3D Mechanics II

- In this lecture, we shall continue with the bond graph description of 3D mechanics.
- We shall complete the description of the joints.
- We shall then describe the problem of closed kinematic loops.
- We present a complete example of a model from 3D mechanics: a bicycle.
- Finally, we shall discuss the efficiency of the generated simulation code both in terms of choices that the user can make (Cardan angles *vs.* quaternions), and in comparing the efficiency of the multi-bond graph solution to the direct multi-body system solution.

Table of Contents

- Translational transformers
- Fixed translation
- Revolute joint
- Spherical joint
- Selection of state variables
- Closed kinematic loops
- Accuracy of simulation results
- Efficiency of simulation runs

The Translational Transformers

- Also for 3D mechanics, we need special translational transformers that describe the effects of transforming a motion along a rigid rod.
- A rotation around one end of the rod leads to a velocity at the other.
- Mathematically, this transformation can be described as:

$$\boxed{\begin{aligned} \mathbf{v}_2 &= \boldsymbol{\omega}_1 \times \mathbf{r} \\ \boldsymbol{\tau}_1 &= \mathbf{r} \times \mathbf{f}_2 \end{aligned}}$$

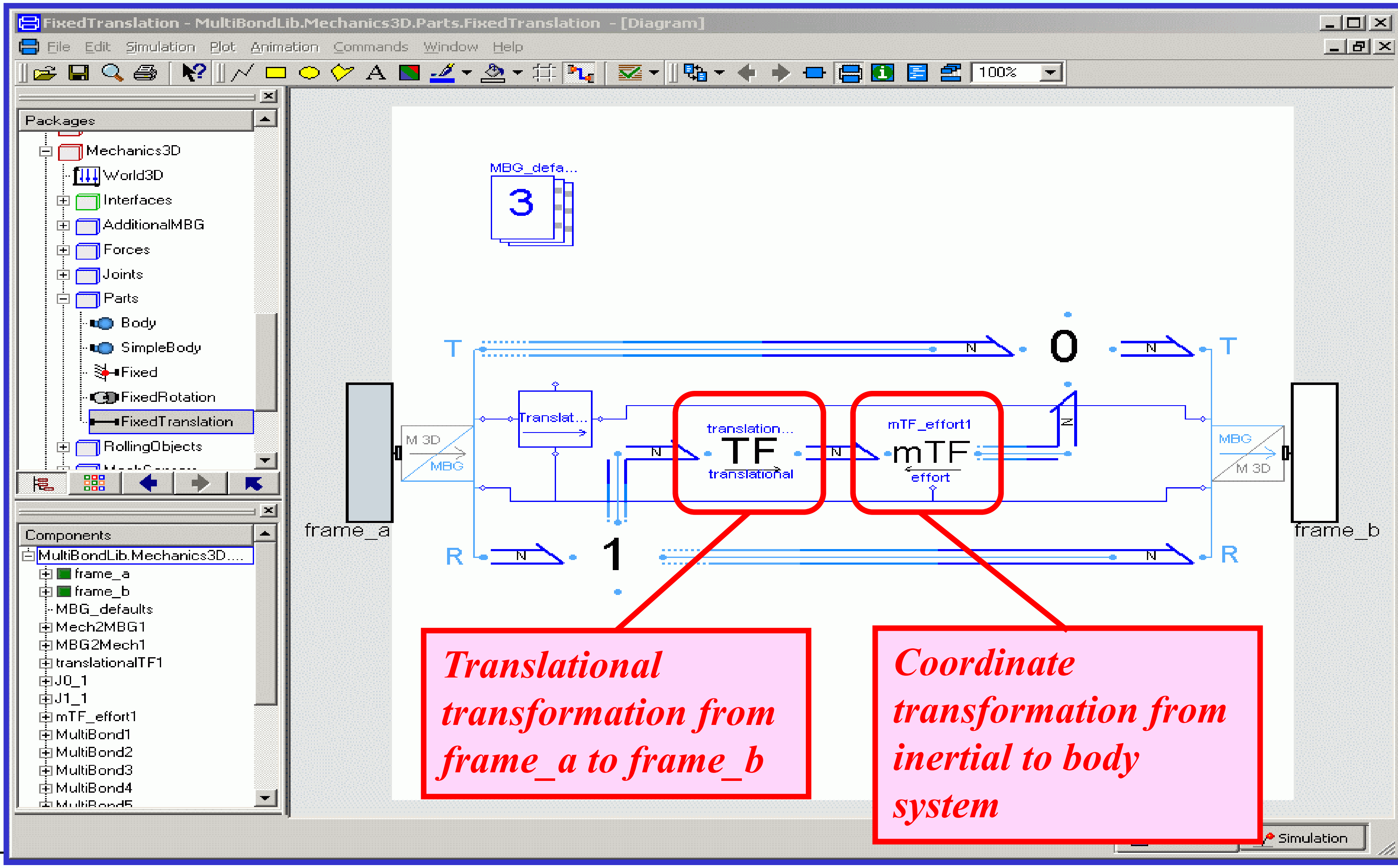
The Translational Transformers II

- The multi-bond graph library offers four different special 3d mechanics transformers:

translationalTF1 TF 1 2 translational	**Name:** translationalTF **Location:** MultiBondLib.Mechanics3D.AdditionalMBG **Parameters:** translation vector $\mathbf{r}$. **Equations:** $\mathbf{f}_2 = \mathbf{f}_1 \times \mathbf{r}$ $\mathbf{e}_1 = \mathbf{r} \times \mathbf{e}_2$
a transl_mTF1 mTF 1 2 translational	**Name:** translational_mTF **Location:** MultiBondLib.Mechanics3D.AdditionalMBG **Parameters:** translation vector $\mathbf{r}$. **Modulating signal:** amplification factor a. **Equations:** $\mathbf{f}_2 = \mathbf{f}_1 \times (a \cdot \mathbf{r})$ $\mathbf{e}_1 = (a \cdot \mathbf{r}) \times \mathbf{e}_2$
r transl_mTF2 mTF 1 2 translational	**Name:** translational_mTF2 **Location:** MultiBondLib.Mechanics3D.AdditionalMBG **Modulating signal:** translation vector $\mathbf{r}$. **Equations:** $\mathbf{f}_2 = \mathbf{f}_1 \times \mathbf{r}$ $\mathbf{e}_1 = \mathbf{r} \times \mathbf{e}_2$
mTF1 mTF 1 2 b M	**Name:** mTF **Location:** MultiBondLib.Mechanics3D.AdditionalMBG **Modulating signals:** transformation matrix $\mathbf{M}$. boolean control signal b. **Equations:** $\underbrace{\begin{matrix}\mathbf{f}_2 = \mathbf{M} \cdot \mathbf{f}_1 \\ \mathbf{e}_1 = \mathbf{M}^T \cdot \mathbf{e}_2\end{matrix}}_{b=true} \quad \begin{matrix}\text{or} \\ \text{or}\end{matrix} \quad \underbrace{\begin{matrix}\mathbf{e}_2 = \mathbf{M} \cdot \mathbf{e}_1 \\ \mathbf{f}_1 = \mathbf{M}^T \cdot \mathbf{f}_2\end{matrix}}_{b=false}$

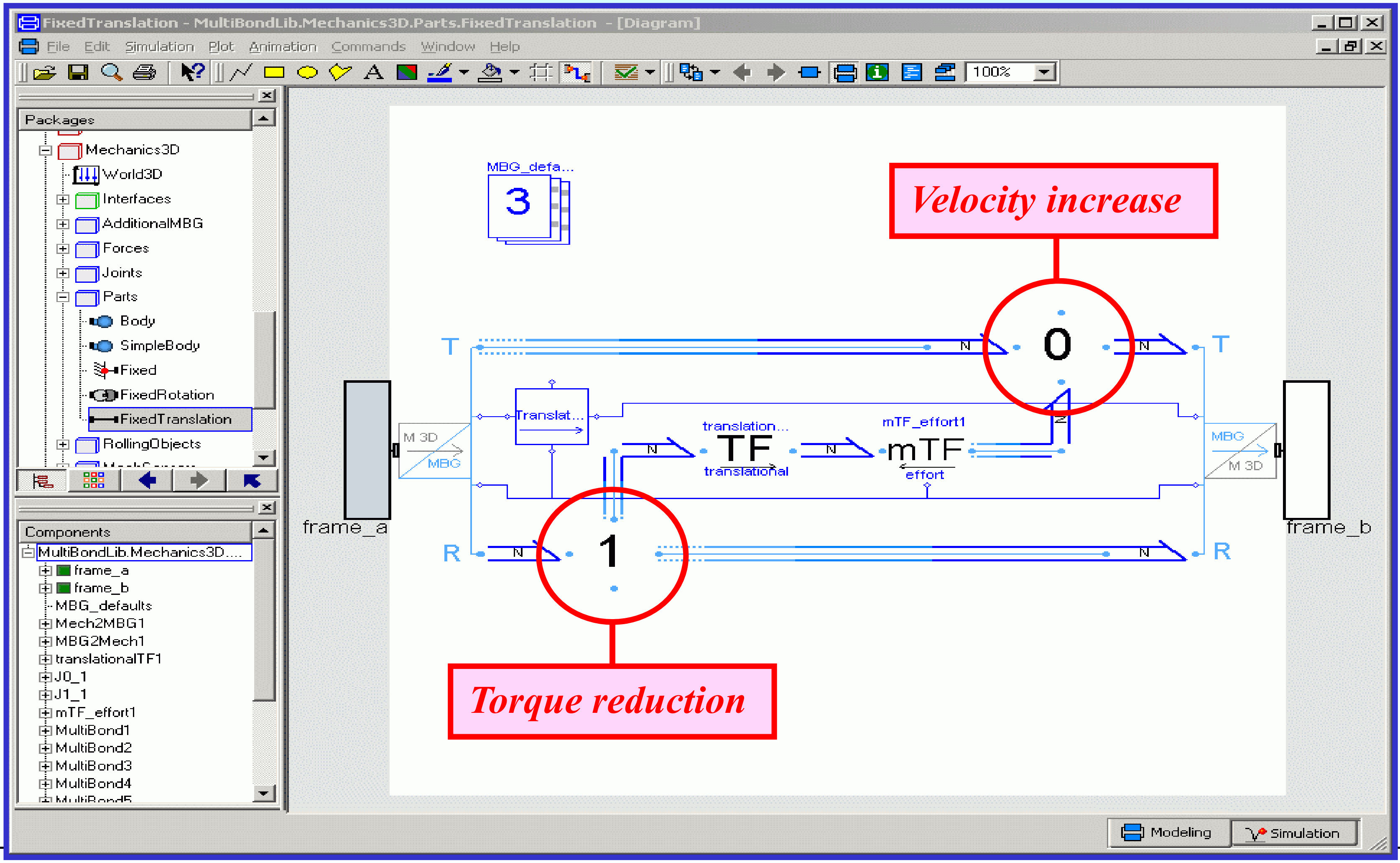
The Fixed Translation

- A rod is modeled by the following multi-bond graph:

