Redundancies in Multibody Systems and Automatic Coupling of CATIA and Modelica

Hilding Elmqvist[1], Sven Erik Mattsson[1], Christophe Chapuis[2]
[1]Dassault Systèmes, Lund, Sweden (Dynasim)
[2]Dassault Systèmes, Velizy Villacoublay, France
{Hilding.Elmqvist, SvenErik.Mattsson, Christophe.Chapuis}@3ds.com

Abstract

Traditionally, multibody systems have been defined in Modelica by connecting bodies and joints in a model diagram. Additionally the user must enter values for parameters defining masses, inertias and three dimensional vectors of positions and orientations. More convenient definition of multibody systems can be made using a 3D editor available, for example, in CATIA from Dassault Systèmes with immediate 3D viewing.

A tool has been developed that translates a CATIA model to Modelica by traversing the internal CATIA structure to get information about parts and joints and how they are related. This information is then used to generate a corresponding Modelica model. The traversal provides information about the reference coordinate system, the center of mass in the local coordinate system, the mass, the inertia, the shape and color of the body exported in VRML format for animation purposes and the icon exported as a PNG file to be used in the Modelica diagrams.

The Modelica diagram layout is automatically generated and is based on the spanning tree structure of the mechanism. Models obtained in this way often contain redundant constraints. A new method has been developed for Dymola to facilitate simulation of such models, i.e. the model reduction is performed automatically.

An important property of the translated model is the possibility to use Modelica extends (inheritance) for adding controllers and other features of the model for dynamic simulation. For instance, the engine model can be extended by introducing models of the gas forces of the combustion acting on the cylindrical joints of the pistons. In that way, the translated model is separated and can be changed independently of the added models.

Keywords: MultiBody systems, Modelica, CATIA

1 Introduction

Traditionally, multibody systems have been defined in Modelica by connecting bodies and joints from the MultiBody library (Otter et.al., 2004) in a model diagram. More convenient definition of multibody systems can be made using a 3D editor, available, for example, in CATIA with immediate 3D viewing. See for example the CAD model with kinematic definition of a four cylinder engine in Figure 1.

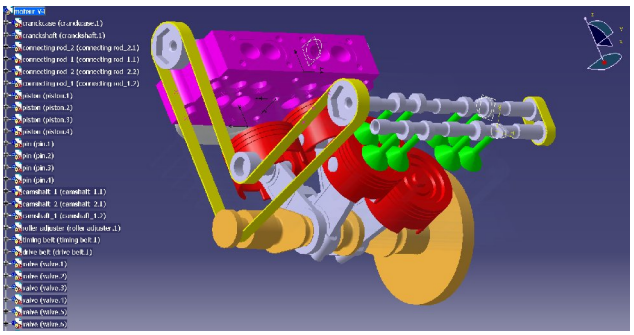


Figure 1. CATIA V5 model of engine

This paper discusses how to *automatically* derive a corresponding Modelica model to enable dynamic simulation of the mechanism. The first part of the paper describes Modelica code generation, layout generation and how a mechanical model can be extended for multi-domain dynamic simulation. The last part discusses how redundancies in the kinematic definition are handled.

Engelson (2000) discusses automatic translation of SolidWorks models to an earlier version of the Modelica MultiBody library. However, this work did not consider automatic handling of kinematic loops, automatic selection of states nor elimination of redundancies in the kinematic definitions. Bowles et.al. (2000) present a converter from ADAMS to Modelica. This converter made automatic layout of the Modelica code in a similar way as to what is described in this paper. Cut joints were inserted auto-

DOI: 10.3384/ecp09430113

matically. However, planar loops and redundant joints were not handled. There are also other simulation software vendors that provide tools for conversion of CAD models to Modelica or similar formalisms. However, they require manual modification after the conversion to derive a valid model. This is a complicated and error prone task and also makes maintenance of models harder.

2 Modelica code generation

A tool has been developed that translates a CATIA V5 and V6 models to Modelica by traversing the internal CATIA structure to get information about parts and joints and how they are related. This information is then used to generate a corresponding Modelica model. Bodies and joints are mapped to Modelica models from a CATIA library written in Modelica. This library is a wrapper library that maps the models to Modelica.Mechanics.Multibody library or to specially written models.

The CATIA parts are mapped to a model called BodyShape. The traversal provides information about the reference coordinate system, the center of mass in the local coordinate system, the mass, the inertia, the shape and color of the body exported in VRML format for animation purposes and the icon exported as a PNG file to be used in the Modelica diagrams. The connector of the BodyShape body represents the center of mass of the body.

The joints are mapped to models in the CATIA.Joints package, see Figure 2.

