing the whole simulation process. The software reads in this script and executes all commands in sequential order.

One of the core ideas of Modelica is to use an object-oriented design (OOD) for models. For transforming LIGGGHTS® functions into an OOD first an object-oriented analysis (OOA) must be done. According to (Coad and Yourdon, 1991) an object is defined as a real world entity related to the problem domain, with "crisply defined boundaries". Objects are encapsulated with attributes and behaviour. For identifying all objects it's helpful to start writing down all functionalities that should be included in future objects. After that object classes and their design parameters have to be defined fulfilling all these functionalities. One principle is that all objects should be self-explaining and easy to understand for the user. The following table shows a selection of defined classes and some of their functions.

Table 1. Selection of DEM object-classes and related functionalities.

Object	Functionalities
SimulationBox	set timestep size set contact model set boundaries of spatial domain get total particle count/mass
SingleParticle	generate a single particle set diameter define material settings
ParticleSet	load saved particle configurations
ParticleSource	generate a particle stream
ParticleSink	remove particles set particle rate / mass rate define material settings
RigidBody	set position and velocity get forces on body
RegionSensor	get particle count/mass in region

As you can see not all of these objects are real world entities, so it would be better to speak of a component-oriented than of an object-oriented design. In order to keep the terminology as simple as possible it was decided to continue speaking of an object-oriented approach.

After classes, functions and parameters are defined they can be implemented in Modelica. Figure 3 shows the structure of the new library and design of the single object models.

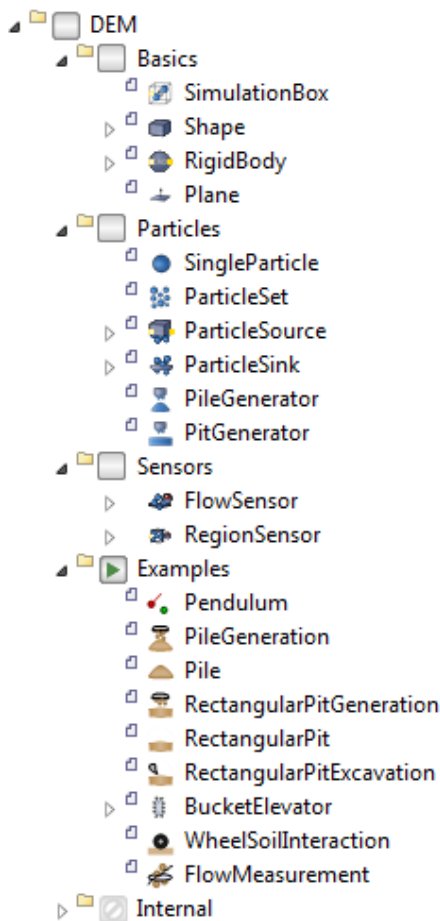


Figure 3. DEM Library in SimulationX®.

3 Simulation coupling

3.1 System architecture

Above library allows the closed modeling of machine and process models. In order to perform a distributed simulation, models have to be subsequently separated again. For a better understanding on how this is done figure 4 shows the system architecture of all simulation components. This structure is divided into front- and back-end.

The front-end essentially consists of the library and a material database, which will be explained more in detail in section 4.2. Each library object contains an internal network client, which is capable to connect and communicate via TCP/IP to a server.

The server is the root node of back-end-structure. It receives the messages coming from the components and forwards them to a special DEM-Slave with an attached compiler. The compiler collects information about all elements in the model and translates them into LIGGGHTS command sequences.

LIGGGHTS itself is not used as an executable but as a shared library with a custom API. Data exchange is much more simplified this way. Furthermore we modified some basic LIGGGHTS function, e.g. for moving meshes, particles sources and sinks during simulation runtime.

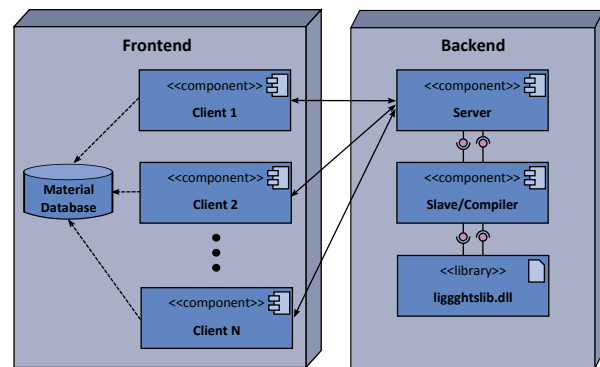


Figure 4. System architecture

3.2 Communication

For communication and data exchange between front- and back-end C-functions accessed by external objects are used. Every object has a *TcpClient*, which is responsible for connecting to the server as well as sending and receiving data. All data is stored in *DataPackages* acting like a send and receive buffer.

```
TcpClient client = TcpClient();
DataPackage outPkg = DataPackage();
DataPackage inPkg = DataPackage();
```

During initialization all clients are connecting to the server. After connection has successfully established initial data is exchanged. This may be for example some positional or geometric information.

```
parameter Boolean isConnected = false;
parameter Boolean isInitialized = false;
```

```
parameter String address = "localhost";
parameter Integer port = 1234;
```

```
initial algorithm
  if isInitialized == false then
    isConnected := connectToHost(
      client, address, port);
  end if;

  setData(outPkg, {/*integer values*/},
    {/*real values*/},
    {/*string values*/});

  if isConnected then
    sendPackage(client, outPkg);
    recvPackage(client, inPkg);
  end if;

  isInitialized:=true;
```

After initialization the main loop starts. Communication between front- and back-end occurs at discrete equidistant time values. For this we use a *sample*-function. At every communication event current model values are pushed to the server. After sending all output data the model waits for the data coming from the server. At the end of the simulation loop some final data is transferred to the server.

```
parameter Real tc(quantity="Basics.Time",
  displayUnit="s") = 0.0001;
Boolean commTrigger(start=false, fixed=true)
  = sample(0, tc);
```

```
algorithm
  if isConnected then
    when commTrigger then
      // set current data to outPkg
      // send and receive packages
      // extract data from inPkg
    elseif terminal() then
      // send final information
    end when;
  end if;
```

4 Pre-processing

4.1 Basic simulation settings

All basic simulation and communication settings are defined in the *SimulationBox*. Similar to the *World* component in *Modelica.Mechanics.MultiBody* package every DEM model must contain one *SimulationBox*. This is ensured by an *outer* construct in the code.

```
outer DEM.Basics.SimulationBox simBox;
```

This way all objects have access to basic parameters like host address and port.

```
equation
  address = simBox.address;
  port = simBox.port;
```

In order to keep computation costs low it's necessary to define boundaries for the DEM space. These boundaries can be fixed (particles will be removed if they leave the spatial domain) or dynamic growing.

4.2 Material definitions

The parameterization of the material properties of DEM models is very complicated and presents a problem that has not been completely solved. In order to increase the operating convenience of the library, a material database has been created, which contains parameter sets for the most realistic description of different granular materials. The valid parameter sets were determined by comparing laboratory measurements and simulation. Various calibration tests were used. Among other things, the shear force, the angle of inclination as well as the transit time of different granular substances were investigated. The selection of the materials to be examined followed the possible future application areas of the total solution. Sand and gravel (construction machinery), hard coal, brown coal, iron ore and potash (mining and conveyor technology) as well as corn and wheat (agricultural machinery and food technology) were investigated.

For the representation of large rocks or boulders a function was implemented, which allows the use of Multisphere materials. In this case, a composite of several spheres is formed, which are inseparably connected to each

other. This function was not provided in the original work package. It has been implemented since it means a considerable added value for the user and thus for the marketing of the final product. The interpolation of a stone by a Multisphere object is shown in figure 5.



Figure 5. Multisphere approximation of a stone

5 Post-processing

5.1 Visualization

Besides representing time-dependent state values in diagrams, 3D visualization is an important part of modern post processing. For this the *ModelicaServices* package comes with some models for animation and visualization of certain predefined shapes such as cylinders, boxes or imported STL- and DXF-geometries. The implementation of this package can vary from one Modelica tool to another.

These capabilities are very limited to basic shapes and not sufficient for the visualization of large particle systems. Though there is an animation body for spheres, it's not very advising to use it, because for n particles it would be necessary to create n animation submodels. This would increase the number of internal equations and downgrade performance.

In our implementation we created a new animation body called *DEMPoints*. We propose to extend *ModelicaServices* with such a model.

For large-scale systems up to one million particles 3D representation itself takes a lot of computation costs. For that reason it's possible to switch between the options *splats*, *diamonds* or *spheres*, which supply different levels of details and performance.

5.2 Sensors

In discrete element simulations it's often necessary to measure particle specific values. For that we enhanced regular LIGGGHTS capabilities by some special sensor functions. Different shaped *RegionSensors* can be used to evaluate the number and mass of particles in a specific volumetric region. Sensor position and size can change during simulation runtime. It's also possible to attach sensors to rigid bodies. One use case would be the measurement of bucket filling level during the digging process of an excavator.

To check if a particle is inside a specific region we use a very simple and efficient algorithm. Consider there's a cuboid region sensor with the position vector \mathbf{x}_S , orientation matrix \mathbf{R}_S and dimensions l_x , l_y and l_z . Now we want

to find out if a particle P is inside the sensor region. First thing we have to do is calculating the absolute difference vector of the particles global position \mathbf{x}_P and the position of the sensor \mathbf{x}_S (eq. 1). After that we transform this absolute difference vector into the relative coordinates of the sensor (eq. 2).

$$\mathbf{x}_{PS_{abs}} = \mathbf{x}_P - \mathbf{x}_S \quad (1)$$

$$\mathbf{x}_{PS_{rel}} = \mathbf{R}_S \cdot \mathbf{x}_{PS_{abs}} \quad (2)$$

To determine if the particle is inside or not we have to check the following logic equation.

$$\begin{aligned} inside_{cuboid} = & |\mathbf{x}_{PS_{rel},x}| < 0.5 \cdot lx \wedge \\ & |\mathbf{x}_{PS_{rel},y}| < 0.5 \cdot ly \wedge \\ & |\mathbf{x}_{PS_{rel},z}| < 0.5 \cdot lz \end{aligned} \quad (3)$$

For spherical region sensors equation 2 can be omitted. Checking is done as shown in equation 4 where r is the radius of the sphere.

$$\begin{aligned} inside_{sphere} = & |\mathbf{x}_{PS_{abs},x}| < r \wedge \\ & |\mathbf{x}_{PS_{abs},y}| < r \wedge \\ & |\mathbf{x}_{PS_{abs},z}| < r \end{aligned} \quad (4)$$

Just mention that there's a second sort of sensors called *FlowSensors*. They are used for measuring the number and mass of particles passing two dimensional surfaces. Computation algorithm for these kind of sensors is basically the same like for contact detection and shouldn't be explained here.

6 Use cases

6.1 Bucket Excavator

As first use case a bucket excavator digging a hole should be simulated. The excavator itself was modeled as multi-body system, which can easily be extended by hydraulic or electric components. For all parts which should interact with the granular material – in this case just the bucket – the new library component *CADPart* was used. As next step a pit of size 6.0 x 2.0 x 1.0 meters was generated by using the *PitGenerator* element. The new library has a direct interface to a database containing predefined materials, as described in section 4.2. So, the material chosen for the pit was *gravel*. Figure 6 shows the 3D view of a running simulation. As you can see particles are visualized directly in SimulationX®.

6.2 Loaded Truck

The in figure 7 shown model of the truck allows to determine the hydraulic forces in the main cylinder of the truck during the loading and unloading of bulk material by taking the elastic suspension into account. It is also possible to determine the influence of the moving bulk material of the drivability during different maneuvers and demonstrates additional features and the capabilities of the developed library. In the background is a *ParticleSource*,

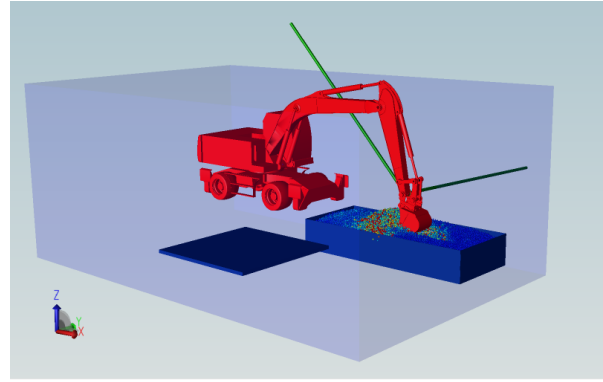


Figure 6. Excavator simulation

which the truck was being loaded with during the simulation. For the *ParticleSource* it is also possible to chose a predefined material of the database. Additionally, the feature is demonstrated that accelerated, rotating and automatically increasing simulation rooms are supported during co-simulation. With the *RegionSensors* the number of particles and the mass of the load can be evaluated which interacts with the *CADParts*. The ground is a *Plane* for the DEM simulation without any feedback to the system simulation.

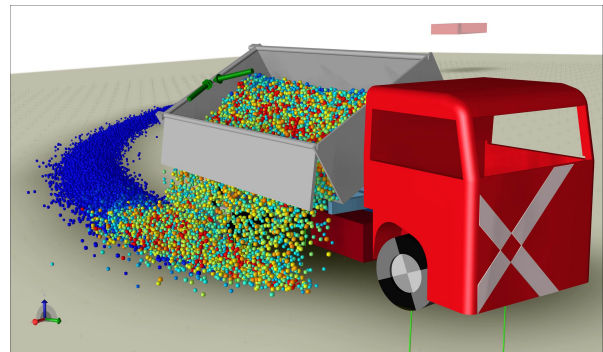


Figure 7. Loaded truck simulation

7 Conclusion and Outlook

In this work, a new concept was presented allowing the closed modelling of machine models and discrete element systems in one simulation tool. For that the command-oriented modeling technique many DEM applications work with was transferred into an object-oriented design approach. This approach allows to perform DEM simulations for inexperienced users who are not familiar with the DEM. But even for very experienced users, the new library will make it much easier to build up DEM models, run coupled simulations and analyze and document the results.

By supporting additional LIGGGHTS® features and additional wizards the modeling could be simplified, the possibilities expanded and the useability of DEM models improved.

Additional LIGGGHTS[®] features could be the breaking and bonding of material. This field of DEM simulation and the determination of the appropriate material parameters is currently the content of some research-projects. But the results of that projects are far away to be used in coupled simulation. Additional wizards could be developed for example to allow the user to vary material parameters, create their own materials, or generate a multisphere body. Furthermore, the analysis possibilities of the LIGGGHTS[®] results in SimulationX could be expanded further in order to increase the added value of the coupled simulation. For this and for all other enhancements, we are looking forward to the feedback of future users and interested parties.

Acknowledgements

This work is part of the project DEM-4-X funded by the BMWi (Federal Ministry for Economic Affairs and Energy, Project No.: 2055606KM4). The authors are deeply grateful for the financial support.

References

Torsten Blochwitz, Martin Otter, Johan Åkesson, Martin Arnold, Christoph Clauß, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauß, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In Martin Otter and Dirk Zimmer, editors, *Proceedings of the 9th International Modelica Conference*, Munich, Germany, September 2012. doi:10.3384/ecp12076173.

Peter Coad and Edward Yourdon. Object oriented analysis. 1991.

Peter A. Cundall. A computer model for simulating progressive, large-scale movements in blocky rock systems. In *Proc. Symp. Int. Rock Mech.*, volume 2, Nancy, 1971.

Hilding Elmqvist, Axel Goteman, Vilhelm Roxling, and Toheed Ghandriz. Generic Modelica Framework for MultiBody Contacts and Discrete Element Method. In Peter Fritzson and Hilding Elmqvist, editors, *Proceedings of the 11th International Modelica Conference*, Versailles, France, September 2015. doi:10.3384/ecp15118427.

Marcus Geimer, Thomas Krüger, and Peter Linsel. Co-Simulation, gekoppelte Simulation oder Simulationskopplung? Ein Versuch der Begriffsvereinheitlichung. *O+P Zeitschrift für Fluidtechnik - Aktorik, Steuerelektronik und Sensorik*, 50(11-12):572–576, 2006.

Christoph Kloss and Christoph Goniva. *Open Source Discrete Element Simulations of Granular Materials Based on Lammmps*, volume 2, pages 781–788. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2011. ISBN 9781118062142. doi:10.1002/9781118062142.ch94.

Günther Kunze, Andre Katterfeld, and Tina Grüning. Simulation maschineller Erdbauprozesse. In *15. Fachtagung Schüttgutförderertechnik*, Munich, Germany, October 2010.

Günther Kunze, Andre Katterfeld, Christian Richter, Hendrik Otto, and Christian Schubert. Plattform- und Softwareunabhängige Simulation der Erdstoff-Maschine Interaktion. In *5. Fachtagung Baumaschinentechnik*, Dresden, Germany, September 2012.

Modeling and Simulation of Wheel Driving Systems based on Terramechanics for Planetary Exploration Rover using Modelica

Hiroki Yoshikawa¹ Takatsugu Oda¹ Kenichiro Nonaka¹ Kazuma Sekiguchi¹

¹Mechanical Systems Engineering, Tokyo City University, Japan, {g1681237, g1591201, knonaka, ksekiguc}@tcu.ac.jp

Abstract

Planetary exploration rovers have to accomplish various missions on uneven and loose terrain. In recent years, systems of rovers adopting terramechanics which determine the force and moment characteristics of the wheel on loose soil is studied. In this study, using Modelica language, we construct a wheel model based on terramechanics, and we identify the wheel characteristics as a linear for a control. We conduct a numerical simulation of the rover using a controller including the identified longitudinal force model. It is shown that when the rover follows a straight line on a plane, the longitudinal force model identified using known soil parameters has sufficient accuracy on the wheel response based on terramechanics and could be used as a control model. *Keywords: terramechanics, modeling, identification, space robots, control system*

1 Introduction

In recent years, research and development of planetary exploration rovers in various configurations have been carried out to investigate the planets. Planetary exploration rovers have to achieve a stable traveling on uncertain and severe terrain. The planet surface is covered with fine deposits, called regolith, and uneven terrain such as craters and rocks. Various planetary exploration rovers have been developed which is equipped with, for example, wheel mechanisms with suspensions to adapt to the planetary surface, crawler mechanisms to enhance the drawbar pull or leg mechanisms to climb over steps (Seeni et al., 2008). Also, NASA is planning to operate a hybrid rover "ATHLETE" which is equipped with wheel and leg mechanisms.

When rovers move on planetary surface, it is important to take into account of terramechanics which governs a relation between soft soil and the driving system of rovers. In order to analyze the effect of the soil, a semi-empirical model proposed by Bekker using the experimental results and a model using Discrete Element Method (DEM) without dependence on wheel parameters are studied (Nakashima et al., 2010). Combining DEM with Finite Element Method (FEM), the simulation using Soil Contact Model (SCM) of Multi-Body System (MBS) which analyzes the more detailed soil movement is proposed (Krenn and Gibbesch, 2011). The deformation of soil and the op-

timal wheel shape are analyzed through these simulations to consider efficient travel on loose soil. However, it is not suitable for the motion analysis of the rover, since it takes large calculation time with FEM and DEM which handle huge complicated elements in order to ensure reasonable accuracy (Taheri et al., 2015).

As for the studies about the control based on terramechanics, designing path (Ding et al., 2014) and analysis of traveling performance while ascending (Ishigami et al., 2007) is conducted. A slip ratio control of the wheels on loose soil using sliding mode control for the rover model considering terramechanics is proposed (Gu et al., 2007). In addition, another slip ratio control of the wheels using PID control to adapt the parameters of terrain surface is studied (Iagnemma and Dubowsky, 2004).

While it is desirable to conduct experiments in space environments to verify these models, computer simulations are preferred considering huge cost. However, it is difficult to compensate for the differences of planetary environments like gravitational field and so on (Pulecchi and Lovera, 2006). To conduct a simulation with minimized the error between the simulation model and the actual equipment is minimized, it is extremely effective for comprehensive analysis through the more detailed rover model and contact model of loose soil. The simulations using Modelica language and modeling tool of physical domains attract a lot of attention. We do not need to care about causality to create the wheel model based on terramechanics such as slip ratio, sideslip angle and velocity of wheel, since Modelica is an equation based language. These features enable us to combine the wheel and rover model effectively.

In previous our study, we conduct simulations considering the space environment using the fundamental control system and the robot model designed by Modelica. In this study, using Modelica language we design a rover model equipped with *the terramechanics model* to conduct a simulation with more detail model. We identify *the identified model* which expresses the relationship between input torque and longitudinal force based on the simulation results of the terramechanics model. Because the terramechanics model is too complex to use in a controller, we design the motion controller using the identified model. We evaluate the effectiveness of identified model through numerical simulations. Therefore,

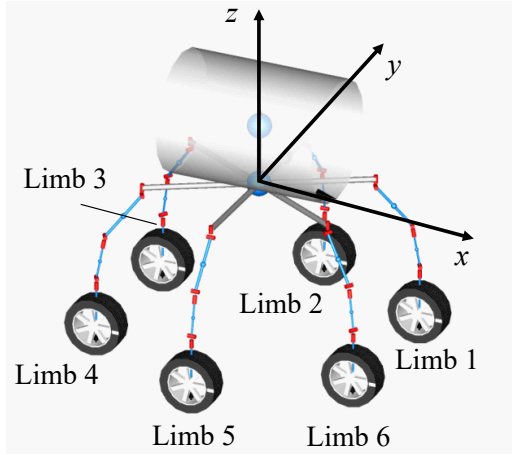


Figure 1. Leg-wheel mobile robot model with six joints of each limb.

the identified model approximates the characteristics of the terramechanics model.

2 Modeling controlled object

2.1 Leg-wheel mobile robot model

Figure 1 depicts a rover model of the controlled object (Yoshikawa et al., 2016). We use a lunar exploration rover "ATHLETE" developed by NASA/JPL as a reference model (Wilcox et al., 2007). This rover is equipped with six limbs with six joints while wheels achieve a high movement performance and accommodate a wide range of tasks using the redundancy. We create this rover model by using Modelica language to control the degree of freedom of the leg-wheel mechanisms with similar movements of ATHLETE. The coordinate system of the rover is attached at the center of the body. The limb has a number to be distinguished from the others in this coordinate system, as depicted in Figure 1.

2.2 Wheel model based on terramechanics

2.2.1 Assumptions of the wheel model

We introduce terramechanics to the wheel model of the controlled objects. We make reference to semi-empirical model (Ishigami et al., 2007) (Wong, 2001) to the wheel model based on terramechanics. Figure 2 depicts the rigid wheel rolling on loose soil. The assumptions of the wheel model are as follows:

- The contact surface between wheels and the ground is flat.
- Radius r and width b of wheel have enough rigidity.
- Wheel rotation does not affect a frontal soil.
- The frontal soil is constricted and released at the rear of the wheel.

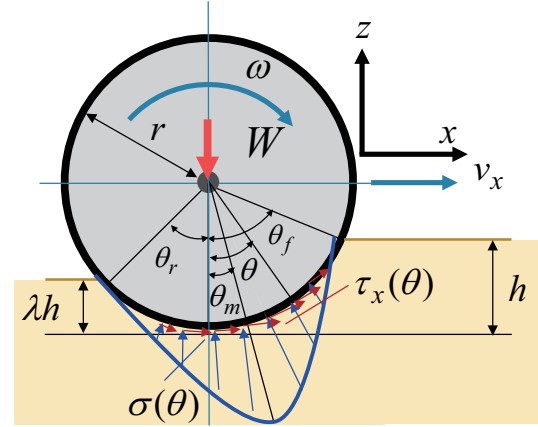


Figure 2. Normal stress and shear stress distribution concept of terramechanics while rolling (Ishigami et al., 2007).