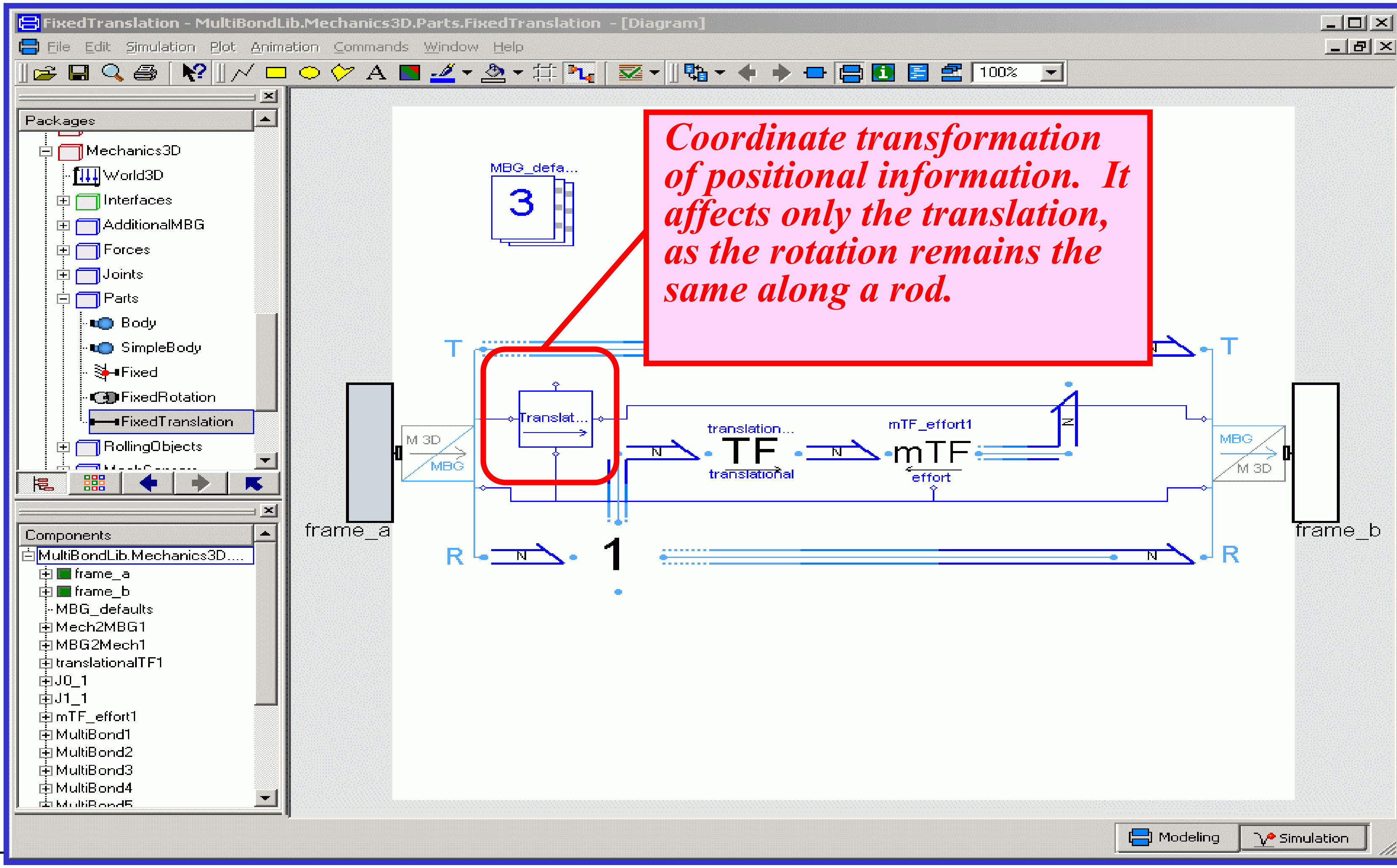


The Fixed Translation II

- A rotation around frame_a leads to a translation at frame_b:



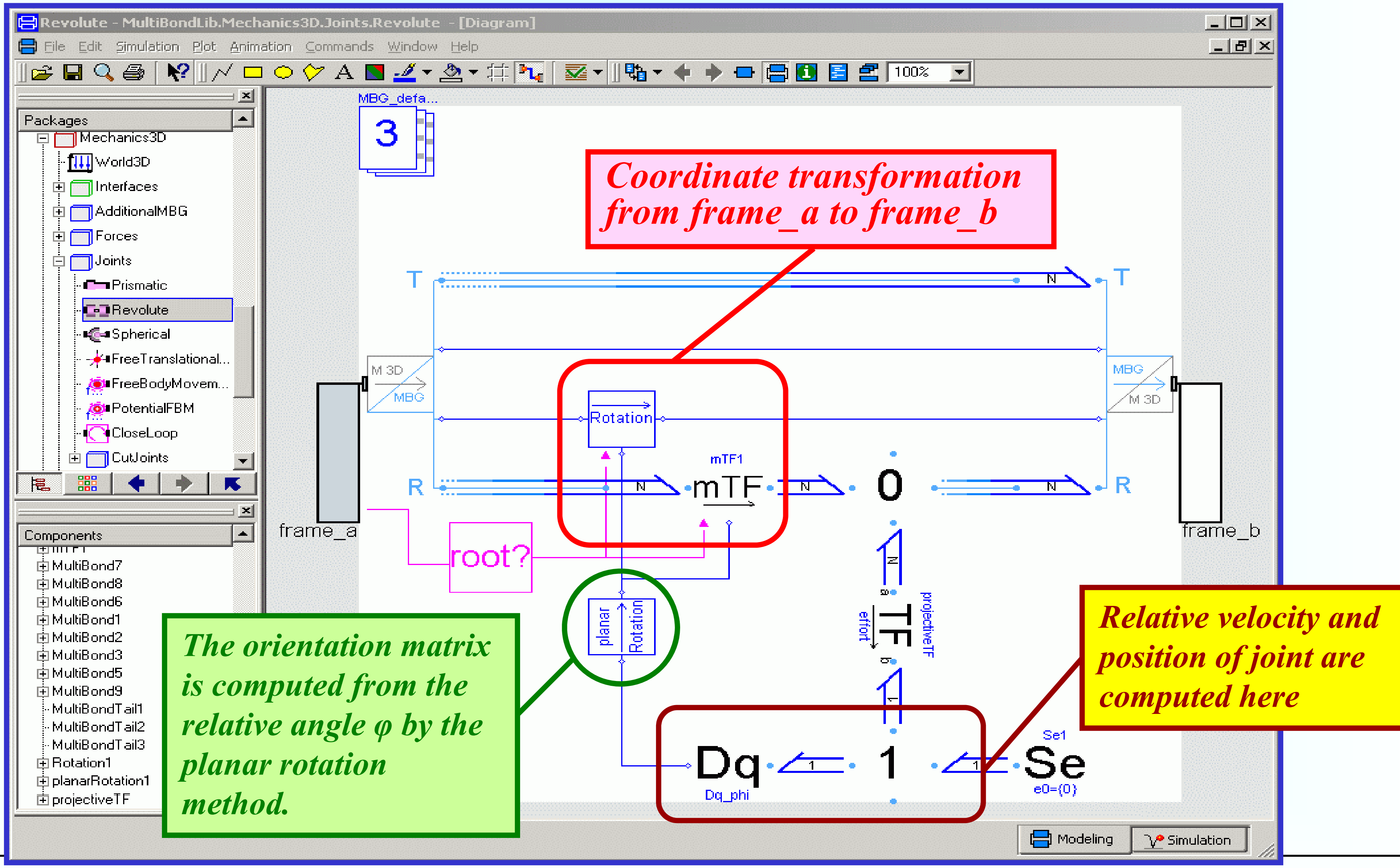
The Fixed Translation III



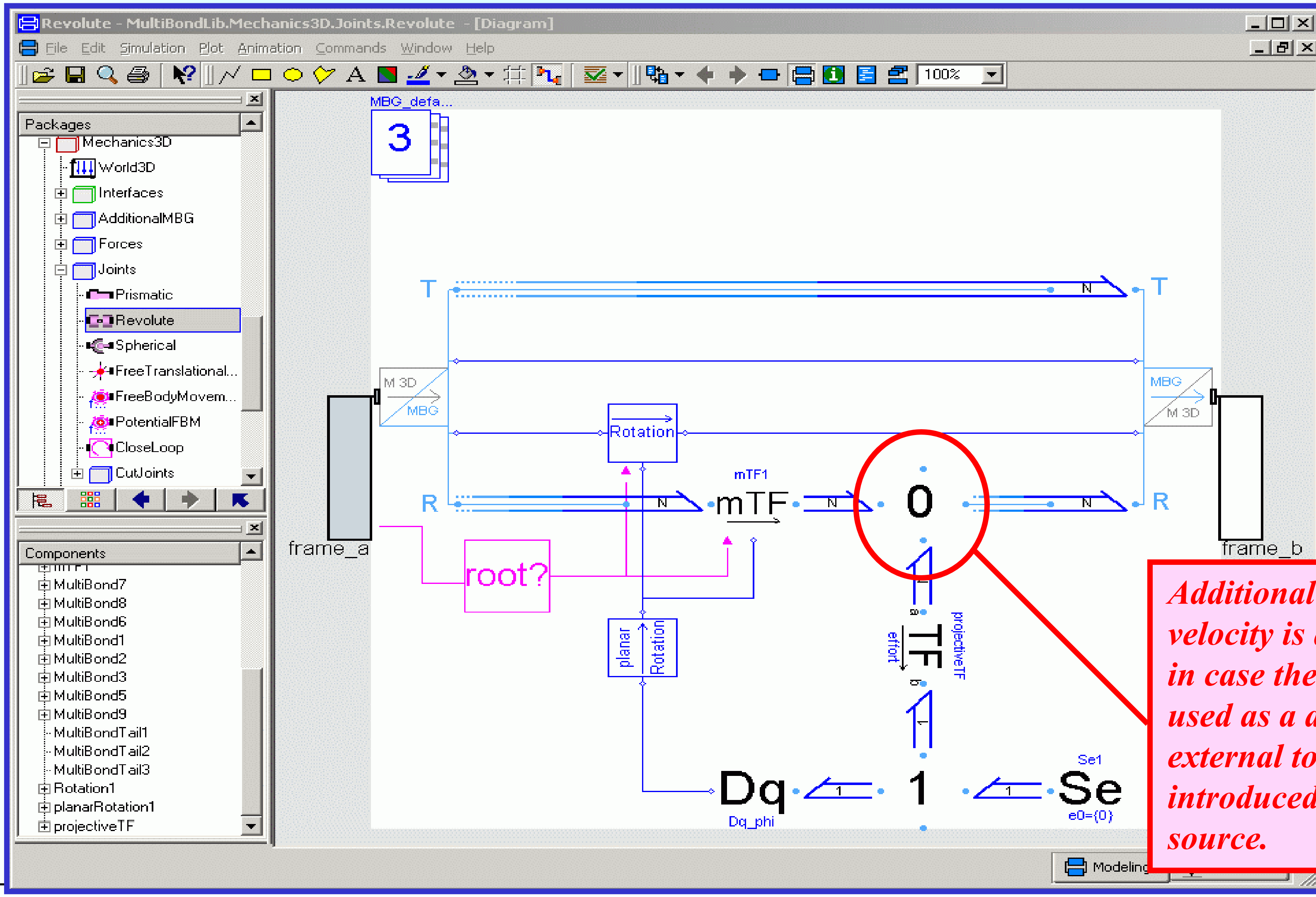
The Revolute Joint

- A revolute joint doesn't affect the translational motion at all.
- A revolute joint can represent a hinge or a drive, depending on how it is being connected.
- If the joint represents a hinge, the external torque at the joint is zero.
- The joint computes the relative angle between frame_a and frame_b, and from it, computes the orientation matrix that is needed to determine the coordinate transformation from frame_a to frame_b.

