January 29, 2018

LINKÖPING UNIVERSITY

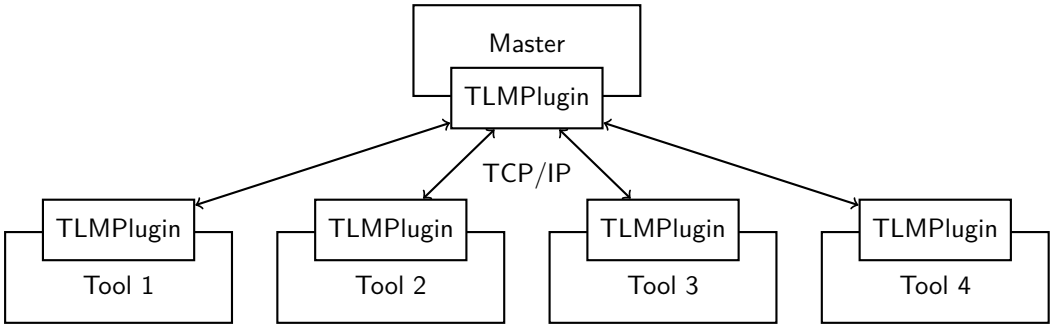
Getting Started With TLM-based Co-simulation

Introduction

With transmission line modelling (TLM), models can be decoupled using physically motivated time delays. Hence, the time delay can be seen as a physical property of the model. This elliminates the need for numerical delays. Thereby, no numerical errors will be introduced. The TLM equations for a mechanical coupling in one dimension is shown below:

$$F_1(t) = c_1(t) + Z_c q_1(t)$$
$$F_2(t) = c_2(t) + Z_c q_2(t)$$

A framework for asynchronous TLM-based co-simulation has been developed by SKF for multi-body simulation of roller bearings. A graphical user interface in OpenModelica Connection Editor has been developed in cooperation with Linköping University. Lately the framework has been extended with support for directional connetions, 1D interfaces, other physical domains and sub-model parameters. Socket communication is used for communication between the master and the external tools.



This tutorial will demonstrate the framework with two examples. First, two models will be coupled with a rotational 1D connection. Second, two multi-body models will be connected between two 3D interfaces.

System Configuration

The "TLMPlugin" files will be distributed during the tutorial.
OpenModelica is a free open-source Modelica environment developed at Linköping University. Install the latest nightly build, which contain support for composite modeling:

http://www.openmodelica.org

You will also need the latest version of Hopsan, an open-source TLM-based system simulation tool developed at Linköping University:

http://flumes.iei.liu.se/hopsan/files/releases/official/

OpenModelica Connection Editor (OMEdit) is used a the graphical interface. It must know the path to the TLMPlugin files.

1. Start OpenModelica Connection Editor
2. Open "Tools → Options → TLM"
3. Make sure all three paths are corrrect

Exercise 1: 1D Connection

The TLM method requires hard typed physical interfaces. For 1D rotational connections, three variables are required:

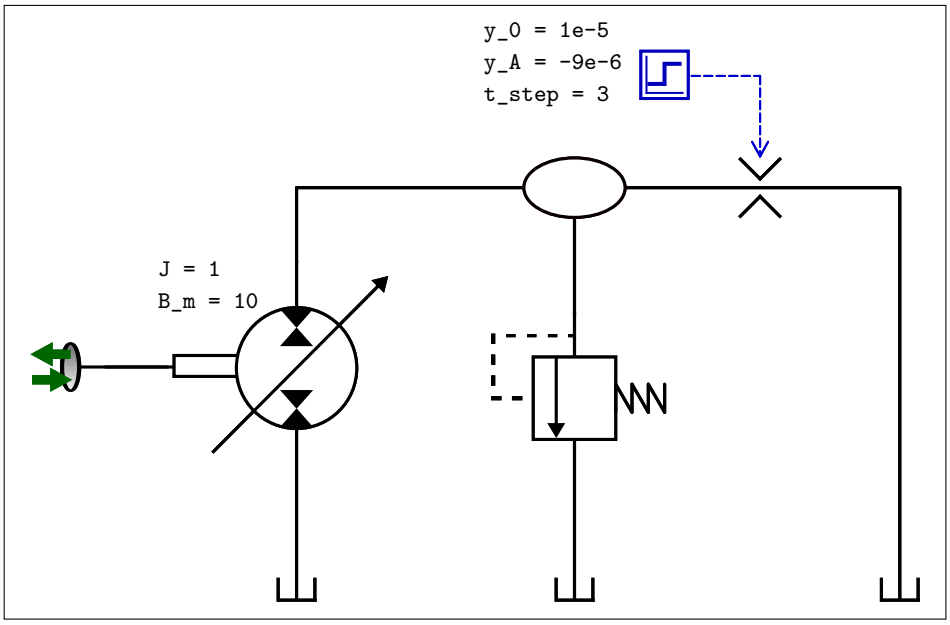
Variable	Unit	Dimensions
Angle	[rad]	1
Speed	[rad/s]	1
Torque	[Nm]	1

Similar variables are used for other 1D domains such as linear mechanics (position, velocity & torque) and hydraulics (volume, flow and pressure).