- Lateral and vertical dynamics of wheels are not considered.

Figure 2 depicts the geometry of the wheel model based on these assumptions; the empirical equation is described in the following section.

2.2.2 Entry and exit angle of wheel

The forces generated from the wheel are calculated by integrating a stress distribution developed between the wheel and terrain surface. Entry angle θ_f and exit angle θ_r are introduced in order to decide the dynamic contact area of the wheel. The entry angle and exit angle are defined as follows:

$$\theta_f = \cos^{-1}\left(1 - \frac{h}{r}\right), \quad (1)$$

$$\theta_r = \cos^{-1}\left(1 - \frac{\lambda h}{r}\right), \quad (2)$$

where h is the sinkage of wheel and λ is the volume ratio of soil.

2.2.3 Specific wheel angle θ_m

The normal stress distribution σ (the blue curve in Figure 2) arises in the normal direction of the wheel while rolling. This normal stress distribution is approximated by the parabolic curve. The maximum stress angle θ_m is an angle at which the value of normal stress is maximum as follows:

$$\theta_m = (a_0 + a_1 \kappa) \theta_f, \quad (3)$$

where a_0, a_1 is a constant value and κ is slip ratio. Slip ratio is represented by using a translational velocity of the wheel v_x and angular velocity of the wheel ω :

$$\kappa = \begin{cases} \left(\frac{r\omega - v_x}{r\omega} \right) & (r\omega > v_x) \\ \left(\frac{r\omega - v_x}{v_x} \right) & (r\omega < v_x). \end{cases} \quad (4)$$

2.2.4 Normal stress distribution model based on bekker's equation

The normal stress distribution model $\sigma(\theta)$ based on soil pressure equation proposed by Bekker is divided into two areas: the front parts of the specific wheel angle $\sigma_f(\theta)$ ($\theta_m \leq \theta < \theta_f$) and the rear parts $\sigma_r(\theta)$ ($\theta_r < \theta \leq \theta_m$). The normal stress distribution model of the wheel is defined as follows:

$$\sigma_f(\theta) = r^n \left(\frac{k_c}{b} + k_\phi \right) [(\cos \theta - \cos \theta_f)]^n, \quad (5)$$

$$\sigma_r(\theta) = r^n \left(\frac{k_c}{b} + k_\phi \right) \left[\cos \left\{ \theta_f - \frac{\theta - \theta_r}{\theta_m - \theta_r} (\theta_f - \theta_m) \right\} - \cos \theta_f \right]^n, \quad (6)$$

where k_c is pressure-sinkage module depending on the viscosity, k_ϕ is pressure-sinkage module depending on the friction and n is the sinkage exponent depending on sinkage of soil.

2.2.5 Shear stress model of wheel

Shear stress model is defined as follows:

$$\tau = \tau_{\max} (1 - e^{-j/k}), \quad (7)$$

$$\tau_{\max} = c + \sigma \tan \phi, \quad (8)$$

where c is the cohesion stress of the soil, ϕ is the internal friction angle of the soil, j is the soil deformation and k is the shear deformation modules. The shear stress of x direction τ_x is obtained by assigning σ to Eq. (8):

$$\tau_x = (c + \sigma(\theta) \tan \phi) (1 - e^{-j_x(\theta)/k_x}), \quad (9)$$

where k_x is the shear deformation modules of x direction and j_x is the soil deformation of x direction as follows:

$$j_x(\theta) = r[\theta_f - \theta - (1 - \kappa)(\sin \theta_f - \sin \theta)]. \quad (10)$$

2.2.6 Vertical and longitudinal force of wheel

The vertical force F_z which is equal to the load of the wheel is calculated by the summation of the normal and shear stress of z direction as follows:

$$F_z = rb \int_{\theta_r}^{\theta_f} \{ \tau_x(\theta) \sin \theta + \sigma(\theta) \cos \theta \} d\theta. \quad (11)$$

The normal and shear stress of the wheel can be calculated using the each contact angle θ_f and θ_r determined by the sinkage of the wheel h . Then, the longitudinal force is calculated by the summation of normal and shear stress of x direction as follows:

$$F_x = rb \int_{\theta_r}^{\theta_f} \{ \tau_x(\theta) \cos \theta - \sigma(\theta) \sin \theta \} d\theta. \quad (12)$$

The rolling resistance torque T_x is calculated using the shear stress as follows:

$$T_x = r^2 b \int_{\theta_r}^{\theta_f} \tau_x(\theta) d\theta. \quad (13)$$

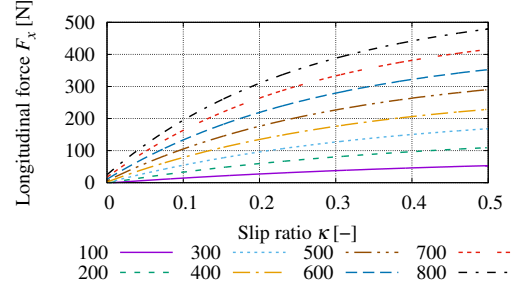


Figure 3. Relationship between sliratio and longitudinal force with respect to load of the wheel.

2.2.7 Longitudinal characteristics

Figure 3 shows longitudinal force F_x with respect to the slip ratio of the wheel when the load of it increases every force 100N within 100N ~ 800N. As the slip ratio increases, the longitudinal force is saturated as Figure 3 indicates. In addition, as indicated in Figure 3, for the same slip ratio, the longitudinal force generated by the wheel depends on the load. It indicates that the increasing ratio of F_x decreases as the load grows.

3 Identification of the wheel model

3.1 Identified model

In this section, to design a rover controller in which the identified model is additionally used, we identify the longitudinal force of the terramechanics model. We approximate the longitudinal force generated at the wheel by a linear first-order system. A step wheel torque is imposed on the wheel, then the wheel response data on the longitudinal force and the slip ratio is sampled. The identified longitudinal force model is depicted in Figure 4. The identified model is separated into two blocks: one for calculating the slip ratio by the wheel torque and the other for calculating the longitudinal force by the slip ratio. This separation helps to capture the feature of the physical relationship.

In the wheel model based on terramechanics, the wheel sinkage which depends on load is decided by the optimization. In order to identify the longitudinal force corresponding to the load change, we represent the parameters of the first order system using a look up table (LUT). Using the LUT in the identified model, we can consider the generated force due to influences of soil deformation caused by load change. Firstly, the gain K_{LUT} and the time constant T_{LUT} are decided using the LUT. The reference values of the LUT are the wheel load W and wheel torque T_w . Secondly, the relationship between slip ratio κ and longitudinal force F_x with respect to load change is depicted in Figure 5. Each point in this Figure represents the reference results of the terramechanics model. To express these relationships in an equation, we approximate it as follows:

$$F_x(W, \kappa) = a(W)\kappa + b(W), \quad (14)$$

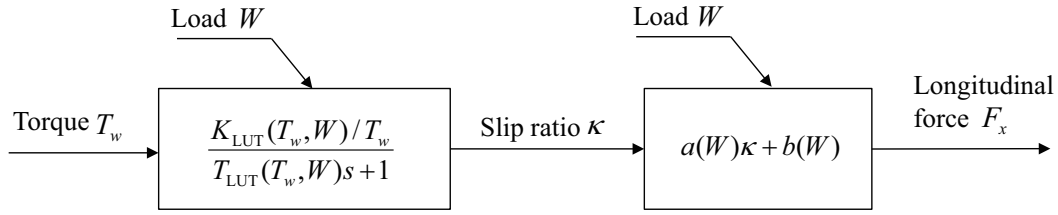


Figure 4. Identification model from wheel torque to longitudinal force of wheel.

Table 1. Precision of identified model to response of wheel based on terramechanics in the load 275, 575, 875 N.

Load	275 N	575 N	875 N
Precision	82.1%	96.8%	96.2%

where $a(W)$ and $b(W)$ are the coefficient derived from the quadratic expressions with respect to load change, as shown in Figure 6 and Figure 7, respectively.

3.2 Verification of identification model

It is noted that the identified model using LUT is an approximation which essentially includes interpolation error. Figure 8 indicates the longitudinal force obtained by the wheel based on terramechanics and the identified model of it in the load $W = 575$ N which is the interpolated region. A precision of identified model is calculated using the following equation:

$$\text{Fit} = \left(1 - \frac{\sqrt{\sum_{k=1}^N [\hat{y}(k) - y(k)]^2}}{\sqrt{\sum_{k=1}^N [y(k) - \bar{y}]^2}} \right) \times 100, \quad (15)$$

where $\hat{y}(k)$ is the output of identified model, $y(k)$ is the output obtained by the controlled object, $\bar{y}(k)$ is the average of it and N is the number of data. In the case that the load is not the reference results of the terramechanics model, for example $W = 275, 575, 875$ N, the precision for the step response of the wheel torque is shown in Table 1. As a result, all of the precision is over 82%. If you need to increase the precision, the degree of the approximate expression will be changed more high degree. Therefore, the identified model sufficiently approximates the longitudinal force of the wheel even when the LUT refers to the interpolated load.

4 Simulation

In this section, to evaluate the accuracy of the identified model for the rover, we design the controller system using the identified model, and confirm the response through the numerical simulation.

4.1 Controller design

To verify whether the rover model with terramechanics wheel model could be controlled using the identified model through the numerical simulation, we construct the

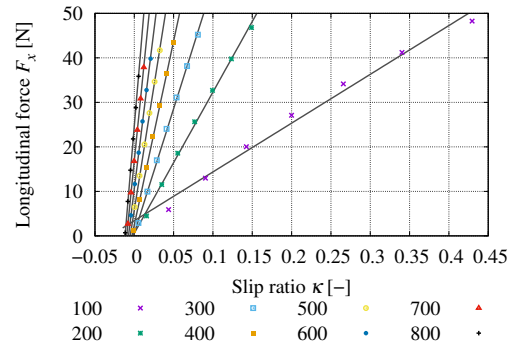


Figure 5. Reference results of the terramechanics model of slip ratio and longitudinal force obtained by step input of wheel torque and linear approximation of them.

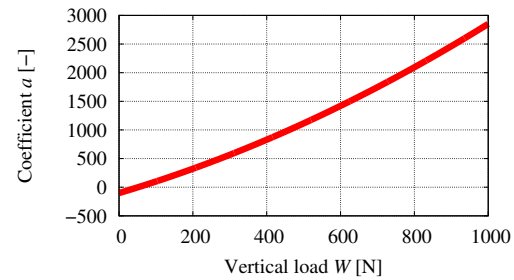


Figure 6. Coefficient a of polynomial.

controller system: the identified model in section 3 is used as the control model to calculate a wheel torque from a velocity controller. Then, the torques are imposed on the rover model (plant model in section 2). The system calculates the wheel torque by feedback control so that the rover achieves the target velocity. In vehicle motion controller, we regard the rover as a mass point model for calculating the rover force on the CoG. To achieve the designed motion, it is assumed that each wheel generate the same longitudinal force as follows:

$$F_{x, \text{all}}/6 = \tilde{f}_{w, i}, \quad (16)$$

where $F_{x, \text{all}}$ is whole longitudinal force of the rover, $\tilde{f}_{w, i}$ is longitudinal force of each wheel and subscript $i = 1 \sim 6$ indicates the limbs number. Each wheel torque $T_{w, i}$ is calculated using the inverse identified longitudinal force model. Then, to realize the inverse model which is the linear first order system, we add the second order filter in front of it so that the model become the strictly proper

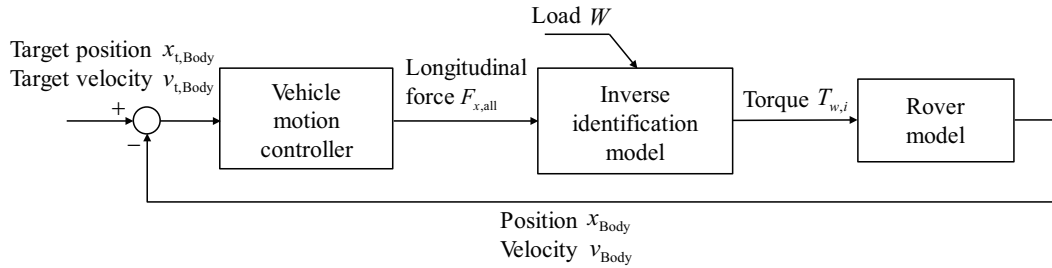
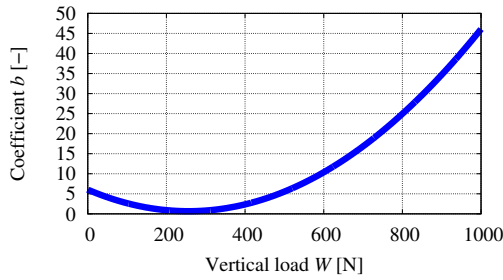
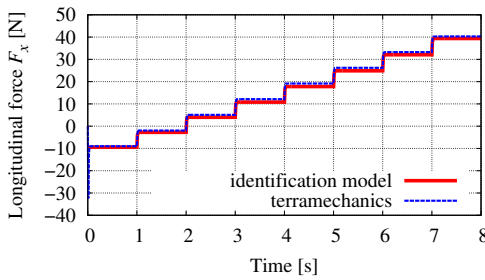


Figure 9. Controller system.

Figure 7. Coefficient b of polynomial.Figure 8. Example of comparison forward identified model with terramechanics wheel model ($W = 575$ N).

model. Using the inverse model, we verify the simple characteristics of the identified model when the model is applied to the rover model. The filter is defined as follows:

$$\frac{\omega_n^2}{s^2 + 2\omega_n s + \omega_n^2}, \quad (17)$$

where ω_n is natural angular frequency and set to be 342 rad/s. The calculated wheel torque is imposed on each wheel of the rover model which indicates the right block depicted in Figure 9.

4.2 Simulation conditions

To verify the response of the wheel model, we conduct a simulation that the rover tracks the target velocity on a plane while keeping the initial posture of the rover. The reference path is the straight line including an acceleration areas. In this simulation, we assume that the lunar surface is covered with regolith uniformly. The parameter of rover mass, target value, soil and wheel shape are indicated in Table 2 (Ishigami et al., 2007).

Table 2. Parameter of rover, wheel and soil.

Parameter	Value	Unit
Rover mass M	1570	kg
Target position $x_{t,Body}$	$1.0 \times \text{time}$	m
Target velocity $v_{t,Body}$	1.0	m/s
Wheel radius r	0.355	m
Wheel tread b	0.175	m
Cohension stress c	0.80	kPa
a_0	0.4	-
a_1	0.15	-
Pressure-sinkage module k_c	1.37×10^3	N/m ^{$n+1$}
Pressure-sinkage module k_ϕ	8.14×10^5	N/m ^{$n+2$}
Soil deformation module k_x	0.036	m
Sinkage exponent n	1.0	-
Friction angle ϕ	37.2	deg
Wheel sinkage ratio λ	0.90	-

4.3 Results and discussions

The simulation results using the identified model are shown in Figure 10. Since the rover moves on a straight line and arranges a symmetric leg position in this simulation condition, we plot the results of the Limb1 ~ 3. Figure 10 (a) through (h) depict the wheel torque, the wheel resistance torque, the vertical force of each wheel, each wheel sinkage, the slip ratio of each wheel, the longitudinal force of each wheel, the velocity of the rover and the desired longitudinal force, respectively.

As shown in Figure 10 (a), the identified model calculates the wheel torque considering the influence of resistance torque, so that the rover enable the wheel to drive smoothly. It is because the controller implicitly considers the effect of resistance which is depicted in Figure 10 (b).

The inertia force due to the acceleration influences that the load distribution of the wheel biases backward of the rover. As a result, Figure 10 (c) indicates that, during the acceleration, the vertical force of the Limb 3 increases while that of the Limb 1 decreases. The load change affects the change of the wheel sinkage h as depicted in Figure 10 (d). The wheel sinkage h is adapted to the vertical force, so that the physical adequacy of the wheel model based on terramechanics can be confirmed. The wheel torque is calculated using the identified model, so that

Table 3. RMS error of longitudinal force.

	Limb 1	Limb 2	Limb 3
RMS error	0.7 N	0.4 N	0.5 N

the wheel arises the different slip ratio, as shown in Figure 10 (e). Accordingly, the longitudinal force of the each wheel is generated uniformly as depicted in Figure 10 (f) even when the load of the each wheel is different. Figure 10(g) indicates that the rover accelerates until the translational velocity reaches 1.0 m/s. Figure 10(h) depicts the actual and desired longitudinal force of limb 1 as a representative example. Table 3 shows the RMS error between the actual and desired longitudinal force in limb 1 ~ 3. This difference in the longitudinal force is caused by the approximation error of the identified model. The precision of the identified model tends to lower as the load decreases, as shown in Table 1. Thus, since the load of Limb 1 decreases during the acceleration, the RMS error becomes the largest. The maximum longitudinal force reaches about 30 N. Nevertheless, all RMS error is below 1.0 N. Although the actual longitudinal force is not equal to the desired, the influence is adequately suppressed by the feedback control.

Each wheel can generate the desired longitudinal force due to calculating the wheel torque corresponding to load change. Moreover, through the use of the identified model, the wheel torque considering the influence of loose soil can be obtained without the optimal calculation for a decision of the wheel sinkage. Therefore, it is shown that the identified longitudinal force model based on the more detailed model has the high accuracy when the model is applied to the rover controller.

5 Conclusions

In this paper, we construct the wheel model based on terramechanics derived from semi-empirical model by using Modelica language. In order to consider the longitudinal force of the constructed wheel model, we approximate it by the linear first order system. Designing the controller using the identified model, we investigate the influence on driving systems of the rover moving on loose soil. The simulation results indicate that the identified model can adapt the influence of load change and consider the soil deformation, so that the identified model has a high accuracy. With reference to the model used in the control, it is important to simplify the structure and identify the characteristic. Consequently, the use of the identified longitudinal force model contributes to a control design for the rover.

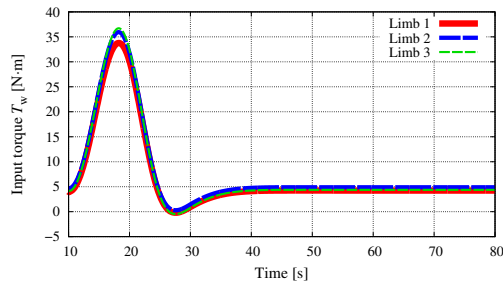
As for the problems to be solved from now on, to enhance the mobility on loose soil, the lateral force of the wheel should be identified to design a controller.

6 Acknowledgments

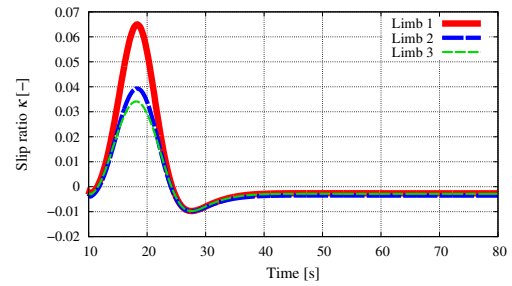
The authors gratefully acknowledge the support of Grant in Aid for Scientific Research (C) No.15K06155 of Japan.

References

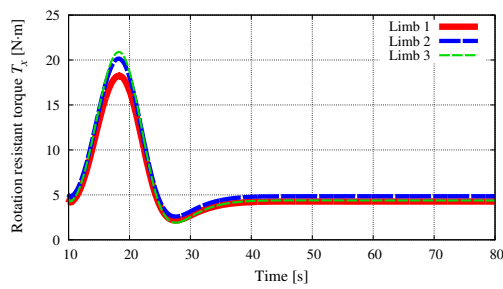
- Liang Ding, Hai-bo Gao, Zong-quan Deng, Zhijun Li, Ke-ruì Xia, and Guang-ren Duan. Path-following control of wheeled planetary exploration robots moving on deformable rough terrain. *The Scientific World Journal*, 2014, 2014.
- Kanfeng Gu, Yingzi Wei, Hongguang Wang, and Mingyang Zhao. Dynamic modeling and sliding mode driving control for lunar rover slip. In *Integration Technology, 2007. ICIT'07. IEEE International Conference on*, pages 36–41. IEEE, 2007.
- Karl Iagnemma and Steven Dubowsky. Traction control of wheeled robotic vehicles in rough terrain with application to planetary rovers. *The international Journal of robotics research*, 23(10-11):1029–1040, 2004.
- Genya Ishigami, Akiko Miwa, Keiji Nagatani, and Kazuya Yoshida. Terramechanics-based model for steering maneuver of planetary exploration rovers on loose soil. *Journal of Field robotics*, 24(3):233–250, 2007.
- Rainer Krenn and Andreas Gibbsch. Soft soil contact modeling technique for multi-body system simulation. In *Trends in computational contact mechanics*, pages 135–155. Springer, 2011.
- H Nakashima, H Fujii, A Oida, M Momozu, H Kanamori, S Aoki, T Yokoyama, H Shimizu, J Miyasaka, and K Ohdoi. Discrete element method analysis of single wheel performance for a small lunar rover on sloped terrain. *Journal of Terramechanics*, 47(5):307–321, 2010.
- Tiziano Pulecchi and Marco Lovera. A modelica library for space flight dynamics. In *In Proceedings of the 5th International Modelica Conference*. Citeseer, 2006.
- Aravind Seeni, Bernd Schafer, Bernhard Rebele, and Nikolai Tolyarenko. Robot mobility concepts for extraterrestrial surface exploration. In *Aerospace Conference, 2008 IEEE*, pages 1–14. IEEE, 2008.
- Sh Taheri, C Sandu, S Taheri, E Pinto, and D Gorsich. A technical survey on terramechanics models for tire-terrain interaction used in modeling and simulation of wheeled vehicles. *Journal of Terramechanics*, 57:1–22, 2015.
- Brian H. Wilcox, Todd Litwin, Jeff Biesiadecki, Jaret Matthews, Matt Heverly, Jack Morrison, Julie Townsend, Norman Ahmad, Allen Sirota, and Brian Cooper. ATHLETE: A cargo handling and manipulation robot for the moon. *Journal of Field Robotics*, 27(5):421–434, 2007.
- Jo Yung Wong. *Theory of ground vehicles*. John Wiley & Sons, 2001.
- Hiroki Yoshikawa, Takatsugu Oda, Kenichiro Nonaka, and Kazuma Sekiguchi. Modeling and simulation for leg-wheel mobile robots using modelica. In *The First Japanese Modelica Conferences, May 23-24, Tokyo, Japan, number 124*, pages 55–60. Linköping University Electronic Press, 2016.



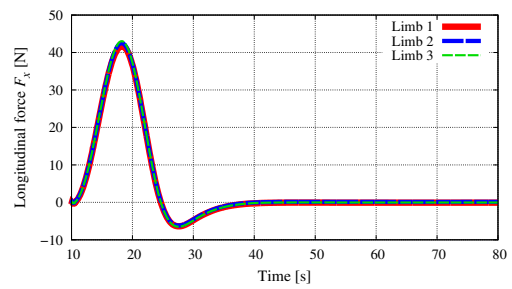
(a) Wheel torque of each wheel.



