On using model approximation techniques for better understanding of models implemented in Modelica

Anton Sodja Borut Zupančič

Faculty of Electrical Engineering, University of Ljubljana
Tržaška 25, 1000 Ljubljana, Slovenia
{anton.sodja,borut.zupancic}@fe.uni-lj.si

Abstract

Modelica enables rapid development of detailed models of heterogeneous and complex systems. However, resulting models are as complicated as reality itself and therefore it may be hard to identify causes for model behavior or verify that model behaves correctly. A traditional engineering approach is to use intuition and experience to identify important parts of the model with the highest impact on model behavior for specific scenario. Numerous model order reduction and simplification techniques (i.e., metrics used by these methods) have been developed to automatically estimate important parts of the models for a certain scenario and thus alleviate reliance on subjective factors, i.e., intuition and past experience.

In this paper are discussed model order reduction and simplification techniques (e.g., metrics used by these techniques for rankings of elements) which are applicable to wide range of Modelica models built from already available libraries. Modelica models are translated to set of differential-algebraic equations and for the latter there are numerous tools for model order reduction already available. However, these tools are not designed for helping users understand the model's behavior and the reduced model may be hard to understand by the user because the structure of the original model is lost. Hierarchical decomposition of the model must be presereved and if the model is developed with a graphical schematics then elements (nodes) of the schematics must be ranked. Therefore we adapted energy-based metrics used in ranking of bond-graphs' elements to much more losely defined Modelica's schematics, so they can be used complementary with ranking methods that work with equations.

Keywords: model order reduction; model simplification; verification

1 Introduction

An important aspect of the Modelica language design is user interaction for efficient modeling of large, complex and heterogeneous physical systems.

Models are usually decomposed in several hierarchical levels. On the bottom of hierarchy are submodels of basic physical phenomena which are most commonly stated as a set of (acausal) differential-algebraic equations and it is thus most conveniently that these equations can be entered directly (e.g., without a need for any kind of manipulation or even transformation to some other description formalism). On higher hierarchical levels, model is described graphically by schematics (i.e., object diagrams) and the obtained scheme usually reflects the topology of the system. Model representation in Modelica is thus understandable also to domain specialists unfamiliar with computer simulation of dynamic system.

Modelica is object-oriented modeling language and thus includes features such as inheritance and replaceable models. This language features are necessary for efficient implementation of model libraries [13], but they increase implementation complexity and make browsing sources of the components from libraries more difficult. For example, component *DynamicPipe* – a model of a straight pipe with distributed mass, energy and momentum balances – from the Standard Modelica Library consists of four base models and three replaceable elements which are also models with complex inheritance hierarchy. The description of the pipe's dynamics, equations of balances and thermodynamic state of the medium in the pipe, is split among more than ten (partial) models to achieve efficient component reuse and prevent code duplication. This kind of model decomposition might not have a physical meaning – it only addresses implementation issues.

In practice, domain specialists usually already have

some calculations (e.g. in Excel) which they want to use for verification of the model implemented in Modelica and also for clarifying unexpected behavior of (usually) much more detailed and complex model in Modelica. So, a matching between calculations they have and model in Modelica is desired. However, modeling environments supporting Modelica currently do not provide many tools that would facilitate investigating and exploring the model. Most of the complex models are build up with use of different model libraries and when the documentation of those libraries do not suffice, especially when the submodels are highly customized components (with replaceable subcomponents and modifications), it is necessary to look under the hood of the used components. But due to complicated implementation of library components, it is undoable for most domain specialists.

Engineers use experience and intuition to determine important parts of the model which have the highest impact on system's dominant dynamics or model's simulation response in specific scenario. In an attempt to diminish reliance on subjective factors such as experience, numerous modeling metrics and methodologies have been developed. They usually require strict modeling formalisms and thus not much effort was put into integrating them into Modelica environments.

2 Model order reduction and simplification techniques

Detailed models of complex systems are also as complex and hard to understand as reality they model. The interpretation of underlying equations or extraction of an in-depth system understanding can get impossible even for relatively small systems [11]. Therefore, symbolic analysis methods, most notably of electrical circuits, incorporate various symbolic approximation techniques which are used to simplify symbolic expression or schematic diagrams and also reduce order (state-space dimension) of the model [10].

An important class of the model order reduction and simplification methods when used in system analysis or for structural design is when they generate a *proper model*, i.e., reduced model with the minimum complexity required to meet the performance specifications and possessing physically meaningful parameters and states [5].

