A Planar Mechanical Library for Teaching Modelica

Dirk Zimmer
Deutsches Zentrum für Luft- und Raumfahrt (DLR)
Münchner Strasse 20, 82234 Weßling, Germany
dirk.zimmer@dlr.de

Abstract

Teaching Modelica to students of a university requires suitable example models. This paper describes a planar mechanical library that is primarily conceived for didactical purposes. It is simple, built out of a few components only, but it enables the modeling of interesting and complex systems. The library is freely available and supported by various Modelica environments.

Keywords:Education;Planar Mechanics;

1 Introduction

1.1 Motivation

This paper presents a planar mechanical library that has been primarily designed for didactical purposes. The idea of such a library is that it is simple and easy to understand. In this way, the students can focus on learning the principles of equation-based modeling and they can avoid the lot of peculiar particularities that have meanwhile become part of the language.

We have used this library in the Modelica course at the technical university in Munich [8]. The course is enlisted in the computer science department. The students of this class mostly study computer science, applied mathematics or physics. Computer science students in Munich do not have any physics course in their basic curriculum. Hence, explaining the modeling of physical systems requires explaining the physics as well, in this particular case: the fundamental laws of motion.

In planar mechanical systems, we describe the physics of a multi body system in a two-dimensional plane. Each body position can be described by the coordinates x and y and its orientation by the angle φ (see Figure 1). Each body has a mass and its inertia can be described by a single scalar.

Planar models of mechanical systems are useful for a number of applications. Very popular is their use for contact problems that are a lot simpler in 2D than in 3D. The modeling of gear wheel interaction is one such example [5]. For this paper their use in teaching is of course the main issue.

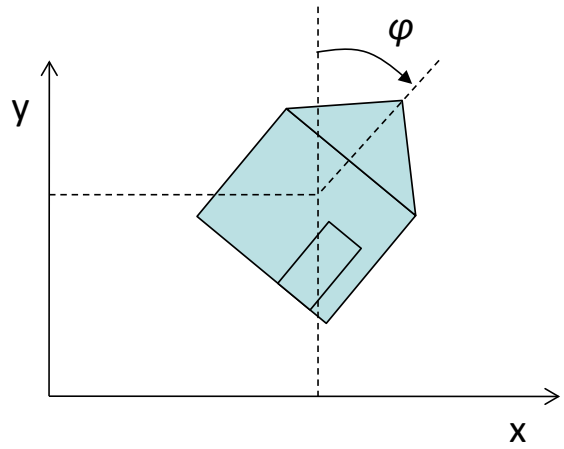

Figure 1: Representation of an object in planar space

1.2 Suitability of planar mechanics

Planar mechanical systems are ideally suited for teaching equation-based modeling, because their components are easy to model and to understand but the resulting systems are often complex in behavior and demanding in their computational aspects. Or to put it in short terms: you can do a lot of cool stuff by simple means.

From the modeling side, planar mechanics offers the following advantages:

- Planar mechanical systems are tangible and visual systems. All students have played with mechanical systems before in their life and everyone has an intuitive (and sometimes wrong) understanding about their motion. This motion can be visualized in an animation, which is more appealing to students than studying plots.
- The physical laws of planar mechanical systems are basically taught already in high-school. D'Alemberst Principle and Newton's Law look familiar to the students. The equations of motion themselves are relatively easy.

- Planar mechanical systems can be steered either by human interaction or by a control law. Again these tasks are very tangible and concrete: everyone has steered a bicycle and everyone has tried to balance a pen in his life.

The resulting system can then be used to demonstrate and study the advantages and difficulties of equation-based modeling.

- First of all, mechanical systems require true non-causal equation-based modeling. Modeling methods that are based on the computational flow such as Simulink are of very limited use in this domain. A kinematic loop can be used as an illustration.
- Also kinematic loops require the solution of non-linear equation systems. The corresponding examples can be used to explain techniques for initialization and state selection.

In contrast to planar mechanical systems, 1D and 3D mechanical system are not so well suited for teaching.

1D mechanical systems are too simple. Of course, We teach both rotational and translational mechanic prior to planar system, but many interesting configurations such as kinematic loops do not naturally exist in 1D. Hence, the topic does not bear long and quickly gets boring unless you enter the specifics of drive-train modeling which is misplaced in a general Modelica course.

