Logical-Physical Modeling Using Abaqus and Dymola

Abaqus/CAE Plug-in

Abaqus 6.9

Legal notices

CAUTION: This documentation is intended for qualified users who will exercise sound engineering judgment and expertise in the use of the Abaqus Software. The Abaqus Software is inherently complex, and the examples and procedures in this documentation are not intended to be exhaustive or to apply to any particular situation. Users are cautioned to satisfy themselves as to the accuracy and results of their analyses.

Dassault Systèmes and its subsidiaries, including Dassault Systèmes Simulia Corp., shall not be responsible for the accuracy or usefulness of any analysis performed using the Abaqus Software or the procedures, examples, or explanations in this documentation. Dassault Systèmes and its subsidiaries shall not be responsible for the consequences of any errors or omissions that may appear in this documentation.

DASSAULT SYSTÈMES AND ITS SUBSIDIARIES DISCLAIM ALL EXPRESS OR IMPLIED REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OF THE CONTENTS OF THIS DOCUMENTATION.

IN NO EVENT SHALL DASSAULT SYSTÈMES, ITS SUBSIDIARIES, OR THEIR THIRD-PARTY PROVIDERS BE LIABLE FOR ANY INDIRECT, INCIDENTAL, PUNITIVE, SPECIAL, OR CONSEQUENTIAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, OR LOSS OF BUSINESS INFORMATION) EVEN IF DASSAULT SYSTÈMES OR ITS SUBSIDIARY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The Abaqus Software is available only under license from Dassault Systèmes or its subsidiary and may be used or reproduced only in accordance with the terms of such license.

This documentation and the software described in this documentation are subject to change without prior notice.

This documentation may be reproduced and/or distributed without permission, as long as it is an exact reproduction, including these notices.

Export and re-export of the Abaqus Software and this documentation is subject to United States and other export control regulations. Each user is responsible for compliance with applicable export regulations.

The Abaqus Software is a product of Dassault Systèmes Simulia Corp., Providence, RI, USA.

U.S. GOVERNMENT USERS: The Abaqus Software and its documentation are “commercial items,” specifically “commercial computer software” and “commercial computer software documentation” and, consistent with FAR 12.212 and DFARS 227.7202, as applicable, are provided with restricted rights in accordance with license terms.

Abaqus, the 3DS logo, SIMULIA, Dymola, and Unified FEA are trademarks or registered trademarks of Dassault Systèmes or its subsidiaries in the United States and/or other countries.

Other company, product, and service names may be trademarks or service marks of their respective owners. For additional information concerning trademarks, copyrights, and licenses, see the Legal Notices in the Abaqus 6.9 Release Notes and the notices at: http://www.simulia.com/products/products_legal.html.

Version block

Version #	Description of Version	Date
1	Initial release	04 Jan 2010

1. *Introduction*

The Abaqus co-simulation technique can be used to analyze complex systems that include electronics, control systems, electro-mechanics, hydraulics, and pneumatics by coupling Abaqus with Dymola, a general-purpose logical modeling software distributed by Dynasim AB. Structural responses computed by Abaqus/Standard or Abaqus/Explicit are coupled at run-time with logical solutions provided by Dymola. See SOSS Answer 3617 for instructions on how to get Dymola and Abaqus ready for co-simulation and for general instructions on how to run the co-simulation process between Abaqus and Dymola.

This guide is a reference for using the Abaqus/CAE plug-in to run co-simulation problems involving Abaqus and Dymola, and provides detailed information on installation and usage. A tutorial with step-by-step instructions for running a co-simulation analysis between Abaqus and Dymola is also included with the guide.

2. *Installation and Setup*

To install the plug-in, download and save the attached archive to one of the following directories:

- *abaqus_dir*`\cae\abaqus_plugins` where *abaqus_dir* is the Abaqus parent directory
- *home_dir*`\abaqus_plugins` where *home_dir* is your home directory
- *current_dir*`\abaqus_plugins` where *current_dir* is the current working directory

Note that if the `abaqus_plugins` directory does not exist in the desired path, it must be created. The `plugin_dir` directory can also be used, where `plugin_dir` is a directory specified in the `abaqus_v6.env` file by the environment variable **plugin_central_dir.** You can store plug-ins in a central location that can be accessed by all users at your site if the directory to which **plugin_central_dir** refers is mounted on a file system that all users can access. For example,

```
plugin_central_dir = r'\\fileServer\sharedDirectory'
```

On Windows platforms, right click on the archive file and select **WinZip →Extract to here**. A folder named `abq_dymola` and a file named `Dymola_plugin.py` will be created. It is recommended that you have only one release of this module at a time. Multiple versions may cause conflicts, and the first detected version in the Python path will be used.

For more details on how you can use plug-ins and the Plug-in toolset to extend the capabilities of Abaqus/CAE please refer to Part VIII: "Using plug-ins," of the 6.9 Abaqus/CAE User's Manual.

The complete path to the executable that launches Dymola, **Dymola.exe**, should be added to the environment variable **PATH**. On Windows machines, a permanent setting can be employed by modifying the system or user environment variable.

1. From the **Control Panel → System** icon, click on the **Advanced** tab and then click **Environment Variables**.
2. Click **New** and enter PATH for the variable name and specify the path to the Dymola executable (something like "D:\Dymola\bin") for the variable value. If there is an existing PATH variable, edit the variable and add the path to the executable to the variable value.
3. Click **OK** and **OK**.

Note that this is the executable that the plug-in will call for launching Dymola. It is recommended that the Abaqus/CAE and Dymola files reside in the same directory and Abaqus/CAE is launched from this directory.

2.1 References

This section contains a list of reference material for co-simulation using Abaqus and Dymola. Documentation references refer to Abaqus 6.9-1.

Abaqus

- SOSS Answer 3617
- "Co-simulation," Section 14.1 of the Abaqus Analysis User's Manual
- Abaqus Analysis User's Manual
- Abaqus Keywords Reference Manual
- Abaqus Installation and Licensing Guide

Dymola

- Dymola User's Manual

2.2 Whom to contact for assistance

You can contact your local Abaqus support office for technical assistance with Abaqus-Dymola problems. Depending on the nature of your support needs, you may be referred to Dynasim AB for further assistance. For the latest information on co-simulation using Abaqus and Dymola, see the SIMULIA Answers database in the SIMULIA Online Support System. The SIMULIA Online Support System is accessible through the **My Support** page at www.simulia.com. You can view answers related to logical modeling by selecting **Abaqus/Standard Analysis Techniques** or **Abaqus/Explicit Analysis Techniques** from the list of **Abaqus Topics**, selecting the subtopic **Cosimulation**, and clicking **Search**.

In case direct support is necessary, include the relevant model database (`.cae`) file, the accompanying replay (`.rpy`) file, the Dymola logical (`.mo`) file and the details of the problem with your support request.

2.3 Typographical conventions

This guide adheres to a set of typographical conventions so that you can recognize actions and items. The following list illustrates each of the conventions:

- Text you enter from the keyboard, or file names: **`abaqus`**, **`jobname.mo`**
- Hyperlinks: www.simulia.com
- Text indicating that you have a choice: *job-name*

3. *Abaqus-Dymola Co-simulation: Using the Plug-in*

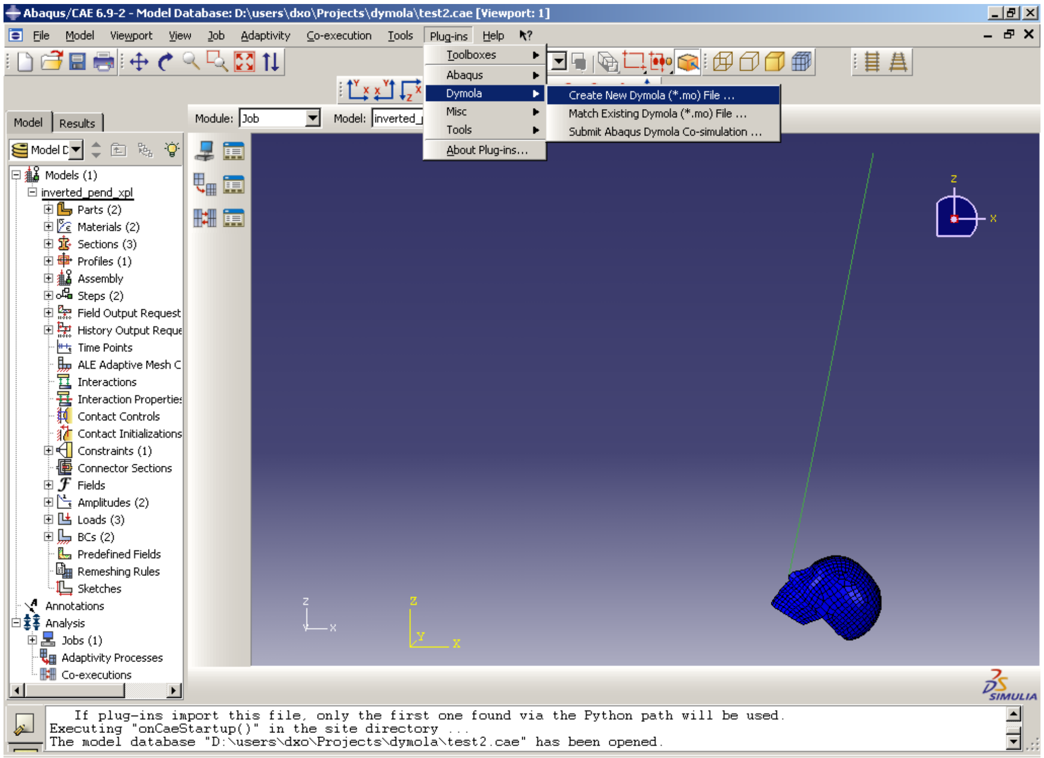
3.1 Overview

Before using the plug-in, it is assumed that both Abaqus and Dymola are properly installed and ready for co-simulation and that you are familiar with the Logical-Physical Modeling using Abaqus and Dymola User's Guides available via SOSS Answer 3617.