A numerous mixed numerical-symbolic model order reduction and simplification techniques have been developed and successfully applied so far [10, 12, 4, 5]. They usually consist of running a series of simulations, ranking the individual coordinates or elements by the appropriate (quantitative) metrics and removing those that fall below a certain threshold [2].

2.1 Equation-based simplification

All analytic models can be described by a system of equations and even if some other modeling formalism is used (e.g., block schemes, bond graphs, etc.), it is possible to export the model as a system of equations. However, model representation in a form consisting of symbolic (algebraic) expressions is meaningful to user only in certain situations, for example, use of transfer functions in control design.

For equation-based simplification, variables of interest must be selected and metrics used for ranking of expressions' terms is then selected as a numerical error with respect to an objective function given by the variables of interest.

Simplification strategies include various algebraic manipulations (e.g., substitution of a variable), where no error is introduced into the simplified equations, and modification of the equations that results in the approximate system (e.g., term deletion, linearization of equations, etc.) which requires a numeric simulation to determine the error caused by modification [14].

Simplification can have a global effect, i.e., affects whole system of equations, when some variables of the system are manipulated or local effect when only single term of one equation is manipulated.

2.2 Structure-based simplification

Most of modern modeling tools provide a graphical interface where models are represented by schematics. Graphical descriptions of the models are based on various modeling formalisms, schematics can be a merely graphical representation of algebraic expressions (e.g., block graphs) and symbolics comprising the schematics represent single or a group of algebraic operations or they can provide additional information about the system (e.g., information about topology of the system). In the latter case, it is sensible to chose customized simplification techniques, although it is possible to map models simplified by equation-based order reduction and simplification techniques to a graphical representation of the original model [12].

Because all physical systems have in common conservation of mass and energy, a widely used class of metrics for order reduction of proper models in physical-systems modeling are related to energy or

power [2]. Energy-based metrics require a modeling formalism where energy of the model's components is easily extracted, for example, bond graphs [9, 5]. Bond-graph modeling is a form of object-oriented physical systems modeling: elements can be seen as object interacting with each others – interactions are described by acausal bonds [1].

Among successfully applied energy-based techniques for bond-graph simplification are ranking of elements based on RMS power of bonds [9], ranking on activity – amount of energy that flow in and out of the element over the given time [5] and comparing the energy associated with each bond to those in neighboring bonds and eliminating those with smallest relative energy [15]. Result of a model simplification by these techniques is also a model described by bond graph. Furthermore, all the energy-based metrics have some physical meaning and can thus help with understanding and addressing modeling issues.

3 Simplification of models in Modelica

According to authors knowledge, there is no modeling environment that provides tools for simplification and order reduction of model implemented in Modelica directly. A Modelica model must be flatten and the resulting DAE system is then exported to a designated tools where model order reduction and simplification is performed.

This approach is suitable for some applications, for example, when reduced model is needed for control design. In such cases, loss of information caused by flattening is not problematic, because only close matching of reduced and original model's behavior is required. However, in applications like model verification and debugging or when model is used to gain insight for system performance improvement, it is desired that simplified model is also a valid Modelica model with the same structure as the original (with the same hierarchical decomposition and topology of schematics).

4 Ranking elements of object diagram

4.1 Choice of metrics

Object diagrams consist of connected symbols representing components (submodels). What kind of interaction a connection defines is determined by type of connectors (i.e., ports) the connected components have. In Modelica is a type of connector very loosely defined. In general, it is a list of variables with some qualifications (e.g., causality, type of variable: intensive – extensive, etc.), but it can also have a hierarchical structure [6].

Although a large number of different kind of schematics can be modeled with appropriately defined connectors, are the most important acausal connections for modeling physical interactions. Each (dynamic) interaction between physical systems results in a energy exchange between the system, so it is very intuitive to chose energy-based metrics for simplification of physical systems models.

Modelica's object diagrams, when modeling physical systems, share some similarities with bond graphs, which are also a form of object-oriented acausal modeling. Therefore it is easy to adapt most of bond-graph simplification techniques to Modelica's object diagrams.