1. **Build the Hopsan model**
   The Hopsan model will consist of a hydraulic circuit. A pump, driven by the OpenModelica motor model, is used to pressurize a volume. The volume is connected to a turbulent orifice, representing the load, and a pressure relief valve. Use a step source to reduce the orifice area from 10 mm^2 to 1 mm^2 after 3 seconds. Set the inertia of the pump to 1 Nms^2. The following components are used in the model:

   Connectivity → Mechanic Rotational Interface of Q-type
   Hydraulic → **Pumps&Motors** → Q-type Variable Machine
   Hydraulic → **Restrictors** → Turbulent Orifice
   Hydraulic → **Sources & Sinks** → C-type Tank
   Hydraulic → **Valves** → **PressureControlValves** → Adjustable Pressure Relief Valve
   Hydraulic → **Volumes&Lines** → Hydraulic Volume Multi Port
   Mechanic → **Rotational** → Rotational Inertia
   Signal → **Source & Sinks** → Step

   When done, save the model as "signal.hmf".



2. **Export the Hopsan model as FMU**
   Click on the Export Functional Mockup Unit toolbar button. Use FMI 2.0 for co-simulation and export to an empty folder.

   FMU ***Export Functional Mockup Unit***

3. **Write the OpenModelica model**
   The Modelica model will be written by hand. It will consist of a torque with a proportional speed control, representing a simple motor. The speed control will generate a torque, acting on an inertia with damping. A counteracting torque will be received from the Hopsan model.

   Below is an example of how the code could look like. Here `w`, `phi` and `t2` are the TLM variables; speed and angle as output and torque as input. Furthermore, `t1` is the torque generated by the controller. Finally, `Kp` is the controller gain and `w_ref` the reference (desired) speed.

```
model motor
  output Real w(start=0);
  output Real phi(start=0);
  input Real t2;
  Real t1(start=0);
  parameter Real J=1 "Inertia";
  parameter Real B=10 "Viscous friction";
  parameter Real w_ref=-200 "Reference speed";
  parameter Real Kp=50 "Controller gain";
equation
  der(w)*J+der(phi)*B = t1-t2;
  w=der(phi);
  t1=Kp*(w_ref-w);
end motor;
```

   Create a new model in OMEdit, name it "motor", write the code and save it.

4. **Export the OpenModelica model as FMU**
   Open "Tools → Options → FMI". Change the settings to export FMI 2.0 for Model Exchange. Close the dialog. Click on "FMI → Export FMU" to start the export.

5. **Create the composite model**
   Click on "File → New MetaModel" to create a new composite model. Right-click on the workspace and rename the model to "MotorSystemModel". **Save the composite model to a new empty folder.**

6. **Load FMUs as external models**
   Load the two FMUs by clicking on "File → Load External Model". The files should appear in the library on the left side.

7. **Add the FMUs to the composite model**
   Add the FMUs to the composite model by dragging and dropping. **Save the composite model.** Each FMU is now copied to a new folder in the composite model directory.

8. **Write FMI configuration files**
   It is necessary to define ports and point out the TLM variables in each FMU. This is done by a small configuration file. Create a file called `fmi.config` in each external model folder. Open each file in a text editor and enter the following text:

```
substeps,10
name,tlm
domain,Rotational
dimensions,1
causality,Bidirectional
position,1
speed,2
force,3
```

The number after "substeps" is used with FMI for co-simulation. It specifies the number of substeps taken during each TLM step. A higher number improves stability, but has a negative effect on performance. The "name", "domain", "dimensions" and "causality" entries are the interface specifications. The numbers after position, speed and force represent the value references of the variables in the FMU. Hence, they must be modified to match the contents of each FMU. Open each FMU with a zip tool. Then open modelDescription.xml in a text editor.

For the motor FMU, locate the variables named "phi", "w" and "t2". For the system FMU, look for variables ending with "_P1_a", "_P1_w" and "_P1_c". Identify the value reference as shown below:

```
<ScalarVariable
  name="phi"
  valueReference="0"
  variability="continuous"
  causality="output"
  initial="exact">
  <Real start="0.0"/>
</ScalarVariable>
```

Name

Value reference

9. **Fetch interface data**
   Yet, OpenModelica has no knowledge about the interfaces or parameters in the external models. To obtain these information, click on the "Fetch interface data" button.

   Fetch Interface Data

10. **Connect interfaces**
    Now one interface should have appeared on each external model. Connect them by dragging a connector from one interface to the other. Use the following connection parameters:

Variable	Value
Delay	1e-4
Zf	100
alpha	0.9

11. **Simulate**
    Open the "Simulation Parameters" dialog and set stop time to 5 seconds.

    Simulation Parameters

    Now click on the "TLM Co-Simulation Setup" icon. A dialog with simulation settings appear. Click on "Simulate".

