Debugging Initialization in Dymola 2019

When starting a simulation of a dynamic model the states of the system must be initialized. The Modelica language permits flexible specification of initial conditions including difficult non-standard initialization problems arising in industrial applications.

If parts are missing in the user-specified initial conditions Dymola automatically completes them. Then it employs symbolic and numeric techniques to try to solve the complete initialization problem.

This special analysis is also used to provide the user with helpful debugging information if the initialization problem is not well-posed. This debugging functionality has been further enhanced in Dymola 2019.

Introduction

The initialization problem of a Modelica model may take many forms. Powerful language constructs allow flexible, non-standard specification of initial conditions. Examples include steady-state initialization, defining initial conditions on non-state variables and initialization of hybrid continuous-discrete systems.

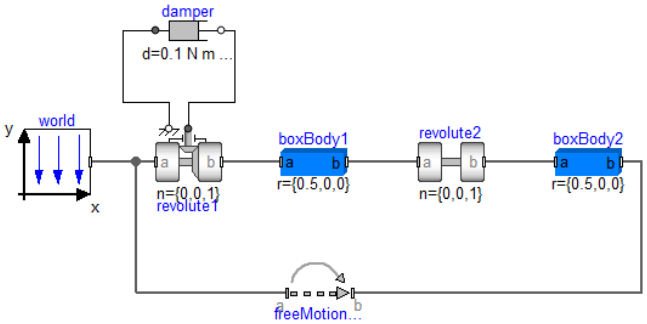
After Dymola has used symbolic techniques to break down an initialization problem, numeric techniques must sometimes be employed, for example to solve nonlinear algebraic loops.

A failure to initialize a model may be caused by modeling errors resulting in an initialization problem lacking solution. Dymola can catch many model problems at an early stage (often during translation) and give detailed error diagnostics. However, we here present an example model where the error is detected later during the initialization process. The new features of Dymola 2019 will help us debug the model.

For a detailed introduction on initialization in Dymola and more debugging help please refer to the Dymola User Manual. Especially, the sections on the *homotopy* operator and the *Save Start Values in Model* functionality. Proper usage of these can greatly increase the robustness and reliability of the initialization process.

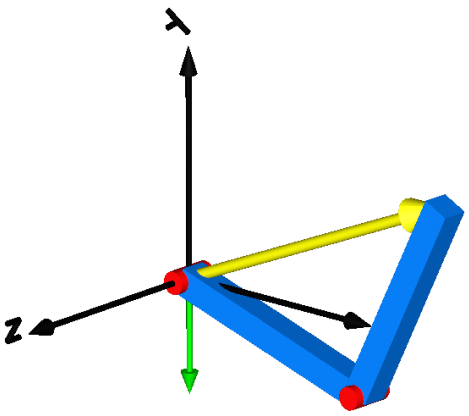
A non-standard initialization example

To demonstrate some of the new debugging features introduced in Dymola 2019 we consider the multibody system example *DoublePendulumInitTip* from the Modelica Standard Library.



The model consists of a double pendulum subject to indirect initialization of the states. The initial configuration is defined by the freeMotionScalarInit component (bottommost in the figure). This component is used to put the tip of the pendulum at the coordinates $(\mathbf{0.7}, \mathbf{0.3}, \mathbf{0.0})$ m and all velocities to zero.

Dymola automatically finds appropriate angles for the revolute joints to satisfy the initial conditions. The yellow arrow in the figure illustrates the desired initial position of the pendulum tip. The bodies and revolute joints visualize Dymola's solution of the initialization problem.