3D mechanical systems on the other side are way too complex. A short look on the components of the standard MultiBody Library [2,7] makes this clear. In 3D, the description of a body orientation can be performed in many different and potentially redundant ways. This redundancy then leads to further difficulties so that kinematic loops require special treatment. In planar mechanics, the orientation is uniquely described by a single angle and kinematic loops do not require special modeling tools.

2 State of the Art in planar mechanical modeling

The library presented in this paper is not the first planar mechanical library that has been developed in Modelica.

Indeed, we have developed one of the first variants as part of the MultiBondLib [7]. It is freely available and it is also well suited for teaching but only in a course where bondgraphic modeling is part of the program. In contrast, the new library is directly based on equations and does not require the knowledge of bondgraphs. Furthermore, because of the use of bondgraphs in the MultiBondLib the connectors contained redundant information and kinematic loops required special handling.

A second planar library has been developed by Höbinger and Otter [4]. In addition to the basic mechanical components (joints and body parts), the library contained models for the contact of curved surfaces. Although, it was envisioned that this library becomes part of the Modelica Standard Library (MSL), this has not yet taken place.

Furthermore new planar mechanical elements have developed by van der Linden [5] for the modeling of gearwheels. This developments use the same interfaces and components as the planar mechanical library presented here.

2.1 Contributions of this Paper

Since already a significant amount of effort has been spent on the development of Modelica code for planar mechanics, it is important to clarify the contribution of this paper. Essentially there are three major objectives for this work:

- **Presentation of a didactical library:** This is the major part of this paper (section 3 to 5). I will present the interfaces and the structure of the library and show how simple the individual components can be modeled.

- **Cross-Platform Library for different compilers:** The ability to compose complex systems out of simple components using only a smaller subset of the language is not only interesting for students but also for compiler developers. The library turns out to be very well suited for testing the abilities of various Modelica environments. Also for teaching purposes, it is good if the material is not bounded to a certain software tool but of general applicability. More on this topic in section 6

- **Establishment of a standard interface for planar mechanics:** The planar mechanical library for didactical purposes is not supposed to become part of the MSL. Libraries that are part of the MSL must be optimized with respect to usability. This in part conflicts with desired level of simplicity for teaching. However, there is no reason why a potential library for planar mechanics in the MSL and the didactical library should use different interfaces.

DOI 10.3384/ecp12076681

3 Structure of the library

The interface of a planar mechanical component represents a flange point. This point is determined by a fixed position the plane (x,y) and a fixed orientation angle (phi). Forces in x and y direction (fx, fy) as well as a torque (t) may act on the flange point. The corresponding Modelica connector is hence designed as follows:

Listing 1: Connector code

```
connector Frame
"General Connector for planar mechanical components"
  SI.Position x "x-position";
  SI.Position y "y-position";
  SI.Angle phi "angle (counter-clockwise)";
  flow SI.Force fx "force in x-direction";
  flow SI.Force fy "force in y-direction";
  flow SI.Torque t "torque (clockwise)";
end Frame;
```

For simplicity, the potential use of vectors in the connector has been omitted. For beginners it is a little easier, to work with x,y, and phi than with a vector r[2] and phi. The same holds for the forces. Given this connector, a variety of planar mechanical components can be implemented. Figure 2 provides an overview of the library content.

The standard components are parts and joints. These elements were designed in strong resemblance to their counterparts in the Modelica MultiBody library. In addition to the standard components, the library contains sub-packages for vehicle wheels and gearwheels.

The wheel models can be used to move with a wheel on the x,y-plane. There are ideal wheel models and simple slip based models inspired by previous works [6].

Future versions of this library may also contain the gear wheel models out of the work of van der Linden [5]. They can for instance be used to assemble a planetary gear box.

All elements in this library contain a suitable visual representation for the animation. For simplicity though, the animation is not as configurable as in the MultiBody library. Another difference to the MultiBody library is that there is no World model available in this library. Again the sheer simplicity is preferred over a more elaborate solution.

The library features a large set of examples that demonstrate the variety of systems that can be assembled from these components: pendulum, crane crab, kinematic loops, or even two-track car vehicle models are included. Also examples of controlled systems and model inversion are contained in this library.