The plug-in provides a GUI-based link between the physical models created in Abaqus/CAE and the logical models created in Dymola. A structural-logical analysis can be completed for a given Abaqus/CAE model database using the plug-in. Before starting the plug-in, you need to do the following in Abaqus/CAE and Dymola:

1) Setup the physical model in Abaqus/CAE. This would involve creating, meshing and assembling the model (or importing an orphan mesh), assigning material and section properties to the geometry, and assigning interaction properties, loads and boundary conditions.
2) If possible, create the interface sensors and actuators that will be exchanged with Dymola. The plug-in also provides the interface to create sensors and actuators interactively if this has not been done already.
3) Ensure that Dymola can be launched upon typing **Dymola.exe** on the command prompt and models can be translated as a `.DLL` file from Dymola without any errors.

The plug-in is available under the Job module of Abaqus/CAE as illustrated in Figure 3-1. While the **Job** module is active, select **Plug-ins** → **Dymola** for a number of options. Each of these options is discussed in detail below.



```
   If plug-ins import this file, only the first one found via the Python path will be used.
Executing "onCaeStartup()" in the site directory ...
The model database "D:\users\dxo\Projects\dymola\test2.cae" has been opened.
```

Figure 3-1: Abaqus/CAE plug-in for Abaqus-Dymola co-simulation

3.2 Co-simulation workflows

An Abaqus Dymola co-simulation analysis requires a physical model created in Abaqus/CAE and a Dymola logical model. The plug-in provides the interface to create or identify sensors or actuators in Abaqus/CAE that can then be written to a new Dymola logical (`*.mo`) file or can be matched with an existing logical file. Based on these two scenarios, there are two different workflows available. A description of these workflows is provided below.

3.2.1 Workflow 1: Build a new Dymola logical model using signals (sensors and actuators) from Abaqus/CAE

This workflow applies when you have built your physical model in Abaqus/CAE and subsequently want to create the logical model using the sensors and actuators created in Abaqus/CAE. The workflow involves selecting the sensors and actuators from the Abaqus/CAE model to be written to the Dymola logical (`*.mo`) file. These signals are then written to the logical file and opened up in Dymola. You are then expected to build the logical model using these sensors and actuators. Upon building the logical model, the model should be translated without errors and the `dynosim.dll` file must be generated successfully. The plug-in can then be used to submit an Abaqus Dymola co-simulation analysis. The workflow is illustrated in Figure 3-2:

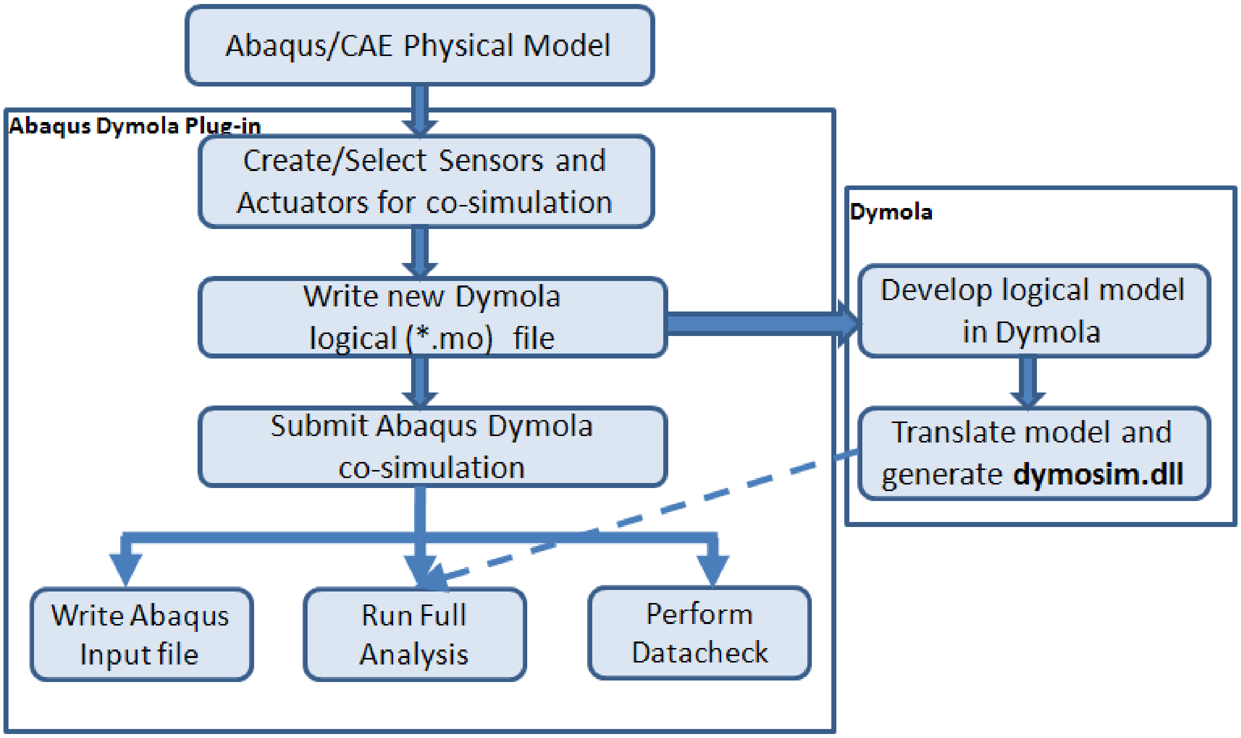
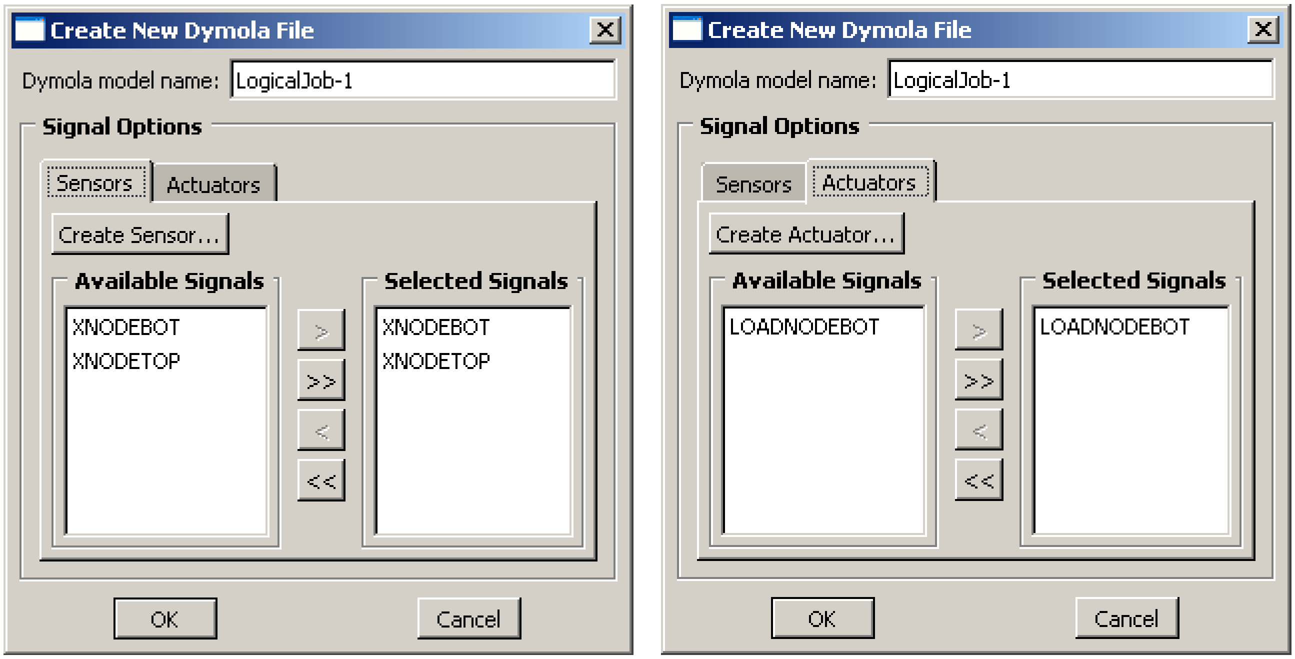

Abaqus/CAE Physical Model
Abaqus Dymola Plug-in
Create/Select Sensors and Actuators for co-simulation
Write new Dymola logical (*.mo) file
Submit Abaqus Dymola co-simulation
Write Abaqus Input file
Run Full Analysis
Perform Datacheck
Dymola
Develop logical model in Dymola
Translate model and generate **dymosim.dll**

Figure 3-2: Workflow1- Building a logical model from Abaqus/CAE signals

This workflow can be enabled by selecting **Plugins → Dymola → Create New Dymola (*.mo) File ...** menu as shown in Figure 3-1. This dialog box is used to create a new Dymola logical (`*.mo`) file by providing the interface to select Sensors and Actuators defined in the Abaqus/CAE model. Upon committing the dialog, the selected Sensors and Actuators are written to a logical file and displayed in Dymola for you to build the logical model. The dialog box (Figure 3-3) has two tabs - one for Sensors and the other for Actuators. The remaining dialog features are listed below:



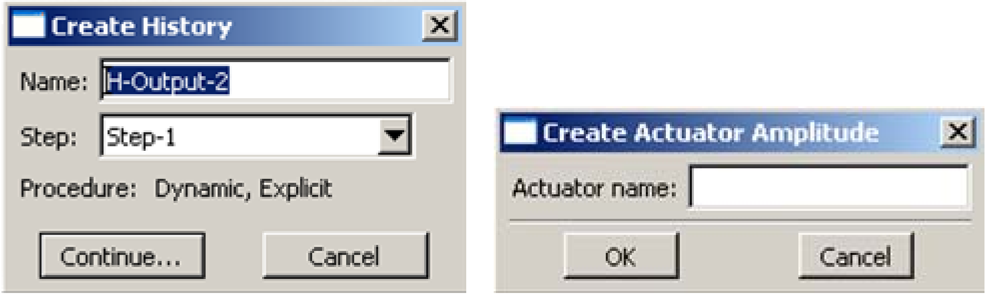
Dymola model name: LogicalJob-1

Sensors – Available Signals: XNODEBOT, XNODETOP; Selected Signals: XNODEBOT, XNODETOP

Actuators – Available Signals: LOADNODEBOT; Selected Signals: LOADNODEBOT

Figure 3-3: Create a New Dymola File dialog box

- **Dymola model name:** Specify the name of the Dymola logical (`*.mo`) file that will be written by the plug-in.
- **Create Sensor:** This button can be used to create a new Sensor History output in the displayed Abaqus/CAE model. Clicking this button launches the Abaqus/CAE dialog to create History Outputs (Figure 3-4). Note that Sensors can be created only when domain **Set** is selected and **Include sensor when available** is toggled on.
- **Create Actuator:** This button can be used to create a new Actuator amplitude in the displayed Abaqus/CAE model. Clicking this button launches a dialog prompting for the Actuator name (Figure 3-4) for the amplitude to be created.
- **Available Signals:** A list of all Sensors or Actuators defined in the displayed Abaqus/CAE model. From this list, select the Sensor(s) or Actuators(s) that will need to written to the Dymola logical (`*.mo`) file. If no Sensors or Actuators have been defined in the model, use the **Create Sensor** and **Create Actuator** buttons to create them.
- **Selected Signals:** A list of Sensors and Actuators that will be written to the Dymola logical (`*.mo`) file.