The Revolute Joint II

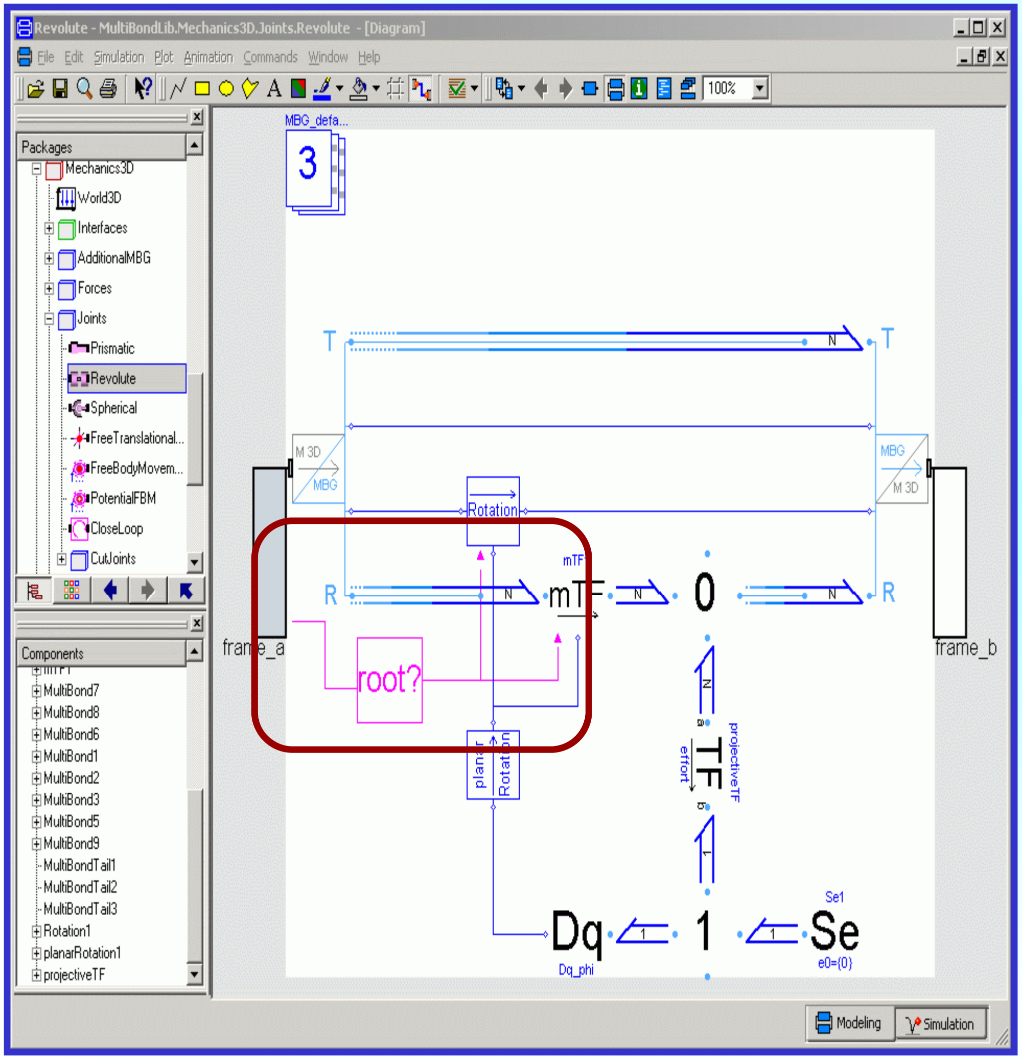


The Revolute Joint III



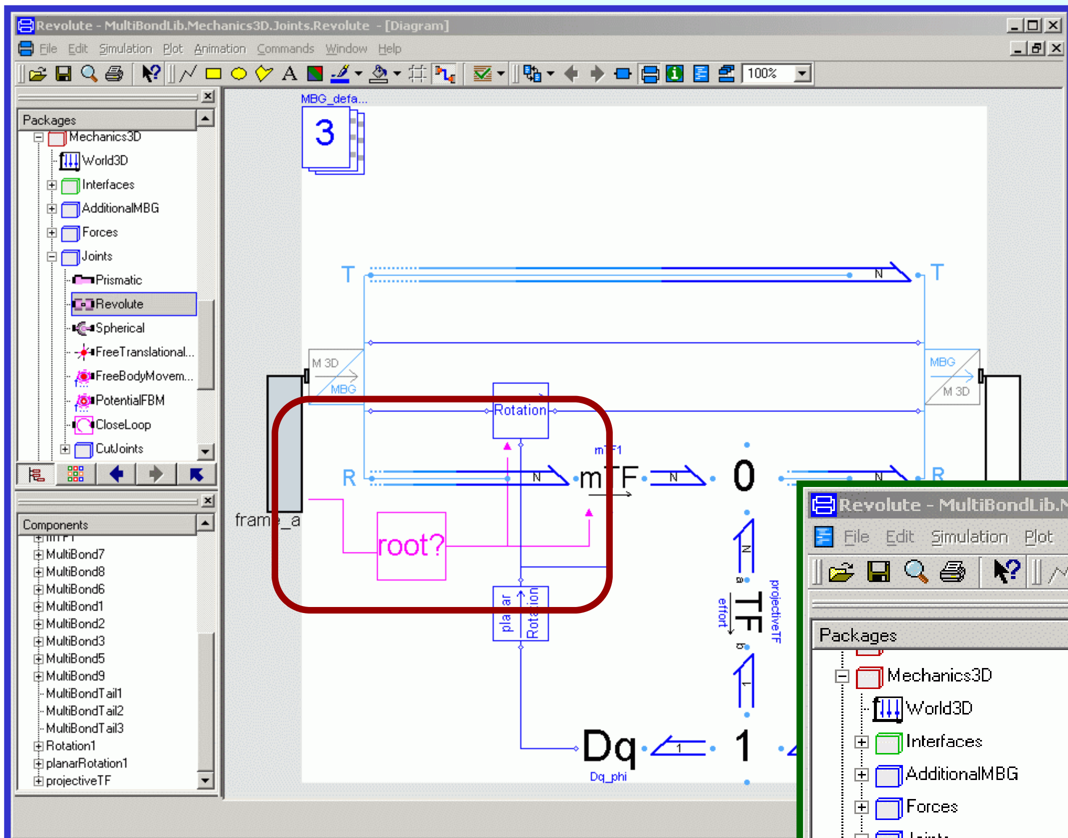
Additional rotational velocity is added here, in case the joint is being used as a drive, i.e., if external torque is being introduced at the effort source.

The Revolute Joint IV

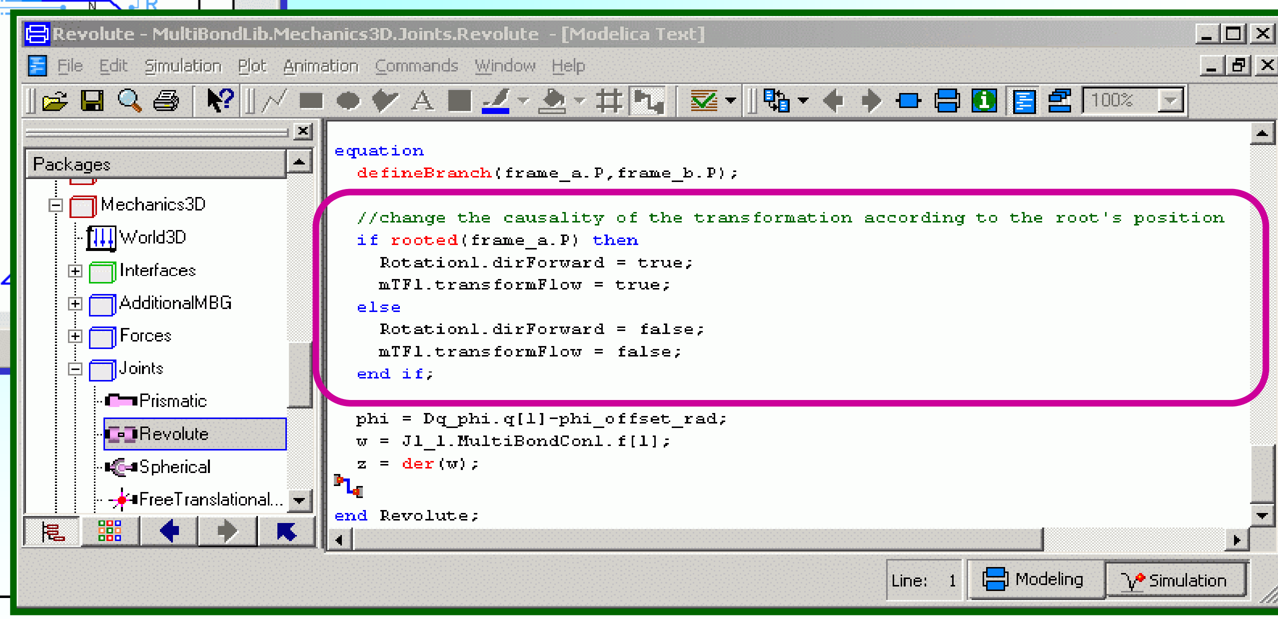


- How is the more suitable computational causality of the coordinate transformations being determined?
- The orientation matrix must be multiplied in starting with the world coordinate system.
- Hence we need to determine, on which side the transformer is connected to the wall.
- That should then be the primary side of the transformer.
- *Dymola* offers a built-in function (***rooted()***) that can be used for such purpose.