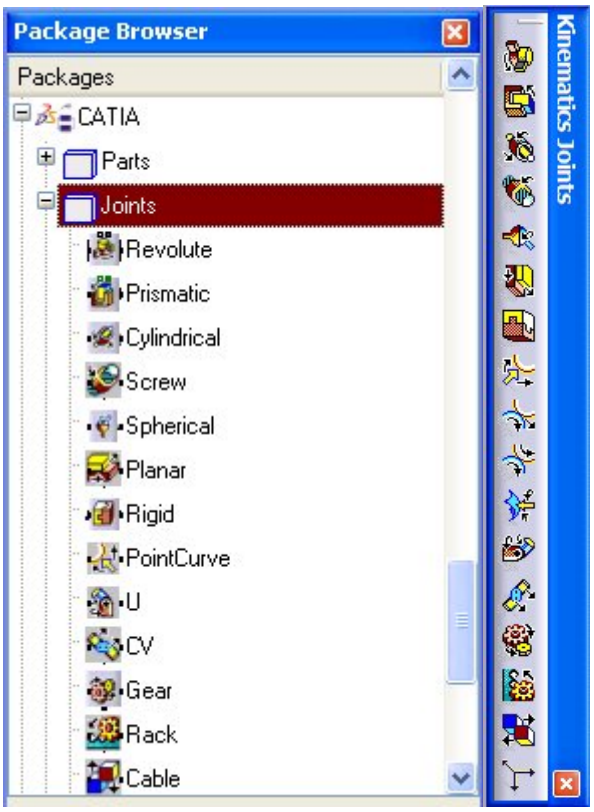
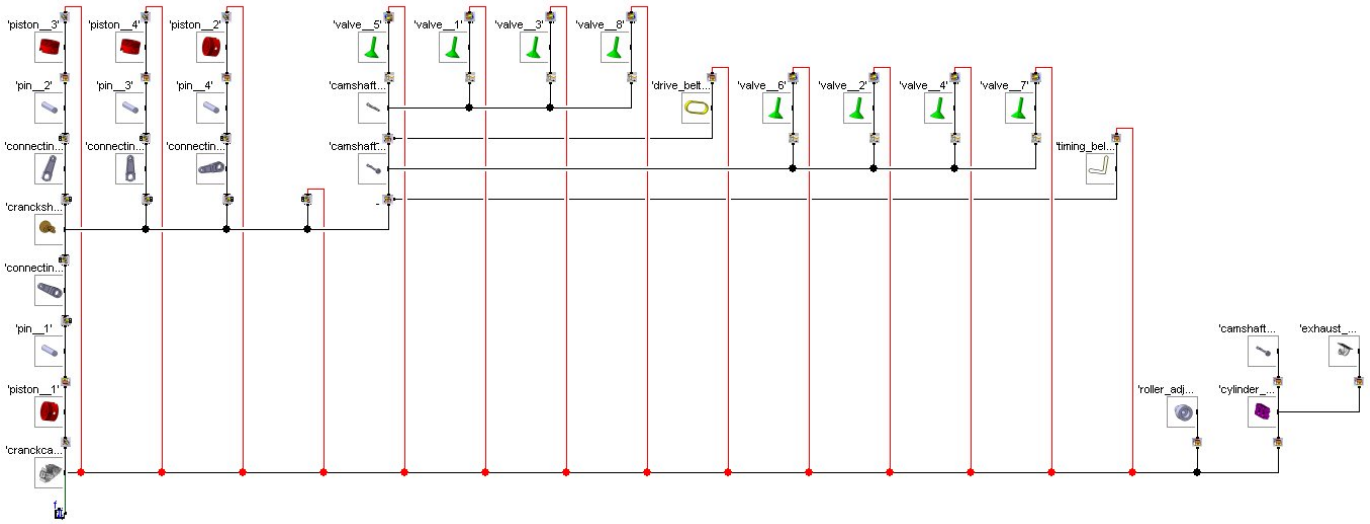


Figure 2. CATIA package structure and Kinematics Joints toolbar in CATIA V5.

There is a one to one mapping from the Modelica Joints to the corresponding Joints in CATIA. The joints include the necessary translations and rotations from the MultiBody joint frames to the center of mass of the respective bodies which the joint is attached to. In addition to the MultiBody joints, some new joint types such as point on curve were developed. The curve data are retrieved from CATIA and table lookup is used in the joint model.

The Modelica model corresponding to the CATIA model in Figure 1 is shown below. It consist of 30 bodies, 9 Revolute, 4 Cylindrical (one for each



Modelica diagram for engine model

cylinder), 8 Prismatic, 2 Gear (to drive the camshafts), 8 PointCurve (one for each valve) and 10 Rigid joints. In addition, the translator introduced 14 CutJoints, one for each kinematic loop. As an example, the Modelica code for the crankshaft is shown below:

```
CATIA.Parts.BodyShape 'crankshaft__1'(
  m = 0.861027543288454,
  r_0_start = {-0.182833605157492,
    0.00863258496368804,-0.0978102539095819},
  R_start=MBS.Frames.Orientation(
    T=[
      0,-0.847538988037033, 0.530733137986643;
      0.0199597528990819, -0.530627407596764,
        -0.84737014496106;
      0.999800784288643, 0.0105933028037161,
        0.0169166684516004],
    w={0,0,0}),
  r_CM = {-0.0584560176906743,0,0},
  I_11 = 0.0116473560180544,
  I_21 = 0.000299454247697959,
  I_31 = -0.000193283295897208,
  I_22 = 0.00484290739689144,
  I_32 = 0.00442125657996889,
  I_33 = 0.00911887589804025,
  shapeName = "2",
  iconName = "./crankshaft__1.png")
  annotation (Placement(transformation(extent=
    {{-120,460},{-60,520}})));
```

The intial position, r_0_start, and orientation, R_start, are retrieved for a consistent initial configuration of the entire mechanism. Mass, m, and inertia, I_11, …, I_33 are calculated for the density given in CATIA. The parameter shapeName refers to the name of a VRML file (2.wrl) exported by CATIA representing the shape and color of the body. The parameter iconName is the name of a PNG-file containing a 2D projection of the body. It is used for the icons in the bodies in the model diagram.