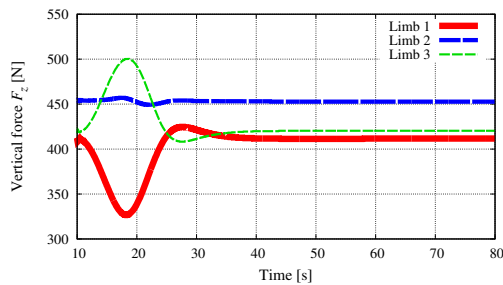
(e) Slip ratio of each wheel.



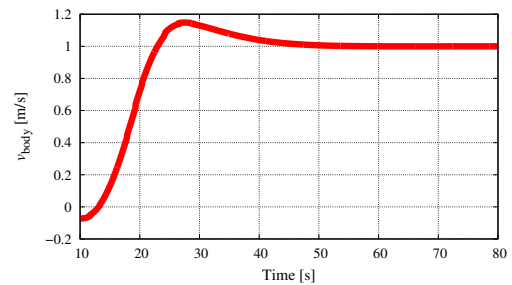
(b) Rolling resistance torque of each wheel.



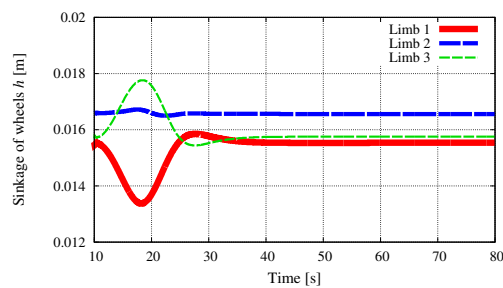
(f) Longitudinal force of each wheel.



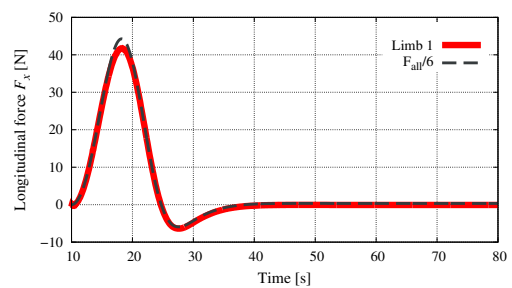
(c) Vertical force of each wheel.



(g) Velocity of rover model.



(d) Sinkage of each wheel.



(h) Actual and desired longitudinal force of limb 1.

Figure 10. Rover driving simulation using longitudinal force model considering terramechanics for driving force distribution.

The Jet Propulsion Library: Modeling and simulation of aircraft engines

Michael Sielemann¹ Anand Pitchaikani² Nithish Selvan² Majed Sammak³

¹Modelon Deutschland GmbH, Germany, michael.sielemann@modelon.com

²Modelon Engineering Pvt. Ltd., India, {anand.pitchaikani, nithish.selvan}@modelon.com

³Modelon AB, Sweden, majed.sammak@modelon.com

Abstract

The Jet Propulsion Library is a new Modelica library that provides a foundation for modeling and simulation of jet engines, and the model-based design of integrated aircraft systems. It provides a fully rigorous foundation for sizing and performance computations, and provides a number of advantages over existing domain-specific solutions due to the use of the Modelica language. This paper provides an introduction and overview of the library and describes an application in the design of a turbo fan engine.

Keywords: *Turbo fan, turbo jet, turbo prop, turbo shaft, performance, model-based design, sizing, secondary power*

1 Introduction

The prime mover of an aircraft, the jet engine, is one of the most important subsystem of an aircraft. Jet engines provide primary power (thrust) and secondary power (to drive flight control, air conditioning, cabin lighting and so on) to the aircraft. Recently, the improvements in the performance and efficiency of jet engines deliver a very large share in the overall platform improvements (for both commercial and military aircraft where for instance super cruise requirements were met). They therefore strongly affect aircraft value and, in case of commercial aircraft, eventually airline competitive edge. The latter is not only driven by costs, but even more so by environmental regulations.

However, these power plant improvements become increasingly difficult to achieve when focusing on the engine in isolation. The reason is that the local improvement potential has largely been leveraged in previous incremental design improvements, and only changes on global aircraft level remain to substantially improve the total aircraft package. For this reason aeronautical systems such as aircraft and their subsystems are becoming more and more integrated. This integration takes place along a number of trends. We mention two of these. The first one is the electrification of secondary power on-board aircraft (Provost, 2002). This trend is also called the “More Electric Aircraft” and has shaped industry road maps since more than two decades. Historically, three different types of secondary power were equal, namely, electric power,

hydraulic power, and pneumatic power. With the “More Electric Aircraft” this is changing in favor of electric power. The main reason lies in the anticipated development potential of power electronics, which is all but exhausted (like that of pneumatic and hydraulic power).

The second trend is more recent and is the electrification of primary power. This is getting increasing interest due to intrinsic limitations in turbofan technology (Kyprianidis et al., 2014) (be it geared or ungeared). Following (Winter, 2013), the overall efficiency of a propulsion system can be considered to be proportional to the product of thermal and propulsive efficiency. To achieve thermal efficiency improvements, the Overall Pressure Ratio (OPR) and the Turbine Entry Temperatures (TET) of the cycles are being increased in an incremental way since the last few decades and are approaching peak values (approximately 1900-2000K TET and around 45-50 cycle OPR). Material limits, turbine cooling, emissions, and losses in the last stage of the high pressure compressor may now impose fundamental limits to the thermal efficiency. Improvements in propulsive efficiency are well achievable via reduction in fan pressure ratio and increases in bypass ratio. However, these improvements are deteriorated by losses through lowered transmission efficiency, increased nacelle weight and higher drag due to larger frontal area (Larsson et al., 2011). When these limits indeed turn out to become fundamental ones, different and more integrated concepts will become of interest. For instance electric ones where power is stored in one way or another on-board and possibly converted to electric power by gas turbines or fuel cells and used to drive distributed propulsion devices. Such aircraft with partially or fully electrified primary power systems are called hybrid or fully electric aircraft.

It is critical that the methods and tools supporting the design of such systems keep pace with the increasing integration on the product side. Only with efficient and robust prediction capabilities it becomes possible to establish model-based design for such solutions introducing new technologies, and cover all relevant “what if” scenarios.

It is therefore evident that if future propulsion systems and technology are to achieve the environmental challenges and performance targets set, rigorous mathematical

analysis of the component physics as well as integrated subsystem physics is important. There exists a critical requirement to develop and adapt models at an appropriate level of fidelity to specific components. One of the key requirement is also to have a generic framework where the user will be able to choose components and characteristics of his choice before integration of the subsystem.

Given the importance of propulsion system simulation as an academic and industrial engineering discipline, the literature on the state of the art is too extensive to be reviewed here. We therefore focus on selected references and tools. First, Gasturb (Kurzke, 1995) is a user-friendly and powerful domain-specific system simulation software for gas turbines. It is mature and provides extensive functionality, but also restricted to the scope defined by the authors. The model equations as implemented in the software can hardly be accessed and adapted by the user. Integration with other simulation and design models is possible but mostly requires process integration and design optimization (PIDO) solutions¹. EnVironmental Assessment (EVA) (Kyprianidis et al., 2008) is an example of a domain-specific simulation software with widened scope (engine system simulation and some aspects of aircraft sizing). Similar to Gasturb it is restricted to the application scope envisioned by its original programmers however. Numerical Propulsion System Simulation (NPSS) (Nichols and Chamis, 1991; Lytle, 1999; Jones, 2007, 2010) covers more than system simulation, as it also works as integration hub between system and field simulation. It can also be labeled as a domain-specific software but it relies on an object-oriented (yet causal) custom language in which component models are written. Model equations as implemented can be accessed and adapted by the user. Integration can also be established via PIDO solutions, but alternatively non-propulsion sub-systems can be modeled in the native NPSS language and be integrated in a computationally more efficient and robust manner than via PIDO solutions. PROpulsion Object Oriented Simulation Software (PROOSIS) (Alexiou and Mathioudakis, 2005; Bala et al., 2007) is a similar system simulation software. It provides the same benefits in terms of access and customization (albeit using an acausal language), and also allows integration using non-PIDO approaches. A number of modeling libraries for non-propulsion sub-systems have been mentioned informally but not documented in the literature (according to the knowledge of the authors). In any case, a main limitation of this platform is the use of an in-house modeling language, which is not widely adopted or openly standardized via a non-profit organization.

While connecting a wide array of tools for multi-disciplinary design optimization of aircraft and sub-

systems is feasible, integrating in a less fragile way based on open standards such as Modelica and FMI would increase flexibility and allowed to substantially increase manageable problem size due to higher computational efficiency. Other proposed interfacing standards such as ARP 4868 (SAE, 2001) are domain specific and work well for model-based efforts in their respective disciplines but not to couple analyses for unconventional designs. However, up to now nobody has proposed a plausible modeling library in Modelica for jet engines. Such a library could eventually be integrated one into a framework for modeling and simulation of aircraft and their components. This enabled time and resource efficient implementation of model-based design processes via reuse of such model assets; a key enabler for model-based design. Additionally, this improved consistency of results.

The objectives of this paper are

1. To suggest a library for modeling and simulation of aircraft jet engines and their sub-systems in Modelica for a broad range of applications ranging from engine and sub-system conceptual design to detailed analysis and design involving transient and real-time simulation².
2. To substantiate why the library can plausibly be applied to industrial-scale problems involving “complex” models and “sophisticated” analyses
3. To apply the framework to an engineering problem, namely, the computation of the full range of cycle performance.
4. To provide an outlook on how a Modelica implementation for modeling and simulation of aircraft jet propulsion provides additional value over existing discipline-specific tools in the design and analysis of unconventional systems.

2 Jet Propulsion Library: Overview and implementation

The Jet Propulsion Library is a modeling framework for gas turbines and jet propulsion of commercial and military aircraft. The comprehensive set of components enables cycle performance analysis and optimization of all types of aerospace gas turbines. On-design and off-design performance can be studied as well as steady-state and transient behavior based on a single model.

The physics of jet engines are governed by the balance equations of thermo-fluid dynamics, the conservation equations of mass, energy and momentum. Thus, some

¹Process Integration and Design Optimization software typically contains numerous CAD/CAE integration adapters that allow the user to link different computation software in a GUI. They often also provide convergence, optimization, and surrogate modeling functionality, which allows to automate analysis and design processes.

²We restrict the scope however to only cover system simulation, i.e., all processes governed by ordinary differential equations (ODE) or differential-algebraic equations (DAE). Processes governed by partial differential equations are beyond the scope of the library, unless their partial derivatives have been suitably discretized to match the formal framework of an ODE or DAE (e.g., one-dimensional discretization of the balance equations of thermo-fluid dynamics).

of the underlying principles have been documented elsewhere (Elmqvist et al., 2003; Casella et al., 2006; Franke et al., 2009a,b). These will not be repeated here. However, given the flow velocities in such engines some assumptions commonly made in the mentioned articles are not appropriate; for instance the assumption that the flow velocity is small and that differences between static and total quantities can be neglected. The following presentation describes some of the differences to the established approaches, and gives a small example of the sophistication in its implementation to address the previously mentioned challenges.

2.1 Why Modelica

At a first glance, domain specific simulation solutions including sophisticated graphical user interfaces are very appealing. We believe however that the use of the generic modeling language Modelica provides advantages that may outweigh the benefits of the former.

First, this is due to the tool support to manage product and model complexity. This relies on the object-oriented nature of Modelica, and allows the tool to conveniently filter what implementations fit in a placeholder on a given model template. Manually choosing from a large library can be surprisingly difficult as industrial size problems are tackled. With Modelica, models can be built rapidly based on pre-configured templates. Additionally, a model architecture can be used once implemented across the system engineering V-cycle even as the user zooms into detailed modeling involving dynamic and real-time analyses. This facilitates creating and maintaining a holistic view even on challenging systems.

Second, given the declarative and symbolic problem description encoded in the Modelica language, a model compiler can transform the model description (equation system) into the form most suitable for a given analysis. This is based on automatic symbolic transformations, and allows executing the same model as dynamic simulation, steady-state simulation, optimization, real-time simulation and so on.

Additionally (and this has already been indicated above), using Modelica it is more straight forward to cover all domains based on first principles. After all, Modelica is one of the native languages of the aircraft sub-system industry. A large community/eco-system exists based on the Modelica language with many commercial and open source model libraries.

Furthermore, interactions become more productive. Based on the open standards, any given model can be made available on multiple tools. This enables model-based collaborations, independently whether based on Modelica or FMI.

Finally, this approach provides full access to the models. After all, while complete documentation of black box component models is great for many cases, reading the actual model code including the exact equations used for simulation in the engineering language Modelica enables

deeper understanding and customization.

2.2 Thermodynamic properties

In the following sections, the distinction between so-called static and total quantities is very important. A static pressure p_s for instance is the actual pressure in the usual sense, which is associated not with fluid motion but with its state. Total and dynamic pressure in turn are closely related to fluid flow, and are a measure of flow velocity. For incompressible fluids, Bernoulli's equation states $p_t = p_s + p_d = p_s + 1/2\rho v^2$. Here, $p_d = 1/2\rho v^2$ is called the dynamic pressure, and p_t total pressure. Total quantities are sometimes also called stagnation quantities as they correspond to the value of the static or thermodynamic quantities if the fluid flow was brought to rest (zero velocity) in a reversible way (isentropically). As we are dealing with compressible fluids in the context of this paper, Bernoulli's equation does not hold. Instead, a compressible formulation has to be used. The details are described in the following sections.

The thermodynamic state is always defined by the static properties such as static temperature and pressure. These are the actual temperatures and pressure observed in the real world. In gas turbine performance computations it is however a tremendous simplification to express the component level equations mostly in total or stagnation quantities (Walsh and Fletcher, 2004). Like this, the exact flow cross section areas and velocities are not necessarily required. There are however also component models, in which the static quantities have to be computed such as mixers and nozzles (Walsh and Fletcher, 2004). In many cases the static quantities are also of interest and are therefore computed in the "output section" (using Modelica parlance).

In any case, the scope of the thermodynamic property computations in the Jet Propulsion Library therefore has to cover both static and total quantities. Additionally, to ensure accurate predictions, this has to be done in what is called the "fully rigorous" way (Kurzke, 2007). From text books, one is tempted use the following equation to relate the total temperature T_t and pressure p_t

$$T_t = T_s \frac{p_t^{\frac{\gamma-1}{\gamma}}}{p_s^{\frac{\gamma-1}{\gamma}}} \quad (1)$$

Or, likewise

$$\frac{p_t}{p_s} = \left(1 + \frac{\gamma-1}{2} M^2\right)^{\frac{\gamma}{\gamma-1}} \quad (2)$$

However, the isentropic exponent γ is not constant across larger temperature or pressure ranges. Therefore, equations (1) and (2) are strictly speaking not applicable. Following (Kurzke, 2007; Sethi, 2008), a fully rigorous approach based on the so-called entropy function Φ can be used instead.

$$\Phi(T) = \int_{T_{ref}}^T \frac{c_p}{R} \frac{dT}{T} \quad (3)$$

Then, the change of the entropy function in an isentropic process is equal to the logarithm of the pressure ratio,

$$\Phi_2 - \Phi_1 = \ln \left(\frac{p_2}{p_1} \right) \quad (4)$$

Based on this approach, we can compute the complete set of the following six static quantities from any two of them plus the complete set of total quantities,

- Mass flow rate w
- Cross section area A_e
- Static pressure p_s
- Static temperature T_s
- Mach Number M
- Flow velocity v

Then, instead of using (2), we can compute the static pressure (and the complete set of static quantities) from the Mach Number as follows (Sethi, 2008) (note that this procedure requires the solution of implicit equation systems and additionally the mass flow rate as input). First, we compute the static temperature T_s from the following implicit equation.

$$M = \frac{\sqrt{2h_t - h_s(T_s)}}{\sqrt{\gamma_s(T_s)RT_s}} \quad (5)$$

Then, the static pressure can be computed explicitly in a fully rigorous way via the following equation

$$p_s = \frac{p_t}{\exp \left(\frac{\Phi_t - \Phi(T_s)}{R} \right)} \quad (6)$$

As written above, the complete set of six static quantities can be computed from any two of them (and the total quantities). Based on which set of two static quantities is given, between zero and two numerical solutions to implicit equations such as (5) are required to compute the full set of static quantities. Therefore, the thermodynamic properties involving rigorous computations of total and static quantities are somewhat different to the state of the art in Modelica (see references above), where the need for solution of implicit equation systems is considered a rare case, which can often be avoided by suitable model reformulations.

To provide convenient access to the computation of total and static thermodynamic properties we have therefore decided to use a package structure similar to Modelica.Media (Elmqvist et al., 2003) but tailored to the application specifics. First, we apply the concept of the thermodynamic state record to both total quantities (which, following the introduction to this section, are required in all component models) and static quantities.

A typical function to compute a total thermodynamic state record has the following interface.

```
replaceable partial function setTotal_pthtX
  "Return total state as function of pt, ht
  and composition X"
input AbsolutePressure pt
  "Total pressure";
input SpecificEnthalpy ht
  "Total specific enthalpy";
input MassFraction X[nS]
  "Mass fractions";
output TotalState total
  "Total state record";
end setTotal_pthtX;
```

Based on a given total thermodynamic state record any total quantity can be computed, for instance total temperature

```
TtIn = Medium.totalTemperature(
  inlet_total);
```

Additionally, a static thermodynamic state record can be computed from a given total state record and any two quantities related to the static quantities (w , A_e , p_s , T_s , M , v as defined above). The rigorous procedure described with (5) and (6) is for instance implement in such a function conforming to the following interface

```
replaceable partial function setStatic_Mnw
  "Return static state as function of total
  state, Mach Number Mn and mass flow
  w"
input TotalState total
  "Total state record";
input Real Mn
  "Mach Number";
input MassFlowRate w
  "Mass flow rate";
input Types.FlowRegime regime
  = Types.FlowRegime.Subsonic
  "Flow velocity regime";
output StaticState static
  "Static state record";
end setStatic_Mnw;
```

Enumeration FlowRegime is optionally used to constrain the solution interval to sub-sonic, sonic, or super-sonic results.

Based on this code structuring concept fully rigorous thermodynamic properties are implemented in the Jet Propulsion Library. See figure 1 for an overview. The underlying model for the entropy function and other related quantities can be exchanged to allow different representations and fidelity levels. The first one implemented in Jet Propulsion Library utilizes the polynomial approach of (Walsh and Fletcher, 2004) and does not capture dissociation effects.

2.3 Connector definition

For the fluid connectors in the Jet Propulsion Library we adapt the concept of stream connectors as proposed in (Franke et al., 2009a,b). As defined there, the static pressure and the static specific enthalpy are used as key connector variables. Given the introduction to section 2.2 we instead opt for using the corresponding total quantities

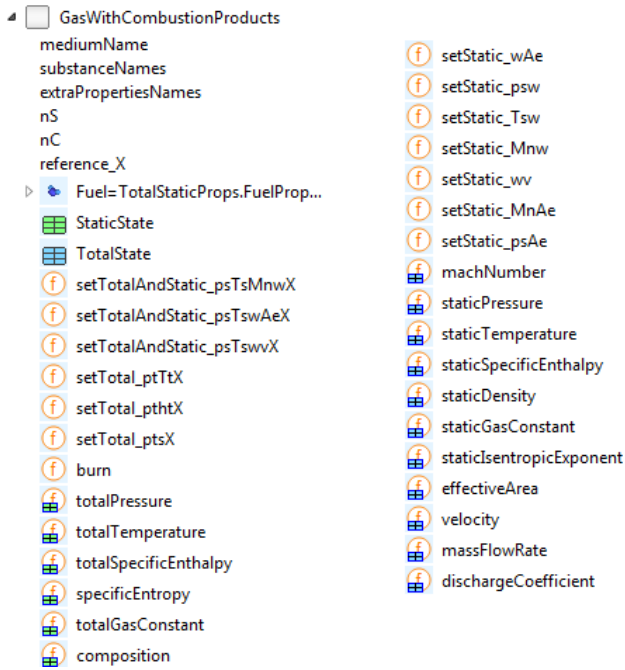


Figure 1. Thermodynamic property functions: Static and total state records plus functions acting on total state records to the left and function acting on static state records to the right

on the connectors. Otherwise, the connector is identical to the well-established and widely adopted fluid connectors. The fluid connector carries flow and thermodynamic state information such as pressure, mass flow rate, specific enthalpy, and composition.

```
connector FluidPort "Fluid connector"
  replaceable package Medium =
    GasWithCombustionProducts
  annotation (choicesAllMatching=true);

  AbsolutePressure p
    "Total pressure";
  flow MassFlowRate m_flow
    "Mass flow rate into the
    component";
  stream SpecificEnthalpy h_outflow
    "Total specific enthalpy of exiting
    fluid";
  stream MassFraction X_outflow[Medium.nS]
    "Mass fractions of exiting fluid";
  stream ExtraProperty C_outflow[Medium.nC]
    "Properties c_i/m in the connection
    point";
end FluidPort;
```

Note how the Modelica naming convention is used for the variables on the fluid connector.

Unfortunately the connector is still not directly compatible with libraries using the standard fluid connector (e.g., from Modelica.Fluid) due to the use of a different package structure for the computation of the thermodynamic properties. Therefore, a simple adapter component was required if connections were to be made to the high speed gas flow path models; for all other interfaces standard con-

nectors are used (fuel flow supply, shaft interfaces etc.).

2.4 Simulation modes: On-design, off-design, transient

Since the beginnings, jet engine performance computations have always considered two main computation problems, design point performance computation (also called on-design performance computations) on one hand and off-design performance computations on the other (Walsh and Fletcher, 2004).

For the design point performance computation, one set of operating conditions has to be imposed. Then, the component performance levels and sizes are selected. Additionally, top level requirements are implemented (e.g., based on cruise at altitude on an ISA day). The design point performance computation then allows to compute important cycle parameters, and to define a specific design. This includes a possibly abstract or estimated engine geometry, based on the fidelity of the analysis. Technically, the output of such a design point performance computation are however scaling parameters on component level.

Given a specific engine design (figuratively in terms of an estimated or abstract geometry, or, more technically, in terms of a complete set of component scaling parameters for a given engine topology), the off-design performance computation then allows to estimate the performance at other key operating conditions (different altitude, Mach Number, day type and so on). Here geometry is fixed and operating conditions are changing. While the literature typically describes off-design performance computation as steady-state analysis, this may as well involve transient simulation.

In order to provide complete functionality in relation to the established methods, these two kinds of computations were also implemented in the Jet Propulsion Library. They can be selected as “simulation modes”.

A closer look at the literature (e.g., (Walsh and Fletcher, 2004)) reveals that the notion of on-design computations is not directly compatible with the rules of balanced modeling in Modelica (Olsson et al., 2008). Typically, the bypass ratio or flow split is imposed on a three-way junction or splitter model. Following the rules of balanced modeling, such a component may however only impose both downstream pressures or impose one downstream pressure and the bypass ratio or a flow rate. In on-design computations it is however required for the scaling procedure to impose both downstream pressures *and* the bypass ratio. Off-design computations in turn are basically the computations classically done in the Modelica language. Therefore, the corresponding simulation problems (be it in steady-state or transient mode) are fully compatible with the concept of balanced modeling.