The Revolute Joint V



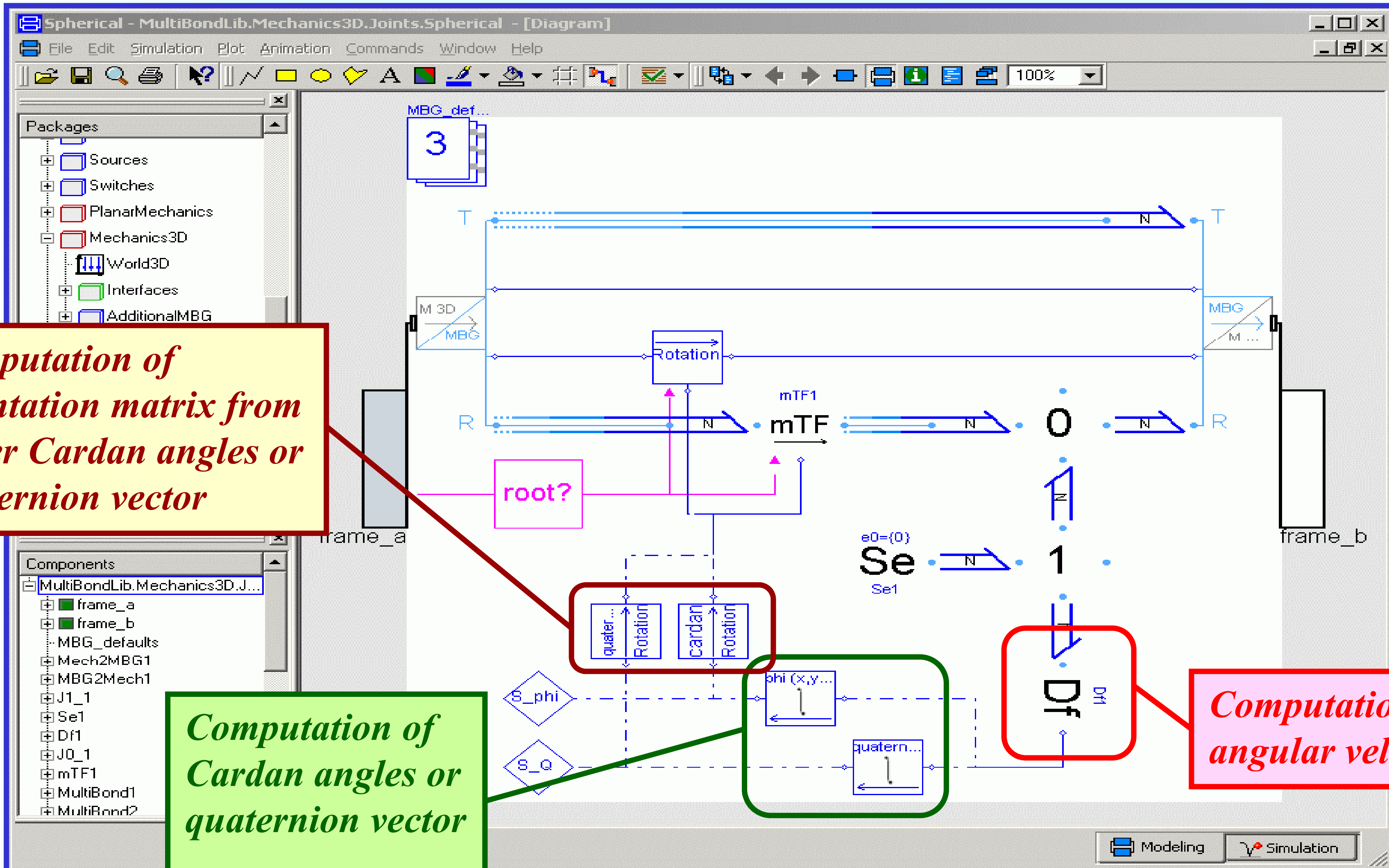
- The computational causality is determined in the equation window. The graphical representation of the root? is only a mnemonic, visible easily, since the connections are not connected through to the connectors.



The Spherical Joint

- A spherical joint is similar to a revolute joint in that it only rotates.
- Yet, a spherical joint has three degrees of freedom, rather than only one. Any rotation is possible.
- Hence we cannot compute easily a plane perpendicular to the rotation, and therefore, the planar rotation method is not suitable.
- We can use either Cardan angles or quaternions. Each method requires to represent the correct vector of angles in a different way.
- Hence the bond graph only determines the angular velocities, using a ***Df*** element. The Cardan angles or the quaternion vector respectively are integrated from the velocities using special elements.

The Spherical Joint II



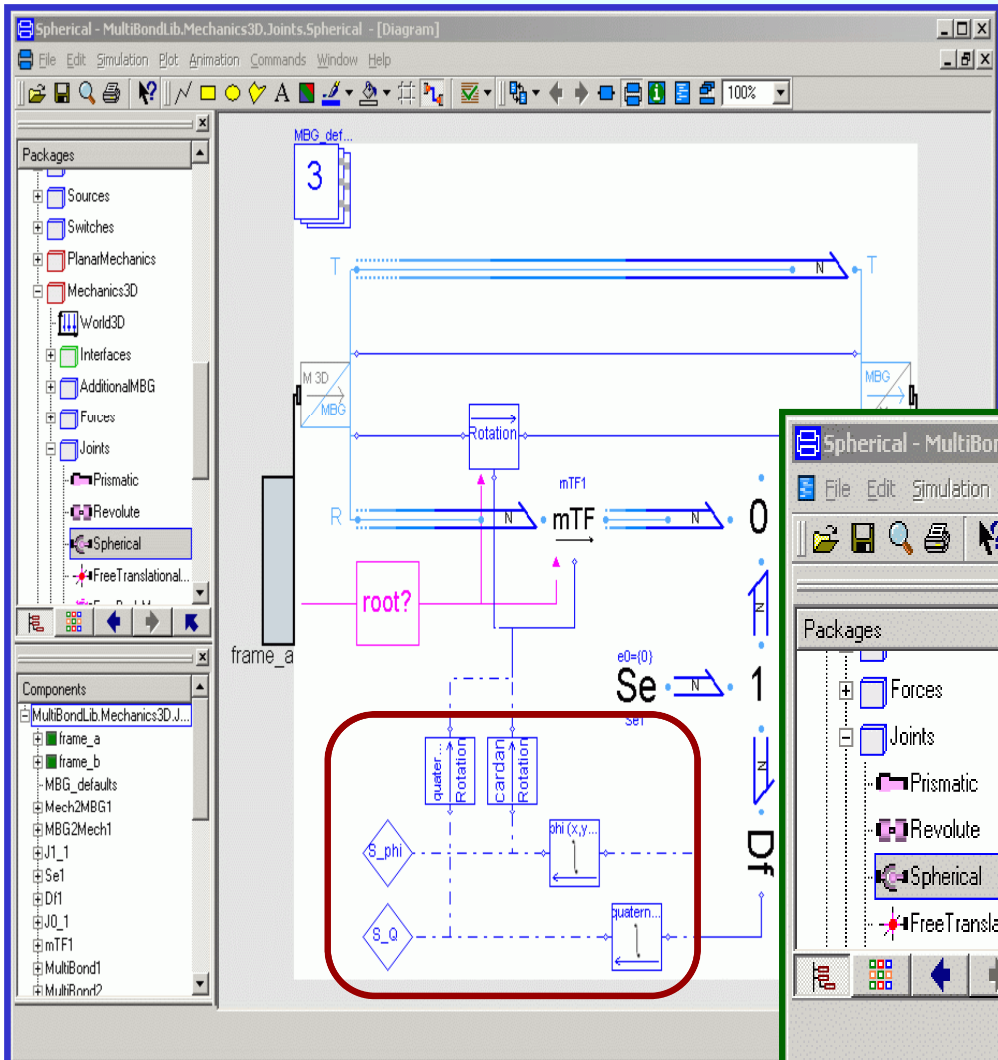
Computation of orientation matrix from either Cardan angles or quaternion vector

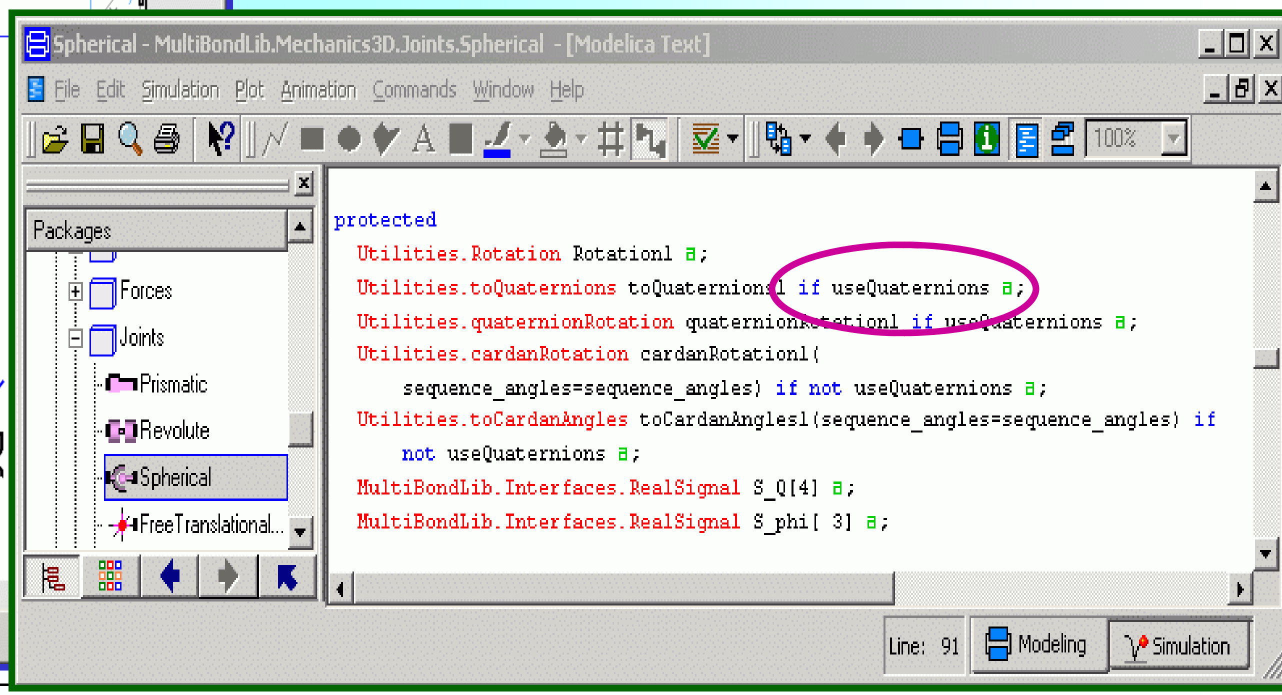
Computation of Cardan angles or quaternion vector

Computation of angular velocities

The Spherical Joint III

- In ***Dymola***, it is possible to invoke a model conditionally.
- This can only be seen and done in the expanded view of the equation window.
- Connections to conditional models are interpreted as conditional.



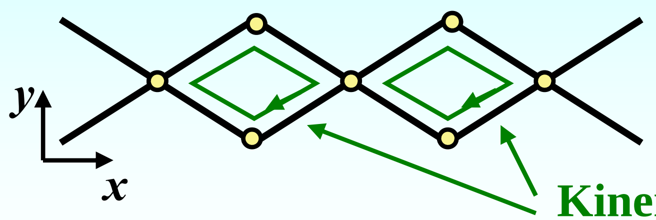


The Selection of State Variables

- When dealing with multi-body systems, it matters greatly, which variables are being selected as state variables, as this will influence strongly the efficiency of the generated simulation code.
- If we choose our state variables wisely, the number of simulation equations of a tree-structured multi-body system grows linearly in the number of degrees of freedom.
- If we make a poor choice of our state variables, the number of run-time equations grows with the fourth power of the number of degrees of freedom.
- To this end, ***we should use the relative positions and velocities of joints as our preferred state variables***.

Closed Kinematic Loops

- The tools that we have learnt to use so far suffice for modeling tree-structured multi-body systems.
- Yet, many practical systems contain ***closed kinematic loops***.
- Kinematic loops offer an improved mechanical stability to a system, and are therefore used frequently.
- As an example, consider a half-timber house. The wooden frames are over-determined and provide the necessary stability that prevents the house from collapsing.
- A typical example may be the pentagraph mounting of an office lamp that can be moved around to provide light where it is most needed.

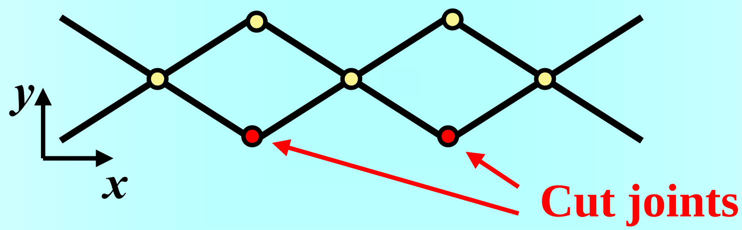


Closed Kinematic Loops II

- Unfortunately, closed kinematic loops lead to redundant system descriptions.
- The reason is that a closed connection exists from one root to another.
- Hence the positions and velocities along this closed path can be computed in two ways, starting with either of the two roots.
- There exist a number of different algorithms that can be used to get around this problem.