A joint instance is shown below. It contains the axis of rotation, n, and fixed translations, ra and rb, and rotations, fixedRotation_a and fixedRotation_b, from the joint to the center of mass of the connected bodies. In addition, the joint instance contains the initial configuration, phi_start, of the joint.

```
CATIA.Joints.Revolute 'Revolute__5'(
  n = {0,0,1},
  ra = {0,0,0.124000000000001},
  rb = {0,0,0},
  fixedRotation_a(
    R_rel=MBS.Frames.Orientation(
      T=[0,0.907781470137235,-0.41944344371495;
        0,0.419443443714955,0.907781470137245;
        0.999999999999986,0,0],
    w={0,0,0})),
  fixedRotation_b(
    R_rel=MBS.Frames.Orientation(
      T=[0,0.81704961687189,-0.576567362560056;
        0,0.57656736256006,0.817049616871915;
        0.999999999999979,0,0],
    w={0,0,0})),
    phi_start = 1.07387718370489e-008)
   annotation (Placement(transformation(extent=
     {{-70,660},{-50,680}}, rotation = 90)));
```

Figure 3 shows the CATIA V5 representation of kinematic joints, i.e. Revolute.5 is connected in between connecting rod_2.1 and pin.2.

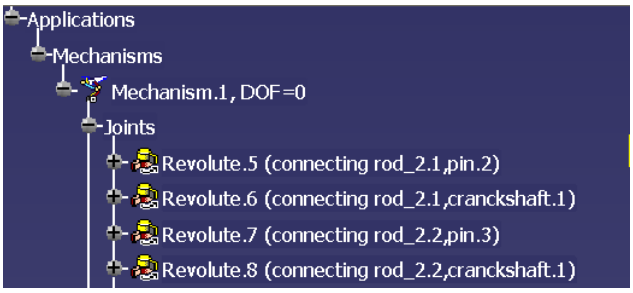


Figure 3. CATIA V5 representation of kinematic joints in engine

The corresponding Modelica representation is two connections:

```
connect('pin__2'.frame_a,
  'Revolute__5'.frame_b );
connect('connecting_rod_2__1'.frame_a,
  'Revolute__5'.frame_a );
```

3 Layout Generation

The Modelica diagram layout is automatically generated and is based on the spanning tree structure of the mechanism obtained by the classical depth-first search algorithm of graph theory. See the layout of the engine four cylinders in the Modelica diagram for engine model above and in Figure 4 which shows the 4 cylinders zoomed in.

In this way, loops are generated and placed aside of the main branches. Notice that the closing of the loops are presented by red connections. For the Engine, it is possible to see the four cylinders and the longest possible loop is starting at the crank case, passing though a cylinder to the crank shaft and then back through another cylinder to the crank case. Then the other two cylinders are detected as loops from the crank shaft to the already-detected crank case and placed correspondingly. Other loops are the synchronization belts and valve shafts. The root of the tree is the fixed body present on the CATIA model.

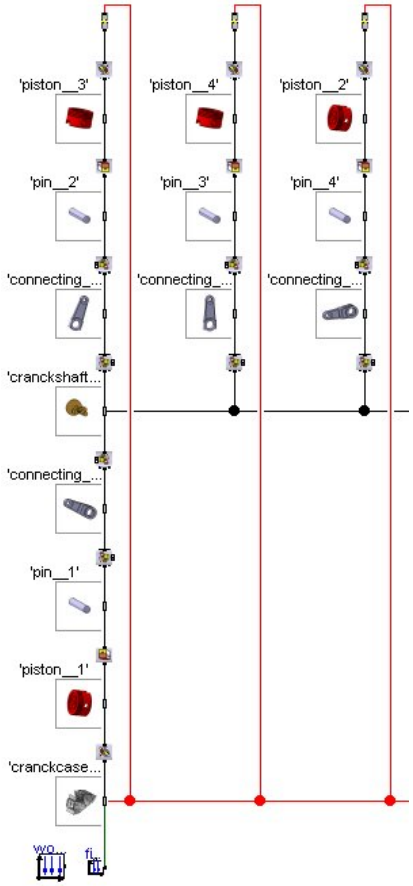


Figure 4. The four cylinders zoomed in.

4 Incorporating dynamics by extending

An important property of the translated model is the possibility to use Modelica extends (inheritance) for adding controllers and other features of the model for dynamic simulation. For instance, the engine model can be extended by introducing models of the gas forces of the combustion acting on the cylindrical joints of the pistons. In that way, the translated model is separated and can be changed independently of the added models.

CATIA models do not contain force aspects. This means that springs, dampers, friction and other force elements have to be added in the Modelica layer. Actuators such as electrical motors or hydraulic systems as well as sensors and control systems are also added to the Modelica layer.

5 Simulating engine

The possibly extended generated Modelica model is symbolically translated to ODE form in the usual way involving finding systems of simultaneous equations, making index reduction and automatic state variable selection. For details, see Otter et.al. (2007) and Mattsson et.al. (2000). The engine model contains point on curve joints. This means a constraint described by tabular data. The index reduction requires the constraints to be differentiated twice. The implication was that the tabular data was used for generation of splines which could be differentiated twice.

In addition, the engine model contains redundant joints. Special methods were developed to handle those, See section 6.