Name: H-Output-2
Step: Step-1
Procedure: Dynamic, Explicit
Actuator name:

Figure 3-4: Create History and Create Actuator Amplitude dialog boxes

Upon clicking **OK**, the following sequence of actions takes place:

1) The dialog box closes after error checks.
2) The Sensors and Actuators displayed in the **Selected Signals** list are written to the Dymola logical (`*.mo`) file specified in the **Dymola model name** field. The file will be written to the directory from which Abaqus/CAE is launched.
3) Dymola is launched (Figure 3-5) and the logical file written out by the plug-in will be displayed. The Dymola model will have the same name as that of the Abaqus/CAE model.

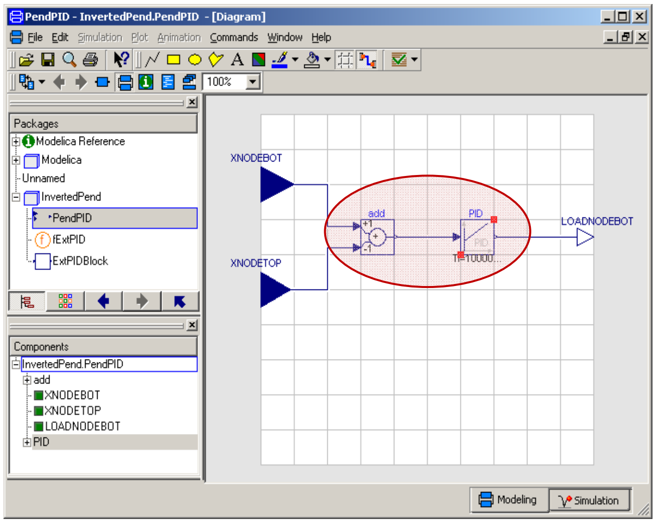
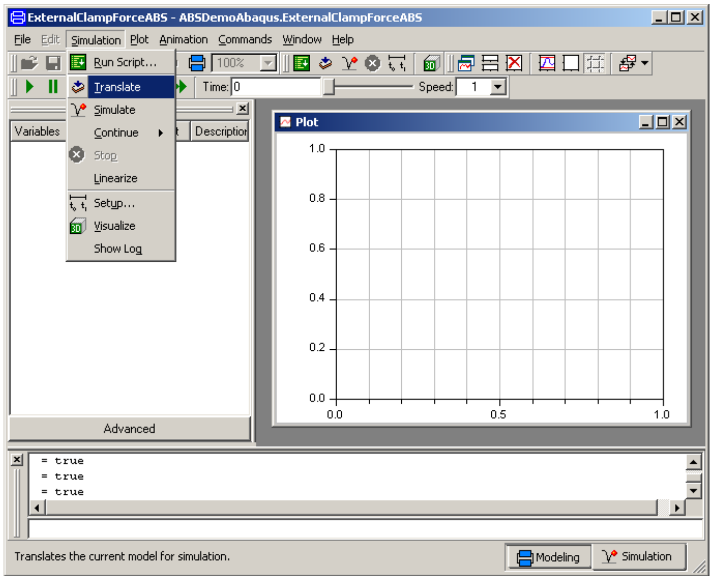


Figure 3-5: Dymola Modeling window

You are expected to build the logical model using the sensors and actuators as shown in the highlighted area in Figure 3-5. Once the model is complete, in Dymola switch to the Simulation tab (bottom right) and translate the model by selecting **Simulation → Translate** (Figure 3-6) to generate the `dymosim.dll` file.



```
= true
= true
= true
```

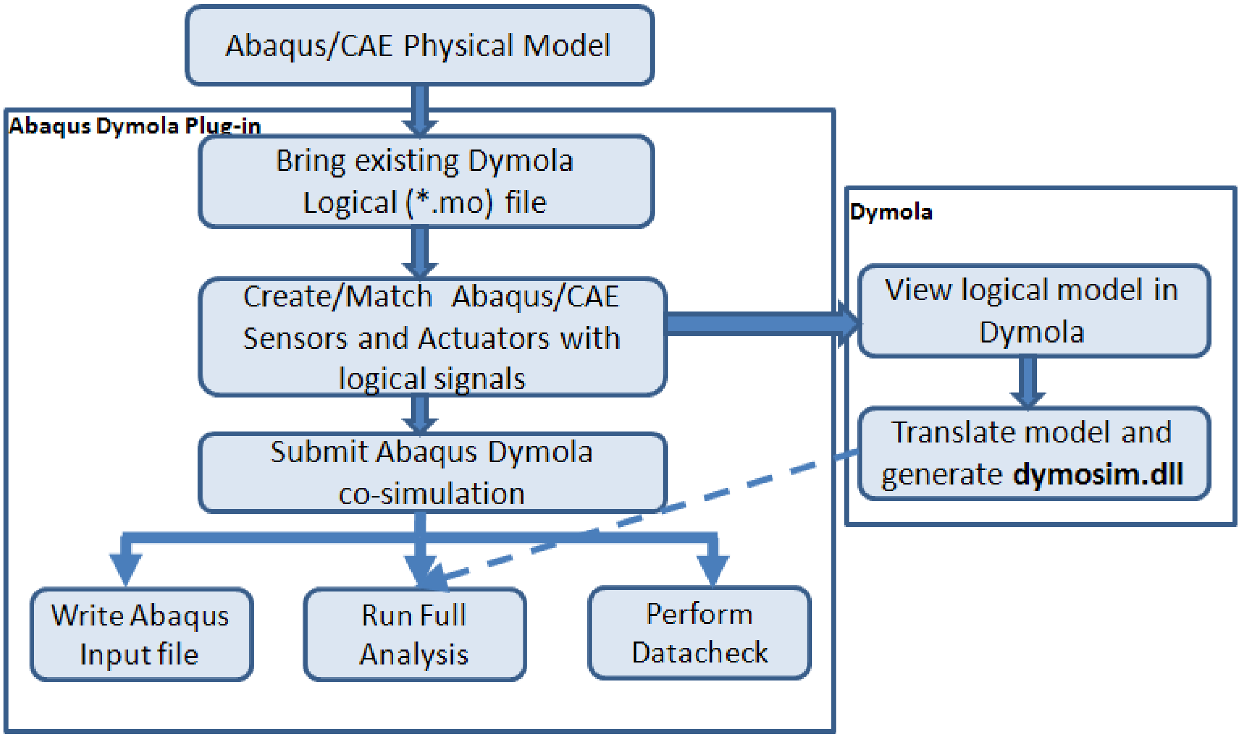
Translates the current model for simulation.

Figure 3-6: Dymola Translate

3.2.2 Workflow 2: Matching Abaqus/CAE Sensors and Actuators with an existing logical model

This workflow applies when you have already built your logical models in Dymola and want to create the corresponding Sensors and Actuators in the Abaqus/CAE physical model so that the two models are consistent. The workflow involves importing and displaying the sensors and actuators from the Dymola logical (`*.mo`) file in the plug-in dialog. You are then expected to create the missing sensors in the Abaqus/CAE model. The plug-in can automatically create the missing actuators in the Abaqus/CAE model. Note that the signal names must be the same in the logical and physical models. For the actuators to be meaningful they must be associated with an Abaqus feature that can reference an amplitude definition such as a load or boundary condition. This can be accomplished in Abaqus/CAE as usual.