The library itself is available at [8] or at the Modelica Website. This is made publicly available and represents the standard version. The examples in this version are all suitable for testing purposes. Furthermore this library is self-contained only requiring a few elements of the standard library but not requiring any other library.

The planar mechanical library that is being used in the lecture course is slightly different. First of all it is developed in several steps as the course proceeds. In its latter stages, it also contains elements from DLR libraries. The lecture course contains also slides explaining the components of this library at great level of detail.

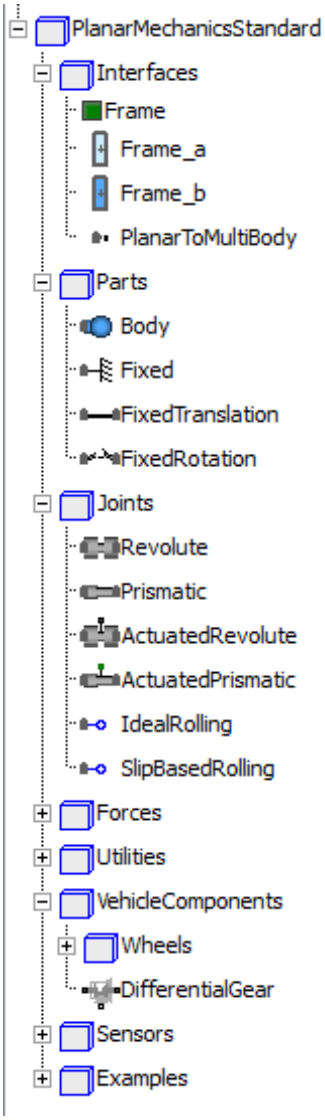

Figure 2: Structure of the planar mechanical library

4 Teaching Modelica

4.1 Context

When the library is used for teaching, it is not presented as a whole but gradually developed together with the students. The goal is that the students learn all relevant processes of modeling in Modelica: from punching in equations, plugging together components to designing a whole library.

In the course "Virtual Physics", the library is being used from lesson 5 on. In the first 4 lessons, the students learn the basics of equation-based modeling and the Modelica language. After going through examples of 1D mechanical systems, we start by the most basic mechanic components.

4.2 Component Modeling

The most important component is of course the body component:

Listing 2: Body component

```
model Body "Body component with mass and inertia"

  Interfaces.Frame_a frame_a;

  parameter SI.Mass m "mass of the body";
  parameter SI.Inertia I "Inertia of the Body";
  parameter SI.Acceleration gx =0
    "gravity acceleration (in x) acting on the mass";
  parameter SI.Acceleration gy=-9.81
    "gravity acceleration(in y) acting on the mass";

  SI.Velocity vx "velocity in x";
  SI.Velocity vy "velocity in y";
  SI.AngularVelocity w "angular velocity";
  SI.Acceleration ax "acceleration in x";
  SI.Acceleration ax "acceleration in y";
  SI.AngularAcceleration z "angular acceleration";

equation
   //The velocity is a time-derivative of the position
   vx = der(frame_a.x);
   vy = der(frame_a.y);
   w = der(frame_a.phi);

   //The acceleration is a time-derivative of the velocity
   ax = der(vx);
   ay = der(vy);
   z = der(w);

   //Newton's law
   fx + m*gx = m*ax;
   fy + m*gy = m*ay;
   frame_a.t = I*z;

end Body;
```

Even with plenty of comments the code remains compact and is very easy to understand. For the first version, everything that may distract the student has been removed. Gravity acceleration is a simple parameter and does not be read out of a strange "world model". There is no animation and there are no options for initialization or state-selection that pollute the code. Just the bare physical equations form the model.

In this version, also no vector notation is used. For students of a technical university it seems to cause no problems in understanding the model code. Teaching experience from universities of applied sciences indicates that vector notation is better introduced later on. Vector notation is used in a subsequent version, where also the code of the animation is added. The students know at this stage that this code is non-essential.

For joint elements, a neutral element is a good starting point. This element implements the lever principle but exhibits no forces on its connectors.