In fact the model has only one degree of freedom, the crank angle. The automatically generated model was extended and a constant torque generator was added to drive the crank shaft. Figure 5 shows plots of torque, acceleration and the position of each of the cylinders. It can be seen that the inertia is not constant since acceleration is time varying. In fact the amplitude of inertia variations increases for higher angular speeds.

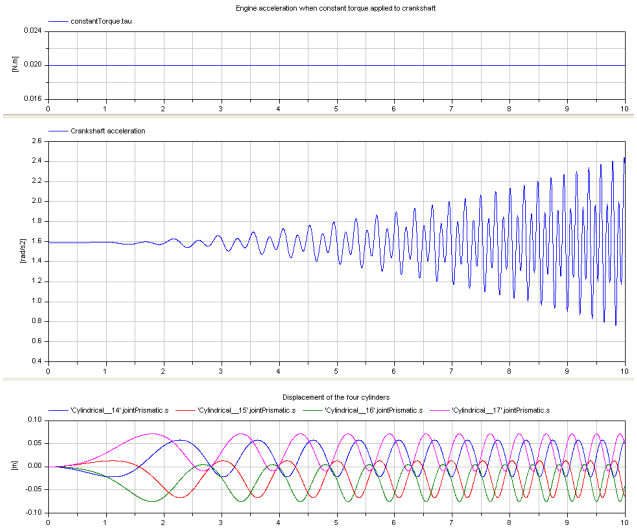


Figure 5. Simulations results.

The results can be animated since all bodies are represented by VRML. The animation in Dymola can be seen in Figure 6.

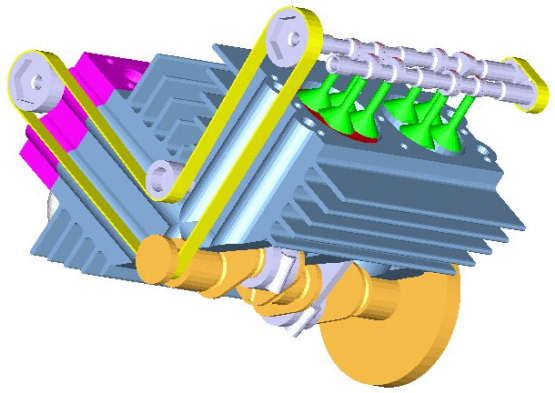


Figure 6. Dymola animation of dynamic behavior.

6 Redundancies in MultiBody Systems

The approximation or idealization that bodies of multibody systems are rigid implies, unfortunately, that it is not possible to determine forces acting on the members of any structure.

As a basic example, let us model a door with two hinges by the simple model Door1 in Figure 7.

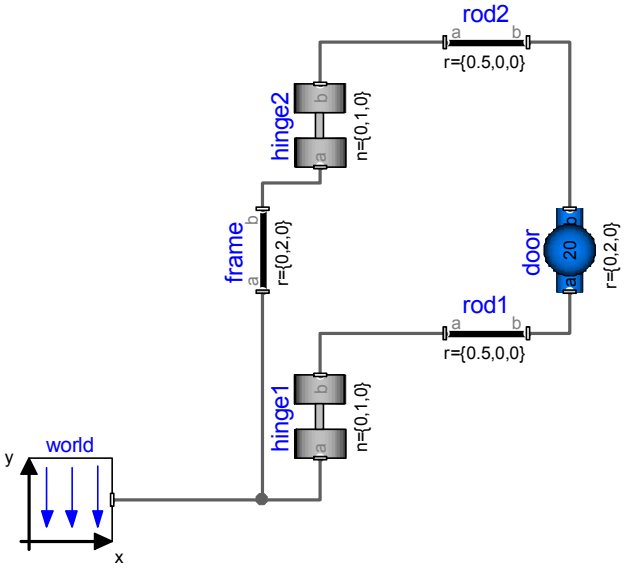


Figure 7. The model Door1

The object *door* represents the door as one rigid body. The house and the door frame are represented by the objects *world* and *frame*. Each of the two hinges is represented by a revolute allowing the door to rotate along a vertical axis, the y-axis. This model does not translate, because it is singular. The error diagnosis from Dymola contains the following message: "The model includes the following hints: All Forces cannot be uniquely calculated. The reason could be that the mechanism contains a planar loop or that joints constrain the same motion."

It is true that joints constrain the same motion in the model. From a kinematics point of view this model is equivalent to the model Door1a in Figure 8.

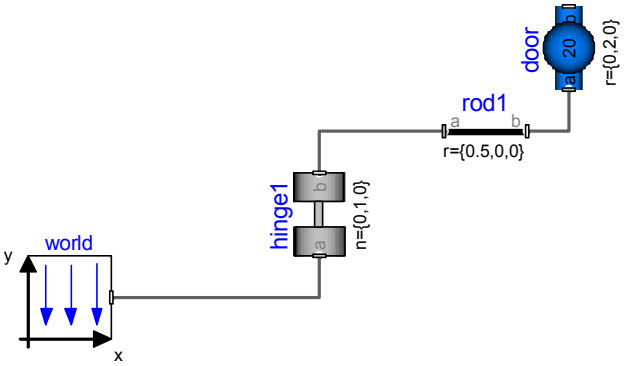


Figure 8. The model Door1a which is a kinematics equivalent to the model Door1.