As the constraints of balanced modeling are imposed for very good reasons (for instance, to improve debugging messages and ensure “plug and play” compatibility when selecting specific implementations during system ar-

chitecting for a given placeholder) it was not an option on the design of the Jet Propulsion Library to rely on locally unbalanced models. Instead, it was decided to restrict the use of on-design computations to initial time and initial equations. Like this, initial equations are used to compute corresponding values of the component parameters that have a fixed attribute equal to false. Based on this decision both on-design computations and off-design computations are in scope of the Library, and its component and system models always remain balanced.

2.5 Component models

With the exception in the connector definition described in section 2.3, the Jet Propulsion Library fully follows the variable naming convention suggested in ARP5571 (SAE, 2005).

2.5.1 Boundary conditions

The types of boundary condition models in the Jet Propulsion Library are similar to those in other thermo-fluid dynamics libraries. Most fundamentally, we distinguish boundary conditions imposing a given mass flow rate, and boundary conditions imposing pressure (obviously all boundary conditions also impose quantities transported by convection). These two kinds of boundary conditions are also required for modeling and simulation of jet engines. However, the prescribed variables may now change from on-design to off-design computations. To provide full flexibility to the user, four different flags are exposed on a boundary condition. These allow to switch on and off the prescription of pressure and mass flow rate for on-design and off-design models respectively. To improve ease-of-use, these four flags are only exposed to the user in the category of advanced component parameters; normally (in simple boundary condition parameterization mode), the user only decides whether the boundary condition is nominally a source or a sink.

- A nominal source prescribes both pressure and flow rate for on-design computations, and pressure for off-design computations, and
- A nominal sink prescribes neither pressure nor flow rate in on-design computation, and pressure in off-design computation.

Figure 2 shows a simple model diagram with such boundary condition instances. Color-coding is used to illustrate whether a component includes over-constrained initial equations (nominal source with green outline) or under-constrained initial equations (nominal sink with blue outline). As long as the number of blue components is equal to the number of green components the system model will be well-posed (actually, any system is well-posed by construction, the color-coding still helps users to double-check their model build-up). The color-coding is also used on other components such as the splitter mentioned in section 2.4 already. Quantities imposed for on-

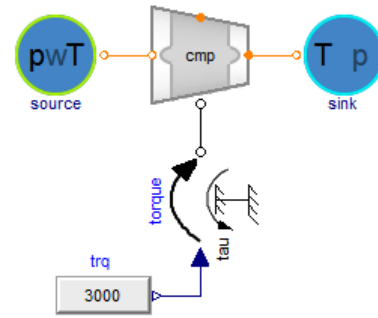


Figure 2. Single component experiment with two boundary conditions (the vectorized bleed port at the top is unconnected and has length zero)

design and off-design have the corresponding letter written in opaque font on the boundary condition, quantities that are either imposed for on-design or imposed for off-design have their corresponding letter written in slightly transparent font.

2.5.2 Compressor

The compressor model is one of the component models that contains the scaling factors mentioned in section 2.4. For the compressor on-design performance computation, the user typically prescribes isentropic efficiency η_{des} and pressure ratio π_{des} at the design point. The corrected mass flow rate $w_{c,des}$ is not imposed directly as a parameter on the compressor model but on the system model as a whole, and then computed from boundary conditions or inlet as well as design bypass ratio BPR_{des} (the same holds for the corrected speed N_c).

The overall compressor performance in terms of isentropic efficiency η (or specific work) and pressure ratio π is encoded in performance maps (Walsh and Fletcher, 2004). Based on a particular point in the performance map that is marked as the design point, four scaling parameters are then computed as described by (Jones, 2007).

- Is. efficiency scaling factor $s_{\eta,des} = \frac{\eta_{des}}{\eta_{des,unscaled}}$
- Pressure ratio scaling factor $s_{\pi,des} = \frac{\pi_{des}-1}{\pi_{unscaled}-1}$
- Corrected flow scaling factor $s_{w_c,des} = \frac{w_{c,des}}{w_{c,unscaled}}$
- Corrected speed scaling factor $s_{N_c,des} = \frac{N_{c,des}}{N_{c,unscaled}}$

Here, quantities with index *unscaled* indicate the value in the original, unscaled performance map. Based on this procedure, one compressor map with its design point can be scaled to represent another compressor (as described by the target design point as prescribed by the user). As long as the design points are close enough, the scaling gives a reasonable approximation of the compressor behavior.

As indicated above, the compressor model requires that the off-design performance is captured in the format of

a performance map. The format of this performance map has to be easy to handle in an computing environment (Walsh and Fletcher, 2004), and avoid vertical or horizontal lines in the table look-up (Jones, 2007). For this reason we use beta or R-line maps³. The method described by (Jones, 2007) in more detail. Basically, the performance maps relates the important thermodynamic variables like corrected mass flow rate, pressure ratio, corrected speed and the efficiency of the compressor. A compressor performance map using R-lines is shown in figure 3. R-lines are family of curves that are parallel to the surge line and evenly spaced among each other. The R-lines ensure unique result in the regions of low corrected air flow where pressure ratio is almost a constant and regions of constant air flow towards the highest air flow region for a given speed line (avoiding table look-up along vertical or horizontal tangents).

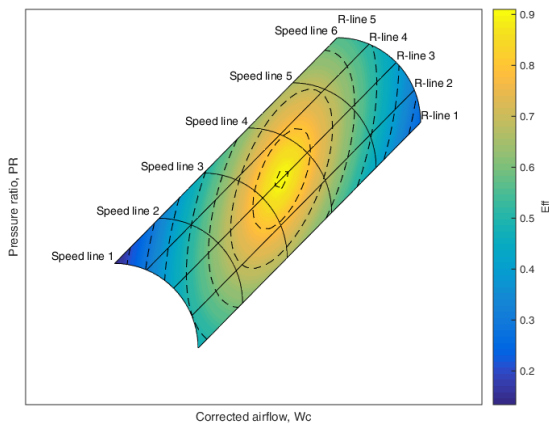


Figure 3. R-line based compressor map with speed lines (solid), r-lines (solid), and efficiency contours (dashed)

Other methods to capture the compressor performance characteristics are described in the literature. One example is the Map Fitting Tool (MFT) method (Sethi et al., 2013). While Jet Propulsion Library currently only implements the R-line or beta line methodology, the object-oriented structure allows for the convenient addition of such additional map format in the future.

Once the key component performance variables were read from the performance map, the component computations continue as known from other thermo-fluid dynamics libraries.

The compressor model also has a mechanical connector through which it can receive shaft power (for instance from the respective turbine models).

The compressor model (like all components in the Jet Propulsion Library) support the modeling of secondary air systems. For instance, bleed can be extracted from this compressor model, routed through an arbitrary network,

³The notion of using an auxiliary coordinate has at least two different names; beta lines (Walsh and Fletcher, 2004), and R-lines (Jones, 2007).

and be supplied for turbine film cooling or for so-called customer purposes. A bleed mass flow rate through the bleed ports can be specified via constant or variable bleed mass flow fractions in the model. In order to capture the stage at which the bleed air is extracted, parameter corresponding to the relative bleed enthalpy and the pressure as a fraction of inlet and outlet conditions are used.

2.5.3 Turbine

The turbines models are built very similar to the compressor models based on the off-design turbine performance map and a set of scaling factors. For the on-design performance computation, the user typically prescribes isentropic efficiency η_{des} and (uncorrected) shaft speed N at the design point. The pressure ratio π_{des} , the corrected mass flow rate $w_{c,des}$ are again computed from boundary conditions and the system model (the pressure ratios at the design point are for instance solved for such that the power balances per shaft are fulfilled).

The turbine performance map used is as shown in figure 4. Given pressure ratio and speed, corrected flow and isentropic efficiency can be uniquely determined in a turbine map. This eliminates the need for R-lines as discussed in the compressor section. The format is again based on (Walsh and Fletcher, 2004; Jones, 2007). The four scaling parameters then computed from the performance map design point and the jet engine design point are similar to the ones used for the compressor and described in section 2.5.2.

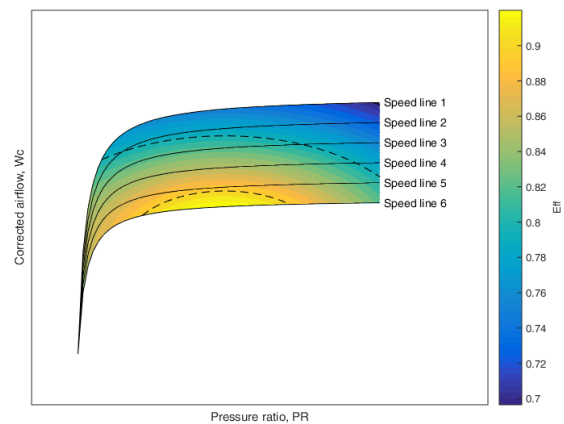


Figure 4. Turbine map with speed lines (solid) and efficiency contours (dashed)

Again, once the key performance variables were read from the map, the component computations basically continue as known from other thermo-fluid dynamics libraries. This means for instance that the turbine model also has a mechanical port through which expansion power is supplied to the compressor through shafts. As indicated before, the turbine model optionally provides bleed ports which can receive secondary cooling air from compressor stages or other sources. The resulting power

from the bleed air flows can be accounted for considering the tip velocity of the turbine blades.

2.5.4 Inlet

A critical part of the inlet models are parametric predictions of the ram pressure recovery η_{ram} and the spillage, bleed, and bypass drag (expressed via the corresponding drag coefficient $C_{d,install}$). These effects can be modeled using the correlations suggested by Kowalski (Kowalski and Atkins, 1979). These are available in two different flavors; a long (and more accurate) form of computation involving 14 tables, and a short form involving 2 compressed tables. The dimensional drag force due to capturing air D_{ram} is eventually computed from

$$D_{ram} = w \cdot v \quad (7)$$

where the ram pressure recovery η_{ram} indirectly influences mass flow rate. The drag force due to installation $D_{install}$ is

$$D_{install} = 1/2 \rho v^2 C_{d,install} \quad (8)$$

The given references contains more details about the implementation.

2.5.5 Nozzle

The ideal gross thrust $F_{g,ideal}$ of a nozzle can readily be computed from the following equation

$$F_{g,ideal} = w \cdot v + (p_{s,exit} - p_{s,amb}) A_{e_{exit}} \quad (9)$$

Here, $p_{s,exit}$ and $p_{s,amb}$ are the static pressures at the nozzle exit section and the ambient respectively. Again, the crucial question for sound model-based design application is how much of the ideal results are achievable. This can be expressed via a number of correlations. One of the quantities to use for this purpose is the nozzle exit gross thrust coefficient C_{F_g} . This coefficient is also used by Kowalski (Kowalski and Atkins, 1979). Beyond C_{F_g} correlations to approximate gross thrust F_g , this methodology also estimates the aftbody drag coefficient $C_{d,ab}$. The former for instance is computed based on pressure ratio and area ratio. Two variations for an axisymmetric nozzle as well as 2-D nozzle exists. Eventually, the actual gross thrust can be computed

$$F_g = C_{F_g} F_{g,ideal} \quad (10)$$

The nozzle model in the library contains replaceable models to compute the contributions individually. This completes the short overview of exemplary component models.

2.6 Interface and template structure

Based on these component models, different cycles can be built up using an interface and template model structure. Different kinds of cycles such turbo fan, turbo jet, geared turbo fan, turbo prop or turbo shaft have been disassembled virtually into reusable sub-system and sub-assembly models. Based on the object-oriented interface and template structure they can be plugged together in a highly

flexible and efficient manner. An unmixed turbo fan for instance consists of the inlet section, fan and compressors, the combustor, the turbines, and the primary and secondary nozzles. An exemplary break-down for such a two spool unmixed turbo fan thus is

- Inlet section
 - Inlet
 - Inlet frame duct
 - Inlet engine duct
- Fan and compressor
 - Fan
 - Splitter
 - Low pressure compressor
 - High pressure compressor
 - Fan duct
- Combustor
 - Diffuser duct
 - Burner
- Turbine
 - High pressure turbine
 - Low pressure turbine
- Primary and secondary nozzle sections
 - Exhaust frame duct and exhaust tailpipe duct, or bypass exhaust frame duct
 - Nozzle

Different to the state of the art described in section 1, this approach uses hierarchy and object-orientation to manage variants and system complexity. Previous art lays all element out on a flat level. With this approach, we can conveniently exchange inlet section models from regular inlets to inlets with inlet particle separator, based on available map data one can conveniently switch between average and split fan models (averaging the core and bypass fan flow, or modeling them via separate fan models using different maps), number of spools, as well as detailed section models for compressor and turbine sections (stage representation, inclusion or removal of case strut ducts, inlet guide vane ducts, transition ducts, exit guide vane ducts and so on).

An example breakdown of a turbo fan engine is illustrated graphically in figure 5. This figure shows the actual view presented to the user in the graphical user interface of a Modelica Integrated Development Environment (IDE).

Each type of component in the break-down above is represented through a class hierarchy of interfaces, templates, and implementations. The implementations use the atomic components described in section 2.5.

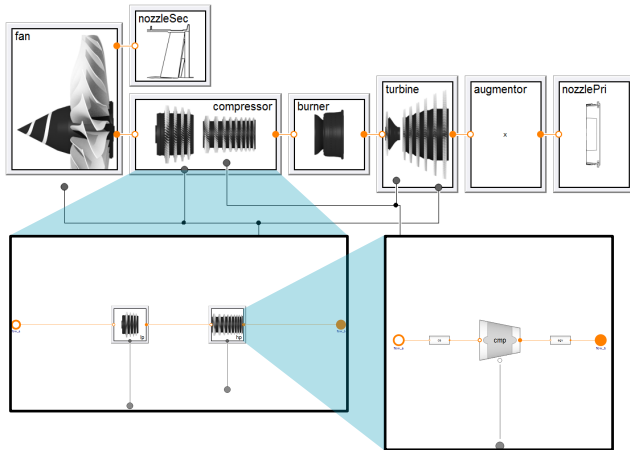


Figure 5. Top-level turbofan model breakdown shown on the top, compressor break-down on the lower left, high pressure compression section break-down on the lower right

3 Application example and results

A complete jet engine model was built using the Jet Propulsion Library of the Pratt & Whitney JT9D. It was created by configuring the two spool unmixed turbo fan template model. Each component starting from inlets, fans, compressors, turbines etc. is redeclared with parameterized models. The respective performance maps are adapted from the open source distribution of the Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS) as described by (Chapman et al., 2014).

Table 1 provides key cycle parameters at the design point. These parameters are approximate but consistent with the given source.

Table 1. Cycle design point parameters.

Parameter	Value
Design point	Sea level static
Day conditions	ISA+15 °C
Inlet flow	698 kg s ⁻¹
Bypass ratio	5.2751
Turbine inlet temperature	1260 °C
Net thrust	223 kN

Figures 6, 7, and 8 show the compressor performance maps. Following the principles described in section 2.5.2, these maps are scaled based on the component design point data given in tables 2, 3, and 4.

Table 2. Fan design point parameters.

Parameter	Value
Efficiency	90.38 %
Pressure ratio	1.60306

Table 3. Low pressure compressor design point parameters.

Parameter	Value
Efficiency	86.575 %
Pressure ratio	2.25

Table 4. High pressure compressor design point parameters.

Parameter	Value
Efficiency	86.2469 %
Pressure ratio	5.67905

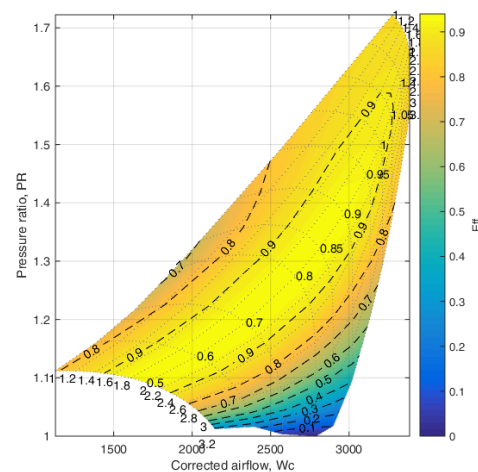


Figure 6. JT9D fan compressor map

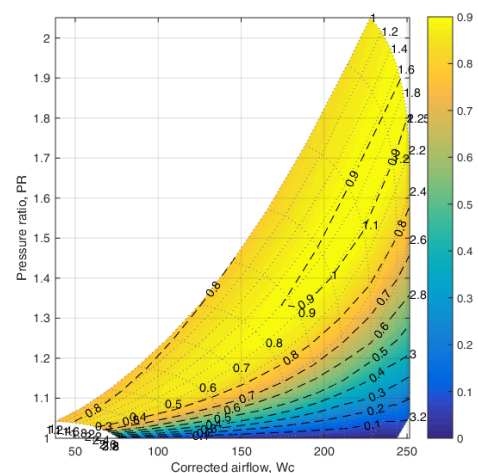


Figure 7. JT9D low pressure compressor map

Figures 9 and 10 in turn show the turbine performance maps. These maps are scaled based on the component design point data given in tables 5 and 6.

Table 5. High pressure turbine design point parameters.

Parameter	Value
Efficiency	91.445 %
Shaft speed	8000 / min

Table 6. Low pressure turbine design point parameters.

Parameter	Value
Efficiency	92.88 %
Shaft speed	3750 / min

In the following, exemplary simulation results are provided. For this purpose, two boundary conditions are imposed on this model, the aircraft Mach number and the fuel flow rate. The design point simulation runs for a sea-level static case. Basic sanity check results show a reasonably good match of the sea level static thrust and specific fuel consumption produced by the engine model with published data (Saarlas, 2007). Then, the boundary condition parameters are varied for off-design simulation. The model was simulated to conduct a full factorial experiment for inputs of inlet Mach numbers (0.5, 0.7 and 0.9) and fuel flow that varied $\pm 50\%$ from the nominal value in steps of 5%. The results of this full factorial experiment is summarized in the two figures below.

Figure 11 plots the relationship between the low pressure spool speed N_L and thrust for different Mach numbers. The thrust is divided by $\delta = p_t/p_{t,ref}$, the normalized inlet total pressure. The low pressure spool speed is divided by the square root of $\theta = T_t/T_{t,ref}$, the normalized inlet total temperature. These corrections are routinely done to normalize the data (Walsh and Fletcher, 2004). Higher Mach Numbers show lower corrected thrust due to the inlet ram drag.

The corrected thrust specific fuel consumption trends are shown in figure 12. Both plots are qualitatively very similar to the charts given in the relevant literature such as (Walsh and Fletcher, 2004; Saarlas, 2007). Illustrative results of the transient simulation mode will be given in a separate reference.

4 Conclusions

The Jet Propulsion Library provides a foundation for modeling and simulation of jet engines, and the model-based design of integrated aircraft system designs. It contains fully models for sizing and performance computations, and has a number of advantages over existing domain-specific solutions due to the use of the Modelica language.

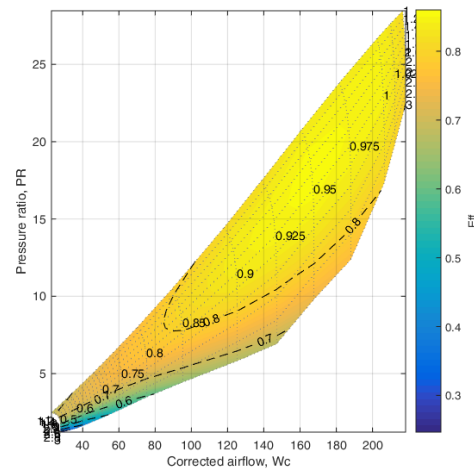


Figure 8. JT9D high pressure compressor map

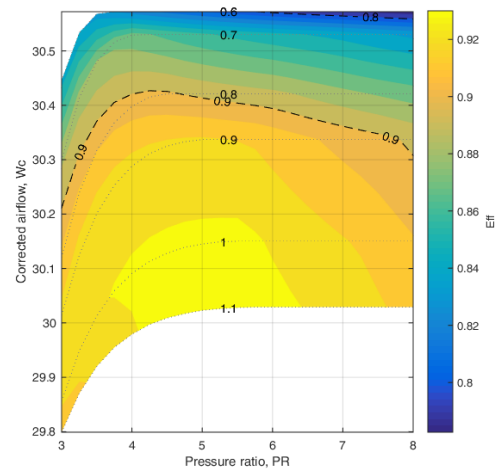


Figure 9. JT9D high pressure turbine map

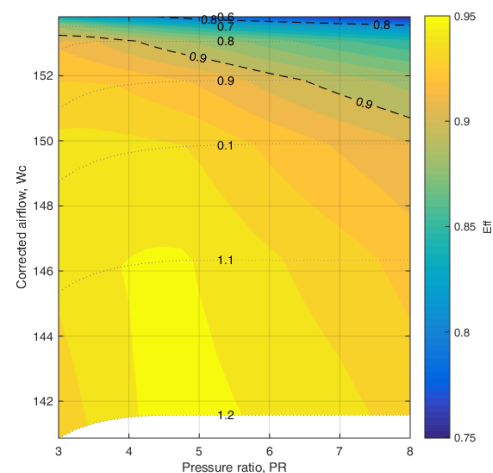


Figure 10. JT9D low pressure turbine map

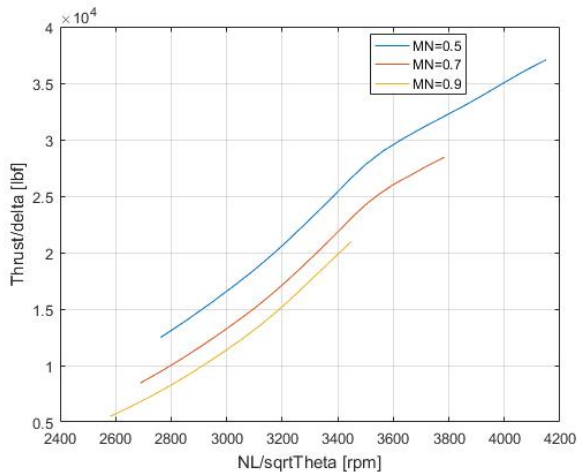


Figure 11. JT9D corrected thrust vs. corrected low pressure spool speed

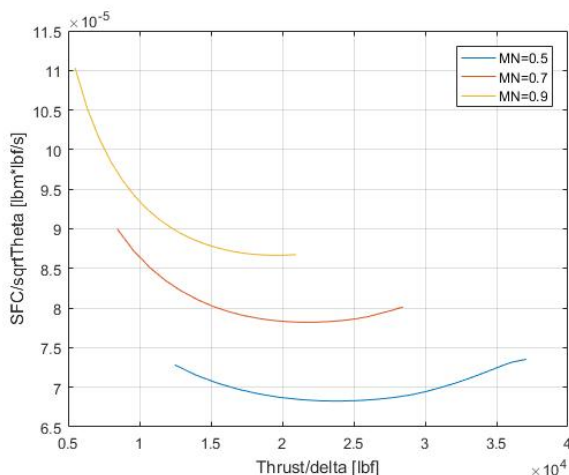


Figure 12. JT9D corrected thrust specific fuel consumption vs. corrected thrust

5 Acknowledgements

We acknowledge the contributions of Shashank Swaminathan who contributed to the interface and template structure described in section 2.6 as summer intern at Modelon, Inc in 2016.