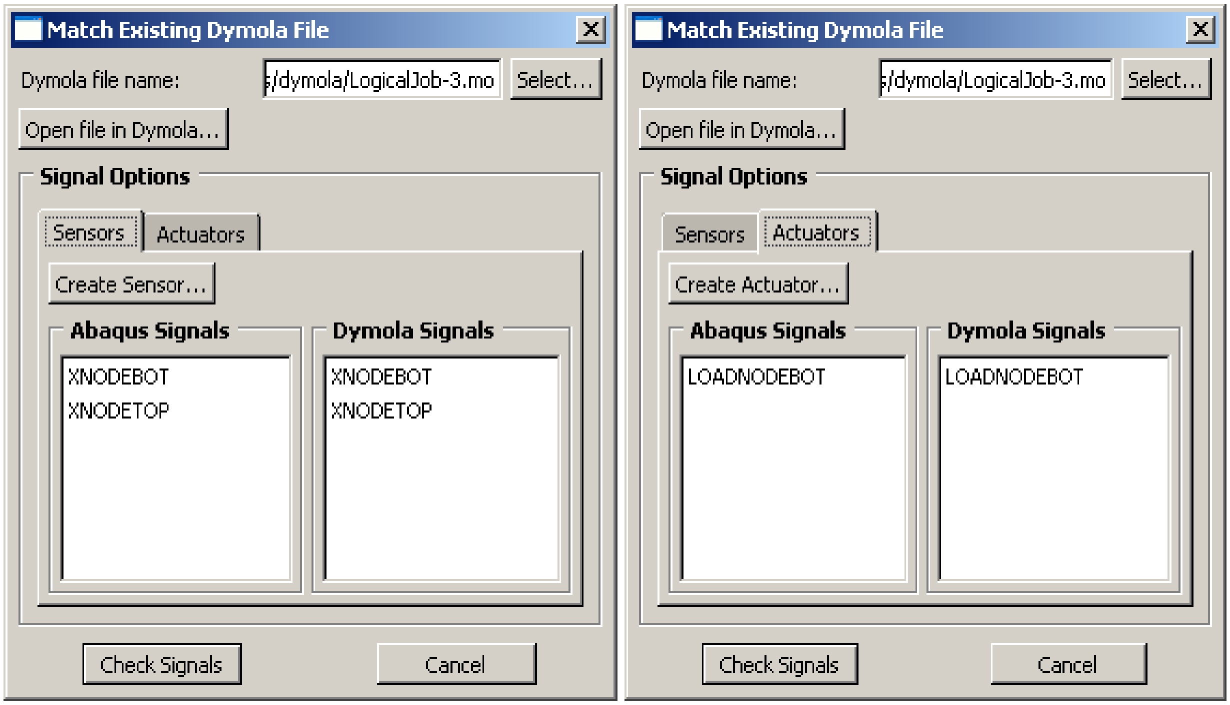
Once the sensor and actuator names have been matched between the Abaqus/CAE and Dymola models, the plug-in can then be used to submit an Abaqus-Dymola co-simulation analysis. The image provided below illustrates this workflow:



Abaqus/CAE Physical Model
Abaqus Dymola Plug-in
Bring existing Dymola Logical (*.mo) file
Create/Match Abaqus/CAE Sensors and Actuators with logical signals
Submit Abaqus Dymola co-simulation
Write Abaqus Input file
Run Full Analysis
Perform Datacheck
Dymola
View logical model in Dymola
Translate model and generate **dymosim.dll**

Figure 3-7: Workflow 2 - Matching Abaqus/CAE signals with an existing Logical model

This workflow can be enabled by selecting **Plugins → Dymola → Match Existing Dymola (*.mo) File.** This dialog box is meant to bring in an existing Dymola logical (`*.mo`) file and match the Sensors and Actuators in the logical model with the ones in the Abaqus/CAE model. The dialog box (Figure 3-8) has two tabs: one for Sensors and the other for Actuators.



Dymola file name: s/dymola/LogicalJob-3.mo Select...

Abaqus Signals: XNODEBOT, XNODETOP

Dymola Signals: XNODEBOT, XNODETOP

Dymola file name: s/dymola/LogicalJob-3.mo Select...

Abaqus Signals: LOADNODEBOT

Dymola Signals: LOADNODEBOT

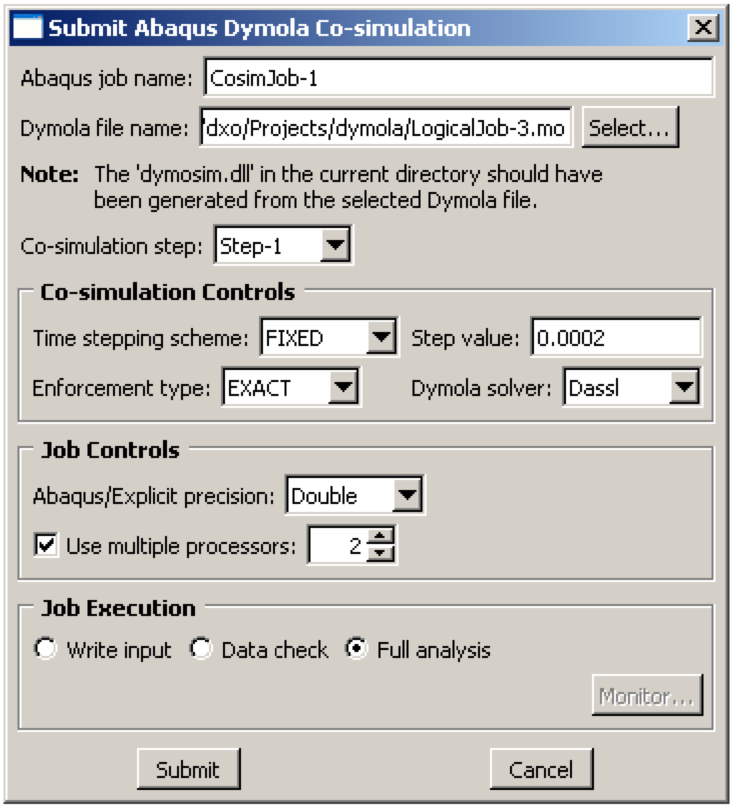
Figure 3-8: Match Existing Dymola File dialog box

Note that the plug-in requires that the Sensor and Actuator names in the physical and logical models must match exactly. Once the logical (`*.mo`) file is imported in the plug-in, an automatic check is done to see if the Abaqus/CAE model has the Sensor and Actuator names matching with the names in the logical model. An information dialog is then posted informing you if signal names are consistent between Abaqus/CAE and Dymola models and if any Sensors or Actuators have to be created in the Abaqus/CAE model. You then have to create these signals in the Abaqus/CAE model with the same name as that of the logical model. Anytime during this process, you can check the consistency of the signals between the Abaqus/CAE and Dymola models. The dialog box, as shown in Figure 3-8, has the following components:

- **Dymola file name:** Specify the name of the Dymola logical (`*.mo`) file that will be read into the plug-in in.
- **Open file in Dymola:** This button can be used to open the selected Dymola logical (`*.mo`) file in Dymola.
- **Create Sensor:** This button can be used to create a new Sensor History output in the displayed Abaqus/CAE model. Clicking this button launches the Abaqus/CAE dialog to create History Outputs (Figure 3-4). Note that Sensors can be created only when domain **Set** is selected and **Include sensor when available** is toggled on.
- **Create Actuator:** This button can be used to create new Actuator amplitude in the displayed Abaqus/CAE model. Clicking this button launches a dialog prompting for the Actuator name (Figure 3-4) for the amplitude to be created.
- **Abaqus Signals:** A list of all Sensor(s) or Actuator(s) defined in the displayed Abaqus/CAE model.
- **Dymola Signals:** A list of all Sensor(s) or Actuator(s) read in from the selected Dymola logical (`*.mo`) file.
- **Check Signals:** A button to check if the Sensor(s) and Actuator(s) read in from the Dymola logical (`*.mo`) file has the equivalent Sensor(s) and Actuator(s) in the displayed Abaqus/CAE model. If they do not match, a message dialog is posted with the list of Sensors and Actuators that have to be created in Abaqus/CAE.

3.3 Submitting an Abaqus-Dymola co-simulation

The dialog box can be accessed from the **Plugins** → **Dymola** → **Submit Abaqus Dymola Co-simulation** menu and allows you to run Abaqus-Dymola co-simulation analyses. It has the following components as illustrated in Figure 3-9:



Submit Abaqus Dymola Co-simulation

Abaqus job name: CosimJob-1

Dymola file name: dxo/Projects/dymola/LogicalJob-3.mo

Note: The 'dymosim.dll' in the current directory should have been generated from the selected Dymola file.

Co-simulation step: Step-1

Co-simulation Controls

Time stepping scheme: FIXED

Step value: 0.0002

Enforcement type: EXACT

Dymola solver: Dassl

Job Controls

Abaqus/Explicit precision: Double

☑ Use multiple processors: 2

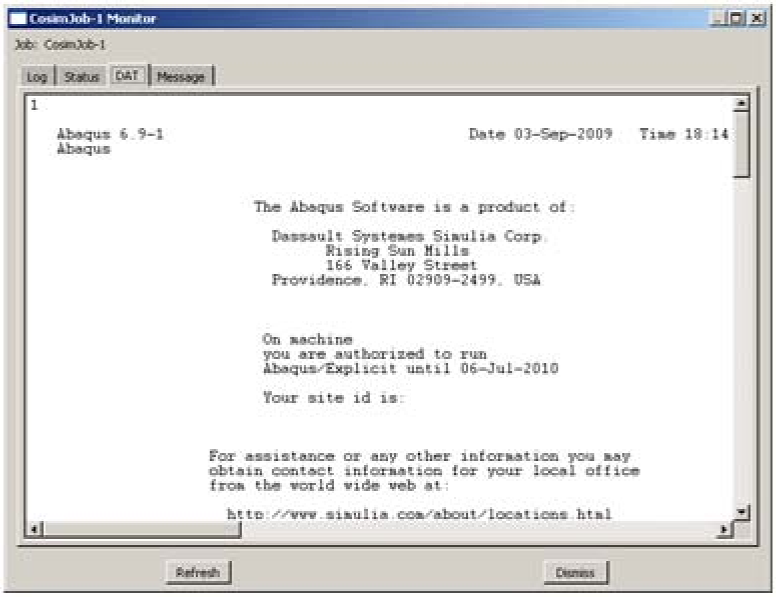
Job Execution

○ Write input ○ Data check ● Full analysis

Figure 3-9: Submit Abaqus Dymola Co-simulation dialog box

- **Abaqus job name:** Specify the Abaqus job name to be used in the co-simulation analysis.
- **Dymola file name:** Select the Dymola logical (`*.mo`) file that will be used with Abaqus for co-simulation analysis. Note that the `dymosim.dll` file in the Abaqus/CAE launch directory should have been generated for the selected Dymola file.
- **Co-simulation step**: Select the co-simulation step from the Abaqus/CAE model.
- **Time stepping scheme**: This can have value of either **VARIABLE** or **FIXED**. Choose **VARIABLE** for Abaqus to set the coupling step size equal to the next suggested increment size computed by the automatic time incrementation scheme. Choose **FIXED** to specify a constant coupling step size.
- **Step value:** This field is applicable only if **Time stepping scheme** is set to **FIXED**. Specify the constant coupling step size to be used for the co-simulation analysis.
- **Enforcement Type:** This can have value of either **EXACT** or **LOOSE**. Setting **EXACT** directs Abaqus to reach the target times in an exact manner (TIME MARKS=YES); that is, Abaqus will adjust the time increment as necessary to exactly meet the target time. Setting **LOOSE** directs Abaqus to meet the target times in an approximate or loose manner (TIME MARKS=NO); performing a co-simulation exchange only after a target time is passed, with time incrementation in Abaqus advancing without regard for the target time.