In other words, the door in the two models can only rotate around the vertical axis, the y-axis. When it comes to forces, the two models are different. The model Door1a, requires hinge1 take all loads, while the model Door1 allows the load to be shared by two hinges.

Why do doors have at least two hinges? The hinges not only balance the vertical force due to gravity, but they need also balance the torque due to gravity. When having just one hinge, this hinge has to provide reactive torques in the x and y directions. There will always be some play in a real hinge and such a door would not behave as desired. When using two hinges, the balancing torque can be realized by a force pair. When having two hinges a more realistic model could be to use spherical joints to describe the hinge. A spherical joint does not produce any reaction torque and we can view as a way of taking the play into account. The balancing torque must be produced by a force pair from the two spherical hinges. Unfortunately, such a model is still singular. In real life, we know that it is important to mount the hinge parts in the frame in the door with good accuracy. If not, the vertical load will not be shared, but one of the hinges has to take all vertical loads. To make the model non-singular, one of the hinges has to include a prismatic joint in the vertical direction, which means that the other hinge has to take all vertical loads. You could think of letting both hinges have two actuated prismatic joints and use springs and dampers to split the vertical loads. However, if we just want to know how the door moves due to external forces, such a model would be unnecessary complex.

When the desire is to model the dynamic behavior of the door, it is just necessary to get the acceleration correct. For a rigid body, we need then to get the to-

tal forces and torques acting on it correct. We need not consider how the forces and torques acting on the body are distributed over the body. For the door, we need only the sum of the forces and torques acting on the body with respect to some common point, say, the center of inertia. We can in fact prescribe any value of the vertical reaction torque from hinge2, because, hinge1, will automatically counter balance it and the sum will be correct. The technique to relax redundant position or orientation constraints by prescribed forces and torques is to introduce cut joints.

Below we will introduce the concept of cut joints and how Dymola handles selection of constraints within cut joints automatically.

6.1 Introducing cut joints in kinematic loops

Redundancies in multibody systems can only appear when there are kinematic loops.

The model Door1 has the loop: frame, hinge2, rod2, door, hinge1, rod1 and back to frame. When traversing the CATIA model structure to generate the Modelica model, the tool keeps track of kinematic loops and introduces a cut joint in each kinematic loop. Thus the model of the door, call it Door2, will include a cut joint. Let us place it between hinge2 and rod2 as shown in Figure 9. Actuators have also been introduced for testing purposes.

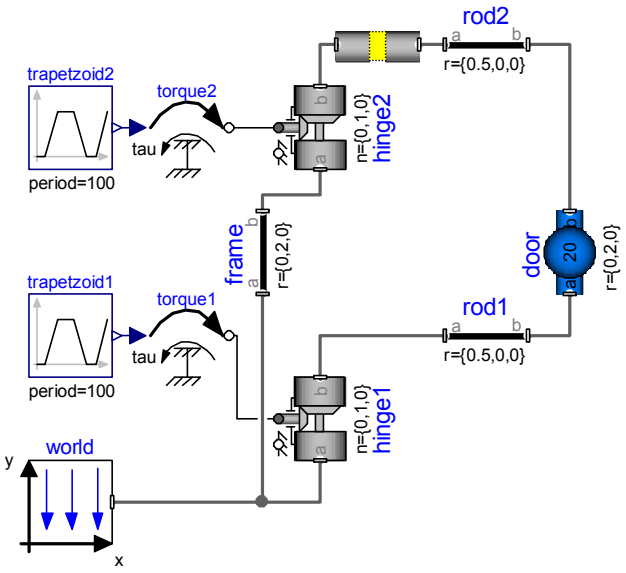


Figure 9. The model Door2.

The basic meaning of the cut-joint is that the position and orientation should be the same. However, it can handle the fact that these constraints might be redundant, i.e., they might be deduced without considering the cut-joint constraints. The cut joint does this by giving expressions for the relative positions and orientations of the two connector frames. The constraint saying that these should be zero is given by a special function with the interface

```
function conditionalConstraint(
  input  Boolean condition;
  input  Real u1;
  input  Real u2;
  output Real y;
  …
end conditionalConstraint;
```

The basic meaning is

```
  y = if condition then u1 else u2;
```

It is used vectorized in the cut joint as

```
fill(0, 6)  =
conditionalConstraint(
  conditions,
  {r_rel_a[1], r_rel_a[2], r_rel_a[3],
   phi[1], phi[2], phi[3]},
  {f_c[1], f_c[2], f_c[3],
   t_c[1], t_c[2], t_c[3]});
```

The first actual argument is a Boolean vector with 6 elements. The second argument includes the relative positions and rotations, while the third argument includes the forces and torques in the corresponding directions. A user could set elements of conditions to false to replace redundant position or orientation conditions by postulating corresponding forces or torques to zero. Unfortunately, realistic applications have many loops and since there $2^6 = 64$ ways of setting the vector conditions for each loop, there is a combinatorial explosion. It is not feasible to set cut-joints manually. Dymola supports automatic selection of which of the constraints to use in order to have a well-posed problem. In simple cases this can be done statically at translation otherwise it is done dynamically at simulation.

Initialization of models with kinematics loop may be a non-trivial issue; in particular when there are conditional constraints to select. Fortunately, CATIA has a powerful kinematics solver, which means that the Modelica models generated from a CATIA model structure can be provided with consistent initial values, so there is no need for generating code for initialization.