Listing 3: Neutral component

```
model Neutral
   //This component has two frames…
  Interfaces.Frame_a frame_a;
  Interfaces.Frame_a frame_b;

equation

   //…but exhibits no effect.
  frame_a.fx = 0;
  frame_a.fy = 0;
  frame_a.t = 0;

   //This is the balance of force and torque
     including the lever principle
  frame_a.fx + frame_b.fx = 0;
  frame_a.fy + frame_b.fy = 0;
  frame_a.t
  + frame_b.t
  + (frame_b.x - frame_a.x)*frame_b.fy
  - (frame_b.y – frame_a.y)*frame_b.fx
  = 0;

end Neutral
```

Any joint can now be implemented by replacing the assignment of zero force with the corresponding positional constraints. Furthermore, the lever principle can often be simplified. Let us for instance look at the revolute joint. Here, two positional constraints are enforced: the position must be equal in direction of x and y. Since there is no distance between the two frames, the lever principle degenerates to a balance of torque.

DOI 10.3384/ecp12076681

Listing 4: Revolute joint, first version

```
model Revolute
  Interfaces.Frame_a frame_a;
  Interfaces.Frame_a frame_b;

equation

   //frame_a.fx = 0 gets  replaced by
  frame_a.x = frame_b.x;

   //frame_a.fy = 0 gets replaced by
  frame_a.y = frame_b.y;

  frame_a.t = 0;

   //since there is no difference in position
     the lever principle can be simplified
  frame_a.fx + frame_b.fx = 0;
  frame_a.fy + frame_b.fy = 0;
  frame_a.t  + frame_b.t = 0;


end Revolute;
```

In a second version, two differential equations and one algebraic equation are added since the joint is well suited to describe the motion of the system.

Listing 5: Revolute joint, second version

```
model Revolute
  Interfaces.Frame_a frame_a;
  Interfaces.Frame_a frame_b;

//These 3 variables help to describe the motion of a system
  SI.Angle phi
  SI.AngularVelocity w;
  SI.AngularAcceleration z;

equation

//For 3 more variables we need 3 more equations:
  frame_a.phi + phi = frame_b.phi;
  w = der(phi);
  z = der(w);

//Known material…
  frame_a.x = frame_b.x;
  frame_a.y = frame_b.y;
  frame_a.t = 0;

  frame_a.fx + frame_b.fx = 0;
  frame_a.fy + frame_b.fy = 0;
  frame_a.t  + frame_b.t = 0;

end Revolute;
```

In this way, also a fixed translation element can be explained. The prismatic joint can then be presented as a translational element of variable length.

4.3 Valuable Examples for Teaching

Having available only five component models for

- body with mass and inertia,
- revolute joint,
- prismatic joint,
- fixed translation,
- and global fixation

enables us to compose already a lot of interesting models.

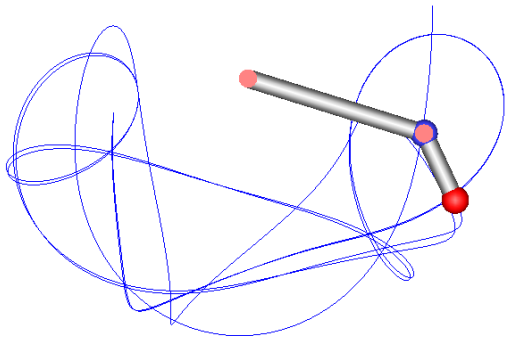

Figure 3: Chaotic trajectory of a double pendulum

The famous double pendulum can be used to demonstrate chaotic system behavior. Figure 3 shows the erratic trajectory of the peak of the pendulum. Simulating with different values for precision yields each time a completely new trajectory and no convergence can be reached. The students learn the important lesson that a simple non-linearity can lead to totally unpredictable and chaotic systems.

