WHAT IS DYMOLA?

What is Dymola?

Features of Dymola

Dymola – Dynamic Modeling Laboratory – is suitable for modeling of various kinds of physical systems. It supports hierarchical model composition, libraries of truly reusable components, connectors and composite acasual connections. Model libraries are available in many engineering domains.

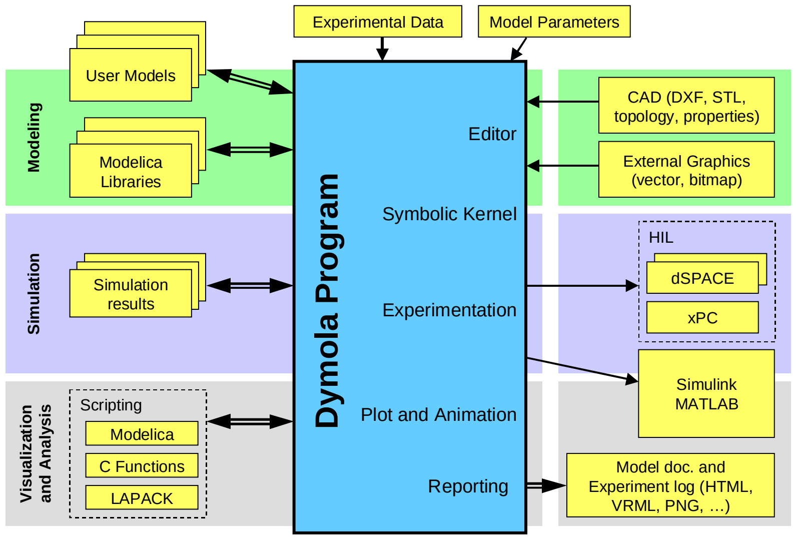
Dymola uses a new modeling methodology based on object orientation and equations. The usual need for manual conversion of equations to a block diagram is removed by the use of automatic formula manipulation. Other highlights of Dymola are:

- Handling of large, complex multi-engineering models.
- Faster modeling by graphical model composition.
- Faster simulation – symbolic pre-processing.
- Open for user defined model components.
- Open interface to other programs.
- 3D Animation.
- Real-time simulation.

Architecture of Dymola

The architecture of the Dymola program is shown below. Dymola has a powerful graphic editor for composing models. Dymola is based on the use of Modelica models stored on files. Dymola can also import other data and graphics files. Dymola contains a symbolic translator for Modelica equations generating C-code for simulation. The C-code can be exported to Simulink and hardware-in-the-loop platforms.

Dymola has powerful experimentation, plotting and animation features. Scripts can be used to manage experiments and to perform calculations. Automatic documentation generator is provided.



Basic Operations

Dymola has two kinds of windows: Main window and Library window. The Main window operates in one of two modes: Modeling and Simulation.

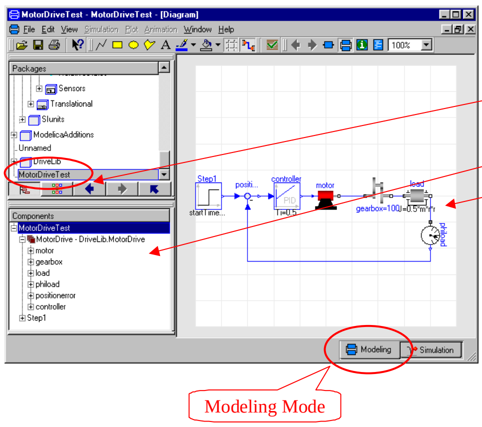
The Modeling mode of the Main window is used to compose models and model components.

The Simulation mode is used to make experiment on the model, plot results and animate the behavior. The Simulation mode also have a scripting subwindow for automation of experimentation and performing calculations.

Simulate an existing model

Find the model

Dymola starts in Modeling mode. The model to simulate is found by using the Open or Demo commands in the File menu. Example models are also found in the libraries: Modelica or ModelicaAdditions or in other libraries opened by the Library command.