Dymola allows plotting of the Boolean conditions vector in the usual way during and after simulation. Dymola also reports the final selection as new model extending from the model simulated with modifiers setting the condition vector of all conditionalConstraint functions:

```
model Door2_Fixed_Constraints
  extends Door2(cutJoint.conditions=
    {false, false, false,
     false, true, false});
end Door2_Fixed_Constraints;
```

Note that all elements are false but the fifth. It means that Dymola has really cut the loop. The fifth element represents rotation along the y-axis. In this respect the cut joint is in series with hinge1 that allow this rotation. Two revolute joints with the same axis of rotation without inertia in between imply ambiguity between the rotation angles of the two revolute joints and such a model would thus be singular.

Typically you need just to simulate the model to have it initialized, i.e., simulate with startTime = stopTime. A model obtained in this way can be used as a basis for further model development and simulation where Dymola's automatic selection is disabled. For example, initialization equations are now supported in usual way.

We have presented a general cut joint with conditional constraints. It can be placed anywhere in a loop with one important exception. It cannot be placed in series with a spherical joint if there is no inertia in between. A spherical joint puts no constraints on the orientation. It is has no Connections.branch(…) defined between its frame connectors to handle the over determined connectors describing the orientation. The same applies for the cut joint. It means that the part between the spherical joint and the cut joint has no root defined to handle the normalization of the overdetermined part of orientation description. To solve this problem, we have introduced a special spherical cut joint. Such a joint only has the position constraints conditional.

6.2 Example: The door model revisited

Let us consider the door model once more and put the cut joint at another place in the loop, namely between rod2 and door as shown in Figure 10.

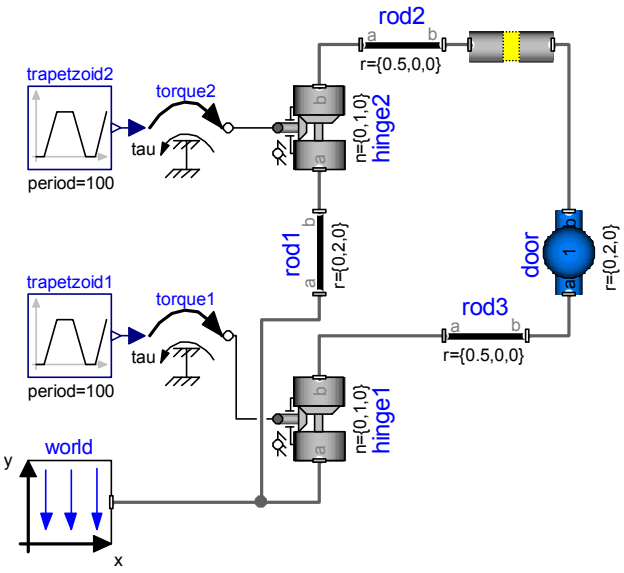


Figure 10. The model Door3.

Assume that the cut-joint would break the loop completely, i.e, all conditions are false. It means that hinge2 allows rod2 to rotate in a plane parallel to the x-z plane. It means that a cut joint cannot break the loop completely, but has to select one constraint. There are two possibilities. First, as for Door2, it can lock the rotation in the direction of the vertical axis. Call this angle φ and thus the active constraint will be φ = 0. Secondly, the motion can be prevented by requiring the relative displacement of the cut joint in the z direction, Δz = |rod2.r|·sin φ to be zero, thus requiring φ = 0. The selection of constraints is based on analyzing the Jacobian of the constraints. For φ ≈ 0, we have Δz ≈ |rod2.r|·φ. It means that if |rod2.r| < 1, the condition, φ = 0 will be selected. Otherwise the conditions Δz = 0 will be selected. The two will give the same accelerations and the door will move in exactly the same way. However, the torques and forces at the connector frames of the cut joint will be different. Choosing φ = 0 as the active constraint in the cut joint, will set all forces and torques of the cut joint to zero, except for the torque in the y direction, cutJoint.frame_a.t[2], which will propagate the torque due to the actuation of hinge2. Choosing Δz = 0 as the active constraint in the cut joint, will set all forces and torques of the cut joint to zero, except for the force in the z direction, cutJoint.frame_a.f[3], which will propagate the effects of the actuation of hinge2. Putting a cut joint close to hinge2 does not influence the possibilities to have actuation on it. This is a better alternative than removing it completely as done in model Door1a.

6.3 Planar kinematics loops

The door model is an example of a model having redundant joints, i.e., joints constraining the same

motion. The error message for the model Door1 indicated another possibility, namely possibility of the loop being planar.

Consider the system PlanarLoop1 in Figure 11.

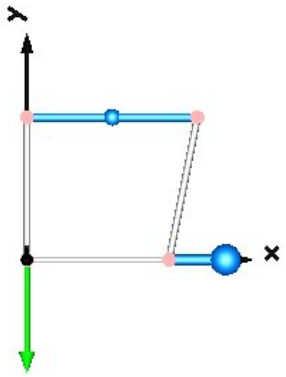


Figure 11. The system PlanarLoop1.

The system is built by bars lying in x-y plane and connected by revolute joints with their axes of rotation parallel with the z axis. A model is shown in Figure 12.