- **Dymola solver:** Select the Dymola solver type.
- **Abaqus/Explicit precision:** Set the precision for the Abaqus/Explicit analysis to **Single** or **Double**.
- **Use multiple processors:** Toggle this on to use multiple processors and select the number of processors.
- **Write input, Data check and Full analysis:** Select one of these options to write the Abaqus input file with co-simulation keywords, to perform a data check on the Abaqus-Dymola co-simulation, or to run the Abaqus-Dymola co-simulation analysis.
- **Monitor:** This button is valid for data check and full analyses and is activated when the job has been submitted. Clicking this button opens a dialog box (Figure 3-10) that allows you to monitor the job. The **Monitor** dialog posts the contents of the log (`.log`), status (`.sta`), data (`.dat`) and message (`.msg`) files of the Abaqus analysis during the co-simulation. Click the **Refresh** button to upload the current file contents in the dialog.
- **Submit:** This button commits the dialog.



```
1

   Abaqus 6.9-1                                    Date 03-Sep-2009   Time 18:14
   Abaqus


                           The Abaqus Software is a product of:

                             Dassault Systemes Simulia Corp.
                                   Rising Sun Mills
                                   166 Valley Street
                             Providence, RI 02909-2499, USA



                            On machine
                            you are authorized to run
                            Abaqus/Explicit until 06-Jul-2010

                            Your site id is:


                      For assistance or any other information you may
                      obtain contact information for your local office
                      from the world wide web at:

                        http://www.simulia.com/about/locations.html
```

Figure 3-10: Co-simulation Job Monitor dialog box

4. *Limitations and Additional Notes*

The following limitations/comments apply:

- The plug-in is currently only available for the 32-bit Windows platform, and both analyses will run on the same machine. You can write the Abaqus input and Dymola mapping files for the co-simulation using the plug-in and run the analysis on remote platforms if necessary.
- The Dymola logical (`*.mo`) files and the Abaqus input files are written to the directory from which Abaqus/CAE is launched. This is consistent with the general behavior of Abaqus/CAE.
- The names of the Sensors and Actuators should match exactly between the Abaqus/CAE model database and the Dymola logical model, though the names are not case sensitive.
- The plug-in only matches the names of the signals between the two applications. It is your responsibility to ensure that the Sensors have been requested at correct locations in the Abaqus/CAE model and that the input Sensors and output Actuators have been used correctly in the Dymola model.
- You should remember to associate the Actuator amplitude with an appropriate option that can reference an amplitude definition in the Abaqus/CAE model for actuation to be meaningful.
- When importing input files with Sensors and Actuators into Abaqus/CAE, you should check if they have been imported correctly into the model and look into possible warning and error messages in the message log.
- When an Abaqus Dymola co-simulation analysis is run using the plug-in, the launch directory of Abaqus/CAE will be searched for a `dymosim.dll` file. This should correspond to the Dymola logical (`*.mo`) file used for setting up the job. If this file does not correspond to the selected Dymola file, results may be incorrect.

5. *Tutorial: Balancing a Vertical Pole*

This chapter contains a tutorial that demonstrates an Abaqus/Explicit-Dymola transient simulation. The problem being modeled is rather simple, but illustrates all aspects of the co-simulation process: the vertical equilibrium of an initially imbalanced pole is being sought (similar to what you would see in a clown act at a circus show).

This tutorial assumes working familiarity with Abaqus/CAE and Dymola. In it, you will learn how to use the plug-in and:

- build a new Dymola logical model using signals from the Abaqus/CAE model
- match Sensors and Actuators between Abaqus/CAE and Dymola models
- set-up and execute an Abaqus-Dymola co-simulation

5.1 Overview

This tutorial requires Abaqus/CAE 6.9-1 or later and Dymola Version 7.2 or later.

5.1.1 Problem description

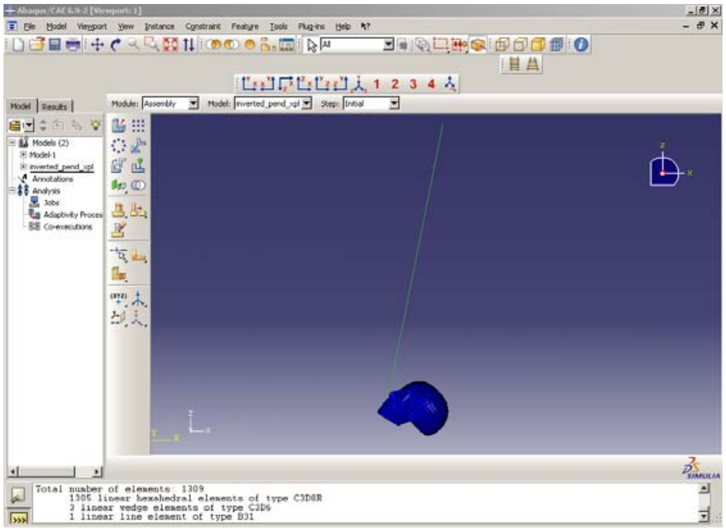
A stiff beam modeling the pole is initially placed in an unbalanced position under a gravity field. If no additional forces are applied to the system, the pole will fall over. The goal is to find the corrective force that needs to be applied at the bottom of the pole in the horizontal direction such that the pole becomes vertical. To make the problem more challenging, two impact loads are applied at the top of the pole, at two distinct points in time, to create further imbalance.

The difference in the top and bottom coordinates in the horizontal direction can be used as a measure of imbalance of the pole: if the pole is in vertical balance the difference is very small (or zero) and only a small corrective force is needed at the bottom of the pole; on the other hand, if the difference is large the imbalance is significant and a larger force needs to be applied.

At a given time, the coordinates (sensors) are passed to Dymola, which computes the magnitude and sign of the imbalance. Dymola then computes the force that must be applied in Abaqus (actuator) to balance the pole. The process is repeated many times during the analysis.

5.1.2 Model description

The pole (and its dynamics) is modeled in Abaqus with a single beam element. For visualization purposes a display body is added to represent the balancing point at the bottom (nose tip of a person) as shown in Figure 5-1:



```
Total number of elements: 1309
        1305 linear hexahedral elements of type C3D8R
        3 linear wedge elements of type C3D6
        1 linear line element of type B31
```

Figure 5-1: Abaqus/CAE structural model

The Dymola control algorithm, shown in the center of Figure 5-2, is quite straightforward. A simple PID controller will produce the desired results.

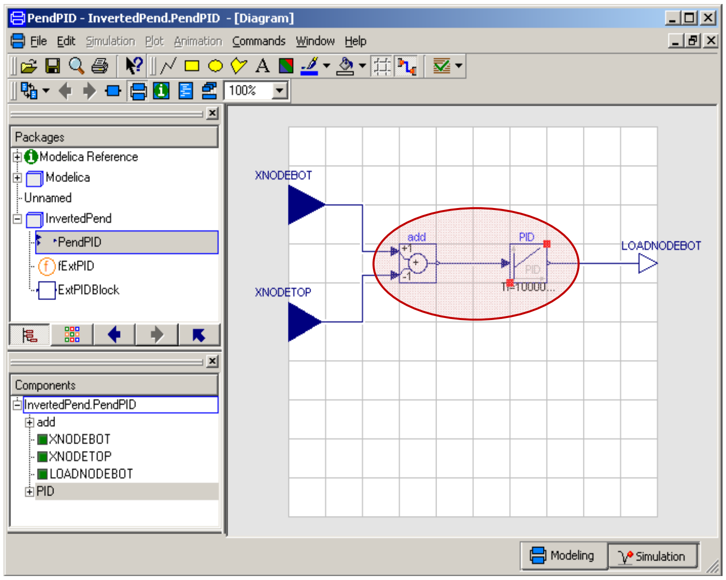


Figure 5-2: Dymola control model

5.1.3 Input files

The model input files for this tutorial are available in a compressed file attached to this SIMULIA Answer. Unzip the files in your work directory. The files are as follows:

- **`inverted_pend_xpl.inp`**: Abaqus input file containing the model
- **`inverted_pend_head.inp`**: Abaqus input file containing a display body for the head model

- **`inverted_pend.mo`**: Dymola input file containing the control model
- **`inverted_pend.py`**: Python script to create the Abaqus/CAE model from the input files

5.2 Using the Abaqus/CAE plug-in for Abaqus Dymola Co-simulation

This section describes the step-by-step procedure for using the Abaqus/CAE plug-in for creating and setting-up an Abaqus Dymola co-simulation with the tutorial files.

5.2.1 Creating the Abaqus/CAE model

First, create the Abaqus/CAE model that will be used as the structural model for the co-simulation.

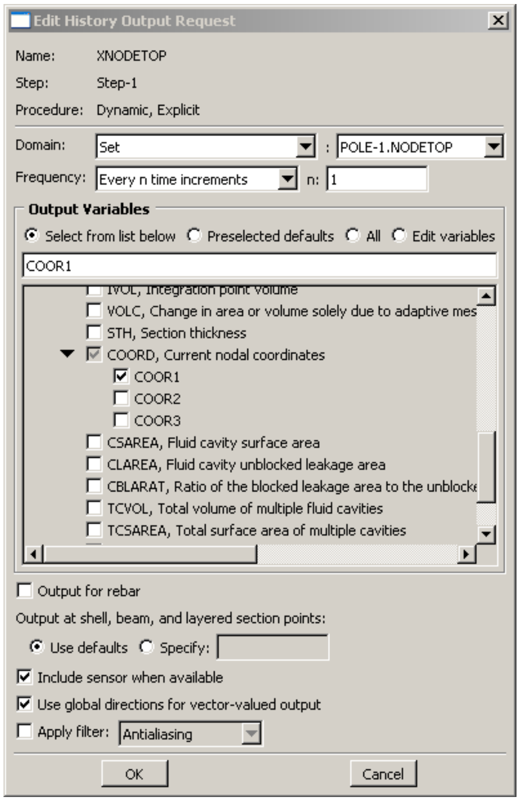
- Unzip the tutorial files to a work directory. Start **Abaqus/CAE** from this work directory.
- Go to **File → Run Script …** and select the Python script **`inverted_pend.py`**. The Python script imports the **`inverted_pend_xpl.inp`** input file into Abaqus/CAE, creates node sets and loads for co-simulation and saves the model database to **`inverted_pend.cae`**. The material properties, step definitions and boundary conditions have already been defined in the model.
- The CAE model **inverted_pend_xpl** has two parts; **HEAD-1** and **POLE-1**. **POLE-1** is the beam that will be used for co-simulation and **HEAD-1** is a display body.