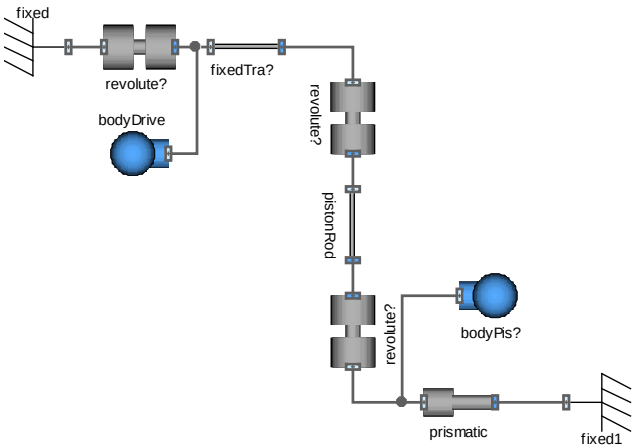

Figure 4: Model diagram of a simple piston engine

Figure 4 displays the model diagram of a piston engine. It represents a kinematic loop: although there are four joint elements, the complete system has just one degree of freedom. This example is used to ex-

plain the mechanism of initialization and state selection to the students. The joint elements are then further enhanced by an initialization section and attributes for state selection. Furthermore, the students learn about the Pantelides algorithm for reducing the differential index of a system.

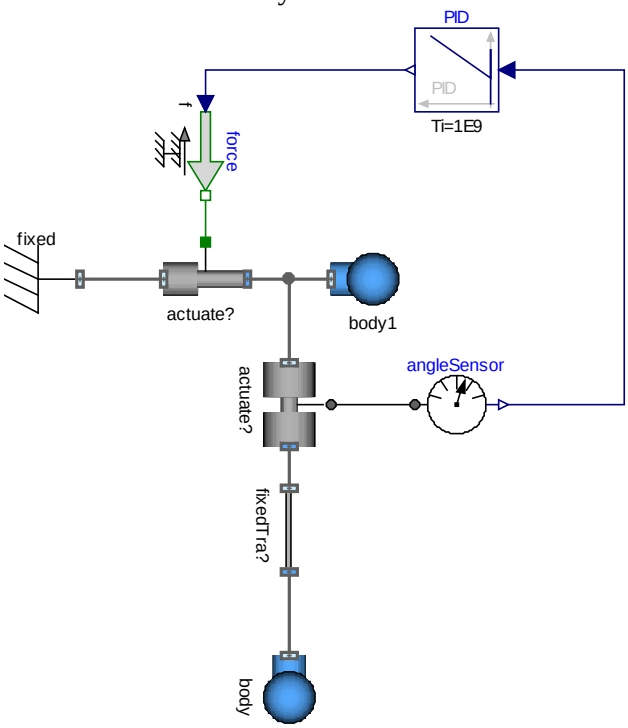

Figure 5: Model diagram of an inverted pendulum controlled by a PID element

The inverted pendulum is a famous example in control theory. It is easy to model by using the planar mechanical components. A simple PID controller can be added to show how a controller can be designed in Modelica. Furthermore it is possible to invert the model by stipulating the trajectory and computing the forces. In this way, the students can learn how flexible a Modelica model can be used: not only for simulation but also for control design and model inversion.

5 Tire and vehicle models

Whereas the standard components already enable the creation of many interesting examples, planar mechanical systems can also be used to model vehicles driving on the plane. To this end three separate wheel models are provided:

- An ideal rolling wheel
- A dry-friction based wheel
- A slip-based wheel

Listing 6 presents the code for the ideal rolling wheel. Although being already significantly more complex, this component is not beyond what a good student can learn to understand if he is supported by sufficient explanations and further material.

Listing 6: Ideal wheel

```
model IdealWheelJoint

  Interfaces.Frame_a frame_a;
  Rotational.Interfaces.Flange_a flange_a;

  parameter SI.Length radius
    "radius of the wheel";
  parameter SI.Length r[2]
    "driving direction of the wheel at angle phi = 0";
  final parameter SI.Length l = sqrt(r*r);
  final parameter Real e[2] =  r/l
    "normalized driving direction";

  Real e0[2]"normalized direction w.r.t inertial system";
  Real R[2,2] "Rotation Matrix";

  SI.AngularVelocity w_roll "roll velocity ";
  SI.Velocity v[2] "transl. velocity";
  SI.Velocity v_long "velocity in longit. direction";
  SI.Acceleration a "accel. of driving velocity";
  SI.Force f_long "longitudinal force";

equation

  //Resolve the normalized driving direction in the
    inertial coordinate system
  R={{cos(frame_a.phi),-sin(frame_a.phi)},
     {sin(frame_a.phi),cos(frame_a.phi)}};
  e0 = R*e;

  //Project the longitudinal velocity in the planar space
   (this implyies that the lateral velocity is zero)
  v = der({frame_a.x,frame_a.y});
  v = v_long*e0;

  //Implement the law of ideal rolling
  w_roll = der(flange_a.phi);
  v_long = radius*w_roll;
  a = der(v_long);

  //Project the force on the longitudinal direction
  {frame_a.fx, frame_a.fy}*e0 = f_long;

  //model the drive torque
  -f_long*radius = flange_a.tau;

  //There is no bore torque
  frame_a.t = 0;

end IdealWheelJoint;
```


