

**A Design, Simulation and Visualization Environment for Object-Oriented
Mechanical and Multi-Domain Models in Modelica**

Vadim Engelson, Håkan Larsson, Peter Fritzson

2. Modelica Language

2.1. Simple Electric Circuit

As an introduction to Modelica we will present a model of a simple electrical circuit. Our goal is to describe features of universal Modelica standard notation, which can be used in applications in various domains (such as electrical, mechanical or chemical). A detailed description of this example can be found in [5, 10]. The system can be broken into a set of connected electrical standard components.

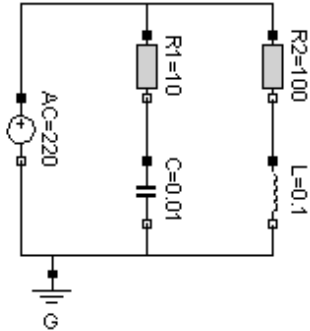


Figure 2. Sample circuit structure in Modelica graphical notation.

Assume that the sample model (Figure 2) consists of a voltage source, two resistors, an inductor, a capacitor and a ground point. Models of such components are available in Modelica standard class libraries for electrical components.

A declaration like the one below specifies R1 to be an instance (i.e. an object) of standard library class Resistor and sets the default value of the resistance, R, to 10 (i.e. R1.R is 10).

```
Resistor R1(R=10);
```

A Modelica description of the complete circuit appears as follows:

```
class circuit
  Resistor R1(R=10);
  Capacitor C(C=0.01);
  Resistor R2(R=100);
  Inductor L(L=0.1);
  VsourceAC AC;
  Ground G;
equation
connect(AC.p,R1.p); connect(R1.n,C.p);
connect(C.n,AC.n); connect(R1.p,R2.p);
connect(R2.n,L.p); connect(L.n,C.n);
connect(AC.n,G.p);
end circuit;
```

2.3. Mechanical System Modeling in Modelica

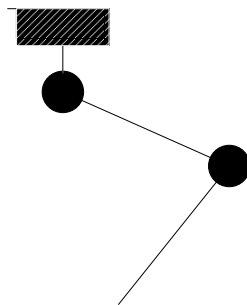
To facilitate mechanical system modeling there exists a standard Modelica class library for modeling multi body mechanical systems (MBS) [9, 13], i.e., systems of rigid bodies connected to each other with certain degrees of freedom.

A model that uses MBS consists of an inertial system (instance of class `Inertial`), joints (instances of classes `RevoluteS` or `PrismaticS`), massless bars (class `Bar`) and bodies (class `Body`) with mass. The objects are connected together with the Modelica `connect` statement.

The `Inertial` object defines the global coordinate system and the gravitational force. All other objects are in some way connected to this object, either directly or through other objects.

The use of the MBS library can be represented by a double pendulum example (see Figure 3):

```
class Pendulum
  Real L = 0.5;
  Inertial I;
  Body P1(rCM={L/2,0,0});
  Body P2(rCM={L/2,0,0});
  RevoluteS rev1(n={0,0,1});
  RevoluteS rev2(n={0,0,1});
  Bar arm(r={L,0,0});
equation
  connect(I.b, rev1.a);
  connect(rev1.b, P2.a);
  connect(rev1.b, arm.a);
  connect(arm.b, rev2.a);
  connect(rev2.b, P1.a);
end Pendulum;
```



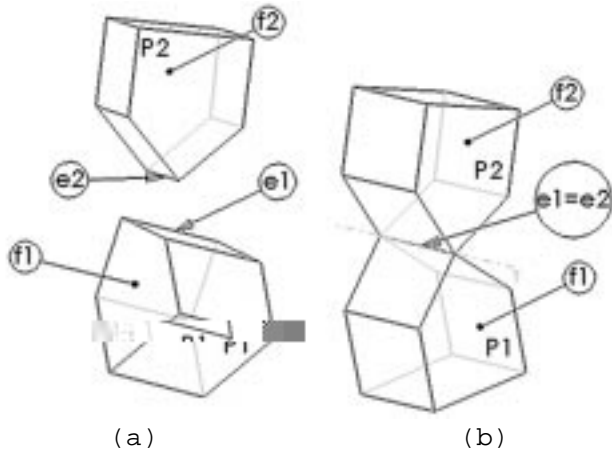


Figure 4. Parts and their mates specification before (a) and after (b) adjustment according to the mates.