References

- A Alexiou and K Mathioudakis. Development of gas turbine performance models using a generic simulation tool. In *ASME Turbo Expo 2005: Power for Land, Sea, and Air*, pages 185–194. American Society of Mechanical Engineers, 2005.
- Arjun Bala, Vishal Sethi, E Lo Gatto, Vassilios Pachidis, and Pericles Pilidis. ProosisUa collaborative venture for gas turbine performance simulation using an object oriented programming schema. In *International Symposium on Air Breathing Engines*, 2007.
- F. Casella, M. Otter, K. Proelss, C. Richter, and H. Tummescheit. The modelica fluid and media library for modeling of incompressible and compressible thermo-fluid pipe networks. In *Proceedings of the Fifth International Modelica Conference*, pages 631–640, 2006.
- Jeffrey W Chapman, Thomas M Lavelle, Ryan May, Jonathan S Litt, and Ten-Huei Guo. Propulsion system simulation using the toolbox for the modeling and analysis of thermodynamic systems (t mats). In *50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, 2014.
- Hilding Elmqvist, Hubertus Tummescheit, and Martin Otter. Object-oriented modeling of thermo-fluid systems. In Peter Fritzson, editor, *Proceedings of the Third International Modelica Conference*, pages 269–286, Linköping, Sweden, 2003.
- Rüdiger Franke, Francesco Casella, Martin Otter, Michael Sielemann, Sven-Erik Mattson, Hans Olsson, and Hilding Elmqvist. Stream connectors—an extension of Modelica for device-oriented modeling of convective transport phenomena. In Francesco Casella, editor, *Proceedings of the seventh International Modelica conference*, pages 108–121, Como, September 2009a.
- Rüdiger Franke, Francesco Casella, Michael Sielemann, Katrin Proelss, Martin Otter, and Michael Wetter. Standardization of thermo-fluid modeling in Modelica.Fluid. In Francesco Casella, editor, *Proceedings of the seventh International Modelica conference*, pages 122–131, Como, September 2009b.
- Scott M Jones. An introduction to thermodynamic performance analysis of aircraft gas turbine engine cycles using the numerical propulsion system simulation code. Technical Report TM—2007-214690, NASA, March 2007.
- Scott M Jones. Steady-state modeling of gas turbine engines using the numerical propulsion system simulation code. In *ASME Turbo Expo 2010: Power for Land, Sea, and Air*, pages 89–116. American Society of Mechanical Engineers, June 2010.
- Edward J. Kowalski and Robert A. Atkins, Jr. A computer code for estimating installed performance of aircraft gas turbine

- engines. Technical report, National Aeronautics and Space Administration, 1979.
- Joachim Kurzke. Advanced user-friendly gas turbine performance calculations on a personal computer. In *ASME 1995 International Gas Turbine and Aeroengine Congress and Exposition*. American Society of Mechanical Engineers, June 1995.
- Joachim Kurzke. About simplifications in gas turbine performance calculations. In *Proceedings of the ASME Turbo Expo*, volume 3, pages 14–17, May 2007.
- Konstantinos G Kyprianidis, Ramon F Colmenares Quintero, Daniele S Pascovici, Stephen OT Ogaji, Pericles Pilidis, and Anestis I Kalfas. Eva: A tool for environmental assessment of novel propulsion cycles. In *ASME Turbo Expo 2008: Power for Land, Sea, and Air*, pages 547–556. American Society of Mechanical Engineers, 2008.
- Konstantinos G Kyprianidis, Andrew M Rolt, and Tomas Grönstedt. Multidisciplinary analysis of a geared fan intercooled core aero-engine. *Journal of Engineering for Gas Turbines and Power*, 136(1):011203, 2014.
- Linda Larsson, Tomas Grönstedt, and Konstantinos G Kyprianidis. Conceptual design and mission analysis for a geared turbofan and an open rotor configuration. In *ASME 2011 Turbo Expo: Turbine Technical Conference and Exposition*, pages 359–370. American Society of Mechanical Engineers, 2011.
- John K Lytle. The numerical propulsion system simulation: A multidisciplinary design system for aerospace vehicles. In *International Symposium on Air Breathing Engines*, September 1999.
- Lester Nichols and Christos Chamis. Numerical propulsion system simulation-an interdisciplinary approach. In *Conference on Advanced Space Exploration Initiative Technologies*, September 1991.
- Hans Olsson, Martin Otter, Sven Erik Mattsson, and Hilding Elmqvist. Balanced models in modelica 3.0 for increased model quality. In *Proceedings of the 6th International Modelica Conference*, 2008.
- Michael John Provost. The more electric aero-engine: a general overview from an engine manufacturer. In *Power Electronics, Machines and Drives*, 2002. *International Conference on*, number 487, pages 246–251. IEEE, 2002.
- Maido Saarlus. *Aircraft performance*. John Wiley & Sons, 2007.
- S-15 Gas Turbine Perf Simulation Nomenclature and Interfaces Committee of SAE. Application programming interface requirements for the presentation of gas turbine engine performance on digital computers (ARP4868). Technical report, Society of Automotive Engineers, 2001.
- S-15 Gas Turbine Perf Simulation Nomenclature and Interfaces Committee of SAE. Gas turbine engine performance presentation and nomenclature for digital computers using object-oriented programming (ARP5571). Technical report, Society of Automotive Engineers, 2005.
- Vishal Sethi. *Advanced performance simulation of gas turbine components and fluid thermodynamic properties*. PhD thesis, Cranfield University, April 2008.
- Vishal Sethi, Georgios Doulgeris, Pericles Pilidis, Alex Nind, Marc Doussinault, Pedro Cobas, and Almudena Rueda. The map fitting tool methodology: Gas turbine compressor off-design performance modeling. In *Journal of Turbomachinery*. American Society of Mechanical Engineers, September 2013.
- Philip P Walsh and Paul Fletcher. *Gas turbine performance*. John Wiley & Sons, 2004.
- Michael Winter. A view into the next generation of commercial aviation (2025 timeframe). In *AIAA Aerospace Today and Tomorrow*, 2013.

Virtual flight testing of a controller for gust load alleviation using FMI for cosimulation

Reiko Müller¹ Markus Ritter²

¹DLR, Institute of System Dynamics and Control, Oberpfaffenhofen, Germany, reiko.mueller@dlr.de

²DLR, Institute of Aeroelasticity, Göttingen, Germany, markus.ritter@dlr.de

Abstract

During aircraft design and certification, one of the most vital development tasks is the calculation of loads and stresses, subsequent structural sizing and iterative mutual adaptation with respect to the aircraft's systems. In an effort to build up a so called virtual flight testing capability in the DLR-wide project *Digital-X* (2012 - 2016), a simulation of a flexible aircraft model coupled with CFD based aerodynamics and a flight control system with included Gust Load Alleviation (GLA) was developed and subjected to a certification relevant gust encounter scenario. Due to the diversity of modeling and simulation tools present in the DLR, the Functional Mockup Interface (FMI) 2.0 model interfacing standard has been successfully employed to cosimulate the control system inside the enclosing simulation framework. **Keywords:** Virtual flight testing, Gust load alleviation, Flight control, FMI, Cosimulation

1 Introduction

An aircraft's flight envelope expresses the admissible region of flight depending on the current state (e.g. variables like angle of attack, Mach number and altitude), with upon exceeding, the aircraft will no longer be flyable (high/low speed stalling, buffeting). In analogy to this, the loads envelope specifies the corresponding limits which the aircraft structure can handle. With the advent of electronic flight control systems, an appropriate means for regulating loads automatically was found and is used to limit the maximum design loads to increase flight safety, as well as the ones due to maneuvering or environmental disturbances like gusts. The benefits are manifold, as for example structural stress and fatigue is reduced on the airframe, passenger comfort is increased and overall aircraft performance can be improved by structural design optimization.

In the following contribution, a novel application for loads analysis is introduced, combining hitherto disconnected simulation steps and forming a high - fidelity "virtual flight testing" - capability. In detail, an aircraft is discretized as Finite Element Method (FEM) - model, with the element's elastic motion solved by methods from Computational Structural Mechanics (CSM). The forces and moments acting on the airframe due to aerodynam-

ics are calculated from the conservation laws of mass, momentum, and energy. These have no closed form analytical solution and can only be solved by employing numerical methods from Computational Fluid Dynamics (CFD). The *Python* - based software framework *FlowSimulator* (Meinel and Einarsson, 2010) and CFD solver *TAU* (Schwamborn et al., 2006) were developed and utilized in the *Digital-X* project for multidisciplinary simulation of transport aircraft with aerodynamics calculated by CFD (Kroll et al., 2016). A *Modelica* - based flight control system with added gust load alleviation functionality had to be integrated in the *FlowSimulator* setup to conduct the virtual flight tests by means of cosimulation using the FMI 2.0 - standard (Mod, 2014). The principal layout of this approach is shown in figure 1. It benefits from the ad-

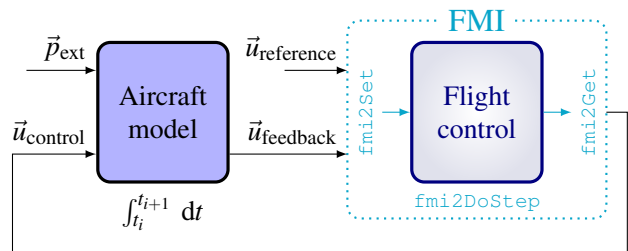


Figure 1. Integration loop of the controller using FMI for cosimulation to conduct virtual flight tests. $\vec{u}_{\text{reference}}$ contains controller reference values e.g. from a Flight Management System (FMS). As well, the aircraft model can depend on external parameters and inputs \vec{p}_{ext} that are not part of the cosimulation loop.

vantages of FMI, that are the time-savings due to omission of user-driven API development, interoperability for various tools and efficient simulation and event/error handling. Due to the large amount of simulations necessary for design and tuning of the controller to a specific test case, a second model based on a faster executing approximative method was established using the loads analysis software *Varloads* (Hofstee et al., 2003). The application scenario is an encounter of a frontal vertical gust, with the control objective of reducing the vertical accelerations and loads on the structure.

The paper is structured as follows: In section 2, the governing equations of motion and the elastic deformation of the aircraft are discussed. The two aerodynamic models necessary for the controller synthesis as well as the high

fidelity simulation are derived in 2.1. The controller is laid out in section 3, including the design in *Modelica* and the gust load alleviation functionality in sections 3.1 and 3.3, along with the integration in the cosimulation setup in section 3.4. Section 4 discusses the application to the gust encounter scenario, while conclusions and an outlook for future work are given in the final section 5.

2 Aircraft modeling

The motion of an aircraft through the air can be described in different levels of detail. The simplest notion is of a point on which the aircraft mass is concentrated, that translates due to external forces and the weight, given by the dynamic equilibrium of forces (d'Alembert principle). When taking into account distributed masses and the rotational movement of the aircraft, one arrives at the rigid-body equations of motion, yielding six degrees of freedom. These are defined in aircraft mean body axes with respect to a ground-fixed inertial Cartesian coordinate system on a local tangent plane (flat earth assumption) with uniform gravity, also known as the Newton-Euler equations (1):

$$\begin{bmatrix} \mathbf{M}_{bb} (\dot{\vec{V}}_b + \vec{\omega}_b \times \vec{V}_b) \\ \mathbf{I}_{bb} \dot{\vec{\omega}}_b + \vec{\omega}_b \times (\mathbf{I}_{bb} \vec{\omega}_b) \end{bmatrix} = \mathbf{T}_{rb}^T \Phi_{gr}^T \vec{P}_g^{\text{ext}} \quad (1)$$

with

\mathbf{M}_{bb}	Mass matrix
\mathbf{I}_{bb}	Inertia tensor
$\vec{V}_b = [uvw]^T$	Body-fixed velocity vector
$\vec{\omega}_b = [pqr]^T$	Rotational velocity vector w.r.t. body fixed system
\mathbf{T}_{rb}	Transformation of Center of Gravity (CG) to grid reference point

In (Waszak and Schmidt, 1988) the equations of motion of the elastic aircraft are derived using the mean axis conditions. These are fulfilled easily by using mode shapes (eigenvectors) of an unconstrained (free-free) structural model and ensure that the rigid body equations (1), and the linear elastic equations of structural mechanics in a modally reduced form (2), are inertially decoupled.

$$\begin{aligned} \Phi_{gf}^T \mathbf{M}_{gg} \Phi_{gf} \ddot{\vec{u}}_f + \Phi_{gf}^T \mathbf{B}_{gg} \Phi_{gf} \dot{\vec{u}}_f \\ + \Phi_{gf}^T \mathbf{K}_{gg} \Phi_{gf} \vec{u}_f = \Phi_{gf}^T \vec{P}_g^{\text{ext}} \end{aligned} \quad (2)$$

with

Φ_{gr}	Modal matrix rigid body modes
Φ_{gf}	Modal matrix of flexible modes
\vec{P}_g^{ext}	Vector of external forces to structural grid points
\mathbf{M}_{gg}	Physical mass matrix
\mathbf{B}_{gg}	Damping matrix
\mathbf{K}_{gg}	Stiffness matrix
\vec{u}_f	Generalized coordinates of elastic modes

Hence equations (1) and (2) are only coupled by means of the external forces \vec{P}_g^{ext} , which in the end allows that

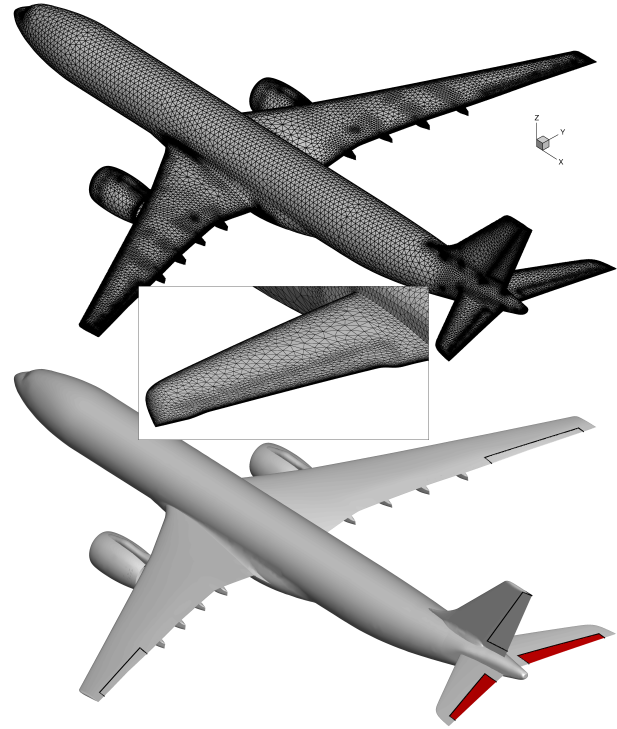


Figure 2. *Digital-X XRF-1* CFD computation mesh with control surfaces and exemplary deflection of the horizontal tail plane control surface of 5 degrees on top. A blending technique was used to obtain a smooth transition at the boundaries of the elevators, which are the only control surfaces used during the gust encounter cosimulation.

both the large nonlinear motions of a maneuver, and the small linear perturbation introduced by the flexible structure, be taken into account.

2.1 Aerodynamic models

The aerodynamic forces included in \vec{P}_g^{ext} are derived from the conservation laws for mass, momentum and energy. While the continuity equation depicts the mass flow through a control volume in the airflow, the Navier-Stokes equations describe the equilibrium of forces, taking into account viscosity, volume forces (e.g. due to gravity) and the momentum flow through the volume. Compressibility of the flow field requires the introduction of the energy equations, formulating the equilibrium between energy flow through the volume, energy produced due to the forces and moments, external energy contributions and inner and kinetic energy of the medium.

In combination, these form an equation system to calculate the forces / the pressure distribution on the aircraft's surface, for which however no closed form analytical solution exists. Numerical methods to solve this kind of problems are grouped under the term of Computational Fluid Dynamics (CFD), with DLR's *TAU* code being a comprehensive software environment for this task and therefore an obvious choice as CFD - solver for the *FlowSimulator*

framework (see description of the simulation setup in 3.4). Due to the nature of turbulent flow, changes can happen on a very small scale, which is why the computational grid for numerically solving the Navier-Stokes equations also may need a very fine resolution locally, to include all turbulent phenomena. As can be seen in figure 2, a higher grid density has been applied especially at geometry changes or regions of expected turbulence.

The complex grids in turn cause a large increase in computation time for calculation of the aerodynamic forces and moments, while during controller and aircraft design, quite often thousands of simulation runs are performed, e.g. to iteratively tune controller gains or to investigate aircraft response to stresses dependent on multidimensional parameter spaces. Due to their high demand on computational power, the Navier-Stokes equations are generally not viable for these kind of tasks and have to be simplified. A first step is to solve only for the unknowns that are most relevant to those applications, e.g. the pressure distribution on the object's surface.

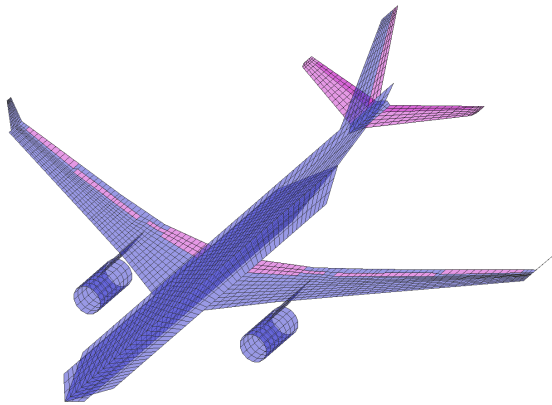


Figure 3. Aircraft aerodynamic model composed of lift surfaces, for use in the Vortex Lattice Method (VLM) or Doublet Lattice Method (DLM), generated by *VarLoads*. Blue panels belong to the aircraft body, purple ones to the control surfaces.

In the beginning of the 20th century, Prandtl found that for flows at higher Reynolds numbers $Re > 10^5$, the effect of viscosity is approximatively limited to a thin boundary layer encompassing the object's body. Consequently, the flow beyond the boundary layer can be considered as inviscid and importantly, the pressure gradient through it normal to the surface as zero ($\frac{\partial p}{\partial z} \approx 0$). In order to obtain the pressure distribution on the object's surface, it is therefore sufficient to calculate it in the inviscid flow just outside of the boundary layer using the inviscid Navier-Stokes or Euler equations. The assumption of isentropic (no energy contribution/drain) and irrotational flow allows to define a velocity potential function

$$\vec{v} = \text{grad } \Phi = [u, v, w] = \left[\frac{\partial \Phi}{\partial x}, \frac{\partial \Phi}{\partial y}, \frac{\partial \Phi}{\partial z} \right] \quad (3)$$

which is inserted into the Euler equations. These can

then be linearized around \vec{v} , with the disturbance velocities $[u', v', w']$

$$\vec{v} = \begin{bmatrix} u_\infty \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} u_\infty + \frac{\partial \Phi}{\partial x} \\ \frac{\partial \Phi}{\partial y} \\ \frac{\partial \Phi}{\partial z} \end{bmatrix} \quad (4)$$

to arrive at the unsteady Prandtl-Glauert equation:

$$(1 - Ma^2) \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2} - \frac{2U}{a^2} \frac{\partial^2 \Phi}{\partial x \partial t} - \frac{1}{a^2} \frac{\partial^2 \Phi}{\partial t^2} = 0 \quad (5)$$

When neglecting the time-dependent terms, the linear second order Laplace equation for the Vortex Lattice Method (VLM) (Hedman, 1966) is obtained:

$$(1 - Ma^2) \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2} = 0. \quad (6)$$

This method calculates a matrix of Aerodynamic Influence Coefficients (AIC) based on (6) to model lift distributed on an approximation of the aircraft consisting of several lift surfaces as shown in figure 3. The unsteady counterpart (in the frequency domain) for solving (5) is the Doublet Lattice Method (DLM). External aerodynamic and propulsive forces are added to the inertial forces by means of the Force Summation method to calculate resultant forces and moments on the aircraft. The loads analysis software *VarLoads*, which was jointly developed by Airbus and DLR (Hofstee et al., 2003), implements all of these modeling and simulation paradigms and was used to prepare the model with simplified aerodynamics for the controller synthesis.

3 Controller design and integration

The Flight Control System (FCS) or short "controller", follows the classical cascaded design layout that is well studied in both theory and practice (see e.g. (Brockhaus et al., 2011)). This layout is based upon the fact that the aircraft's equations of motion can be separated into parts that play a role on different timescales (timescale separation principle). For example, the body-fixed rotational rates $[p \ q \ r]_B$ as fast states are directly linked to the deflection of the control surfaces and resulting moments. On the other hand, states referring to orientation and even more position have a slower progression. This allows to dissect the flight control system into smaller parts, as shown in figure 4: The inner loop or Stability and Control Augmentation (SCA) - block can be designed to stabilize the aircraft and to dampen the dynamic aircraft modes (e.g. phugoid, dutch roll - modes). The autopilot in turn generates a reference orientation for the modified plant of the aircraft stabilized by the inner loop. It is designed to achieve a high tracking precision for the desired trajectory variables. The additional Gust Load Alleviation (GLA) is

arranged at the level of the SCA, since it is assumed here that the gusts cannot be sensed ahead of the aircraft (e.g. by LIDAR like in Hecker and Hahn (2007)) and necessitate fast reactions of the controller / the control surfaces.

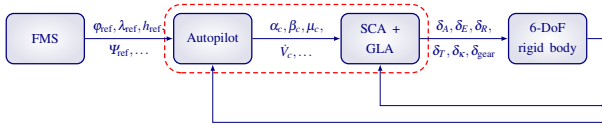


Figure 4. Structure of an electronic flight control system, consisting of the FMS and the FCS with autopilot and inner loop SCA. The framed blocks are the considered parts in this work.

As the scenario only considers a frontal vertical gust encounter, the GLA operates on the longitudinal dynamics, and can symmetrically deflect ailerons and elevators to attenuate the gust. Discrimination between inner and outer ailerons and elevators as well as distributed spoilers are incorporated in the GLA - layout, but only uniform and symmetric deflections of likewise δ_A and δ_E act as inputs. No actuator dynamics are modeled, due to their absence in the aircraft model of the cosimulation. Acceleration measurements are available at the Center of Gravity (CG) and form the single feedback variable:

$$n_{z,m} = \frac{\text{Lift}}{\text{Weight}} = \frac{V_K \dot{\gamma}}{g \cdot \cos(\Phi)} + \cos(\gamma) \quad (7)$$

with load factor $n_{z,m}$, kinematic velocity V_K , trajectory pitch angle γ and roll angle Φ . The load factor is fed into the parameterized GLA filter structure which generates control surface deflection commands, for the elevator δ_E^g with a filter structure containing e.g. a tunable time-constant. These are then super-positioned to the commands of the flight controller:

$$\delta_i = \delta_i^c + \delta_i^g, \quad i \in [A, E]. \quad (8)$$

Hence the autopilot and inner loop also contribute to the load alleviation due to the gust, by acting to hold altitude and speed.