DOI 10.3384/ecp12076681

The code for the other two wheel models is only a little more complex. The students have to learn about friction characteristics and regularization techniques. Given these wheel models, a simple one-track car model can be composed in five minutes:

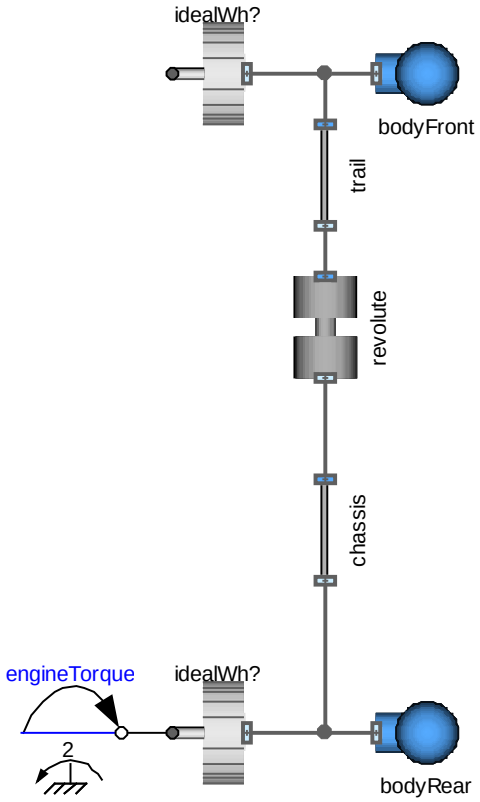

Figure 6: Model diagram of a simple one track vehicle

Such a model is sufficient to study the influence of the trail or the basic difference between front-wheel drive and rear-wheel drive.

The highlight of the course is a two-track car model with slip-based wheels. It is enhanced by a simple 3D chassis that computes the load balance on the four wheels. The car model can be simulated in real-time. It is also visualized in real time by the use of the SimVis Library [1] (see Figure 7) and can be controlled online by the keyboard using components from the Modelica Device Drivers library [3]. As a result, the students can drive their own car model in 3D just as in a computer game. Such an example attracts many students to the course and helps to keep up their motivation during the course.

6 Cross-platform compatibility

Since the library uses only a subset of the Modelica language that consists entirely out of well-established language constructs, it can be supported by a large set of different Modelica compilers already now. 15 examples have been selected for testing the results of various Modelica simulation environments. The current test results are summarized in figure 8. It shows the test results for all 17 examples and for for different compilers.

First of all, Dymola[9] offers full support of the library. It is also the environment that has been used for the development of the library and that I use for teaching.

JModelica[11] is also able to parse and process the entire library. It does not offer dynamic state-selection as in Dymola but this feature is not so essential for a didactical library.

OpenModelica[10] can also parse the entire library. The correct translation and simulation is possible for large set of examples but not for all of them. In some more complex examples, the back-end of the compiler still has some problems with the non-holonomic constraints equations that originate from ideal rolling parts.

Also SimulationX[12] offers almost full support of the library. Some examples require a non-standard solver but these are this was the only small problem that occurred. For one example of a kinematic loop, SimulationX started with the wrong initial position but this might be due to modeling ambiguity.

In all cases the compiler developers are working on the occurring problems and there is a fair chance that a complete support of the library can be realized soon.

Test of MapleSim[13] have not yet been completed. First results indicate that MapleSim parses the code correctly and that the simulator is capable of simulating the test cases. The current problems concern the usability of the models but these problems should be solved for the new version of MapleSim.