Closed Kinematic Loops III

- To avoid the redundancies associated with these closed connections, it is possible to declare one joint of each kinematically closed loop as ***cut joint***.



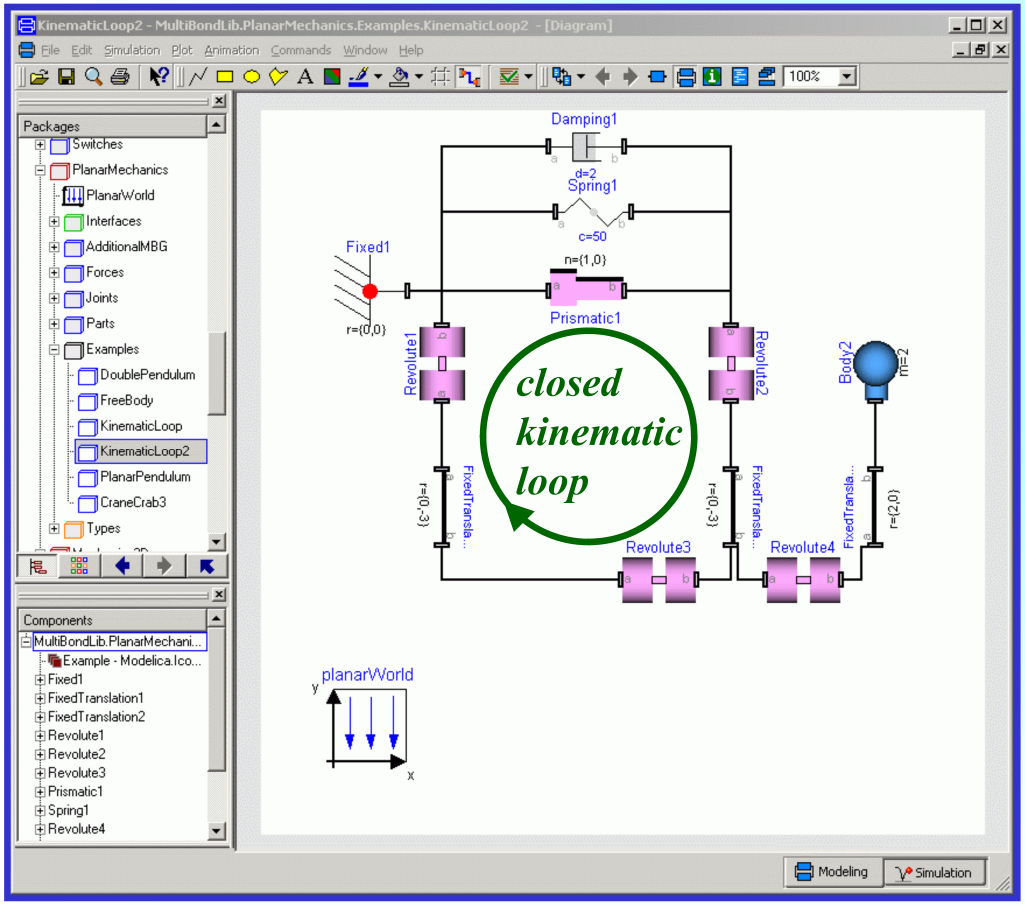
- Cut joints do not define any integrators, thereby eliminating the introduction of redundant equations.
- ***Dymola*** used to support this idea by offering a number of ***cut joints***.
- The approach was given up, because it required a manual analysis of the system topology by the user.

Closed Kinematic Loops IV

- A different solution is to cut the closed kinematic loops somewhere at a connection, rather than within a joint.
- The disadvantage of this method is that we allow the use of surplus integrators that wouldn't be truly necessary.
- This reduces the efficiency of the resulting simulation code a little, but the approach is much more convenient, as it can be automated.

Planar Kinematic Loop – An Example

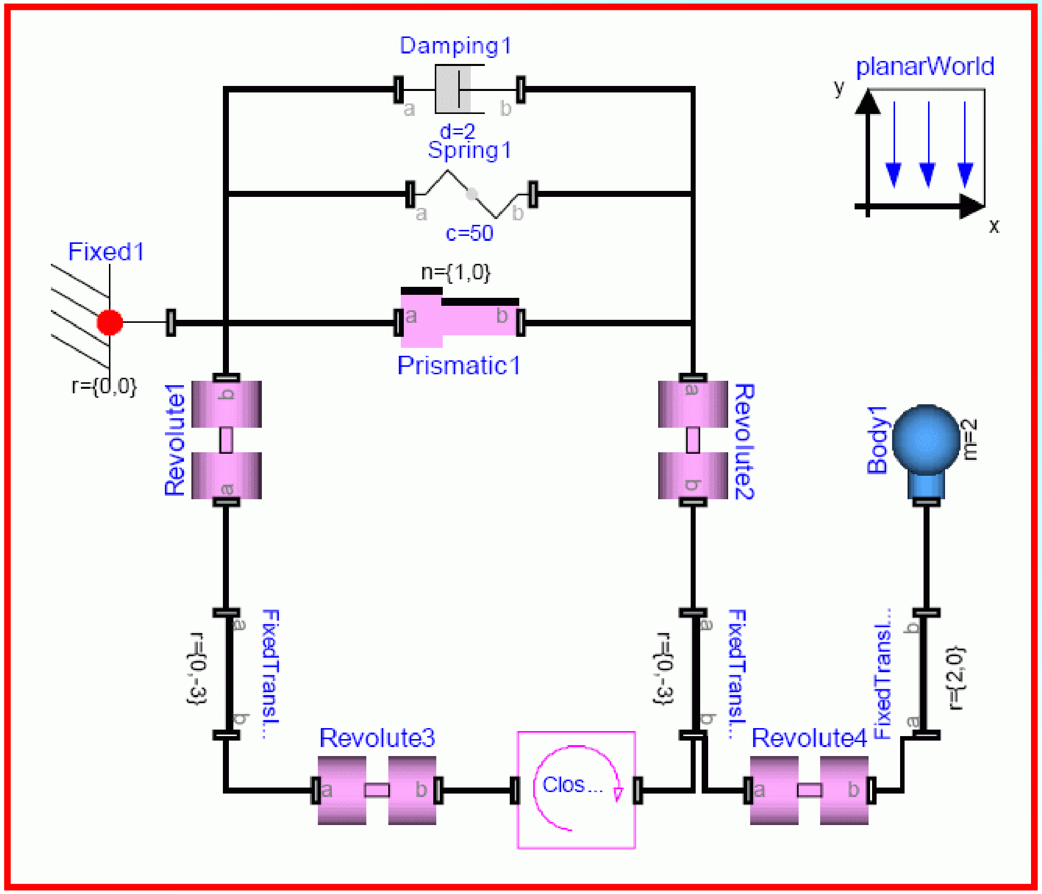
- Let us start with an example of a planar kinematic loop:



- Each of the revolute joints defines one mechanical degree of freedom.
- The prismatic joint takes two of these degrees of freedom away.
- A closed kinematic loop is being formed by three of the revolute joints, two fixed translations, and the prismatic joint.
- We need to break that loop somewhere.

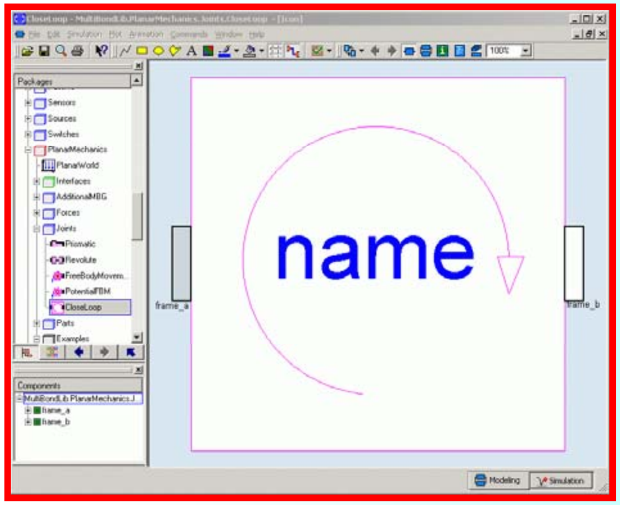
Planar Kinematic Loop – An Example II

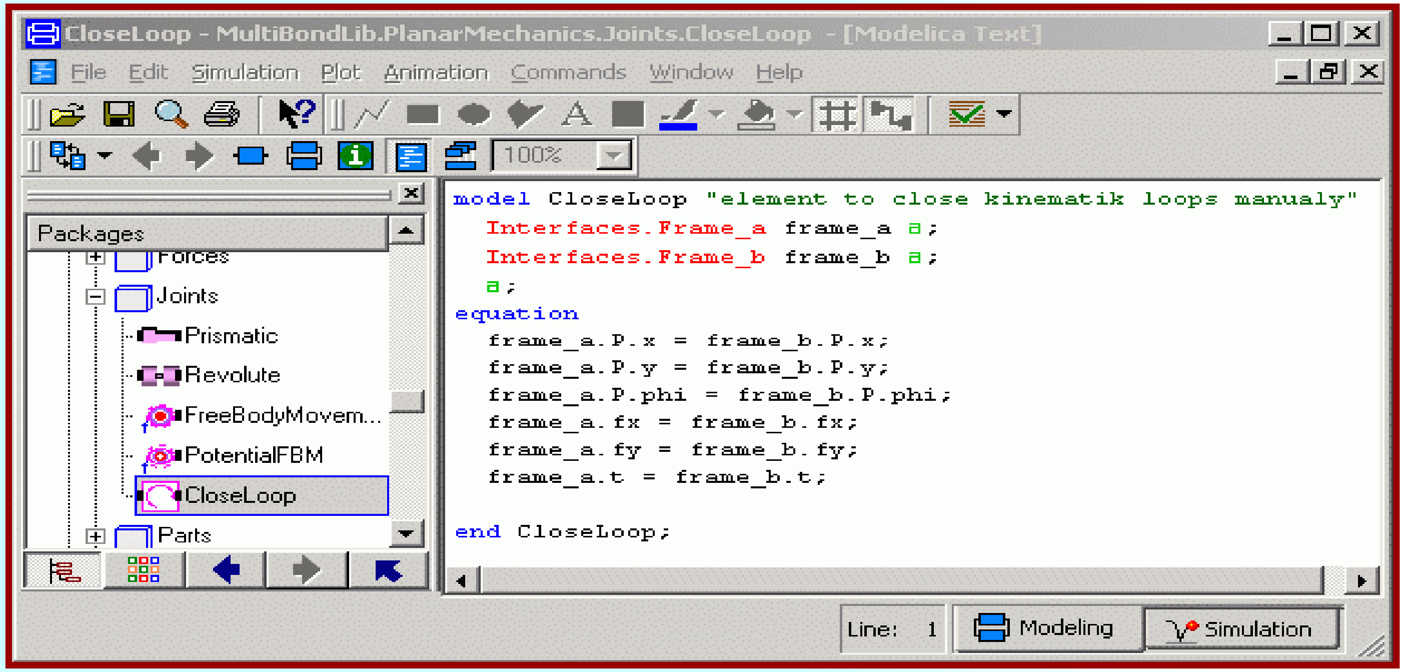
- We can introduce a loop-breaker model anywhere along the loop:



- The loop-breaker model avoids connecting the velocities on both sides.
- This returns enough freedom to the system to eliminate the redundancy among the equations.