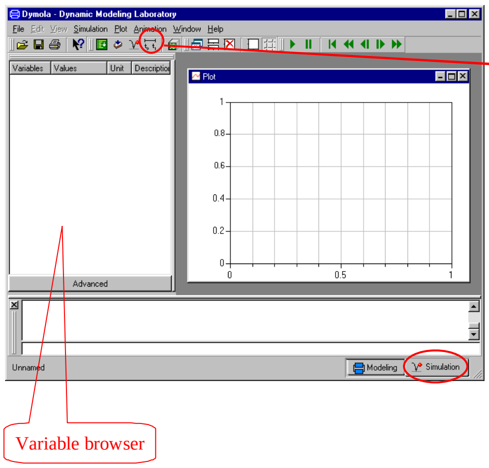


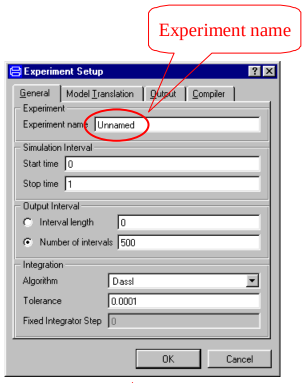
Browsing

- Tree Browser for model packages and models
- Component hierarchy
- Viewer
  - diagram
  - documentation

Simulation Mode

The Simulation Mode is used for experimentation. It has simulation setup to define duration of simulation, etc., plot windows, animation windows and variable browser.





General | Model Translation | Output | Compiler

Experiment
Experiment name: Unnamed

Simulation Interval
Start time: 0
Stop time: 1

Output Interval
Interval length: 0
[✓ Number of intervals]: 500

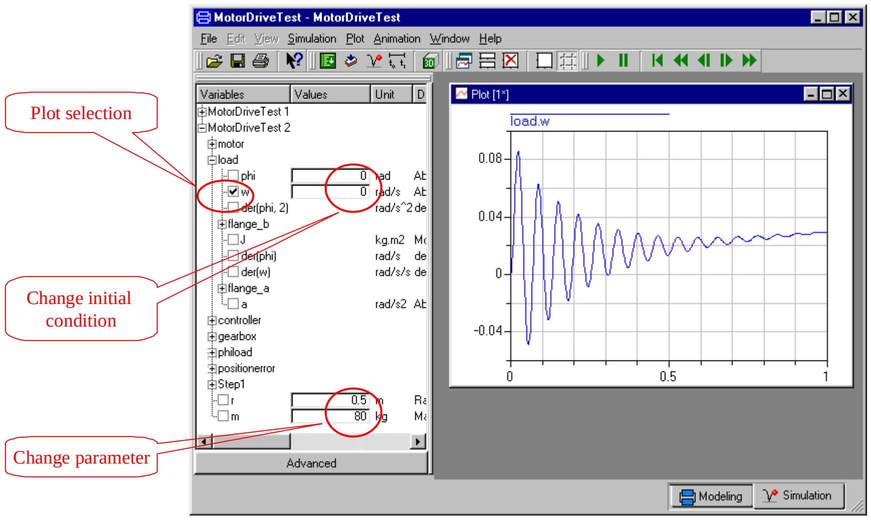
Integration
Algorithm: Dassl
Tolerance: 0.0001
Fixed Integrator Step: 0

OK | Cancel

Simulate Mode

- Simulate controls
  - Setup
  - Simulate
- Plot windows
  plot window controls
- Animation windows
  animation window controls

The variable browser allows selecting plot variables, changing parameters and initial conditions.