3.1 Controller model in *Modelica*

The resulting Flight Control System (FCS) was implemented in *Modelica* using *Dymola* 2016, and is shown in figure 5. The *Modelica Standard* - and *LinearSystems* - libraries provide all of the needed models, with which a flight control library had been established. It consists of modules arranged in longitudinal, lateral, inner and outer loop controllers and is also prepared for use in conjunction with DLR's *FlightDynamics* library (Looye, 2008). The *Dymola* simulation tool makes use of the object oriented features of *Modelica*, allowing easy testing and interchange of different modules and furthermore offers an implementation of the FMI standard.

In the diagram view depicted in figure 5, the middle left and the lower center grey rectangular blocks represent

the FCS and the GLA respectively. The FCS consists of four channels for the four individual control effectors of the airplane (throttle, elevator, aileron and rudder). The autopilot modes are set to speed -, altitude - and course - hold, while the commanded sideslip angle β_c is zero. The inner loop receives orientation commands from the autopilot and calculates corresponding rates and control surface deflections. Each of the channels includes a set of several cascaded linear controllers. To ensure robustness over the flight envelope, multiple gust - and load - cases, a robust controller synthesis process would normally be appropriate. However, since only one gust encounter case is considered in this study, a simple tuning of the controller gains has been performed to minimize the effect on the wing root bending moment (see section 3.3).

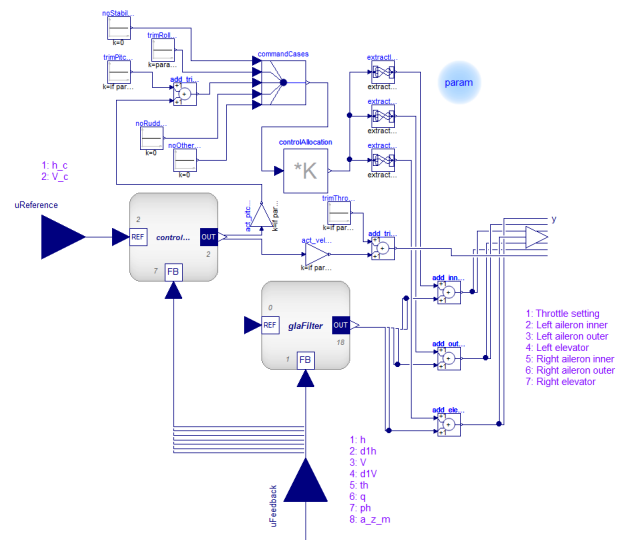


Figure 5. *Modelica* model of the longitudinal controller with gust load alleviation.

3.2 Initialization of the FMU

Each of the four channel's inner loops mentioned in the last section contain either integrator or derivative blocks with internal states that have to be initialized correctly to avoid transient oscillations in the beginning of the simulation. Furthermore the cosimulation must be compatible with both aircraft models and their respective trim algorithms. The given variables of the initialization process are the reference inputs $\vec{u}_{\text{reference}}$ and feedback inputs $\vec{u}_{\text{feedback}}$ from the aircraft, while the unknowns are the FMU outputs \vec{u}_{control} (see figure 1). Hence a two-step initialization of the closed-loop simulation setup is performed:

- At first, the aircraft is trimmed separately for steady horizontal flight at a given speed and altitude (see table 1 for a set of characteristic state values), without the controller. This yields values for e.g. α and Θ and also for elevator deflection δ_E and throttle δ_T ¹.

¹The initial values of the lateral effectors δ_A and δ_R are zero.

- In the second step, the Functional Mockup Unit (FMU) outputs need to be set to the aircraft control input values (\vec{u}_{control}) obtained in step 1. Yet due to FMI design, which prevents variables with output causality from the assignment of any value, additional trim parameters have to be defined.

Table 1. Trim values for aircraft model used for controller synthesis

Property	Unit	Value
Mach number	-	0.83
Altitude	ft	35000
Reference velocity	m/s	246.1
Aircraft mass	kg	198540
Angle of attack	°	4.55
Gust gradient H	m	85.9
Gust velocity in z - direction	m/s	-4.296

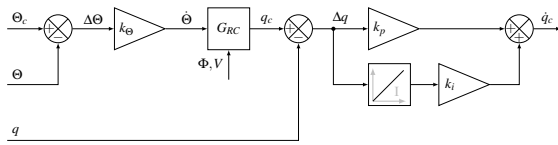


Figure 6. Inner loop pitch channel, with pitch angle Θ , pitch rate q and G_{RC} as transfer function containing the correction for turning flight (increase in pitch due to rotated lift vector).

This second step is further explained using the example of the inner loop pitch channel shown in figure 6: With given $\delta_{E,\text{trim}}$ and Θ_{trim} , the single degree of freedom is the initial state value of the integrator. The trim pitch angle is added to the autopilot command

$$\Theta_c = \Delta\Theta_{\text{AP}} + \Theta_{\text{trim}}, \quad (9)$$

where $\Delta\Theta_{\text{AP}}$ is zero here due to initial $h = h_c$. Likewise the elevator command consists of

$$\delta_E = \dot{q}_c + \delta_{E,\text{trim}} + \delta_{E,\text{GLA}}. \quad (10)$$

With the constraint of steady state integrator initialization ($\dot{x}_{\text{int}} = 0$), and similar provisions for the velocity channel, the initial equation of the controller model has to be specified as shown in listing 1. By calling the `initialize()` - method of the FMU, the controller can then match the preceding aircraft trim.

Listing 1. Initial equation of the controller model

```

initial equation
controllerLongitudinal.y[1] = trim_de_T;
controllerLongitudinal.y[2] = trim_de_E;

```

3.3 Synthesis of the GLA controller

The controller and GLA gains were adapted to the gust encounter scenario using the fast executing simplified model of section 2. The single objective of this process was the minimization of the bending moment around the aircraft's longitudinal x - axis (see figure 2) at the wing root station, M_x . In contrast to the high-fidelity simulation, both the elevators and the ailerons were actuated by the GLA. Figure 7 compares three gust encounters, one open loop, one with flight controller only, and one combined with additional GLA. The undisturbed case is added for reference and shows the bending moment at the trim condition.

The control objective is to reduce the initial maximum amplitudes of M_x . This is satisfied by the FCS and the GLA as expected, with the most notable difference in the second peak at $t \approx 0.7$ s. Due to several filter time-constants, the GLA does not act against the first falling peak, which is why the controller - and GLA - variants reduce the moment about the same amount. At the second rising gust peak, the GLA is able to reduce the moment around 45 %, however the GLA inputs generate an increased preceding moment. Using this highest GLA - peak, the reduction over the open loop case is still as large as 39 % with the steady trim moment value as baseline.

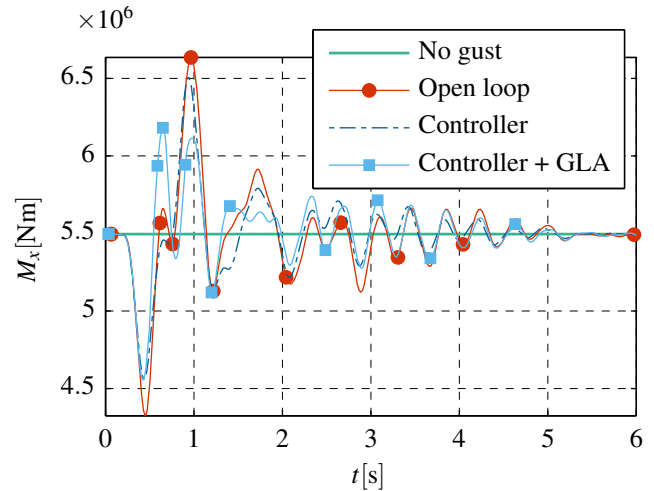


Figure 7. Wing-root bending moment for uncontrolled and two controlled gust encounter simulations.

3.4 Integration in simulation environment

The *FlowSimulator* framework allows to specify, integrate and simulate all sub-models necessary for the controlled coupled CSM - CFD application. A special *FlowSimulator* - plugin called *FSDynaflly* has been developed at DLR's Institute of Aeroelasticity to model the process chain for the controlled cosimulation, see figure 8.

After initialization, the governing equations of motion of the free-flying elastic aircraft (equations (1) and (2)) are solved by *FSDynaflly* for the current time step. Their outputs and the command references are passed on to the

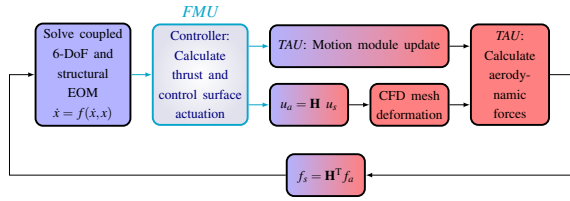


Figure 8. Time domain solution process of *FSDynaflty_6DOF* including the flight controller.

FMU to form the control error. The deflection commands calculated by the FMU are mapped onto the structural grid at the respective control surface positions, yielding modal deformations u_s . An unstructured mesh has been built for the *Digital-X XRF-1* configuration using the meshing software *CENTAUR*, with the control surfaces being cut into the CAD geometry based on locations provided by Airbus. Each control surface thus has a separate boundary marker in order to be deflected properly in the unsteady gust encounter simulation. As the structural grid does not coincide with the aerodynamic one, the deformations are multiplied with the splining matrix \mathbf{H} , which is built from Radial Basis Functions (RBFs). It is then morphed according to the deformations u_a using the submodule *FS-Deformation*, as is shown in figure 2 with the example of the Horizontal Tail Plane (HTP) - deflection. In parallel, another submodule of the CFD solver *TAU* calculates an update of the aircraft motion, followed by the actual call of *TAU* to solve for the new aerodynamic forces f_a of the next time step. To solve the equations of motion, these are transformed back into forces f_s relating to the structural grid by multiplication with \mathbf{H}^T .

The controller interfaces to *FSDynaflty* through the Functional Mockup Interface (FMI), in the working principle shown in figure 1. The FMI for cosimulation methodology was adopted, since deployment as model exchange - type FMU would have been far more complicated (e.g for integration and event handling). The complete controller model shown in figure 5 is exported together with the Sundials *CVode* ODE - solver compiled in a FMI - compliant library (64-bit .so for UNIX - type target simulation environment). As the application and interfacing layer of *FSDynaflty* is written in *Python*, the DLR - developed *Python* FMI - API of *PySimulator* (Pfeiffer et al., 2012) is used to address the FMU. Finally, a fixed-time step master algorithm enables communication between the two cosimulated models, a *Python* code representation is given in listing 2.

Listing 2. Master algorithm for the cosimulation of aircraft with the controller in *Python* (only the most relevant commands are displayed).

```
# Load the FMU
fcs = FMUInterface.FMUInterface(
    "Controller_GLA.fmu")
fcs.fmiInstantiate()
```

```
# Trim the aircraft
[x_tr, u_tr, dx_tr] = aircraft.trim(
    x0, u0, dx0,
    ix, iu, idx0)

# Set the trim parameters in the FMU ...
# ... to achieve u equal to u_tr
fmu_setReal_inValueAndReference(
    fcs, pars_trim,
    ["trim_de_T", "trim_de_E",
     "trim_de_A", "trim_de_R"], u_tr)

# Initialize the FMU
fcs.initialize()

# Integration loop
while aircraftODE.successful()
    and status == 0
    and t <= stopTime:
        # Set u to u_tr for the first ...
        # ... timestep
        if t == t0:
            u_in = u_tr
        else:
            u_in = u
        # Evaluate aircraft right hand side ...
        # ... and retrieve feedback
        der_x, out = aircraftODE.f(
            t, aircraftODE.y, u_in)
        # Set controller reference inputs
        fmu_setReal_inValueAndReference(
            fcs, u_ref, ["h", "V"], u_ref.val)
        # Set controller feedback inputs
        fmu_setReal_inValueAndReference(
            fcs, u_feedback,
            u_feedback.varNames, out)
        # Integrate one step for FMU
        status = fcs.do_step(t, dt, True)
        # Retrieve controller commands
        u = fmu_getReal_fromValueAndReference(
            fcs, y_out, y_out.varNames)
        # Set aircraft model inputs
        aircraftODE.set_f_params(u)
        # Integrate one step for aircraft
        aircraftODE.integrate(t + dt)
        # Increment the master time
        t = t + dt
    # End of integration
```

4 Vertical gust encounter simulation

As mentioned before, the only scenario covered in this contribution is an encounter of a discrete vertical gust with the assumption that all points in planes normal to the aircraft's velocity are affected (as defined in (Joint Aviation Authorities, 1994)). The vertical velocity profile is shaped according to equation (11)

$$w_{\text{wind}} = \frac{U_{ds}}{2} \left[1 - \cos\left(\frac{V}{H}\pi \cdot t\right) \right], \quad (11)$$

and therefore denoted as "One-minus-cosine" - gust. The parameters of this function are the design gust velocity U_{ds} , the gust gradient H , which is the distance parallel to the aircraft's flight path for the gust to reach its peak velocity, and $V \cdot t$ as the distance traveled into the gust.

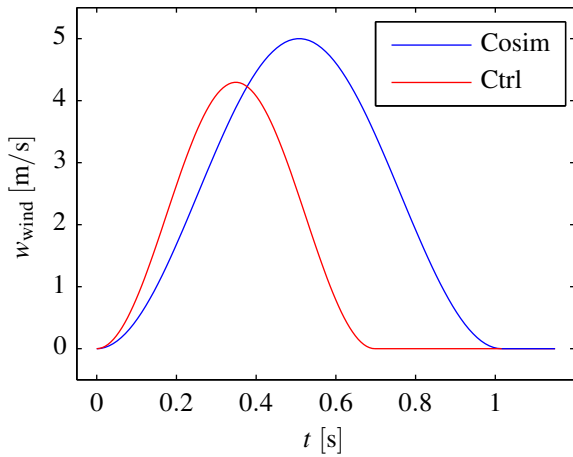


Figure 9. One minus cosine gust definitions used for the complete cosimulation and for the controller synthesis.

The gust parameters are slightly differing between the high fidelity cosimulation and controller simulation as shown in figure 9, similarly the angle of attack and HTP trim values, see table 2. Furthermore, in the high fidelity simulation results presented in the following, only the control surfaces of the horizontal tail plane were used as primary control surfaces to reduce the loads acting on the airframe. This was partly due to project time constraints and availability of other control surface geometries like spoilers and ailerons. The gains in overall load reduction can therefore not be compared between the high- and lower fidelity models as of now, yet this was not the goal of this specific application anyway.

Table 2. Trim values for high fidelity simulation, only values differing from those in table 1 are listed.

Property	Unit	Value
Angle of attack	°	3.39
HTP trim angle	°	2.58
Gust gradient	m	125
Gust velocity in z - direction	m/s	-5
Communication time stepsize	s	0.01

Two gust encounter simulations are presented in the following, one without gust attenuation, and another one with the flight controller in the loop. The results are shown in figure 10 in terms of selected states measured in the body fixed coordinate system with the pitch rate q , its time derivative \dot{q} , the velocity in the z - direction w and the acceleration in the z - direction \dot{w} .

As can be seen from the time function of the states plotted, the actuation of the controller markedly reduces the accelerations of the airframe's center of gravity, thereby reducing structural loads as well. A reduction of the heave accelerations of more than 20% is achieved. This simulation is a purely symmetric maneuver, meaning that no

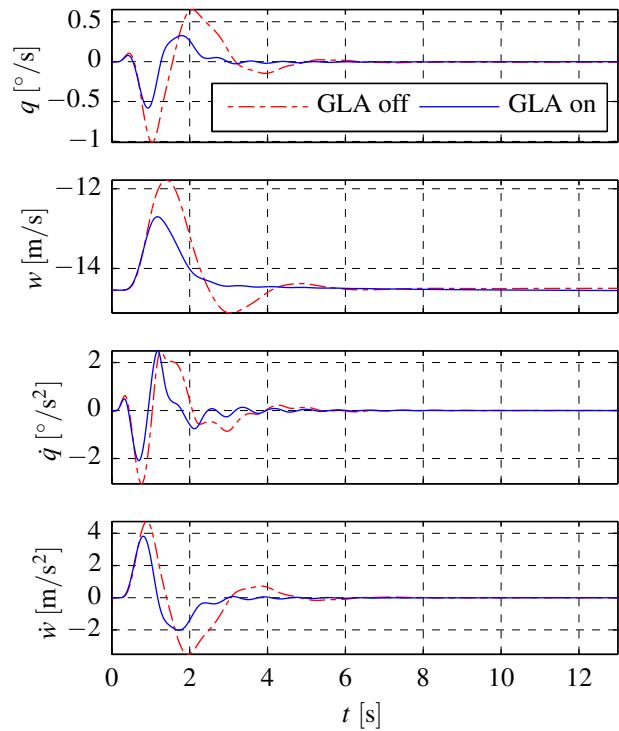


Figure 10. Selected states of the aircraft as function of time, showing a reduction of accelerations due to the gust load alleviation.

distinctive lateral motions are excited during the gust encounter. Small but negligible lateral motions occur due to non-negative values for I_{xy} , and I_{yz} of the tensor of inertia of the aircraft. These entries can be attributed to the fact that the mass model is not purely symmetric. The output of the controller in terms of the time dependent rotation of the horizontal tail plane's control surface is shown in figure 11. The maximum deflection of the HTP is about 1.3° . This value is comparatively low, but the gust disturbance velocity is small as well.

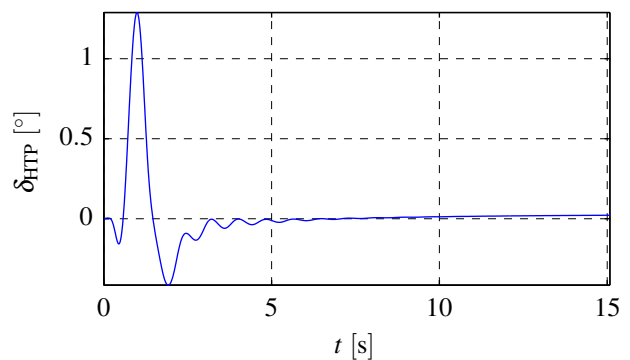


Figure 11. Controller output in terms of the rotation of the HTP control surface.

5 Conclusion and outlook

In this contribution, a new methodology for loads analysis and virtual flight testing of flight controllers is presented. Usually disconnected simulation steps are combined into a single process chain, including elastic structural aircraft modeling, full Navier-Stokes aerodynamics and a flight control system with gust load alleviation.

A flight controller with outer and inner loop, as well as gust load alleviation system was set up in *Modelica* using *Dymola*. It was tuned for a gust encounter scenario using a Vortex Lattice Method (VLM) based aerodynamic model, allowing the required large number of simulations during controller synthesis. A final reduction of up to 45% in the wing root bending moment was found there.

A setup for the cosimulation was developed in *Python*, including a fixed-step master algorithm connecting the simulation framework *FSDynaflly* with the controller. By employing the FMI standard to interface the controller, dedicated API development for the dissimilar aircraft models could be omitted. Furthermore the functionalities of FMI for cosimulation allowed an easy setup and efficient operation of the controlled high fidelity simulation. Reductions in the vertical and the pitch accelerations of up to 20 % were achieved, also consequentially leading to a reduction in the structural loads.

After successfully completing this first proof of concept, future work will be directed towards functionality in larger simulation studies with multi-parameter or even multi-model test cases and different scenarios (e.g. maneuver loads, flight performance analysis). Ensuring the robustness of the controller for the entire flight envelope will be an important prerequisite for these applications, and could be achieved by employing methods from robust control design (e.g. H_∞ or robust LPV control).

An immediate next step will be the addition of new control surfaces (ailerons and possibly spoilers) to the CFD mesh, since currently the only means of controlling the aircraft and the loads is the horizontal tail plane. Based on the results of the simplified aerodynamics simulation, it is expected that loads on the main wing can be further reduced by this approach. As well, it should be worthwhile to incorporate additional design criteria in the controller synthesis process. By treating it as a multi-objective optimization problem, the investigation of trade offs between e.g. load reduction, passenger comfort and flying qualities is made possible.

6 Acknowledgments

This work was prepared during the course of DLR project *Digital-X*. The authors would like to express their thanks to the following colleagues for their support and input: Martin Leitner, Hans-Dieter Joos, Andreas Pfeiffer and Thiemo Kier.

References

- Rudolf Brockhaus, Wolfgang Alles, and Robert Luckner. *Flugregelung*. Springer, 2011. ISBN 9783642014437. URL <http://books.google.de/books?id=2IKXH3skXBwC>.
- Simon Hecker and Klaus-Uwe Hahn. Advanced gust load alleviation system for large flexible aircraft. In *Proceeding 1st CEAS Konferenz*, 2007.
- Sven G Hedman. Vortex lattice method for calculation of quasi steady state loadings on thin elastic wings in subsonic flow. Technical report, DTIC Document, 1966.
- Jeroen Hofstee, Thiemo Kier, Chiara Cerulli, and Gertjan Looye. A variable, fully flexible dynamic response tool for special investigations (VarLoads). In *2003 CEAS/AIAA/NVvL International Forum on Aeroelasticity and Structural Dynamics*, 2003.
- Joint Aviation Authorities. Joint aviation requirements. JAR-25. Large aeroplanes. *Civil Aviation Authority Printing & Publication Services, Greville House*, 37, 1994.
- Norbert Kroll, Mohammad Abu-Zurayk, Dilianna Dimitrov, T Franz, Tanja Führer, Thomas Gerhold, Stefan Görtz, Ralf Heinrich, Caslav Ilic, Jonas Jepsen, et al. DLR project Digital-X: towards virtual aircraft design and flight testing based on high-fidelity methods. *CEAS Aeronautical Journal*, 7(1):3–27, 2016.
- Gertjan Looye. The new DLR flight dynamics library. In *Proceedings of the 6th International Modelica Conference*, volume 1, pages 193–202, 2008.
- Michael Meinel and Gunnar O Einarsson. The FlowSimulator framework for massively parallel CFD applications. *PARA 2010*, 2010.
- Functional Mock-up Interface for Model Exchange and Co-Simulation, Version 2.0*. Modelica Association, July 2014.
- Andreas Pfeiffer, Matthias Hellerer, Stefan Hartweg, Martin Otter, and Matthias Reiner. PySimulator-A Simulation and Analysis Environment in Python with Plugin Infrastructure. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, pages 523–536. Linköping University Electronic Press, 2012. 76.
- Dieter Schwamborn, Thomas Gerhold, and Ralf Heinrich. The DLR TAU-Code: recent applications in research and industry. In *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics, Egmond aan Zee, The Netherlands, September 5-8, 2006*. Delft University of Technology; European Community on Computational Methods in Applied Sciences (ECCOMAS), 2006.
- Martin R Waszak and David K Schmidt. Flight dynamics of aeroelastic vehicles. *Journal of Aircraft*, 25(6):563–571, 1988.

The DLR Environment Library for Multi-Disciplinary Aerospace Applications

Lâle Evrim Briese¹ Andreas Klöckner¹ Matthias Reiner¹

¹Institute of System Dynamics and Control, DLR German Aerospace Center, Oberpfaffenhofen, Germany,
Lale.Briese@dlr.de · Andreas.Kloeckner@dlr.de · Matthias.Reiner@dlr.de

Abstract

Environment models are vital elements for any type of vehicle dynamics simulations, such as aircraft or satellites. Recently, applications have been developed, where these previously unrelated regimes of operation need to be integrated, for example in end-to-end simulations of launch vehicles. This paper therefore introduces the new DLR Environment Library, which implements common models of planets, geospheres, currents, kinematics, and physical effects for such applications. It provides a set of environment models with minimal dependencies, complete compatibility to the Modelica Standard Library, and convenient drag & drop usage. The DLR Environment Library is expected to immensely aid developing end-to-end simulation models integrating components from DLR's SpaceSystems and FlightDynamics Libraries. In particular, it will importantly decrease modeling errors due to its consistent environment models.