The ClosedLoop Model

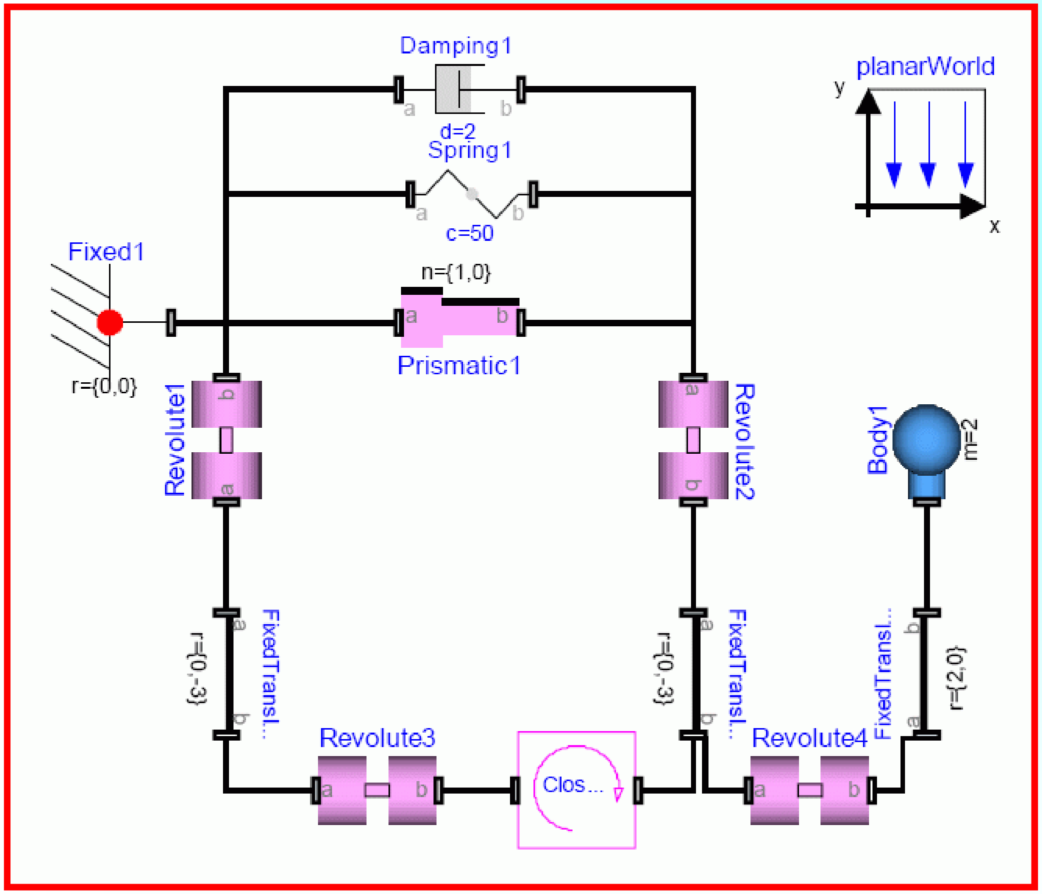


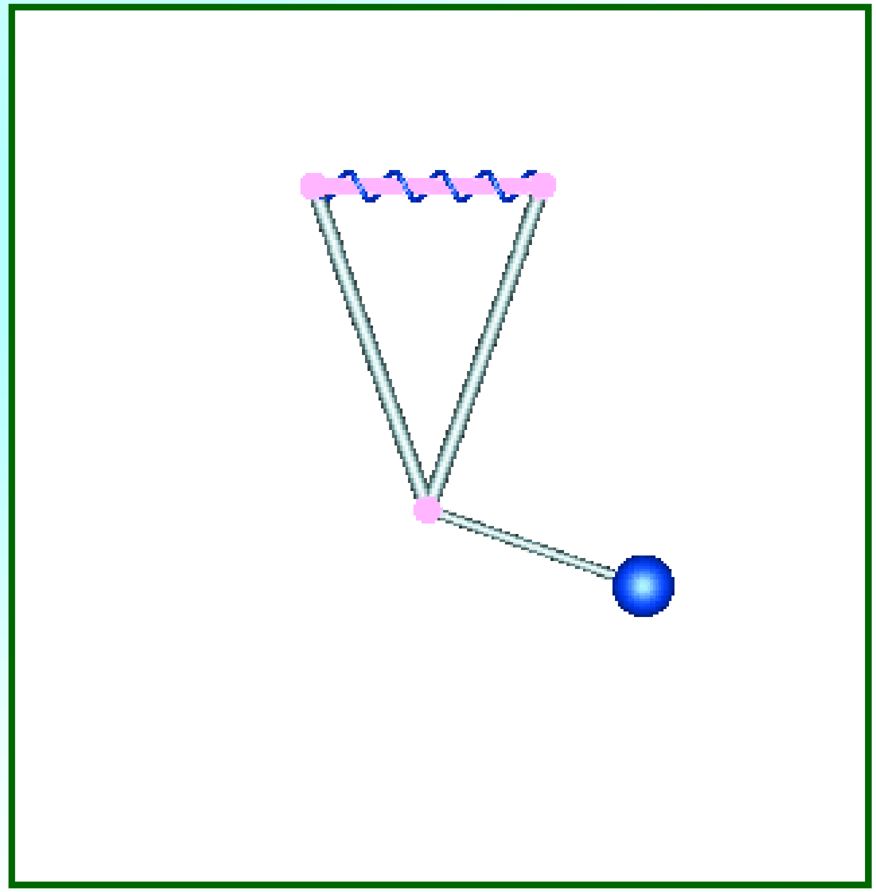


- By eliminating the velocity equations on the connection, we allow the velocities to be computed separately by differentiation on both sides, although we are in fact computing the same quantity twice.

Planar Kinematic Loop – An Example III

- Let us simulate the model to see what it is doing:





Planar Kinematic Loop – An Example IV

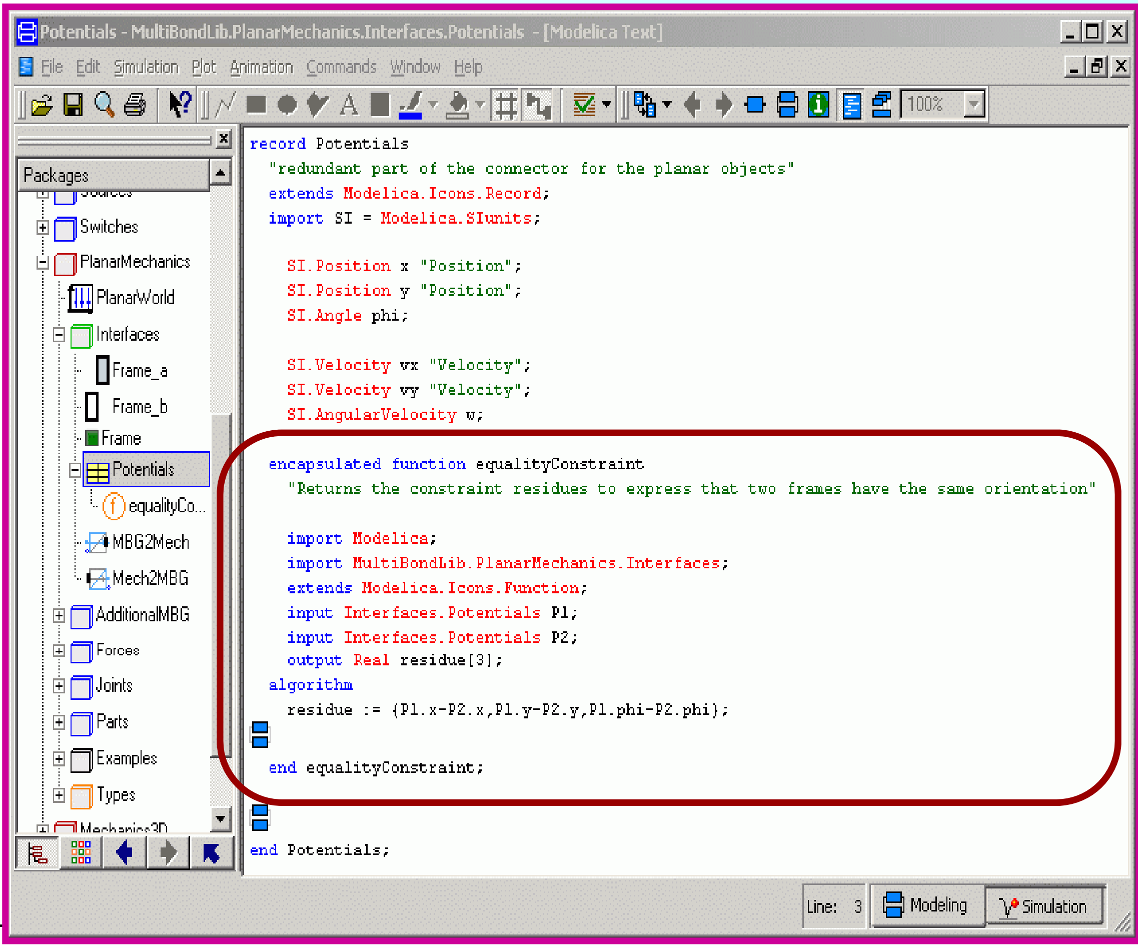
- With a little bit of help, ***Dymola*** is capable of inserting the loop-breaker model on its own.
- To this end, the user needs to declare components that can form a closed kinematic loop by use of the ***defineBranch()*** function:

```
equation
  defineBranch(frame_a.P, frame_b.P);
```

- This is necessary anyway for the ***rooted()*** function to work properly.
- On its own, ***Dymola*** will cut the loop open as far away from the root as it can.
- In this way, the two paths are made equally long.