5.2.2 Creating a New Dymola Logical (`*.mo`) file

Now, create sensors and actuators in the Abaqus/CAE model and write them to a Dymola logical file from the plug-in.

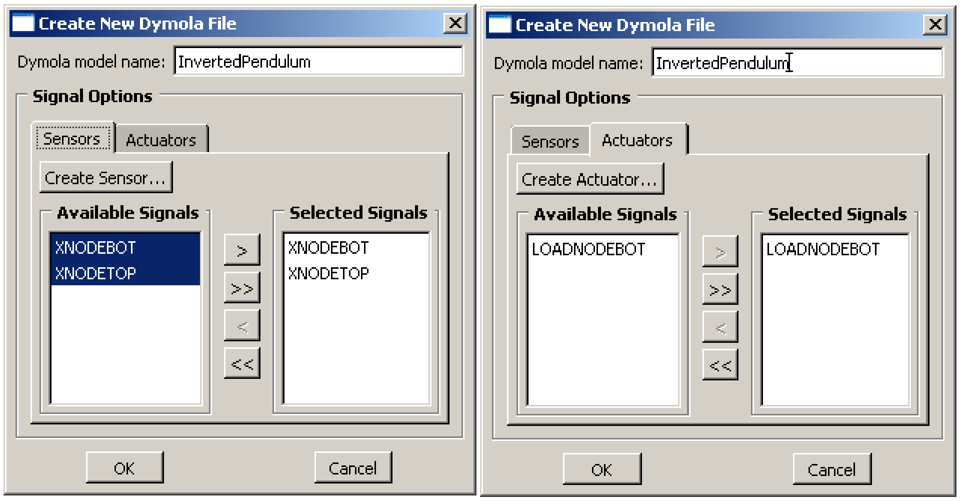
- Go to the **Job** module of Abaqus/CAE
- Click **Plug-ins → Dymola → Create New Dymola (*.mo) File…** The **Create New Dymola File** dialog box will be launched.
- The default name for Dymola models will be prefixed by *`LogicalJob`*. Rename this if necessary.
- Under **Signal Options**, go to the **Sensors** tab. Two sensors need to be created for the output variable *COOR1*; one for each node of the beam. Click the **Create Sensor** button. The **Create History** dialog box will be posted. Enter **XNODETOP** as the history output name and click **Continue….** In the ensuing **Edit History Output Request** dialog, choose **Set** as the Domain and select the set **POLE-1.NODETOP** from the drop-down box. Choose **Every n time increments** for Frequency and enter a value of **1** for n. For the output variables, select **COOR1**. This will be under **COORD, Current nodal coordinates**, which in turn, can be found under **Volume/Thickness/Coordinates**. Toggle on **Include sensor when available** option to use this history output request as a sensor. Click **OK** to create the sensor. The dialog box will be as shown in Figure 5-3. The created sensor will now be displayed under the **Available Signals** list of the plug-in dialog. Create another sensor with the name **XNODEBOT** for the node set **POLE-1.NODEBOT** and output variable **COOR1**.
- Add the two sensors that have been created to the **Selected Signals** list. This can be done by selecting both the sensors and clicking on the >> button. The sensors will now be visible under the **Selected Signals** list and will be written to the Dymola logical (`*.mo`) file.
- Next, go to the **Actuators tab** to create an actuator. Click **Create Actuator**. In the **Create Actuator Amplitude** dialog, enter **LOADNODEBOT** as the actuator name and click **OK.** The actuator will now be displayed under the **Available Signals** list. Select the actuator and move it to **Selected Signals** list using the > or >> buttons.

The dialog box should look like the one shown in Figure 5-4:



Name: XNODETOP
Step: Step-1
Procedure: Dynamic, Explicit
Domain: Set : POLE-1.NODETOP
Frequency: Every n time increments n: 1
● Select from list below ○ Preselected defaults ○ All ○ Edit variables
COOR1
☐ IVOL, Integration point volume
☐ VOLC, Change in area or volume solely due to adaptive mes
☐ STH, Section thickness
☑ COORD, Current nodal coordinates
  ☑ COOR1
  ☐ COOR2
  ☐ COOR3
☐ CSAREA, Fluid cavity surface area
☐ CLAREA, Fluid cavity unblocked leakage area
☐ CBLARAT, Ratio of the blocked leakage area to the unblock
☐ TCVOL, Total volume of multiple fluid cavities
☐ TCSAREA, Total surface area of multiple cavities
☐ Output for rebar
Output at shell, beam, and layered section points:
● Use defaults ○ Specify:
☑ Include sensor when available
☑ Use global directions for vector-valued output
☐ Apply filter: Antialiasing

Figure 5-3: Creating history output sensor



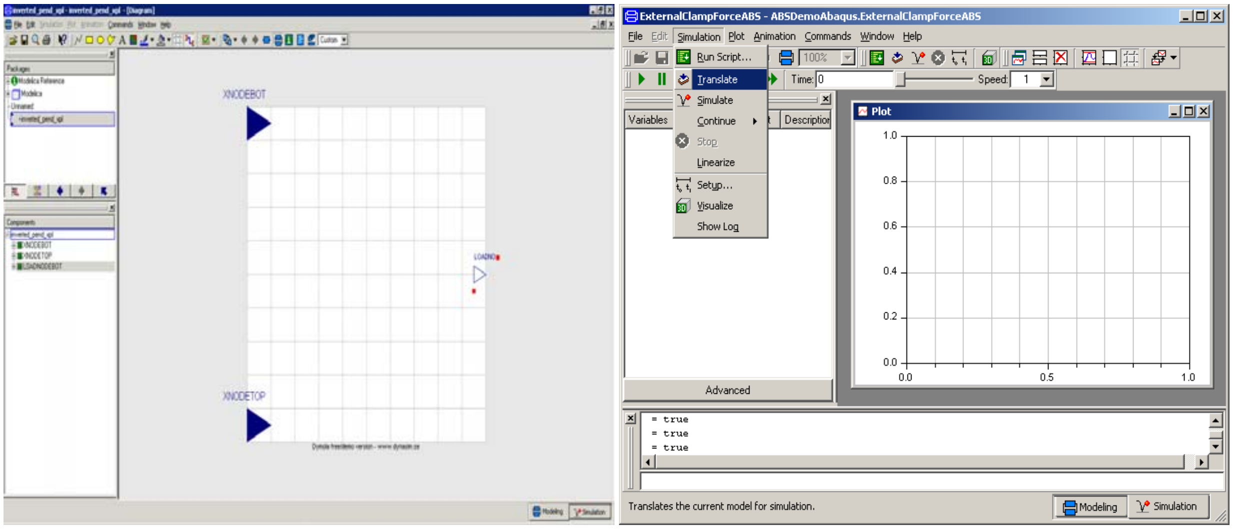
Dymola model name: InvertedPendulum
Available Signals: XNODEBOT, XNODETOP
Selected Signals: XNODEBOT, XNODETOP

Dymola model name: InvertedPendulum
Available Signals: LOADNODEBOT
Selected Signals: LOADNODEBOT

Figure 5-4: Create New Dymola File dialog

Once the signals to be written to the Dymola logical file have been selected for the **Sensors** and **Actuators** tabs, click **OK**. This will write a Dymola logical (`*.mo`) file with the specified name to the work directory and also launches Dymola to display the model as shown in Figure 5-5. Build the desired logical model in Dymola. Once the model is complete, switch to the **Simulation** tab and translate the model by selecting **Simulation** → **Translate**

(Figure 5-5) to generate the `dymosim.dll` file. Review Section 6-1, **Translating the Dymola Model** in the Appendix for more information on generating the DLL file. Save the Abaqus/CAE model.



```
= true
= true
= true
```

Translates the current model for simulation.

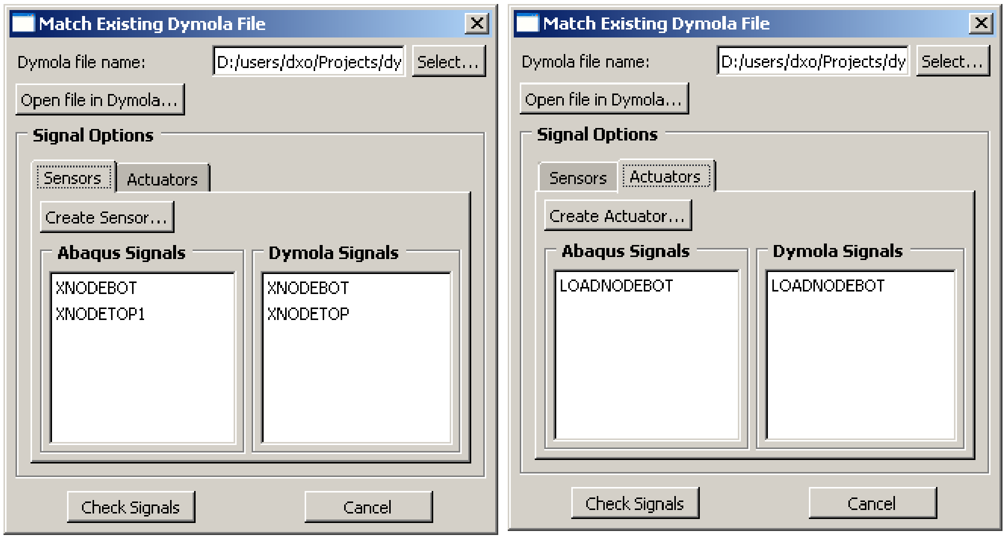
Figure 5-5: Dymola logical model and Dymola Translation Window

5.2.3 Matching an Existing Dymola Logical (`*.mo`) file

In this section of the tutorial, an existing Dymola Logical (`*.mo`) file will be read into Abaqus/CAE using the plug-in and the sensors and actuators will be matched between the Abaqus/CAE and Dymola models. The missing signals will be created in the Abaqus/CAE model. The existing Dymola logical model file **`inverted_pend.mo`**, will be used for this exercise.