Connectors usually contains a pair of effort and flow variable (however, their product is not necessarily an energy flow like in bond graph formalisms), as can be seen by inspecting Modelica Standard Library [7] where elementary connector definitions for almost all physical domains are gathered:

- Interaction between components in analog circuits (*Modelica.electric*) is determined by voltage v and current i, the latter is a flow variable, and the power of the interaction is product of both variables: $p = v \cdot i$.
- Similar is connector in *Modelica.Magnetic* composed of variables for magnetic potential difference V_m and magnetic flux Φ, an effort and flow variable respectively. Power of the connection is product of variables: $p = V_m \cdot \Phi$.
- Connectors used for modeling of 1-D mechanics, translational and rotational, consist of position s and angle ϕ respectively, and force f and torque τ respectively. However, product of connector's effort and flow variable is no longer power. For determination of the power of connection, displacement variable has to be differentiated: $p = \frac{d}{dt}s \cdot f$ and $p = \frac{d}{dt}\phi \cdot \tau$ for translational and rotational mechanics respectively.
- In *Modelica Multibody* library, which deals with 3-D mechanics, are effort and flow variables no longer scalars, they are 6-dimensional vectors,

so a state of a free-body (having 6 degree-of-freedom) can be determined. Furthermore, due to computational restrictions, implementation of connector takes also into account a suitable selection of a frame of reference (forces, torques and orientation are expressed in local, while position is in global frame of reference). A definition of the connector is the following:

```
connector Frame
  SI.Position r_0[3];
  Frames.Orientation R;
  flow SI.Force f[3];
  flow SI.Torque t[3];
end Frame;
```

Position is determined with variable *r_0*, while orientation R is a structure containing transformation matrix T from global to local frame of reference and vector of angular velocities ω in local frame of reference. Forces and torques are given by vectors f and t respectively. Power of the connection can be calculated by expression: $p = \frac{d}{dt}(\mathbf{T} \cdot r_o) \cdot f + \omega \cdot t$, where again, there is a need to differentiate position after transformation to local frame.

- Connector for modeling heat transfer in 1-D consists of effort variable temperature T and flow variable for heat-flow rate Q_{flow}. The energy transfer is in this case equal to flow variable, $p = Q_{flow}$.

- Library *Modelica.Fluid* deals with modeling of heat and mass transfer. The connector used in library's components which covers also mass transfer is implemented as following:

```
connector FluidPort
  replaceable package Medium =
   Modelica.Media.Interfaces.PartialMedium;

  flow Medium.MassFlowRate m_flow;
  Medium.AbsolutePressure p;
  stream Medium.SpecificEnthalpy
      h_outflow;
  stream Medium.MassFraction
      Xi_outflow[Medium.nXi];
end FluidPort;
```

Besides effort and flow variable, pressure p and mass-flow rate m_{flow} respectively, the connector includes also additional information about properties of the substance which is being exchanged in the interaction modeled by a connection of type *FluidPort*: specific enthalpy h and composition of substance (vector of mass fractions X_i if substance is a mixture). The thermodynamic state of the substance is uniquely determined by the variables of connector and all the other (thermodynamic) properties can be calculated by using functions provided by package *Medium* which is a parameter of the connector. However, thermal diffusion is not covered by this connector (it is neglected).

Energy flow associated with the connector is composed of thermal, hydraulic and chemical term and could be calculated as following [3]: $p = \dot{m} \cdot s \cdot T + \dot{m} \cdot p/\rho + \sum \mu_i \cdot \dot{N}_i$. Quantities specific entropy s, temperature T, density ρ, chemical potential μ_i and molar flow $\dot{N}_i$ can be calculated from thermodynamical state equations provided by package *Medium*.

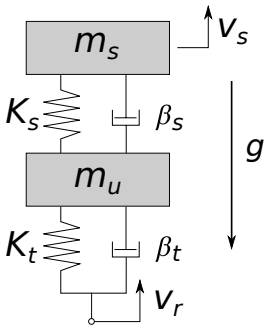

Figure 1: Scheme of car a suspension.

Although it is possible to calculate energy flow of the connector from the variables of the connector, this is not always possible to do as a post-processing the simulation results. For example, derivative of the position or angle in connector of the library for 1-D mechanics may not be available if this variable is not chosen for state variable. This implies instrumentation of the model.