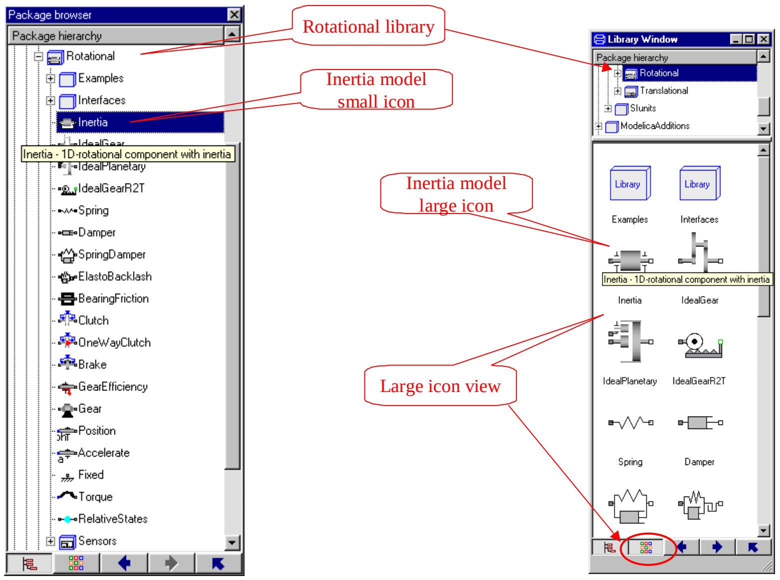
An animation window shows a 3D view of the simulated model. The animation can be run at different speeds, halted, single stepped and run backwards.

Building a model

The graphical model editor is used for creating and editing models in Dymola. Structural properties, such as, components, connectors and connections are edited graphically, while equations and declarations are edited with a built-in text editor.

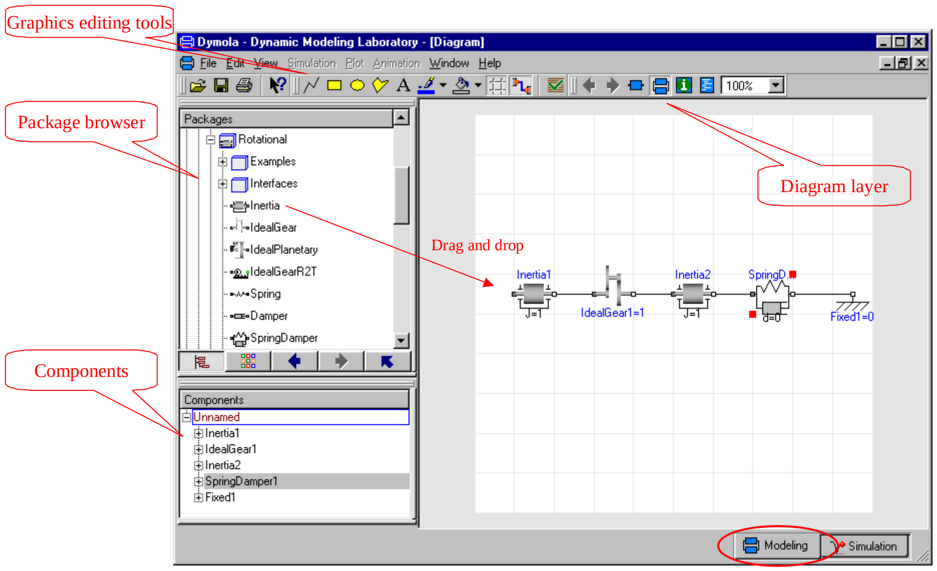
Find a component model

The package browser allows viewing and selecting component models from a list of models with small icons. It is also possible to get a large icon view