3.1. Example

The example (Figure 4(a)) describes a fragment of the pendulum model. The part P1 has front face (f1) and upper edge (e1). The part P2 has the front face (f2) and bottom edge (e2). There is a mate M_1 that specifies that planes of f1 and f2 are coincident. The mate M_2 specifies that the edges e1 and e2 are coincident. The SolidWorks system analyzes the mates and adjusts positions of the parts (Figure 4(b)). The system automatically rejects invalid mate combinations. Our translator [7] finds that there is a joint with one rotational degree of freedom between the parts P1 and P2, and calculates the position and orientation of the rotation axis. This pair of mates corresponds to an instance of class `RevoluteS` from Modelica MBS library with attached `Body` instance.

3.2. Modelica Model

Each SolidWorks assembly consists of a set of parts, and it stores a set of mates. All these are validated and translated to a set of Modelica MBS class instances and appropriate *connections* between them. Mass, position of center of mass and inertia tensor are extracted from the corresponding part documents by SolidWorks. The result of a Modelica model simulation is position, rotation, velocity, acceleration and other physical properties of each `Body` as functions of time during the simulated time period.

4. Translation and simulation

Figure 5 represents components of the environment needed for visualization. Our translator from SolidWorks to

Modelica takes information about the mates and produces a corresponding set of Modelica class instances with connections between them. The mass and inertia tensors for each part are computed by SolidWorks. These are extracted and used in Modelica model. Geometry information is saved in a separate STL [17] file for each part.

By default the gravity force is applied to the mechanical model. Usually this is not enough for simulation. All external forces that are applied to the bodies, as well as motor forces that are applied to revolute and prismatic joints should be specified. This is done outside the SolidWorks model by adding code for new class instances to the Modelica model.

A control subsystem that controls the forces according to a certain plan (mission) can be written in Modelica. If necessary, external code in C can be added to the model.

When a Modelica model is simulated, the position, orientation, velocity and acceleration for each part (`Body` instance) is computed. For Modelica simulation we use the Dymola tool with Modelica support[3].

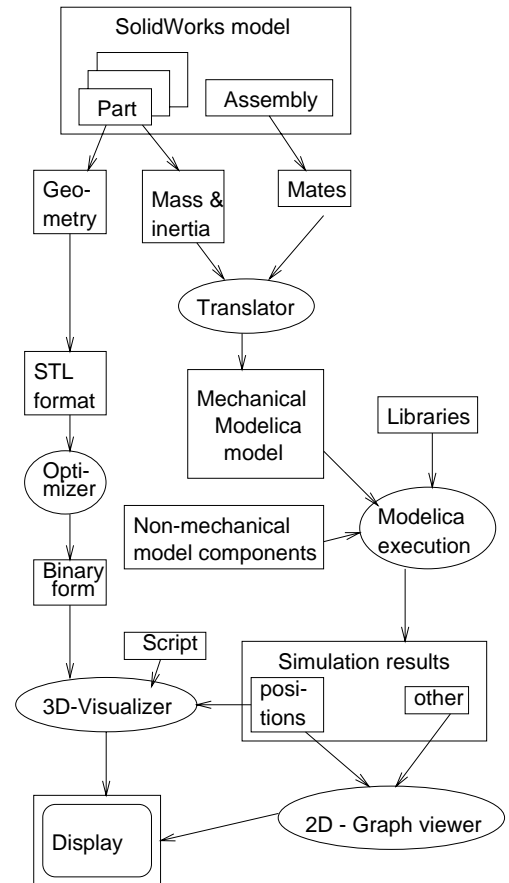


Figure 5. The path from SolidWorks model to dynamic system visualization.

5. Visualization

The integrated environment includes a visualizer that provides online dynamic display of the assembly (during simulation) or offline (based on saved state information for each time step).

The STL format[17] is a very simple format suitable for visualization. All surfaces are divided into triangles and the the coordinates of the triangle vertices, as well as normal vectors of the triangles, are listed in the STL-file.

The visualizer loads the corresponding STL file for each part and optimizes it for rendering. After that, rendering is performed by OpenGL [8] library functions. During optimization the vertices positioned very close are merged together. Optimized STL code is stored in a binary file for future use.

The user of the visualizer can alternatively utilize the pop-up menu system, keyboard shortcuts, or a command

of contact between parts or if a representation of the part geometry on closed form was to be included in the Modelica model.

7.2. True Multidomain Applications

Modelica is suitable for multiple application domains. Components from several domains (mechanical, electrical, hydraulic simulation) can be used within the same Modelica model. For instance, electrical components can be combined with mechanical components in the model. The electrical parts can be written by hand, or designed using a block-oriented editor, or extracted from an electric CAD system. The generated Modelica code for electrical components in combination with a mechanical design tool produces a multidomain Modelica model.