Most bond-graphs energy-based metrics, like [5], require energy flow of the element. Fg. 1 illustrates a scheme of a car suspension for one wheel. Corresponding representation of a model with a bond graph is depicted in Fg. 2. In Fg. 2 can be seen that each element (e.g., tire stiffness) is represented with a bond which have an element symbol on one end and with another it is connected to the 1-junction. A model of the car suspension from Fg. 1 build from Modelica Standard Library's components is shown in Fg. 3. Because bond-graph and Modelica's object diagram preserve system topology, there are some analogies between them. A connection node in Modelica, when two or more connectors are connected together), is equivalent to 0-junction in bond-graph representation

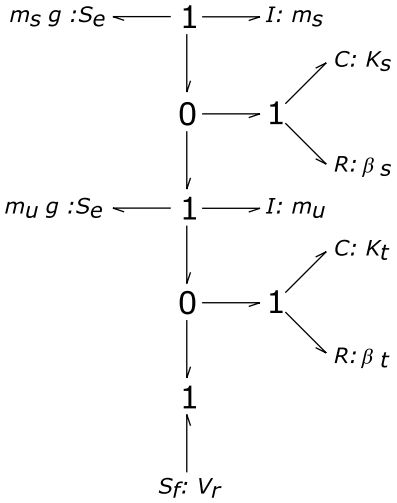

Figure 2: Bond graph of a car suspension

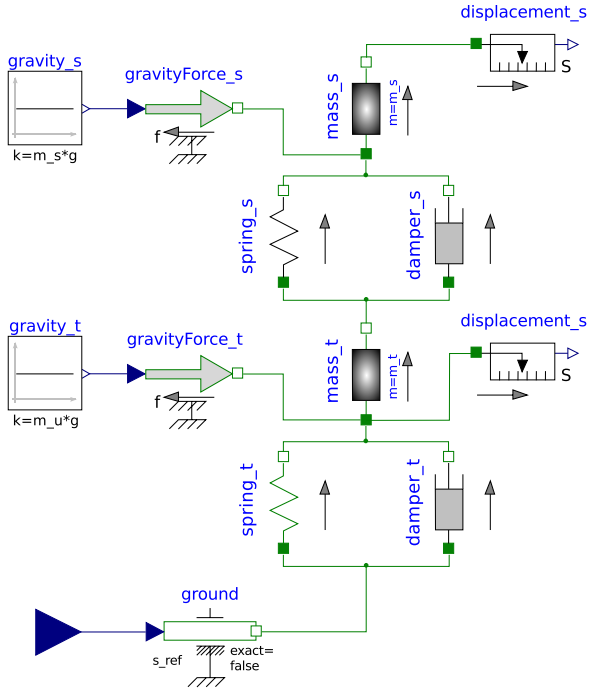

Figure 3: Car-suspension model represented by a Modelica object diagram.

– the effort variables of connected connectors are the same and the flow variables sum to zero. According to the analogy, each Modelica's component in object diagram should be analogous to 1-junction – effort variables of the component's connectors should sum to zero. However, this is not the case in Fg. 3 where effort variable defined in connectors is absolute position. Nevertheless, energy conservation law implies that energy flow of all component's connectors and change of energy stored, added or removed by component must sum to zero. Therefore, the energy flow which corresponds to energy flow of bond representing the element can be in Modelica object diagrams calculated as sum of energy flows of the component's connectors. For example, activity A of a component, weighting factor used in metrics proposed by [5], is thus calculated for Modelica components as following:

$$A = \int_{t_1}^{t_2} |\sum_{i=0}^{N-1} -p_i(t)| \cdot dt \qquad (1)$$

In Eq. 1, $p_i(t)$ designates power flow into i-th connector, N is the number of connectors in component and $[t_1,t_2]$ is the time window of observation.

4.2 Model instrumentation

In order to assure that all the necessary data for selected (energy-based) metrics evaluation are provided in the simulation results, model must be instrumented, i.e., additional equations must be inserted into the model.

It is possible to insert equations for weighting factors (e.g., Eq. 1) directly. However, this introduces many new equations and states into the simulation model and can have a very negative impact on simulation's duration and also on numerical stability. This can be problematic especially with large models, where the use of model approximation methods is the most sensible.

Therefore we decided to use the least instrumentation possible and do most of calculations of weighting factors as a post-processing of simulation results. Model instrumentation was implemented in OpenModelica's shell [8]. Before model can be simulated, it must be loaded into the environment together with all the libraries it requires. Upon loading, abstract syntax tree (AST) of the model is generated and saved into the environment. So, instrumentation was implemented as a separate function which traverses the AST in the environment and for each connection encountered inserts an equation for energy-flow calculation. What kind of equation needs to be inserted is determined by inspecting the type of connector used in the connection equation. For this purpose, a special library of components is provided to the instrumentation function. Each component of the library have as an annotation provided a fully-qualified path to the connector-type definition of which equation for an energy-flow calculation provides. If there is no corresponding component found in the library for the connection's connector-type, that connection is skipped.