Keywords: environment modeling, gravitational models, planet models, atmosphere models, kinematic state models, space mission simulation, multi-disciplinary modeling

1 Introduction

Modeling of environmental effects is highly relevant for vehicle simulations, such as aircraft (Klöckner et al., 2013; Looye, 2008), satellite (Reiner and Bals, 2014; Pulecchi et al., 2006), or launch vehicle simulations (Acquatella, 2016). While these domains have mostly been treated as independent in the past, latest developments point towards even more integrated simulation needs. For instance, reusable launchers will require accurate modeling of aircraft-like and satellite-like flight phases. Especially, combined multi-disciplinary simulations, including several vehicle types like launch vehicles and satellites with corresponding environmental conditions as well as ground stations, are of great interest within end-to-end space mission simulations.

For several years, the Institute of System Dynamics and Control at the DLR German Aerospace Center has been developing Modelica-based libraries for the modeling and simulation of flight vehicles (DLR FlightDynamics Library) and satellites (DLR SpaceSystems Library) as shown in Figure 1. These libraries can operate either in-

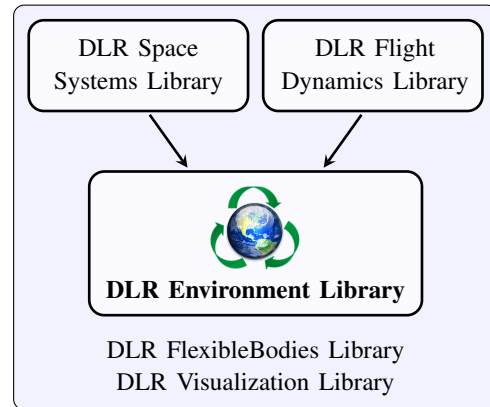


Figure 1. An overview of the interaction of application-based libraries with the new DLR Environment Library.

dependently from each other or in combination with other libraries. For example, the DLR FlexibleBodies Library is used for modeling flexible structures and the DLR Visualization Library is used for visualizing multibody systems.

Although these libraries share one common need for the modeling of environmental effects, there have been different application- and library-specific environment models for each library. Certainly, not every application requires the same level of detail or the same type of environment models for the specific design regime. For example, a satellite system in Low Earth Orbit (LEO) can neglect gravitational effects of another planet and a flight vehicle with a cruise flight altitude of 40.000ft is hardly influenced by the solar radiation pressure, unlike spacecraft in a deep space environment.

In general, most environment models are stored inside subpackages of application libraries, providing just the minimal amount of data needed for the realistic simulation of the desired application. For this purpose, most environment models take into account gravitational acceleration, atmospheric parameters, and specific influences which are relevant for use cases as presented for example in Reiner and Bals (2014), Looye (2008) or Pulecchi et al. (2006). The advantages of application- and library-dependent environment models are clearly the reduction of the level of detail and the simplification of complex environmental effects. This leads to a smaller amount of available models for individual purposes and consequently to less required maintenance.

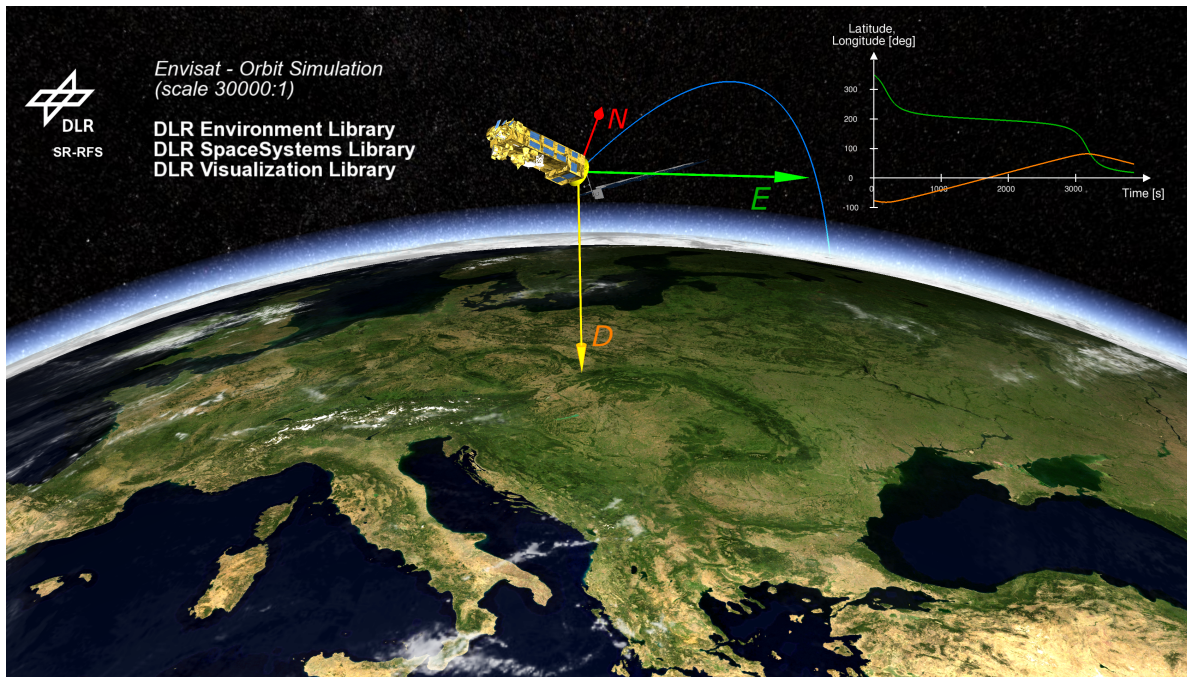


Figure 2. Visualization of Envisat in an orbit simulation using the DLR Environment Library.

A major disadvantage of this strategy arises when multiple application-based libraries must be used for one comprehensive problem. The combination of several multi-disciplinary libraries can then lead to redundancies or discrepancies in the overall environmental formulation. This is especially the case if two or more application libraries are combined for the end-to-end simulation of e.g. reusable launch vehicles. For example, depending on the specific application, coordinate systems or rotation sequences can be defined differently and therefore can lead to often unmanageable errors. Also, different gravitational models can result in mismatched data of the Inertial Measurement Unit (IMU) of each system, finally increasing the overall error between vehicles in a multi-disciplinary simulation.

To prevent these problems, it has been decided to build a common library, based on knowledge of environmental effects inside application libraries. The overall goal of the library is to provide a modular, non-redundant and user-friendly formulation of environmental effects. It has to be compatible with the Modelica Standard Library (MSL) as well as the application libraries developed at the Institute of System Dynamics and Control.

Within this paper, the new Environment Library is presented. In Section 2, an overview of the library is given, including its purpose, its main characteristics, its basic structure as well as the verification of its compatibility with the MSL. Based on this section, some selected features of the library are further introduced in Section 3. The functionality of the models is demonstrated within Section 3 with specific examples provided by the application libraries mentioned before. The main advantages of the proposed library are summarized in Section 4.

2 Overview of the Library

For multi-disciplinary end-to-end simulations regarding all kinds of vehicles (from Earth-based flight and launch vehicles to spacecraft in deep space environment), specific but consistent environmental conditions have to be considered. The library is developed to fulfill these multi-disciplinary requirements and is therefore based on environment models from two application-related Modelica-based libraries developed by the Institute of System Dynamics and Control:

- DLR SpaceSystems Library (SSL) (Reiner and Bals, 2014), and
- DLR FlightDynamics Library (FDL) (Looye, 2008; Klöckner et al., 2014a)

The library in its current version is fully tested within the simulation environment Dymola 2017. Although it is designed as a stand-alone library, it is based on the Modelica Standard Library (3.2.2) and builds on the DLR Visualization Library (1.4) for optional visualization as presented in Figure 2 for an orbit simulation of the environmental satellite Envisat. The DLR Visualization Library is not required for the functionality of the provided environment models, but it enables drag & drop visualization of all parts of the simulation. With the visual effects, a better understanding of the overall model behaviour can be provided, especially when flight dynamics are considered.

The main characteristics of this library regarding its basic structure and the implementation of the provided models were determined by considering the following goals.

- **Modularity**

Models inside the library shall be able to work independently. A limited amount of interdependent models is considered acceptable.

- **Adaptivity**

The library shall be able to grow and has to allow individual modification. The library shall adapt easily to changes in other used libraries (e.g. MSL).

- **Generality**

The library shall be seen as a common ground for multiple disciplines. It shall therefore provide models which can adapt to requirements of multiple disciplines without affecting unrelated disciplines.

- **Reusability**

Models provided by the library shall be usable in different domains, at the best as drag & drop models.

- **User-friendliness**

The library and its models shall be designed without the need of intensive maintenance or without excessive user interaction or configuration.

- **Simplicity**

The amount of configurable parameters shall be reduced as much as possible and conditional changes of the model behaviour shall be implemented in separate models instead of using enumerated types.

2.1 Basic Structure of the Library

An overview of the top-level structure of the Environment Library is shown in Figure 3. The library provides a documentation with information about the library itself including contact information, references, release notes and a tutorial for beginners. Additionally, examples to demonstrate the functionality of the provided models are implemented. Environment models are stored in the main subpackages. Further dependencies between the main subpackages are kept minimal in order to achieve maximum modularity.

The main subpackages for the modeling of planets, geospheres and currents are created based on an object-oriented structure as shown in Figure 4. All main subpackages contain the package `BaseClasses` in which partial models for each discipline are implemented. These partial models provide specific functions to be accessed from anywhere within the simulation model corresponding to the `inner` and `outer` concept in Modelica. From these partial models, drag & drop models in the top-level of each subpackage are extended. This planet-independent library structure enables a highly modular, user-friendly, object-oriented and consistent modeling of environmental conditions. These advantages are vitally important for multi-disciplinary simulations regarding multiple vehicles in different environments. Especially, many separate models instead of one general model containing all possible

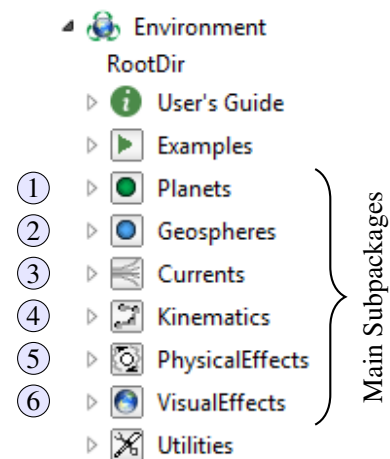


Figure 3. An overview of the top-level library structure.

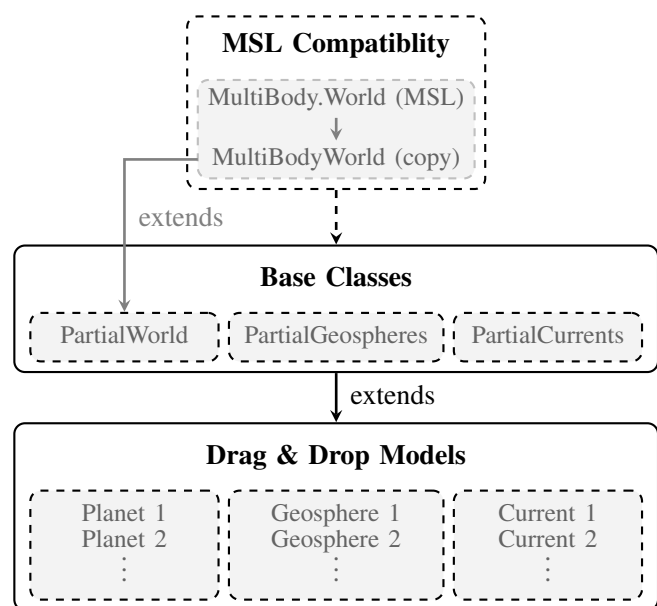


Figure 4. An overview of the object-oriented library structure.

environmental effects, offer a better understanding of the overall model behaviour. A brief summary of these subpackages is given below:

- **Planets ①**

This subpackage contains generic MSL-based planet models, providing relevant planet frames, the global simulation time, the time-dependent rotation angle of rotating planets, the planet constellation inside the solar system as well as the advanced replaceable `gravityAcceleration` function.

- **Geospheres ②**

Models which represent general geospheres (e.g. atmosphere) are included inside this package, providing inner models with replaceable functions similar to the `gravityAcceleration` function to calculate specific geospheric parameters as well as the mean current of the geosphere.

- **Current** ③

Inside this package, components for modeling of additional currents like user-defined wind are stored, which are designed as drag & drop models to induce currents (or wind effects in terms of the atmosphere) to a connected body frame.

- **Kinematics** ④

The package `Kinematics` contains several functions to describe kinematic relationships and coordinate transformations as well as models for automatic state selection for MSL-based bodies.

- **Physical Effects** ⑤

Inside the package `PhysicalEffects` specific models representing certain physical effects are implemented which provide for example forces due to the solar radiation pressure or information about the geomagnetic field.

- **VisualEffects** ⑥

Several models to visualize the Earth, the Moon and the Sun are implemented inside this package. They are based on components in the DLR Visualization Library. With the visualization software *SimVis* (Bellmann, 2009) and the provided high-resolution visualization data of the Earth, the simulation results for a certain problem can be shown in a realistic and easily recognizable way (see Figure 2).

Subpackage-specific functions are implemented inside the `Functions` packages, whereas components which are not intended for further usage are stored inside the `Internal` packages. Using common base classes for all drag & drop models, two possible modeling approaches can be followed by the user (van der Linden et al., 2014). First, the partial models from the `BaseClasses` packages can be placed inside generic simulation models such that they can be replaced dynamically for each application model. Second, the drag & drop models from the top level can be used as fully functional and stand-alone components of application models.

The `Utilities` package provides environment related constants, enumeration types, icons as well as general functions. Inside the subpackage `User`, individual user options can be stored as additional constants. Especially for large files that cannot be saved inside the `Resources` folder, like the visualization data of the Earth, a modifiable path name to the source directory can be supplied and managed by each user individually.

3 Selected Features of the Library

Within this section, some of the main subpackages and features of the Environment Library are emphasized. Selected use cases are provided in thematically related subsections to demonstrate the functionality of the presented models as well as the wide range of available model variants inside the library.

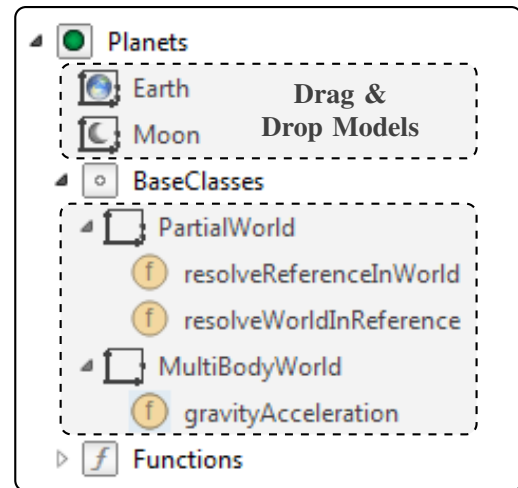


Figure 5. An overview of the subpackage `Planets`.

3.1 The Planets subpackage

The `Planets` subpackage as shown in Figure 5 provides planet models, which are compatible to the standard multi-body world component of the MSL (Otter et al., 2003). This facilitates the switching to enhanced world models in application libraries without changing the application library structure or code.

The model `MultiBodyWorld` with the replaceable function `gravityAcceleration` is a modified copy of the original world component with two important changes. On the one hand, the parameters have been rearranged in additional tabs without changing their content. This is done in order to provide a better overview of the parameters inside the world component and to enhance the user-friendliness of the overall model with a thematically structured graphical user interface. On the other hand, the equations to define the position and orientation of the frame `frame_b` in the original world component are no longer defined as equations. Instead, they are integrated as variable declarations to the definition of the frame itself such that these values can be changed while using an `extends` statement. With this new modeling approach for an extended world model and its modified frame definition, moving planets within the solar system can be integrated into the simulation taking into account their influences on each other.

From this modified world component a partial model `PartialWorld` for planetary objects is extended introducing an additional frame and two internal functions to resolve any vector from the inertial frame `frame_b` to this new moving reference frame. Because the world component is defined as an `inner` model, it is possible to call the two internal functions inside the `PartialWorld` from anywhere inside the simulation model under the condition that an `outer` command referencing the world component is used. Although this modeling strategy is applicable to any rotating planetary object, the implementation of the Earth will be explained further in this section.

The Earth component is extended from the partial model `PartialWorld`. The frame `frame_b` is defined as an inertial frame (*Earth Centered Inertial*, ECI), whereas the new additional frame represents the *Earth Centered Earth Fixed* (ECEF) coordinate system with an attitude depending on the simulation time and the Earth's angular velocity. Additionally, the parameter `gravityType` is redefined with Earth-specific gravity types and the function `gravityAcceleration` is redeclared with the corresponding functions to calculate the gravity acceleration vector of the Earth.

With this modeling concept, components from the MSL can be used with the new world definition. This can be demonstrated with the example *Double Pendulum* placed on the Earth's surface. The original world component inside the MSL and the modified `MultiBodyWorld` are based on the same gravity model and therefore provide the same results as presented in Figure 6. Additionally, the resulting acceleration of the component `boxBody2` using a more advanced gravitational model based on the `Earth` is presented. It is shown, that the behaviour of the *Double Pendulum* changes significantly over time depending on the chosen gravitational model, demonstrating the consequences of using different gravitational models within end-to-end simulations.

Some planet-specific features implemented inside the extended planet models are further introduced using the `Earth` component as an example.

Absolute Simulation Time

The new world component adds an absolute simulation time `julianDate`, which is initialized with parameters provided by the user either in Julian date format or in years, months, days and hours with minutes and seconds as fractions.

Rotation of the Earth

The Earth's rotation angle at a certain time is determined as a function of the absolute simulation time as proposed inside the *Naval Observatory Vector Astrometry Software* (NOVAS) (Bangert et al., 2011). The transformation matrix from the ECI frame to the rotating ECEF frame can either be defined as a simplified rotation between these frames using only the rotation angle (ERA) around the Earth's rotation axis (z -axis) with respect to the ECI frame, or it can be calculated considering the nutation and precession depending on the Julian date as well as the difference in seconds between *Universal Time* and *Universal Coordinated Time* (Bangert et al., 2011). The leap seconds are automatically computed based on tabular data (Astronomical Almanac, 2010), but can also be provided by the user as input values.

Gravity Acceleration

To calculate the gravity acceleration vector $\mathbf{g}_0 \in \mathbb{R}^3$ with respect to the inertial frame, the `Earth` model provides a `gravityAcceleration` function comparable to the stan-

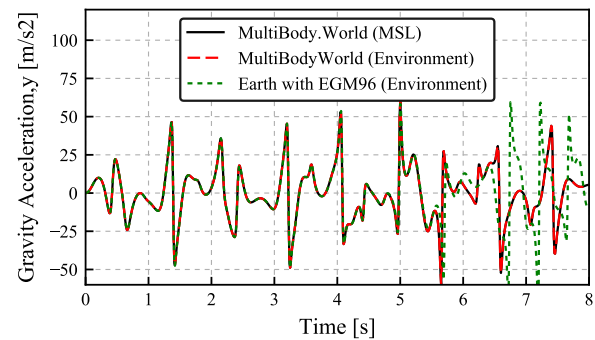


Figure 6. Compatibility of the planet models with the MSL.

dard world component. The basic gravity types from the original world component are still available to maintain the compatibility to the MSL. Additionally, more precise gravity models like the EGM96 (Lemoine et al., 1998) and the Vinti Order 6 (Bate et al., 1971) gravity models can be chosen for the calculation of the gravity acceleration vector $\mathbf{g}_E \in \mathbb{R}^3$ of the Earth. The EGM96 model uses terms up to the second degree of the zonal harmonic coefficients of the gravitational potential as discussed in Reiner and Bals (2014). Those are only dependent on the symmetrical mass distribution along the z -axis of the Earth. The Vinti Order 6 potential function takes into account the perturbation accelerations due to the Earth's nonsphericity based on Bate et al. (1971).

If needed, the gravity acceleration from the Moon and the Sun can be considered inside the precise gravity models. Therefore, the current positions of the Moon \mathbf{r}_M and the Sun \mathbf{r}_S with respect to the Earth are calculated analytically with low precision formulae for planetary positions (van Flandern and Pulkkinen, 1979). As an alternative, they can also be obtained from the DE405 ephemeris files (Standish, 1998). Relying on the current Julian date as an input parameter, the DE405 ephemeris coefficients are extracted from an external C-code to calculate the positions of the Moon and the Sun.

The total gravity acceleration vector \mathbf{g}_0 is finally calculated as the sum of the gravity acceleration from the Earth \mathbf{g}_E , the Moon \mathbf{g}_M and the Sun \mathbf{g}_S . The gravity acceleration vectors \mathbf{g}_M and \mathbf{g}_S are calculated according to the Equation (1) depending on the position $\mathbf{r}_{M,S} \in \mathbb{R}^3$ and the gravitational constant $G_{M,S}$ of the Moon or the Sun.

$$\mathbf{g}_{M,S} = G_{M,S} \left(\frac{\mathbf{r}_{M,S} - \mathbf{r}_E}{\|\mathbf{r}_{M,S} - \mathbf{r}_E\|^3} - \frac{\mathbf{r}_{M,S}}{\|\mathbf{r}_{M,S}\|^3} \right) \quad (1)$$

The absolute gravity acceleration for different gravity models is shown in Figure 7 for the given latitude of the International Space Station (ISS). The position of the ISS is calculated from orbital elements provided by *NORAD Two-Line Element sets* (TLE) (NASA, 2011) for a given point in time as implemented inside the SSL. The models provide very similar results, except for the expected disturbances.

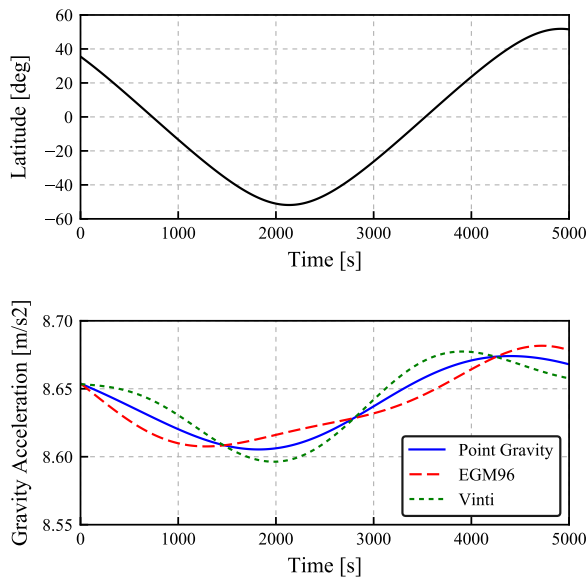


Figure 7. Comparison between different gravity models (ISS).