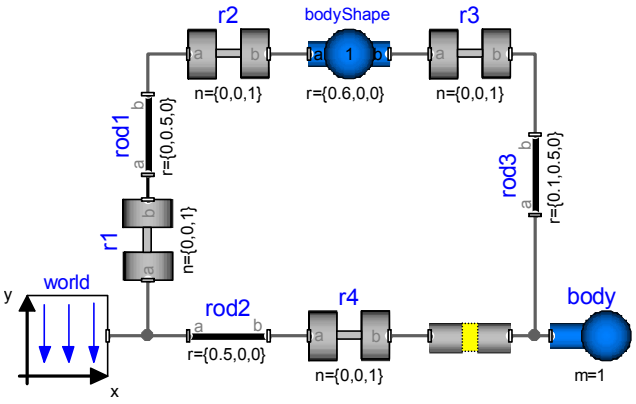


Figure 12. The model PlanarLoop1.

Dymola selects the constraints in the following way:

```
model PlanarLoop1_Fixed_Constraints
  extends PlanarLoop1(cutJoint.conditions=
  {true, true, false, false, false, true});
end PlanarLoop1_Fixed_Constraints;
```

Since the third argument is selected to false, the loop is cut in the z-direction, i.e., the position constraint in the z direction is redundant. First, the system cannot move in the z direction at all, because all joints are revolute joints with their axes of rotation parallel with the z axis. Starting from world going the path rod2, r4 we can deduce that the left end of the cut joint has always cutJoint.frame_a.r[3] = 0. Going the other way from world we find in similar way that cutJoint.frame_b.r[3]. This means that the position constraint in z-direction of the cutJoint is redundant. The components cannot rotate along any axes in the x-y plane. This explains why Dymola selected

```
cutJoint.conditions[4] = false;
cutJoint.conditions[5] = false;
```

The model PlanarLoop1 is built nicely in a coordinate plane z = 0. Far from all Modelica models generated from the CATIA models structure is that well-behaved. Dymola must be able to find any planar loop independent of which plane it exists in. Let us define the mechanism in the plane

-sin(a)*y + cos(a)*z = 0

It means a tilting of the mechanism along the x-axis an angle, a, as shown in Figure 13.

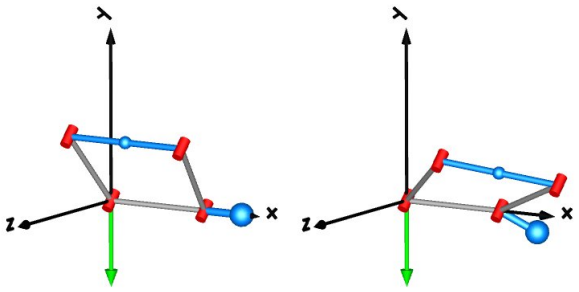


Figure 13. The system PlanarLoop2 in two positions.

The model is shown in Figure 14.

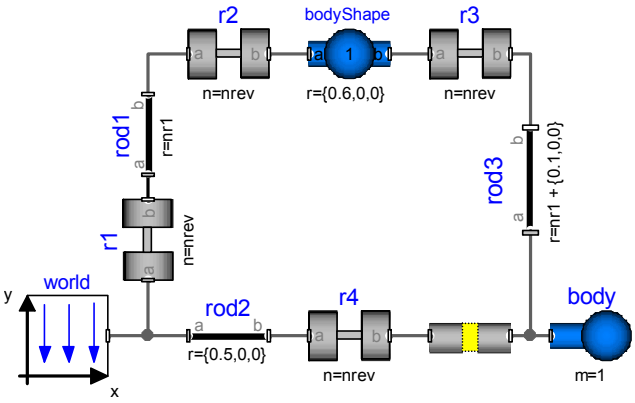


Figure 14. The model PlanarLoop2.

The model PlanarLoop2 is an extension of PlanarLoop1, introducing the parameters a, nrev = {0, -sin(a), cos(a)} and nr1 = 0.5*{0, cos(a), sin(a)}, and modifying the axes of rotations of the revolute joints and the axes of rod1 and rod3.

When the angle is small i.e., -45º<a <45º, Dymola selects the constraints as for the model PlanarLoop1, which is the special case a=0. For -90º<a <-45º or 45º<a <90º for Dymola selects the constraints in the following way:

```
model PlanarLoop2_Fixed_Constraints
  extends PlanarLoop1(cutJoint.conditions=
  {true, false, true, false, true, false});
end PlanarLoop2_Fixed_Constraints;
```

Simply said, you get the selection of the coordinate plane to which you have least tilt angle.

6.4 Example: A crank shaft, piston and cylinder mechanism.

Consider the model Engine1a, which is found in Modelica.Mechanics.MultiBody.Examples.Loops. see Figure 15 and 16..

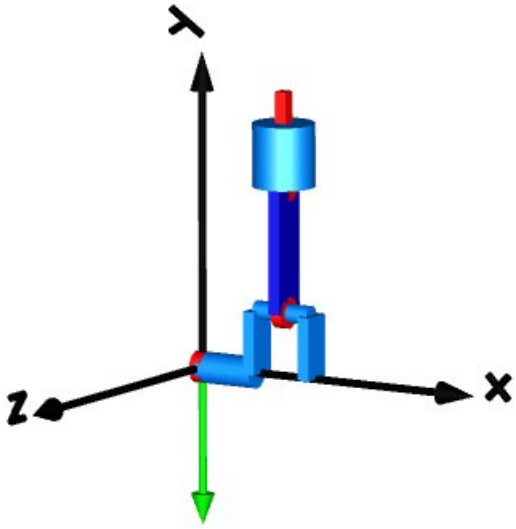


Figure 15. An animation of Engine1a.