Besides instrumentation of the model, connection graphs for each hierarchical level of the models are

added to the environment.

After a model is instrumented, it can be simulated in the usual way (with command *simulate()*).

4.3 Ranking and presentation of results

Ranking of components is performed as post-processing of simulation results. This enables possibility of switching ranking metrics without repeating (possibly time-consuming) instrumentation and simulation. Furthermore, ranking of components which are not of current interest can be avoided.

In our current implementation, activity-metrics (Eq. 1) is used for ranking. Each hierarchical level is considered separately. To determine activity of the component, a connection graph of the object-diagram on given hierarchical level is taken from environment (where it was put by instrumentation function) and energy flows of component's connections are extracted from it. The absolute sum of connection's energy flows (as determined by Eq. 1) are numerically integrated by a quadrature form. However, because integration is done as post-processing, there is a significant loss of accuracy. Quadrature formulas have a much higher truncation-error then solvers used for simulation of the model. Furthermore, such integration is affected by the chosen communication interval. Nevertheless, because accuracy of weighting factors is not of critical importance, use of quadrature formulas suffice in most cases.

The results of ranking are currently provided only in printed form (in a tableau), because there was no graphical interface suitable for adaptation available. Simplification of the model based on obtained ranking is not implemented yet.

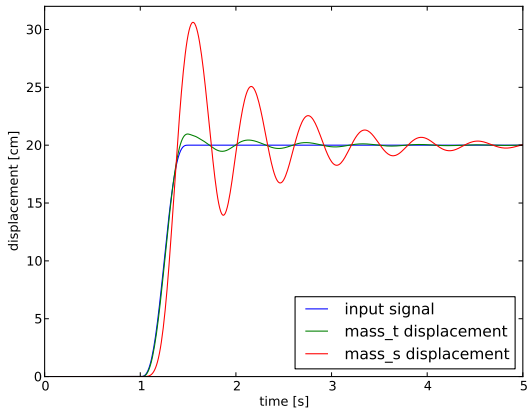

Figure 4: A car hits a smooth curb: low-frequency excitation signal is given as an input to model on Fg.3. Also a response – displacement of a unsprung (*mass_t*) and sprung mass (*mass_s*) is depicted.

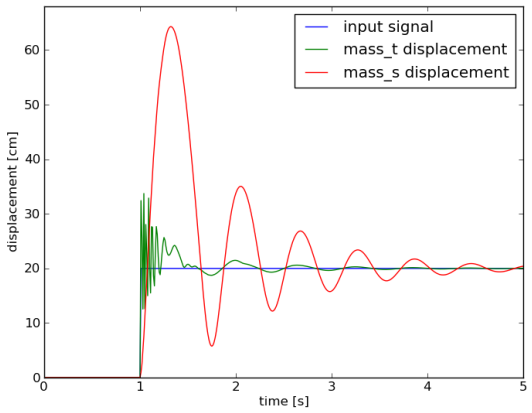

Figure 5: A car hits a sharp curb: step signal is given as an input to model on Fg.3. Also a response – displacement of a unsprung (*mass_t*) and sprung mass (*mass_s*) is depicted.

Element	activity [J]	relative [%]	accumulated [%]
gravityForce_s	2,270.06	37.06	37.06
spring_s	1,763.33	28.79	65.85
ground	795.02	12.98	78.82
mass_s	787.65	12.86	91.68
damper_s	198.82	3.25	94.93
spring_t	192.57	3.14	98.07
gravityForce_t	92.98	1.52	99.59
mass_t	24.53	0.40	99.99
damper_t	0.53	0.01	100.00
displacement_s	0.00	0.00	100.00
displacement_t	0.00	0.00	100.00

Table 1: Ranking of components when model from Fg. 3 is given input shown in Fg. 4.

Element	activity [J]	relative [%]	accumulated [%]
mass_t	528,914.61	43.25	43.25
spring_t	481,340.66	39.36	82.61
damper_s	115,233.47	9.42	92.03
spring_s	49,128.08	4.02	96.05
damper_t	32,039.56	2.62	98.67
mass_s	9,124.75	0.75	99.42
gravityForce_s	5,916.76	0.48	99.90
gravityForce_t	1,196.47	0.10	100.00
ground	0.00	0.00	100.00
displacement_s	0.00	0.00	100.00
displacement_t	0.00	0.00	100.00

Table 2: Response of components when model from Fg. 3 is given a step signal as input.