Planar Kinematic Loop – An Example IV

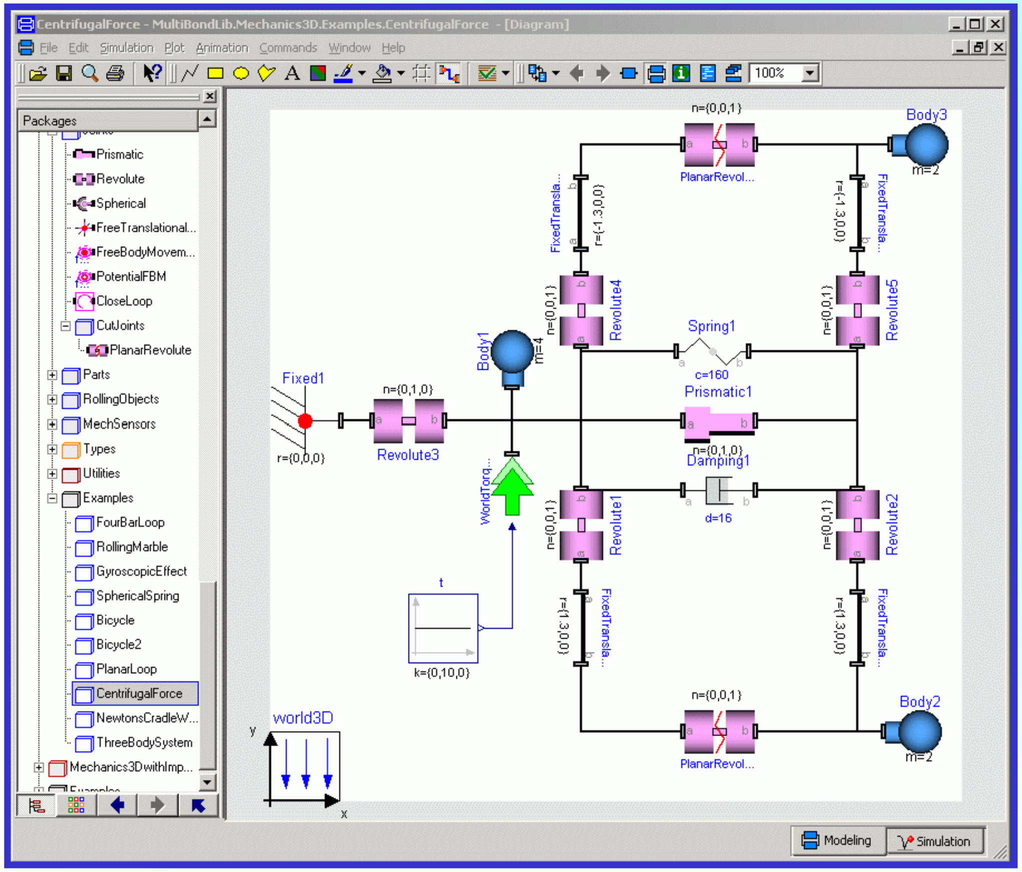
- In fact, ***Dymola*** doesn't actually insert the ***ClosedLoop*** model.
- It solves the problem in a different way:



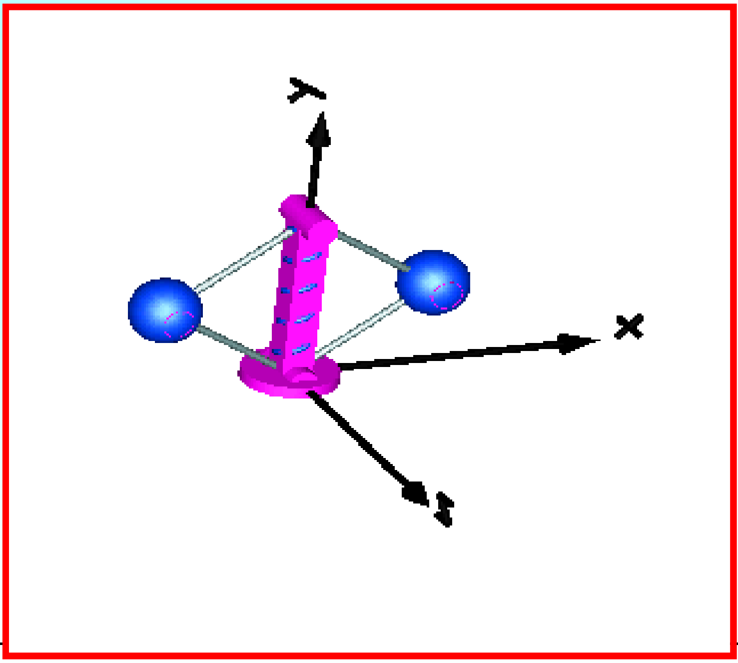
- At the place, where ***Dymola*** decides to break the loop, it uses an alternate set of equations, as formulated in the encapsulated ***equalityConstraint()*** function.
- A residue vector is being introduced that effectively provides the additional degrees of freedom needed to get around the redundancy problem.

Planar Loops in 3D Mechanics

- The automated loop-breaking algorithm doesn't always work. The following example demonstrates the problem:



- Let us look first at the simulation results to better understand what this system is doing:

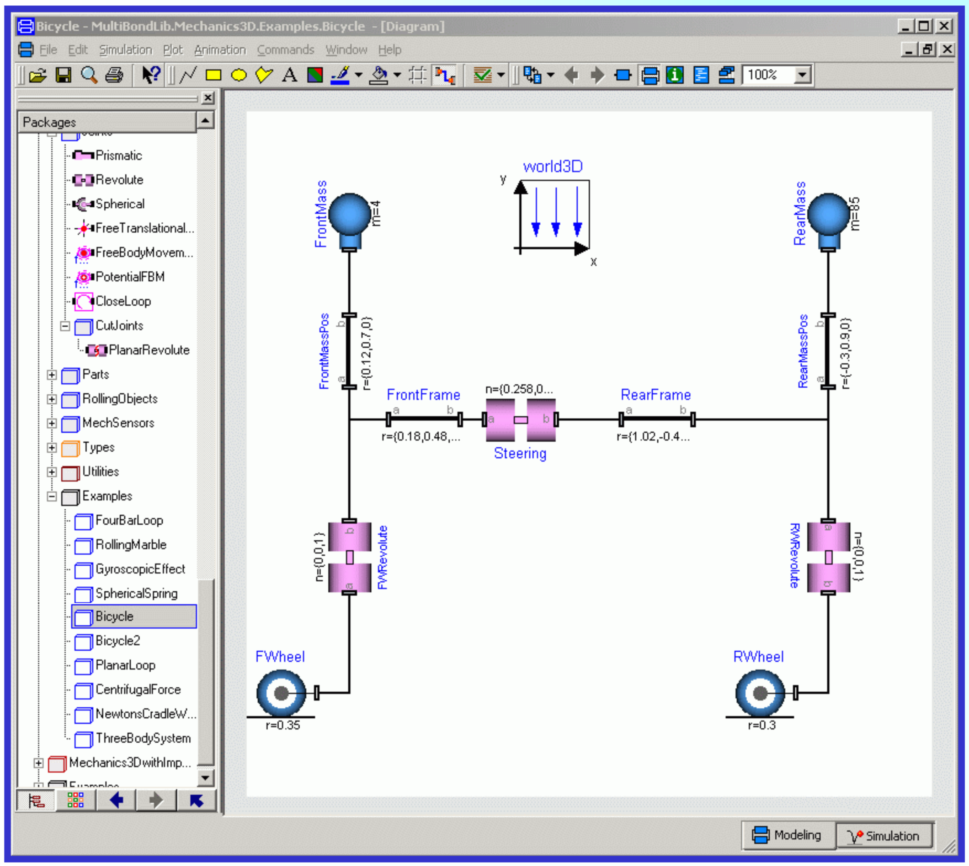


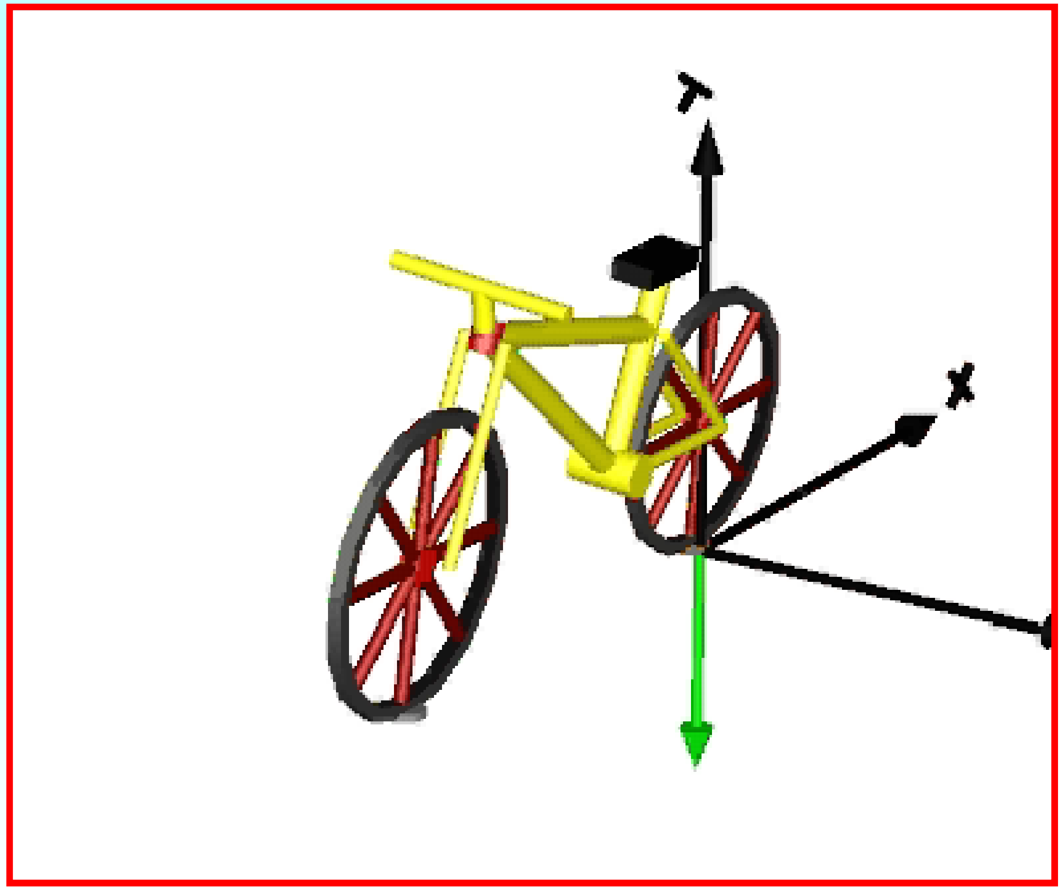
Planar Loops in 3D Mechanics II

- The problem is the following: There are two planar closed kinematic loops each defined by three revolute joints and a prismatic joint.
- Two revolute joints with the same rotation axis suffice to restrict the freedom of motion to a single axis. The constraint of the third revolute joint is therefore superfluous, which leads to an additional redundancy that doesn't get removed by the automated loop-breaker algorithm.
- For this reason, a special ***revolute cut joint*** was introduced in the 3D mechanics library that can be used to break ***planar closed kinematic loops in 3D mechanics***.