    TLM Co-Simulation Setup

12. **Plot results**
    When simulation is complete, the plotting perspective should open automatically. You can plot the interface variables from the variables tree.

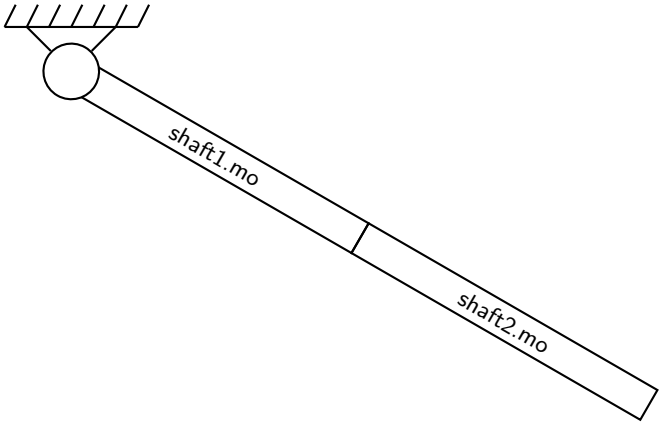
    CSV files with internal variables in each FMU has also been generated. Look in the external model folders for "logdata.csv".

Exercise 2: 3D Connections

Mechanical 3D connections consist of 24 variables, representing 6 degress of freedom:

Variable	Unit	Dimensions
Position	[m]	3
Orientation	[-]	3x3
Speed	[m/s]	3
Angular speed	[rad/s]	3
Force	[N] & [Nm]	3+3

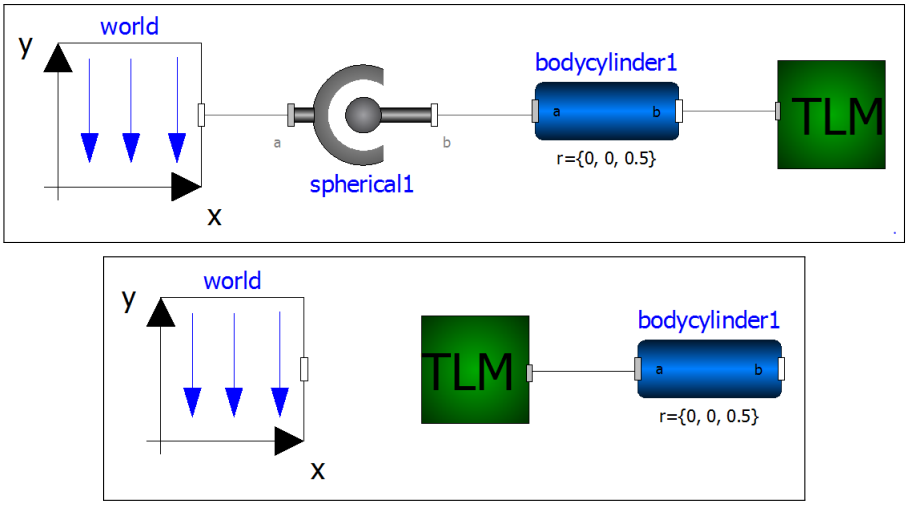
The second tutorial will demonstrate how to connect and simulate two existing 3D models. The external models represent the upper and the lower half of a swinging pendulum arm.



1. **Create a new composite model**

   Click on "File → New MetaModel" to create a new composite model. Right-click on the workspace and rename the model to "PendulumModel". **Save the composite model to a new empty folder.**

2. **Load the shaft models as external models**
   Choose "File → Load External Model" and load "`shaft1.mo`" and "`shaft2.mo`". Since these are pure Modelica models, you can open them and view them directly in OMEdit. As you can see, there is one missing icon in each model. Load "`OM_TLM.mo`" from the "`.../TLMPlugin/bin/OpenModelica`" folder to resolve this.



3. **Add the shafts to the composite model**
   Drag each external model to the composite model. 3D animation will require a geometry file. Double-click on each model and browse for the geometry files. Use "`shaft1.stl`" for the first shaft and "`shaft2.stl`" for the second. **Save the composite model.**

4. **Fetch interface data**
   Click on the "Fetch Interface Data" icon to fetch information about interfaces and parameters. The two arms should now appear in the 3D viewer.

    ***Fetch Interface Data***

5. **Connect interfaces**
   Connect the two interfaces to each other by dragging a connector between them.

6. **Align interfaces**
   As can be seen in the 3D viewer, the two shafts appear inside each other. This is because the connected interface points are not aligned. Thus, they have different positions and/or orientations relative to the inertia system. Click on the "Align Interfaces" button and align "shaft2.tlm" to "shaft1.tlm". Verify the alignment in the 3D viewer.

    ***Align Interfaces***

7. **Simulate**
   Click on the "TLM Co-Simulation Setup" icon and start the simulation.

    ***TLM Co-Simulation Setup***

8. **Analyze results**
   After a successful simulation OMEdit will automatically switch to the plotting perspective. Click on the play button to start a replay animation of the simulation. You can also plot variables from the variable tree as usual.