Tests within Wolfram SystemModeler [14] have not yet been done.

Figure 7: 3D-Realtime visualization of the two track vehicle

	Name	Dymola	Open Modelica	JModelica	SimulationX
1	FreeBody	OK	OK	OK	OK
2	Pendulum	OK	OK	OK	OK
3	DoublePendulum	OK	OK	OK	OK
4	CounterSpin	OK	OK	OK	OK
5	CraneCrab	OK	OK	OK	OK
6	CraneCrabControlled	OK	OK	OK	OK
7	InvertedCraneCrab	OK	OK	OK	OK
8	WheelBasedCraneCrab	OK	OK	OK	OK
9	PistonEngine	OK	OK	OK	OK
10	KinematicLoop	OK	F?	OK	F?
11	TestIdealWheel	OK	F	OK	OK
12	TestDryFrictionWheel	OK	OK	OK	OK
13	TestSlipBasedWheel	OK	OK	OK	OK
14	SingleTrack	OK	F	OK	OK
15	TwoTrack	OK	OK	OK	OK

Categories		
	OK	Runs succesfully
	F	Fails

Figure 8: This table displays the current support of the library among different Modelica environments

7 Conclusions

Ultimately, the goal is to have a didactical library available that can be used to teach Modelica in different modeling and simulation environments.

I personally hope that this library helps other lecturers to create their Modelica courses. It can be used for free under the Modelica 2 license. Suggestion (or even better: contributions) that help to improve the quality of the library are always highly welcome.

Acknowledgements

I would like to acknowledge the effort of Thomas Schmitt and Markus Andres who gathered further teaching experience using planar mechanical models at the University of Applied Sciences Vorarlberg.

I would like to thank Franciscus van der Linden for his contribution to the standardization of the interface and the gear wheel models in the library.

Many thanks to Francesco Casella for having the idea to use this library as test-case for different compilers. Johan Åkesson, Jens Frenkel and Adrian Pop helped in testing the library on OpenModelica and JModelica.

Thanks for Ingrid Bausch-Gall and Jakob Tobolar for the help with SimulationX and thanks to Matthias Reiner for a first investigation in MapleSim.

References

[1] Bellmann, Tobias (2009) Interactive Simulations and advanced Visualization with Modelica. Proc. of the 7th International Modelica Conference, 20.-22. Sept. 2009 , Como, Italy

[2] Cellier, F.E. and D. Zimmer (2006), *Wrapping Multi-bond Graphs: A Structured Approach to Modeling Complex Multi-body Dynamics*, keynote presentation, 20th European Conference on Modeling and Simulation, Bonn, Germany, May 29-31, 2006

[3] Elmquist, H., et. al. (2009) Modelica for embedded systems. Proc. of the 7th International Modelica Conference, 20.-22. Sept. 2009 , Como, Italy

[4] Höbinger, M. and M. Otter (2008), *Planar-MultiBody - A Modelica Library for Planar Multi-Body Systems*. Proc. 6th International Modelica Conference, Bielefeld, Germany

[5] van der Linden, F. (2012), *Modelling of Elastic Gearboxes Using a Generalized Gear Contact Model*. In review for the Proc. of the 9th Modelica Conference, Munich, Germany

[6] Zimmer, D. and M. Otter (2010), Real-Time Models for Wheels and Tires in an Object-Oriented Modelling Framework. *Journal of Vehicle System Dynamics*. Volume 48, Issue 2, pp. 189-216 .

DOI 10.3384/ecp12076681

[7] Zimmer, D. and F.E. Cellier (2007), The Modelica Multi-bond Graph Library, *Simulation News Europe*, Volume 17, No. 3/4, pp. 5-13.

[8] Zimmer D. *Virtual Physics*. Lecture Notes available at: www.robotic.dlr.de/dirk.zimmer

Tool References

[9] Dymola: www.3ds.com/products/catia/portfolio/

[10] OpenModelica: www.openmodelica.org

[11] JModelica: www.jmodelica.org

[12] SimulationX: www.itisim.com

[13] MapleSim: www.maplesoft.com/products/maplesim

[14] Wolfram SystemModeler: http://www.wolfram.com/system-modeler/

DOI
10.3384/ecp12076681