A Bicycle – Another Example

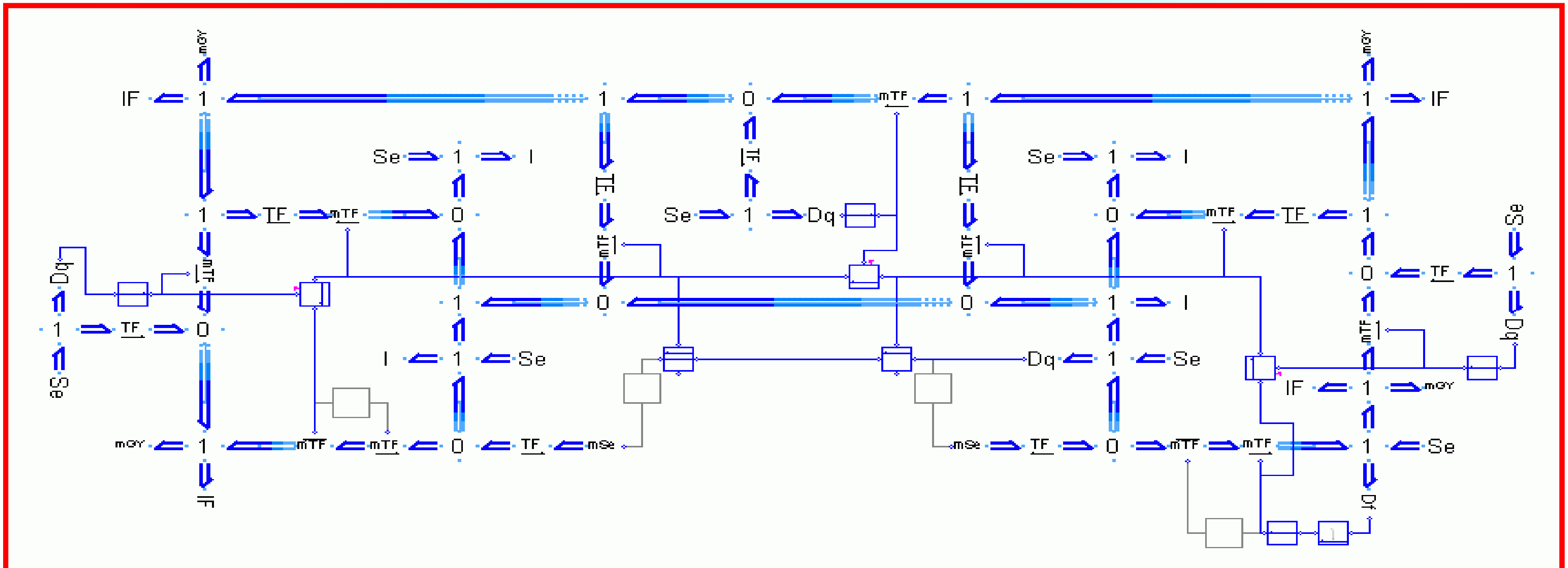
- Let us now model a bicycle using the built-in wrapped multi-body component models of the 3D mechanics sub-library of the multi-bond graph library:





A Bicycle – Another Example II

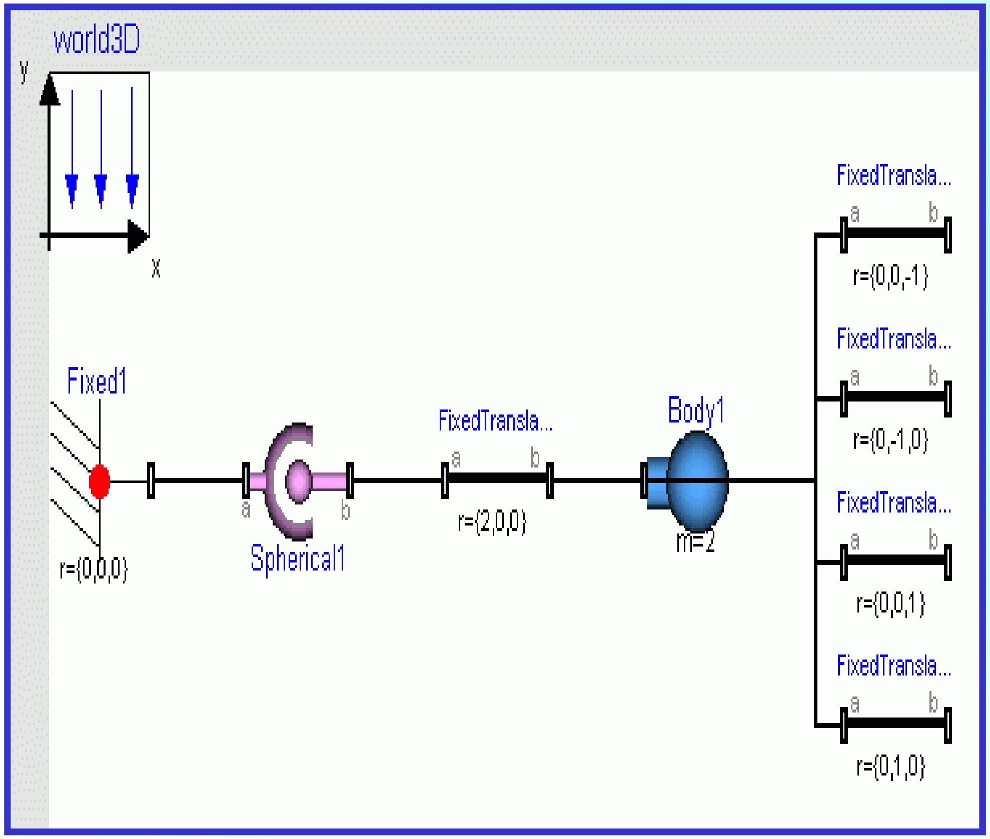
- The use of the multi-bond graph library for modeling systems from 3D mechanics is quite simple. Here is the corresponding multi-bond graph model without wrapping:

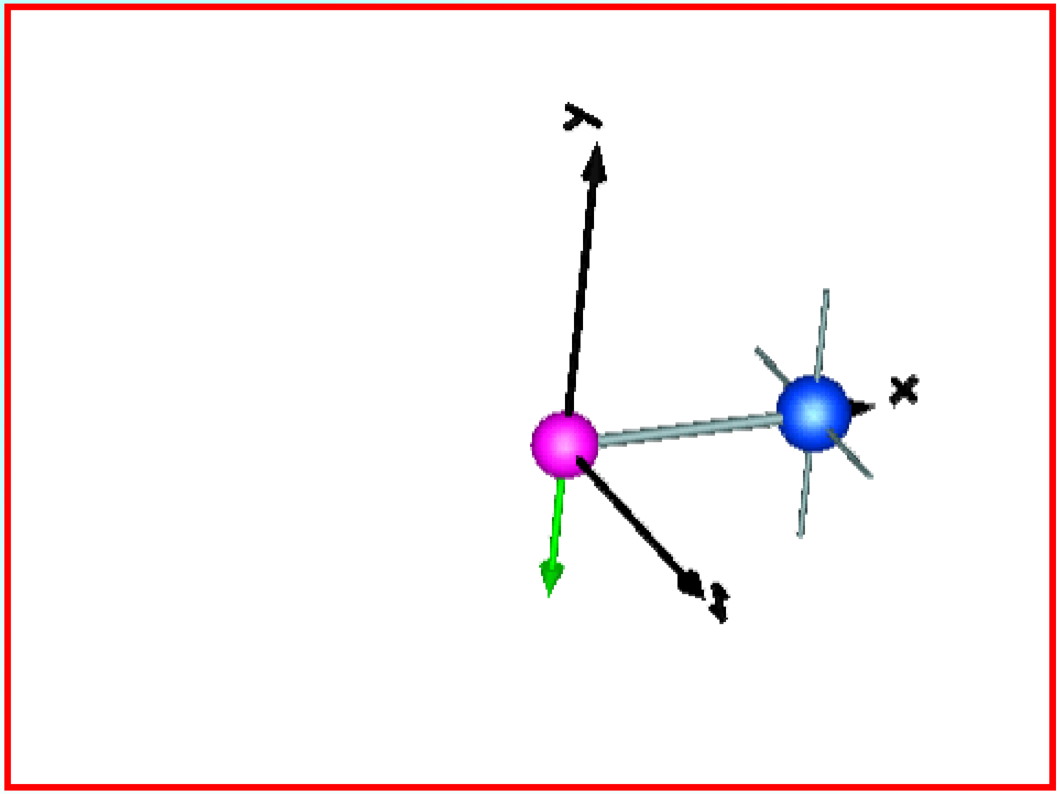


- No comments are necessary!

Accuracy of Simulation Results

- Let us simulate yet another model. It shows beautifully the gyroscopic effects.





Accuracy of Simulation Results II

- The model was simulated three times, while changing the parameter settings:

	good cardan angle seq.		quaternions		bad cardan angle seq.	
tolerance	error	steps	error	steps	error	steps
$1.0 \cdot 10^{-4}$	$4.9 \cdot 10^{-4}$	$2.9 \cdot 10^{3}$	$5.0 \cdot 10^{-3}$	$2.6 \cdot 10^{4}$	$1.8 \cdot 10^{-0}$	$5.4 \cdot 10^{4}$
$1.0 \cdot 10^{-6}$	$9.7 \cdot 10^{-6}$	$6.2 \cdot 10^{3}$	$3.1 \cdot 10^{-4}$	$4.8 \cdot 10^{4}$	$2.9 \cdot 10^{-4}$	$9.5 \cdot 10^{4}$
$1.0 \cdot 10^{-8}$	$1.2 \cdot 10^{-7}$	$1.4 \cdot 10^{4}$	$1.1 \cdot 10^{-5}$	$8.4 \cdot 10^{4}$	$3.5 \cdot 10^{-5}$	$2.0 \cdot 10^{5}$
$1.0 \cdot 10^{-10}$	$1.2 \cdot 10^{-7}$	$2.3 \cdot 10^{4}$	$1.1 \cdot 10^{-6}$	$1.4 \cdot 10^{5}$	$3.0 \cdot 10^{-6}$	$4.4 \cdot 10^{5}$

- Making the correct choice of which method to use for computing the orientation matrix of the spherical joint had a huge influence both on the execution speed (number of integration steps) and on the accuracy of the simulation.
- It is sometimes worthwhile experimenting with these model parameters to get the most out of the simulation.

Efficiency of Simulation Run

- The following table compares the efficiency of the simulation code obtained using the multi-body library contained as part of the standard ***Modelica*** library with that obtained using the 3D mechanics sub-library of the multi-bond graph library.

	MultiBody			Mechanics3D		
experiment	linear equ.	non-lin. equ.	steps	linear equ.	non-lin. equ.	steps
Pendulum	0	0	207	0	0	207
Double pendulum	2	0	549	2	0	549
Crane crab.	2	0	205	4	0	205
Gyroscopic exp. with Cardans	2,2	0	294	3,2	0	294
Gyroscopic exp. with Quaternions	4,3	4	24438	4,2	4	25574
Planar Loop	8,2	2	372	6,2,2	2	372
Centrifugal exp.	10,2,2	2,2	70	16,2,2	2,2	70
Four bar loop*	10,5,2	5	446	9,5,2	5	625
Bicycle*	15,5,3,2	1	97	15,3	1	84

References I

- Zimmer, D. (2006), *A Modelica Library for MultiBond Graphs and its Application in 3D-Mechanics*, MS Thesis, Dept. of Computer Science, ETH Zurich.
- Zimmer, D. and F.E. Cellier (2006), "The Modelica Multi-bond Graph Library," *Proc. 5th Intl. Modelica Conference*, Vienna, Austria, Vol.2, pp. 559-568.

References II

- Cellier, F.E. and D. Zimmer (2006), "Wrapping Multi-bond Graphs: A Structured Approach to Modeling Complex Multi-body Dynamics," *Proc. 20th European Conference on Modeling and Simulation*, Bonn, Germany, pp. 7-13.

- Andres, M. (2009), *Object-Oriented Modeling of Wheels and Tires in Dymola/Modelica*, MS Thesis, Mechatronics Program, Vorarlberg University of Science and Technology, Dornbirn, Austria.

References III

- Andres, M., D. Zimmer, and F.E. Cellier (2009), "Object-Oriented Decomposition of Tire Characteristics Based on Semi-empirical Models," *Proc. 7th International Modelica Conference*, Como, Italy, pp. 9-18.

- Schmitt, T. (2009), *Modeling of a Motorcycle in Dymola/Modelica*, Mechatronics Program, Vorarlberg University of Science and Technology, Dornbirn, Austria.

- Schmitt, T., D. Zimmer, and F.E. Cellier (2009), "A Virtual Motorcycle Rider Based on Automatic Controller Design," *Proc. 7th International Modelica Conference*, Como, Italy, pp. 19-28.