- Switch to the **inverted_pend_xpl** model in Abaqus/CAE. Delete the actuator amplitude **LOADNODEBOT** created earlier. This can be done directly from the model tree of Abaqus/CAE, or switch to the **Load** module and go to **Tools** → **Amplitude** → **Delete** → **LOADNODEBOT**. Next, delete the sensor history output **XNODEBOT** created earlier. This can be done directly from the model tree, or switch to the **Step** module and go to **Output** → **History Output Requests** → **Delete** → **XNODEBOT**. This can also be done by clicking on the **History Output Request Manager** icon on the toolbox area. Finally, rename the other sensor history output **XNODETOP** to a different name (for example**, XNODETOP1)**. This can be done directly from the model tree, or from the **Step** module after selecting **Output** → **History Output Requests** → **Rename** → **XNODETOP**.
- Next, switch to the **Job** module. Select **Plug-ins** → **Dymola** → **Match Existing Dymola (*.mo) File…**.The **Match Existing Dymola File** dialog box will be posted.
- Click on the **Select** button across **Dymola file name** field to bring an existing Dymola logical model file into Abaqus/CAE. Clicking on the button will bring up a file selection dialog. Select the **`inverted_pend.mo`** file and click **OK**. A dialog box as shown in Figure 5-6 will be posted.
- The dialog box states that the sensors **XNODETOP** and **XNODEBOT** and the actuator **LOADNODEBOT** have to be created in the Abaqus/CAE model **inverted_pend_xpl**. Click **OK** for creating the amplitude. It is not possible to automatically create the sensors in the Abaqus/CAE model as there are unknowns like node set name, output variable and output frequency.
- Click the **Open file in Dymola…** button. This will launch Dymola and display the opened Dymola Logical model file. Switch to the Simulation tab of Dymola and translate the model by selecting **Simulation** → **Translate** to generate the `dymosim.dll` file. Review Section 6-1**, Translating the**

Dymola Model in the Appendix for more information on generating the DLL file. The sensor history outputs have to be created in the Abaqus/CAE model to match with the Dymola sensors. As described in Section 5.2.2, click the **Create Sensor…** button and create a sensor with the name **XNODEBOT** for the node set **POLE-1.NODEBOT** and output variable **COOR1**. The sensor will now be displayed on the **Abaqus Signals** list of the **Sensors** tab as shown in Figure 5-6. Next, click the **Check Signals** button at the bottom of the dialog box. A dialog message stating that Dymola sensor **XNODETOP**

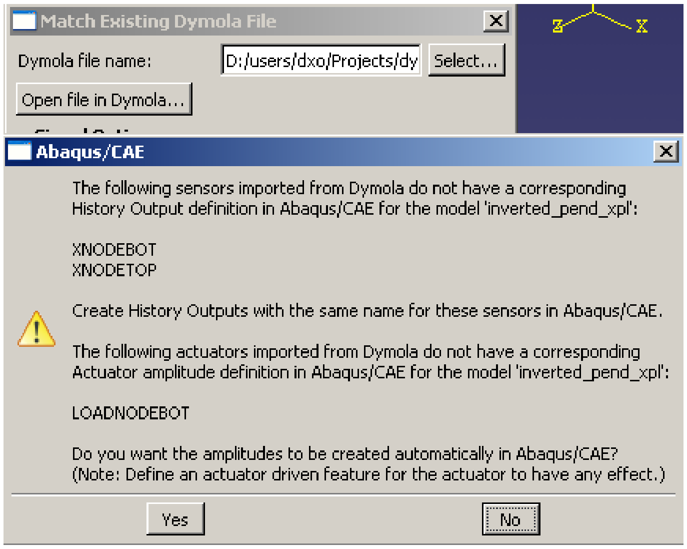


Dymola file name: D:/users/dxo/Projects/dy

Sensors tab: Abaqus Signals: XNODEBOT, XNODETOP1; Dymola Signals: XNODEBOT, XNODETOP

Actuators tab: Abaqus Signals: LOADNODEBOT; Dymola Signals: LOADNODEBOT

Figure 5-6: Match signals between Abaqus/CAE and Dymola



Dymola file name: D:/users/dxo/Projects/dy

The following sensors imported from Dymola do not have a corresponding History Output definition in Abaqus/CAE for the model 'inverted_pend_xpl':

XNODEBOT
XNODETOP

Create History Outputs with the same name for these sensors in Abaqus/CAE.

The following actuators imported from Dymola do not have a corresponding Actuator amplitude definition in Abaqus/CAE for the model 'inverted_pend_xpl':

LOADNODEBOT

Do you want the amplitudes to be created automatically in Abaqus/CAE?
(Note: Define an actuator driven feature for the actuator to have any effect.)

Figure 5-7: Warning dialog for Dymola Logical file

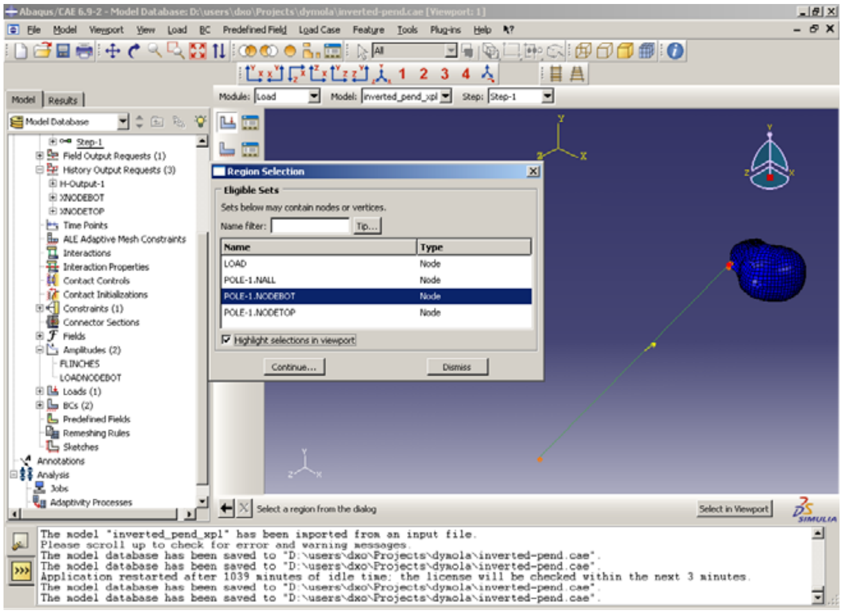
does not have a corresponding history output in Abaqus/CAE model will be displayed, as in Figure 5-7. Click **No**. In the model tree of Abaqus/CAE, expand **History Output Requests** for the model **inverted_pend_xpl**. Rename the previously changed sensor back to **XNODETOP** by selecting the history output **XNODETOP1** followed by right-click mouse button and choose **Rename…**. The name will be updated in the **Abaqus Signals** list of **Sensors** tab.

- Once again, click **Check Signals** to see if the signal names are consistent between the Dymola and Abaqus/CAE models. The signals should now be consistent. Click **No** on the info dialog and then click **Cancel** to close the **Match Existing Dymola File** dialog box.
- Save the Abaqus/CAE model.

5.2.4 Submitting the Abaqus-Dymola co-simulation

In this section of the tutorial, an Abaqus-Dymola co-simulation will be defined and run from the plug-in.

- Before running the co-simulation, it is very important to define an actuator driven feature (like a load) in the Abaqus/CAE model – otherwise, the actuator will not have any effect. A load that will use the **LOADNODEBOT** actuator will be created in the Abaqus/CAE model. Switch to the **Load** module. Click the **Create Load** icon from the toolbox area. Give an appropriate name for the load, select **Mechanical** for **Category** and select **Concentrated force** under **Types for Selected Step**. Click **Continue…** to select the points for the load. Click **Sets…** in the prompt area of the viewport. In the ensuing **Region Selection** dialog box, select the **POLE-1.NODEBOT** node set (as shown in Figure 5-8) and click **Continue…**. In the subsequent **Edit Load** dialog box, specify a value of **-1** for **CF1** and choose **LOADNODEBOT** for **Amplitude**.



Module: Load Model: inverted_pend_xpl Step: Step-1

Sets below may contain nodes or vertices.