4.4 Example

A model from Fg. 3 is excited by two different signals, depicted in Fg. 4 and Fg. 5 respectively. For each experiment, components of the model are ranked with activity metrics (Fg. 1) and results are shown in Table 1 and 2 respectively.

As it can be seen, both rankings are very different, but so are the excitation signals. In the first example, the highest ranked components belong to the part of the model with slow dynamics, while in the second example is the part with faster dynamics much more excited and therefore also highly ranked. However, in the second example, simulation's communication interval is too large and thus there is a large error in weighting factors estimation.

5 Conclusion

Presentation of a model to user is an important aspect of modeling environments that helps with model understanding and maintenance. However, many Modelica's language features (e.g., inheritance) are important for effective implementation and to prevent code duplication, but may worsen the clarity of the model implementation. Furthermore, detailed models of complex systems are as hard to understand as reality. Therefore, we believe that there should be integrated a tool into the modeling environment which would help users, non-modeling specialist, to better understanding the model and provide effective means for explaining the model behavior and model verification (and debugging). As it proposed in the paper, model order reduction and simplification techniques (e.g., ranking metrics used by these techniques) can be used for this purpose. It is important that the results are presented in the same form as the original model. Modelica's models are represented graphically, by object diagrams, or as a set of acausal differential-algebraic equations. Therefore, model order reduction and simplification techniques for both representation must be used. There are already many methods for simplifying (ranking) models represented with DAE system. We also showed that method for simplifying bond graphs (graphical modeling formalism) can be adapted to work with Modelica's object diagrams.

References

[1] J. F. Broenik. Introduction to physical systems modeling with bond graphs. In *SiE whitebook on Simulation Methodologies*, pages 1–31, 1999.

[2] Samuel Y. Chang, Christopher R. Carlson Carlson, and J. Christian Gerdes. A lyapunov function approach to energy based model reduction. In *Proceedings of the ASME Dynamic Systems and Control Division – 2001 IMECE*, pages 363–370, New York, USA, 2001.

[3] Modeling Chemical Reactions in Modelica By Use of Chemo-bonds. Cellier, f. e. and greifeneder, j. In *Proceedings of the 7th Modelica Conference*, pages 142–150, Como, Italy, 2009.

[4] Sanjay Lall, Petr Krysl, et al. Structure-preserving model reduction for mechanical systems. *Physica D*, 284:304–318, 2003.

[5] Loucas Sotiri Louca. *An Energy-based Model Reduction Methodology for Automated Modeling*. PhD thesis, University of Michigan, 1998.

[6] Modelica Association. *Modelica Specification, version 3.2*, 2010. `http://www.modelica.org/documents/ModelicaSpec32.pdf`.

[7] Modelica Association. *Modelica Standard Library 3.1, User's Guide*, 2010. `https://www.modelica.org/libraries/Modelica`.

[8] Open Source Modelica Consortium. Openmodelica. `http://www.openmodelica.org`.

[9] R. Rosenberg and T. Zhou. Power-based model insight. In *Proceedings of the ASME WAM Symposium on Automated Modeling for Design*, pages 61–67, New York, USA, 1988.

[10] P. Schwarz et al. A tool-box approach to computer-aided generation of reduced-order models. In *Proceedings EUROSIM 2007*, Ljubljana, Slovenia, 2007.

[11] R. Sommer, T. Halfmann, and J. Broz. Automated behavioral modeling and analytical model-order reduction by application of symbolic circuit analysis for multi-physical systems. In *Proceedings EUROSIM 2007*, Ljubljana, Slovenia, 2007.

[12] Ralf Sommer, Thomas Halfmann, and Jochen Broz. Automated behavioral modeling and analytical model-order reduction by application of symbolic circuit analysis for multi-physical systems. *Simulation Modelling Practice and Theory*, 16:1024–1039, 2008.

[13] Hubertus Tummescheit. *Design and Implementation of Object-Oriented Model Libraries using Modelica*. PhD thesis, Lund Institute of Technology, 2002.

[14] T. Wichmann et al. On the simplification of nonlinear dae systems in analog circuit design. In *Proceedings of CASC'99*, pages 485–498, Munich, Germany, 1999.

[15] Y. Ye and K. Youcef-Youmi. Model reduction in the physical domain. In *Proceedings of the American Control Conference*, pages 4486–4490, San Diego, CA, USA, 1999.