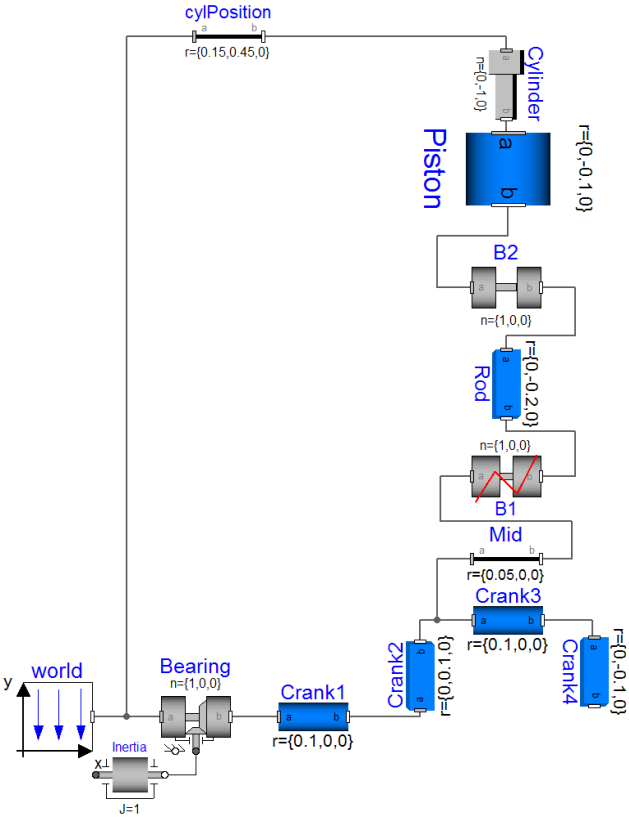


Figure 16. The model Engine1a.

The model is intricate. First, the revolute joint B1 is a revolute planar cut joint indicating that it is part of a planar loop. This is may be not that difficult to understand when looking at Figure 15. However, the component cylinder at the top is not a cylindrical joint as might be expected, but a prismatic joint. This joint does not allow the piston to rotate. If B1 had been a real revolute joint then the crank mechanism had prevented rotation of the piston along its length axis. Unfortunately, when B1 is a planar cut joint, then the piston can rotate. The component Cylinder being just a prismatic joint fixes that.

The new universal cut joint and the automatic constraint selection of Dymola relives the user from such complex "fixing" of the model. Let us make a more natural, straightforward model, call it Engine. It is similar to Engine1a, but B1 is an ordinary revolute joint and Cylinder is a cylindrical joint. Additional we add a cutJoint between cylPosition and Cylinder, because there is a kinematics loop. Dymola translates and simulates this model without any problems. Dymola reports the following selection of constraints for cutJoint:

```
model Engine_Fixed_Constraints
   extends Engine(cutJoint.conditions=
   {false, true, true, true, true, false});
end Engine_Fixed_Constraints;
```

Note, that the conditions vector has only two elements selected to be false. Reconsider the model Engine1a. The revolute cut-joint B1 means selecting the condition vector to be

```
   {false, true, true, true, false, false}
```

The fifth element is here false. It means that the revolute planar cut joint is cutting the loop a bit too much. If Cylinder in model Engine1a would have been a cylindrical joint, the part between B1 and Cylinder had been free to rotate along the y axis. Selecting Cylinder to be a prismatic joint prevents the rotation along the y-axis.

7 Conclusions

A method for coupling kinematic CAD models, in particular CATIA models, to Modelica has been described. Since such models often contain redundant joints, a new combined structural, symbolic and numerical technique has been developed for Dymola to handle such models. Details were given about typical such redundant kinematic definitions and how the new method handles those.

8 Acknowledgements

The authors would like to thank Martin Otter (DLR) and Hans Olsson (Dynasim) for their contributions.

References

Bowles P., Tiller M., Elmqvist H., Brûck D., Mattsson S.E., Möller A., Olsson H., Otter M (2000).: **Feasibility of Detailed Vehicle Modeling**. SAE World Congress 2000.

Dymola (2009). **Dymola Version 7.3**. Dassault Systèmes, Lund, Sweden (Dynasim). Homepage: www.dymola.com.

Engelson V. (2000): **Tools for Design, Interactive Simulation, and Visualization of Object-Oriented Models in Scientific Computing**. Department of Computer and Information Science, Linköping University, Sweden.

Mattsson S.E., Elmqvist H., and Olsson H. (2000): **Dynamic Selection of States in Dymola**. Proceedings of Modelica Workshop 2000, pp. 61-67. Download: http://www.modelica.org/events/workshop2000/proceedings/Mattsson.pdf

Otter M., Elmqvist H., Mattsson S.E. (2003): **The New Modelica MultiBody Library**. Proceedings of Modelica'2003, ed. P. Fritzson, pp. 311-330. Download: http://www.modelica.org/events/Conference2003/papers/h37_Otter_multibody.pdf

Otter M., Elmqvist H., Mattsson S.E. (2007): **Multidomain Modeling with Modelica.** Handbook of Dynamic System Modeling, Chapter 36, Ed. P. Fishwick, Taylor & Francis Group LCC .