Debugging an initialization failure

We choose another initial configuration and try to simulate. This time the integrator fails to initialize the model. An error message in the simulation log tells us

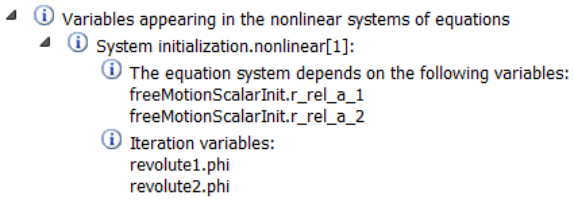
```
ERROR: Failed to solve nonlinear system using Newton solver.
  During initialization at time: 0
  Tag: initialization.nonlinear[1]
```

A continued reading of the message (not shown here) reveals that Dymola created a nonlinear system of equations to find the initial values for the angles

`revolute1.phi` and `revolute2.phi`

of the two revolute joints.

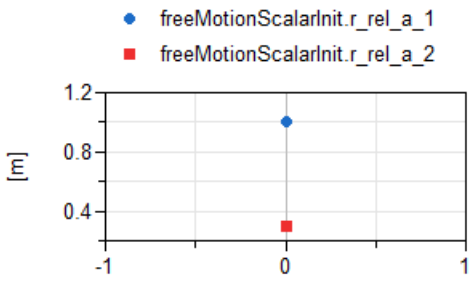
Furthermore, we can read that Dymola suspects that the initialization problem lacks solutions. To debug, let's first learn more about this nonlinear system of equations. Go to Simulation>Setup… and enable "List non-linear iteration variables" under the Translation tab. After re-running the simulation we find more useful information about the nonlinear system under "Statistics" in the translation log.



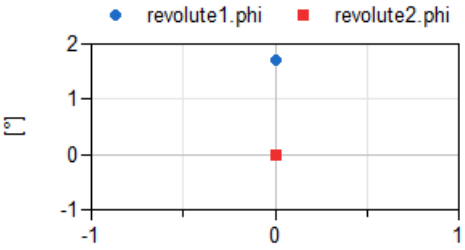
The information provided in the translation log has been extended in Dymola 2019 to also include dependencies. We conclude that the failing nonlinear system is directly dependent on the desired initial x- and y-coordinates for the tip of the double pendulum.

Plotting an initialization failure

As a new feature of Dymola 2019 the results of failed initializations are now available in the Variable Browser ready for plotting. Enable "Store variables after failed initialization" under the Debug tab in the simulation setup. We can for example plot our desired initial position for the tip of the pendulum.

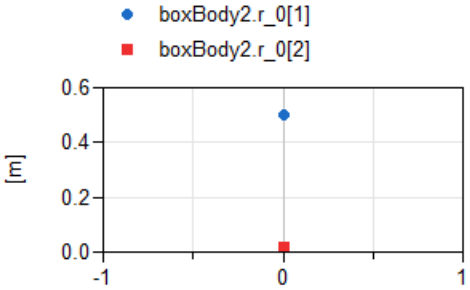


The last values of the iteration variables are also interesting for our debugging effort.



The angle `revolute2.phi` is almost zero indicating that a straight line was the last configuration tried during the attempt to solve the nonlinear system. Indeed, in the model diagram above we see that the length of each body is $\mathbf{0.5}$ m. Thus, the pendulum is too short to satisfy the desired initial configuration $(\mathbf{1.0}, \mathbf{0.3}, \mathbf{0.0})$ m, even when the bodies are put in a straight line.

To learn even more about the translated model and the initialization process we can enable "Generate listing of translated Modelica code in dsmodel.mof" under the Translation tab in the simulation setup. In dsmodel.mof we can for example read that the position of the outermost body is updated during the iterations to solve the nonlinear system. The resulting global x- and y-coordinates of this body can also be plotted.



For ease of presentation we have here used a simple example model. The initialization failure was directly dependent on the problematic parameter values. However, the new features truly shine when debugging complex models encountered in industrial applications where the connection between the point of failure and the root cause is not that apparent. By plotting variables from the failure and up the dependency chain the problem can be traced back to its root cause.

Conclusion

Several new features for debugging initialization failures have been introduced in Dymola 2019. Among them we find enhanced log messages, extended information about nonlinear system of equations and the capability to plot the results of a failed initialization.