3.2 The Geospheres subpackage

The subpackage `Geospheres` is implemented to provide specific types of geosphere models, like atmosphere models which can be used in combination with flight or launch vehicles, respectively.

All main geosphere models are extended from the partial model `PartialGeosphere`. This model contains two replaceable functions, which can be called from anywhere inside a simulation model. Both functions require the position of an object with respect to the rotating reference frame. The outputs of the function `baseProperties` are the absolute pressure, temperature, density and speed of sound, corresponding to the `BaseProperties` implementation inside the MSL package `Media`. The output of the second replaceable function `meanCurrent` is the velocity of the geosphere-specific current.

Similar to the implementation of planets, the user can choose between two modeling concepts using either the `PartialGeosphere` or the stand-alone geosphere models from the top-level of this subpackage. The replaceable functions `baseProperties` and `meanCurrent` can be re-declared with advanced functions for each specific geosphere model.

In terms of atmosphere models, geodetic parameters such as latitude, longitude and altitude have to be calculated from the given input position. For this purpose, consistent kinematic functions inside the `Kinematics` package are used (see Section 3.5) approximating the shape of the planet according to the World Geodetic System '84 (WGS'84) (NIMA, 2000). Optionally, the user can decide if the geoid undulation between the calculated altitude and the *Mean Sea Level* shall be taken into account. For this reason, the geoid information based on the EGM96 model is computed with an external C-code (Lemoine et al., 1998).

In the Environment Library, different geosphere models for the Earth's atmosphere are implemented. For example, a constant atmosphere with user-provided parameters or a user-defined atmosphere with input values based on tabular data can be chosen. For the latter option, the tabular data is interpolated using the altitude of the object. Other geosphere components use standard atmosphere models as explained in the following list:

- **StandardAtmosphere (ISA)**

Within this component, two atmosphere models covering several regimes are implemented. The *Two Zones Model* by Schänzer (1969) can be chosen especially for flight vehicles with an altitude up to 40.000ft. The *Three Zones Model* (NASA, 2015) can be used if atmospheric conditions between the troposphere and the upper stratosphere are needed. This model is based on atmospheric measurements with separate curve fits for the troposphere, the lower and the upper stratosphere.

- **StandardAtmosphere76**

This component is based on the U.S. Standard Atmosphere model from 1976 where the atmospheric parameters can be determined for altitudes from -5 km up to 1000 km. For altitudes above 50 km, the data for this atmosphere model is based on rocket and satellite measurements (NASA, 1976).

- **NRLMSISEAtmosphere**

The NRL-MSISE-00 model is a highly accurate empirical model developed by the *U.S. Naval Research Laboratory* (NRL) (Picone et al., 2001). It is primarily used by spacecraft due to its accuracy in altitudes above 100 km and its range from the ground to the exosphere. The density and temperature at a certain position are computed using the NRL-MSISE-00 database with an external C-code. As inputs, the Julian date provided by the world component as well as the geodetic parameters latitude, longitude and altitude are required.

For all atmosphere models, the mean current is based on a logarithmic approach to determine the velocity vector in the Earth's boundary layer with respect to the ground. In Figure 8, a comparison between the provided atmosphere models is shown for the ascent phase of a generic launch vehicle depending on its current altitude. All atmosphere models provide similar results for the atmospheric density but significantly different results for example for the temperature corresponding to the approximation methods used inside particular atmosphere models. Especially in multi-disciplinary simulations, one common atmosphere model instead of many application-specific atmosphere models for several vehicles can therefore reduce errors in the overall model behaviour.

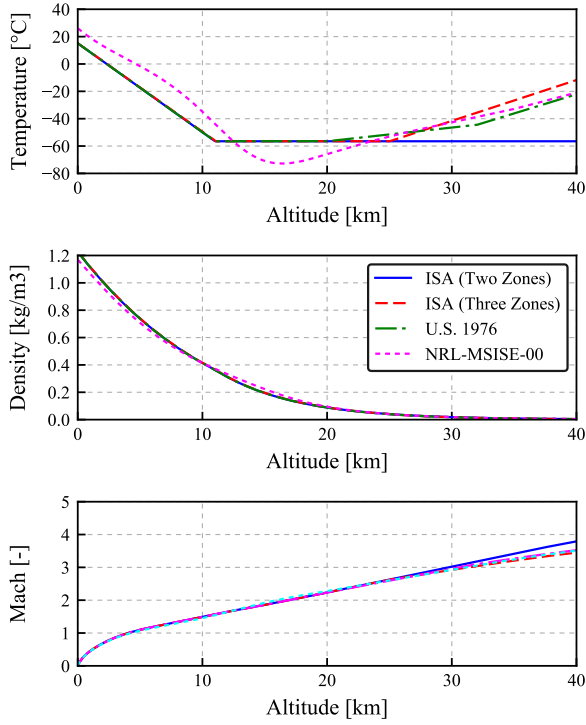


Figure 8. Comparison between different atmosphere models.

3.3 The Currents subpackage

Currents are used as a supplement to geosphere models as presented in Section 3.2. While geosphere models provide mean currents for any location with their internal function `meanCurrent`, a current model provides simplified models of local flow velocities such as turbulence or gusts. Each submodel (e.g. aircraft or spacecraft) can have its own local current model. All currents retrieve the mean current from the geosphere and add local effects to it.

A simple example is the continuous Dryden turbulence model (MIL-STD-1797A, 1990), which adds a low-pass-filtered white noise to the mean current (The MathWorks, 2016). Such an approach is illustrated in Figure 9 for a user-defined wind profile, where the filtered noise is added to the mean current from the geosphere model. This approach makes it possible to also cover distributed flow effects, such as wake vortices or delayed turbulence.

Like for the geosphere models, the user has the option to also use local wind or gust effects based on tabular data, which interpolates the velocity components according to the position of the object connected to the current's frame.

3.4 The PhysicalEffects subpackage

The subpackage `PhysicalEffects` provides stand-alone drag & drop models to automatically induce forces and torques due to physical effects on the attached frames. In contrast to the previous subpackages, these models are not based on a common partial model. However, all models fulfill the same goals as defined in Section 2 for instance in terms of modularity, simplicity and user-friendliness. Selected features of this subpackage are described below.

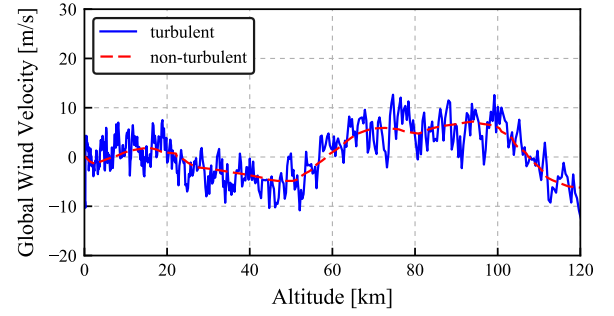


Figure 9. Application of a turbulent current on a wind profile.

Gravity Gradient Torque

The gravity gradient torque is modeled as a torque τ_a that acts on a connected frame with the position $\mathbf{r}_{0,a}$ and the rotational transformation matrix \mathbf{T}_a . The torque is caused by the allocation of the mass with respect to its center and depends on the inertia tensor $\mathbf{I}_B \in \mathbb{R}^{3 \times 3}$ (Larson and Wertz, 1999). In Equation (2), the gravity acceleration vector $\mathbf{g}_0 \in \mathbb{R}^3$ is a function of the position $\mathbf{r}_{0,a}$ and the Julian date t_J which is retrieved for the position of the connected frame directly from the world component.

$$\tau_a = \left(\mathbf{T}_a \mathbf{g}_0(\mathbf{r}_{0,a}, t_J) \frac{3}{\|\mathbf{r}_{0,a}\|} \right) \times \left(\mathbf{I}_B \mathbf{T}_a \frac{-\mathbf{r}_{0,a}}{\|\mathbf{r}_{0,a}\|} \right) \quad (2)$$

Solar Radiation Pressure

The effect of the solar radiation pressure is modeled as a force \mathbf{f}_{sp} that acts on the connected frame. Shadows of the Moon and the Sun are considered with the shadow factor $\chi_{sp} \in [0, 1]$ using a cylindrical shadow model. The equations are implemented as proposed in Montenbruck and Gill (2000). Required parameters are the effective area A_{sp} of the solar radiation pressure and its normal vector \mathbf{n}_{sp} as well as the coefficient of reflectivity of the material $\xi_{sp} \in [0, 1]$ (total absorption to total reflection) as shown in Equations (3) to (5). The distance between the Sun and the frame is defined as $\mathbf{d}_{sp} \in \mathbb{R}^3$. The solar radiation pressure p_\odot is assumed to be constant for spacecraft near Earth.

$$c_{sp,\theta} = \frac{\mathbf{n}_{sp}}{\|\mathbf{n}_{sp}\|} \frac{\mathbf{d}_{sp}}{\|\mathbf{d}_{sp}\|} \quad (3)$$

$$\mathbf{c}_{sp,R} = \left[(1 - \xi_{sp}) \frac{\mathbf{d}_{sp}}{\|\mathbf{d}_{sp}\|} + 2 \xi_{sp} c_{sp,\theta} \frac{\mathbf{n}_{sp}}{\|\mathbf{n}_{sp}\|} \right] \quad (4)$$

$$\mathbf{f}_{sp} = -\chi_{sp} p_\odot A_{sp} c_{sp,\theta} \mathbf{c}_{sp,R} \frac{AU^2}{\|\mathbf{d}_{sp}\|^2} \quad (5)$$

Geomagnetic Field

The geomagnetic field can be computed for a connected frame by the `GeoMagneticField` component, using the US/UK World Magnetic Model (WMM) from 2010 or 2015 (Maus et al., 2010). The model provides a magnetic field vector $\mathbf{B}_m \in \mathbb{R}^3$ that depends on the latitude, longitude and altitude of the component as well as the current

Julian date provided by the world model. The output vector is calculated in the local *North-East-Down* (NED) frame which can be transformed to any other coordinate system like ECI or ECEF with the provided functions inside the package `Kinematics` (see Section 3.5). The resulting geomagnetic field can be used for example in simulations with magnetic actuators or *Inertial Navigation Systems* (INS).

Atmospheric Drag for Spacecraft

The atmospheric drag is caused by friction with the remainder of the atmosphere depending on the altitude. Like the solar radiation pressure, the atmospheric drag is modeled as a force and torque element acting on the attached frame which should be located at the center of mass of the object. The density ρ of the atmosphere is provided by the NRL-MSISE-00 model due to its accuracy in near Earth orbit. The drag force \mathbf{f}_{ad} and torque $\boldsymbol{\tau}_{ad}$ can be computed using Equations (6) and (7) where \mathbf{v}_{rel} is the relative velocity of the object with respect to the rotating Earth.

$$\mathbf{f}_{ad} = -0.5 c_d A_{ad} \rho \|\mathbf{v}_{rel}\| \mathbf{v}_{rel} \quad (6)$$

$$\boldsymbol{\tau}_{ad} = 0.5 c_d A_{ad} \rho \|\mathbf{v}_{rel}\|^2 \left[\frac{\mathbf{v}_{rel}}{\|\mathbf{v}_{rel}\|} \times \mathbf{T}_a \mathbf{d}_{cp} \right] \quad (7)$$

Required parameters are the drag coefficient c_d , the effective area A_{ad} and the vector from the center of pressure to the center of mass \mathbf{d}_{cp} , resolved in the attached frame.

3.5 The Kinematics subpackage

The Environment library includes a comprehensive toolset for coordinate transformations and kinematics simulation. These can be used for `MultiBody` models to flexibly

- compute kinematic states in different notations,
- define different notations of continuous states, and
- constrain the kinematics to lower-order models.

In order to compute kinematic states of a `MultiBody` model, functions are provided which transform the standard state set of a `MultiBody` frame (i.e. position \mathbf{r}_0 , attitude \mathbf{R}, \mathbf{T} , and rotational velocity \mathbf{R}, \mathbf{w}) to a broad variety of different notations. The structure of the provided models and functions reflects this distinction in separate collections of conversion functions. Since many notations, such as WGS'84 positions or an aircraft's attitude, are given relative to the planet's reference system, a further discrimination is made between conversions in the inertial and the world reference system (see Figures 10 and 11).

The conversion functions are used by the provided sensor models, which retrieve the states of a frame. The implemented functions to calculate the kinematic relationships are also generalized such that they can be used to compute the required parameters given in any other geodetic system instead of the WGS'84. The simplified and user-friendly structure of the kinematic functions provides

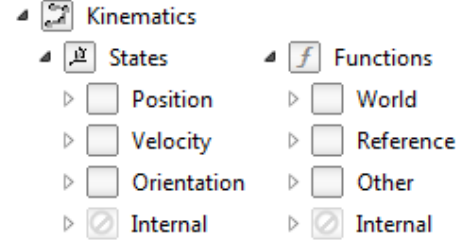


Figure 10. An overview of the package `Kinematics`.

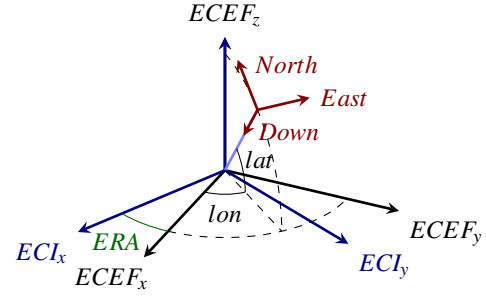


Figure 11. An overview of some basic coordinate systems.

a better overview for the user. Common functions can reduce errors due to different implementations of coordinate systems within application-based libraries.

In addition to computing the states in a required notation, the library provides models to define the very same notations as actual continuous time states of a `MultiBody` model by simply dragging the component into a model and connecting it to a frame connector. This is accomplished by first, defining the desired states with `stateSelect=StateSelect.always`, second, transforming the desired states into standard `MultiBody` notation, and finally, setting the frame variables to the result. As for the transformation functions, there are components to set position, velocity and attitude states independently from each other.

Finally, constraint models are provided, which interface seamlessly with the standard `MultiBody` models. This includes a generalization of the quasi-steady flight kinematics inside the FDL to general `MultiBody` models. Therefore, any six degrees of freedom (DOF) model can be transformed effectively into a three DOF model removing the rotational states. The transformation is accomplished by explicitly setting $\mathbf{R}, \mathbf{w} = \{0, 0, 0\}$ and creating new unknown variables for the orientation \mathbf{Q} . The model thus interrupts the usual flow of calculation (conceptually a double integration from torques to rates and attitude) as shown in Equations (8) to (10).

$$\text{der}(\mathbf{R}, \mathbf{w}) := \mathbf{f}(\mathbf{t}) \quad \Rightarrow \quad \mathbf{0} := \mathbf{f}(\mathbf{t}), \quad (8)$$

$$\text{der}(\mathbf{Q}) := \mathbf{f}(\mathbf{R}, \mathbf{w}) \quad \Rightarrow \quad \text{no kinematics}, \quad (9)$$

$$\mathbf{t} := \mathbf{f}(\mathbf{Q}, \mathbf{R}, \mathbf{w}) \quad \Rightarrow \quad \mathbf{Q} := \mathbf{f}(\mathbf{t}). \quad (10)$$

By rooting `frame_a.R` in the model, the kinematics equations in the `MultiBody` components are disabled and by setting $\mathbf{R}, \mathbf{w} = \{0, 0, 0\}$, the dynamics equations implicit

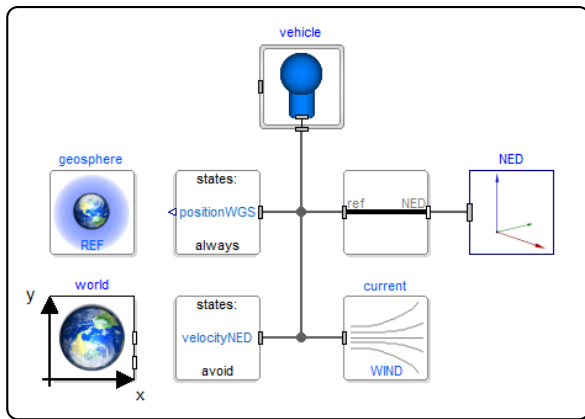


Figure 12. An overview of the structure of a simulation model using the presented environment and kinematics models.

itly force the torques to be zero. In order to achieve a well-defined model, external torque equations are required to compute the unknown attitude variables Q .

As for the usage of the models provided by the subpackage `Kinematics`, the structure of a simulation model for a generic vehicle is shown representatively in Figure 12 in combination with the previously presented models within the DLR Environment Library.

4 Conclusion

The DLR Environment Library is a Modelica-based library for modeling environmental effects for application-specific libraries. Over the past years, environment models, optionally including planet definitions or atmospheric parameters, have been developed independently and in a smaller scale within each application library. These separate developments have induced problems due to redundant declarations, mismatched level of detail, accuracy and precision within multi-disciplinary projects. With the DLR Environment Library, these problems are solved, as introduced in previous sections.

Especially, the new library and modeling concept based on an object-oriented library structure provides several advantages as listed below:

- modular, reusable and comprehensible structure,
- easily adaptable to new requirements & applications,
- consistent definition of environmental conditions,
- simple, user-friendly and understandable models,
- reduced maintenance demands.

Although the development of general terrain, weather and aerodynamic models is in progress, these packages have been excluded from the content of this paper, since the models are not yet fully implemented and tested inside the Environment Library. The implementation of all planets within the solar system is also planned for the future.

Acknowledgements

We would like to thank Dr. Gertjan Looye (DLR German Aerospace Center) for his contributions to former implementations of environment models inside the DLR FlightDynamics Library.

References

- P. Acquatella. Launch Vehicle Multibody Dynamics Modeling Framework for Preliminary Design Studies. *6th International Conference on Astrodynamics Tools and Techniques (ICAAT)*, 2016.
- Astronomical Almanac. *The Astronomical Almanac for the Year 2011*. United Kingdom Hydrographic Office, 2010. ISBN: 978-07-0774-103-1.
- J. Bangert, W. Puatua, G. Kaplan, J. Bartlett, W. Harris, A. Fredericks, and A. Monet. User's Guide to NOVAS Version C3.1. Technical report, U.S. Naval Observatory, 2011.
- R. Bate, D. Müller, and J. White. *Fundamentals of Astrodynamics*. Dover Publications, Inc., 1971. ISBN: 0-486-60061-0.
- T. Bellmann. Interactive Simulations and advanced Visualization with Modelica. In *Proceedings of the 7th International Modelica Conference*, pages 541–550, 2009. doi:10.3384/ecp09430056.
- A. Klöckner, M. Leitner, D. Schlabe, and G. Looye. Integrated Modelling of an Unmanned High-Altitude Solar-Powered Aircraft for Control Law Design Analysis. In *Advances in Aerospace Guidance Navigation and Control - Selected Papers of the Second CEAS Specialist Conference on Guidance, Navigation and Control*, pages 535–548. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-38252-9.
- A. Klöckner, G. Looye, R. Müller, R. Kuchar, F. Re, and M. Leitner. Object-Oriented Aircraft Modeling with the DLR FlightDynamics library. In *9th AIRTEC 2014 International Congress*, 2014a.
- A. Klöckner, F. L. J. van der Linden, and D. Zimmer. Noise Generation for Continuous System Simulation. In *Proceedings of the 10th International Modelica Conference*, pages 837–846, 2014b. ISBN: 978-91-7519-380-9.
- A. Klöckner, A. Knobloch, and A. Heckmann. How to Shape Noise Spectra for Continuous System Simulation. In *Proceedings of the 11th International Modelica Conference*, pages 411–418, 2015. ISBN: 978-91-7685-955-1.
- W. J. Larson and J. R. Wertz. *Space Mission Analysis and Design*, volume 3. Microcosm Press and Kluwer Academic Publishers, 1999. ISBN: 1-881883-10-8.
- F. G. Lemoine, S. C. Kenyon, J. K. Factor, and R. G. Trimmer et al. The Development of the Joint NASA GSFC and National Imagery and Mapping Agency NIMA Geopotential Model EGM96. Technical report, National Aeronautics and Space Administration (NASA), 1998.

- G. Looye. The New DLR Flight Dynamics Library. In *Proceedings of the 6th International Modelica Conference*, volume 1, pages 193–202, 2008.
- S. Maus, S. Macmillan, S. McLean, B. Hamilton, A. Thomson, M. Nair, and C. Rollins. The US/UK World Magnetic Model for 2010-2015. Technical report, National Oceanic and Atmospheric Administration (NOAA), 2010.
- MIL-STD-1797A. *Flying Qualities of Piloted Aircraft*. U.S. Department of Defense, 1990. Military Standard.
- O. Montenbruck and E. Gill. *Satellite Orbits - Models, Methods and Applications*. Springer Verlag, Heidelberg, 2000. ISBN: 978-3-642-63547-2.
- NASA. U.S. Standard Atmosphere, 1976. Technical report, National Aeronautics and Space Administration, 1976.
- NASA. Definition of Two-line Element Set Coordinate System, 2011. National Aeronautics and Space Administration, http://spaceflight.nasa.gov/realddata/sightings/SSapplications/Post/JavaSSOP/SSOP_Help/tle_def.html.
- NASA. Earth Atmosphere Model, 2015. National Aeronautics and Space Administration, <https://www.grc.nasa.gov/WWW/K-12/airplane/atmosmet.html>.
- NIMA. World Geodetic System 1984 - Its Definition and Relationships with Local Geodetic Systems. Technical report, National Imagery and Mapping Agency, 2000.
- M. Otter, H. Elmqvist, and S. Mattsson. The New Modelica MultiBody Library. In *Proceedings of the 3rd International Modelica Conference*, pages 311–330, 2003.
- J. M. Picone, A. E. Hedin, and A. C. Aikin D. P. Drob. NRLMSISE-00 empirical model of the atmosphere: Statistical comparisons and scientific issues. *Journal of Geophysical Research*, 107, 2001. doi:10.1029/2002JA009430.
- T. Pulecchi, F. Casella, and M. Lovera. A Modelica Library for Space Flight Dynamics. In *Proceedings of the 5th International Modelica Conference*, pages 107–116, 2006.
- M. J. Reiner and J. Bals. Nonlinear inverse models for the control of satellites with flexible structures. In *Proceedings of the 10th International Modelica Conference*, pages 577–587, 2014. doi:10.3384/ECP14096577.
- G. Schänzer. *Einführung in die Flugphysik*. Institut für Flugführung, TU Braunschweig, 1969. Lecture notes.
- E. M. Standish. JPL Planetary and Lunar Ephemerides, DE405 / LE405. Technical report, Jet Propulsion Laboratory, 1998.
- The MathWorks. Dryden Wind Turbulence Model, 2016. <http://de.mathworks.com/help/aeroblks/drydenwindturbulencemodelcontinuous.html>.
- F. L. J. van der Linden, C. Schlegel, M. Christmann, G. Regula, C. I. Hill, P. Giangrande, J.-C. Maré, and I. Egaña. Implementation of a Modelica Library for Simulation of Electromechanical Actuators for Aircraft and Helicopters. In *Proceedings of the 10th International Modelica Conference*, pages 757–766, 2014. doi:10.3384/ECP14096757.
- T. C. van Flandern and K. F. Pulkkinen. Low-Precision Formulae for Planetary Positions. *The Astrophysical Journal Supplement Series*, 41:391–411, 1979.