Name filter:

Name	Type
LOAD	Node
POLE-1.NALL	Node
POLE-1.NODEBOT	Node
POLE-1.NODETOP	Node

☑ Highlight selections in viewport

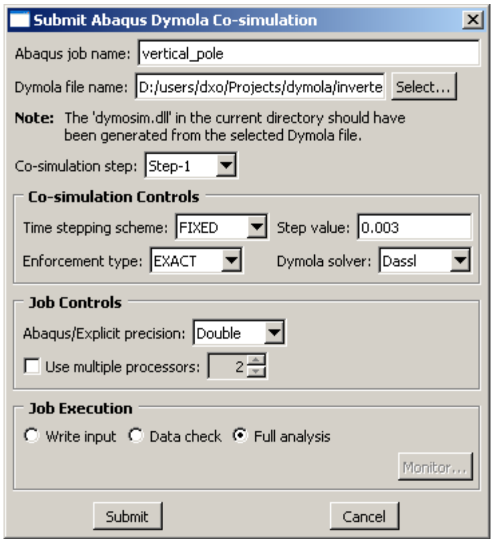
Select a region from the dialog

```
The model "inverted_pend_xpl" has been imported from an input file.
Please scroll up to check for error and warning messages.
The model database has been saved to "D:\users\dxo\Projects\dymola\inverted-pend.cae".
The model database has been saved to "D:\users\dxo\Projects\dymola\inverted-pend.cae".
Application restarted after 1039 minutes of idle time; the license will be checked within the next 3 minutes.
The model database has been saved to "D:\users\dxo\Projects\dymola\inverted-pend.cae".
The model database has been saved to "D:\users\dxo\Projects\dymola\inverted-pend.cae".
```

Figure 5-8 Node set selection for CLOAD

- Now, the Abaqus-Dymola co-simulation can be specified and run. Switch to the **Job** module. Go to **Plug-ins → Dymola → Submit Abaqus Dymola Co-simulation….** The **Submit Abaqus Dymola Co-simulation** dialog box will be posted.
- Specify `vertical_pole` as the **Abaqus job name**. Click **Select…** and select **inverted_pend.mo** as the **Dymola file name**. Choose **Step-1** as the **Co-simulation step**.
- Under **Co-simulation Controls**, choose **FIXED** as **Time Stepping Scheme**, specify **0.003** for **Step value**, choose **EXACT** for **Enforcement type** and choose **Dassl** for **Dymola solver**.
- Under **Job Controls**, choose **Double** for **Abaqus/Explicit precision**. Select the no. of multiple processors if required.
- Under **Job Execution**, select **Data check**.

The models have now been set-up for co-simulation. Make sure that the Dymola model has been translated properly and the `dymosim.dll` file exists in the work directory. The dialog box should look like the one in Figure 5-9. Click **Submit** to run a data check on the co-simulation. Once the jobs start to execute, the **Monitor** button will be activated. Click **Monitor…** to open the co-simulation job monitor dialog box. Click the **Refresh** button at the bottom to update the job files. Look at the DAT and Status files for possible warning and error messages. Once the data check is complete and successful, select **Full analysis** under **Job Execution** and re-submit the complete co-simulation analysis. Monitor the Abaqus job files from the job monitor dialog. The co-simulation should take about a minute to complete. Once the job completes successfully, open the output database in Abaqus/Viewer.



Submit Abaqus Dymola Co-simulation

Abaqus job name: vertical_pole

Dymola file name: D:/users/dxo/Projects/dymola/inverte

Note: The 'dymosim.dll' in the current directory should have been generated from the selected Dymola file.

Co-simulation step: Step-1

Co-simulation Controls

Time stepping scheme: FIXED

Step value: 0.003

Enforcement type: EXACT

Dymola solver: Dassl

Job Controls

Abaqus/Explicit precision: Double

☐ Use multiple processors: 2

Job Execution

○ Write input ○ Data check ● Full analysis

Figure 5-9: Submit Abaqus Dymola Co-simulation dialog box

5.3 Results

In Abaqus/Viewer open **`vertical_pole.odb`**. If you animate the deformed shape in Abaqus/Viewer, you will see that the pole becomes vertical after which it tilts again since impact-like loads are applied at given times.

History outputs of the coordinates at the top and bottom nodes of the pole should appear as in Figure 5-10. The coordinates are quite different to start with after which they become very similar at 0.314 sec. Because of the impact load, they become different again only to become the same at about 0.68 seconds and so on.

The control force computed by Dymola is plotted in Figure 5-11.

As an exercise, reduce the fixed time increment size in the **Submit Abaqus Dymola Co-simulation** dialog box to 0.001 and re-run the co-simulation analysis. It is not necessary to translate the Dymola model again as it did not change. Also, try using the **Write input** option and compare the keywords of the generated input file with the **`inverted_pend_xpl.inp`** input file.

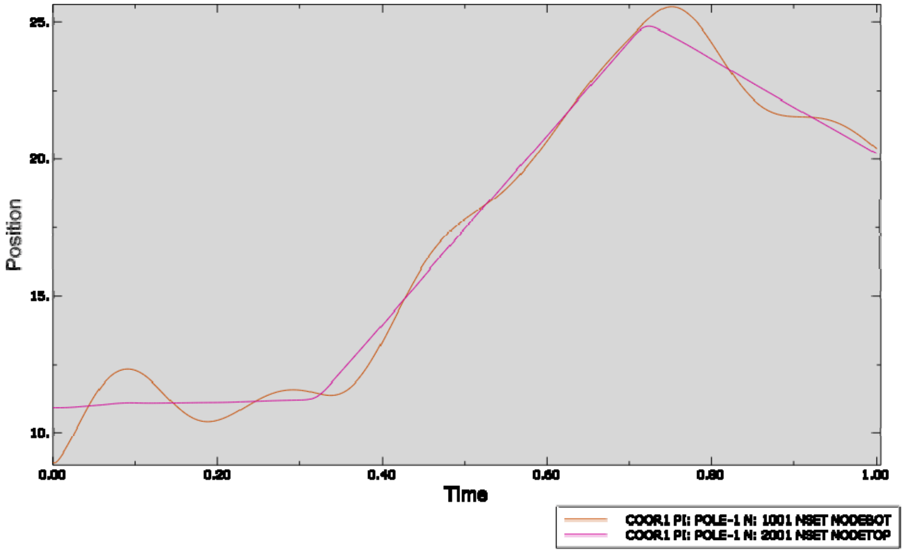


Figure 5-10: Top and bottom node coordinates (X-direction)

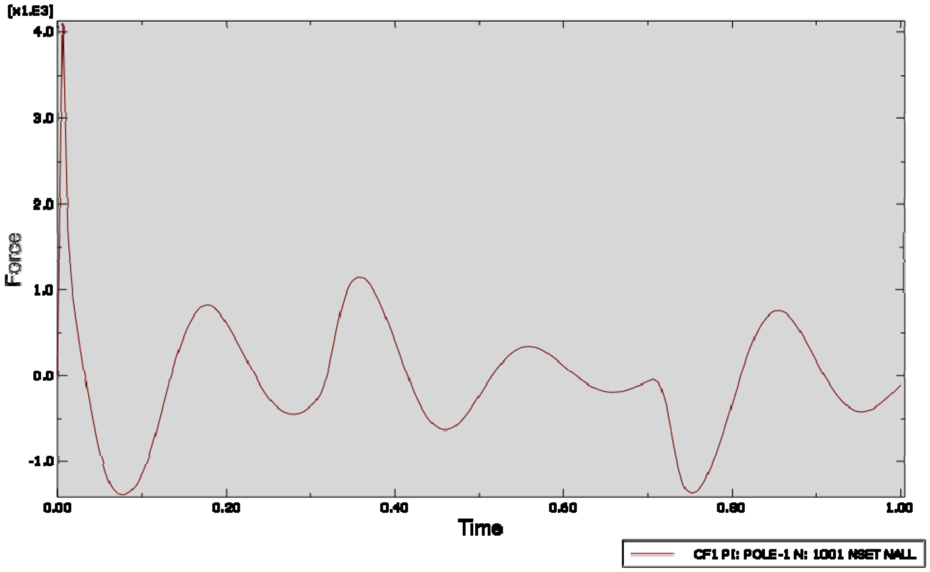
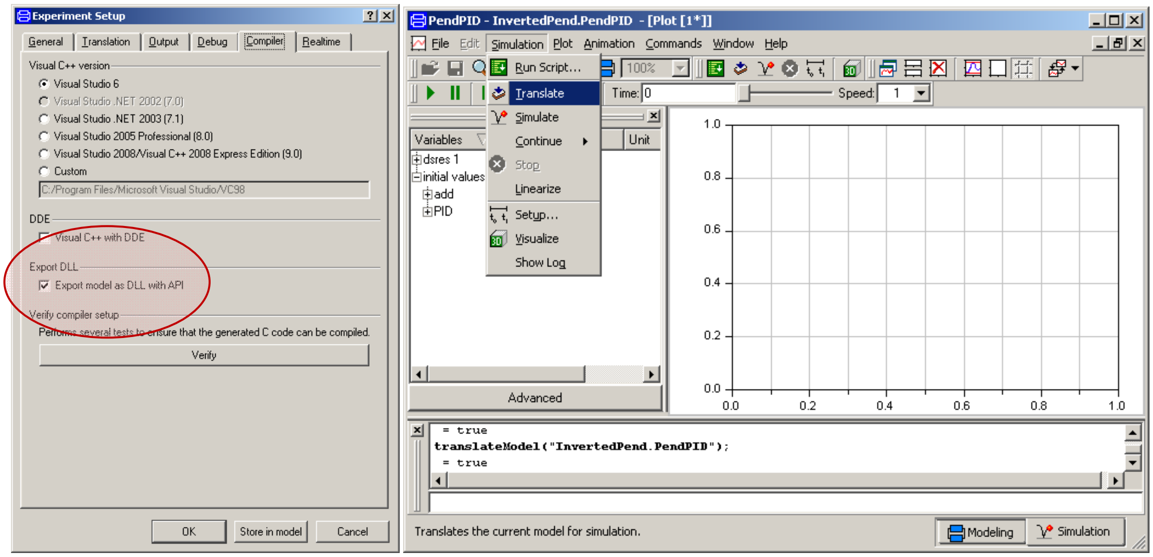


Figure 5-11: Control force computed by Dymola

6. *APPENDIX*

6.1 Translating the Dymola Model

From the bottom right of the **Modeling** panel in Dymola, click **Simulation** to switch to the **Simulation** panel. Select **Simulation → Setup** and then click the **Compiler** tab. Toggle on **Export model as DLL with API**, and click **OK**. From the **Simulation** panel, select **Simulation → Translate**. The current working directory should now have a file called `dymosim.dll` that was created by the translation.



● Visual Studio 6

C:/Program Files/Microsoft Visual Studio/VC98

☑ Export model as DLL with API

Performs several tests to ensure that the generated C code can be compiled.

```
 = true
translateModel("InvertedPend.PendPID");
 = true
```

Translates the current model for simulation.

Time: 0

Speed: 1

Figure 6-1: Translating the controller model