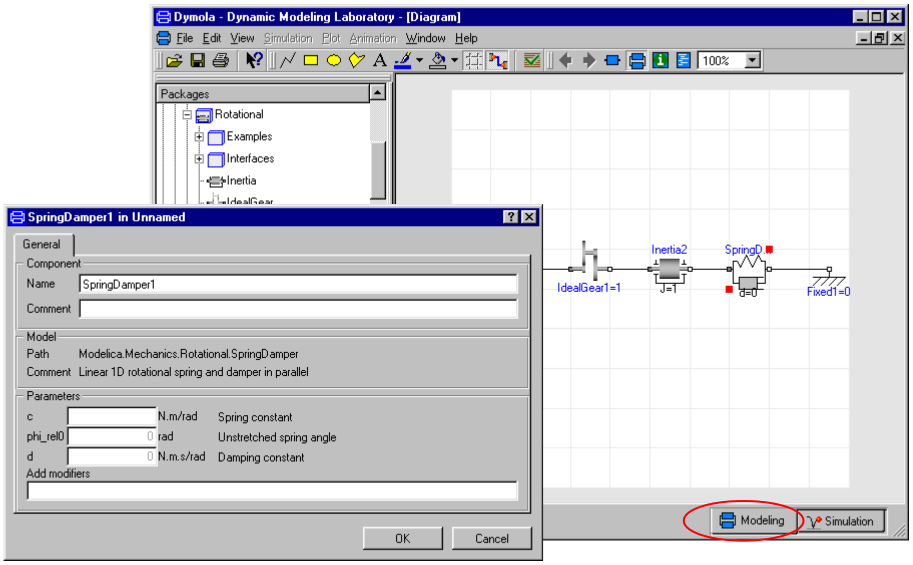


.

Components are dragged from the package browser to the diagram layer and connected. The component hierarchy is shown in the component browser.



By double clicking on a component, a dialog for giving the name of the component and its parameters are shown.



SpringDamper1 in Unnamed

General

Component

Name	SpringDamper1
Comment	

Model

Path	Modelica.Mechanics.Rotational.SpringDamper
Comment	Linear 1D rotational spring and damper in parallel

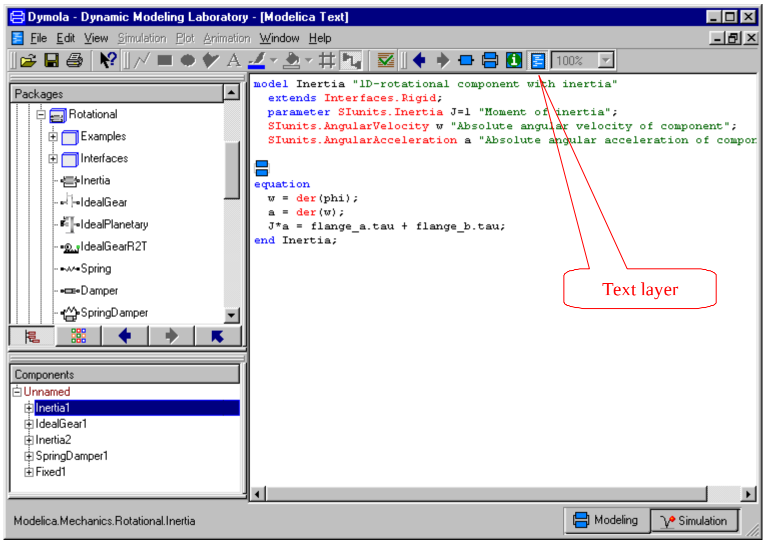
Parameters

c		N.m/rad	Spring constant
phi_rel0	0	rad	Unstretched spring angle
d	0	N.m.s/rad	Damping constant

Add modifiers

OK Cancel

The model browser allows opening a component for inspection of the documentation or the model itself, for example, by looking at the underlying Modelica code which is shown in the Modelica Text layer. This is also the editor for entering Modelica code, i.e. declarations and equations for low level models.



```
model Inertia "1D-rotational component with inertia"
  extends Interfaces.Rigid;
  parameter SIunits.Inertia J=1 "Moment of inertia";
  SIunits.AngularVelocity w "Absolute angular velocity of component";
  SIunits.AngularAcceleration a "Absolute angular acceleration of compon

equation
  w = der(phi);
  a = der(w);
  J*a = flange_a.tau + flange_b.tau;
end Inertia;
```

Features of Modelica

Modelica is an object-oriented language for modeling of large, complex and heterogeneous physical systems. It is suited for multi-domain modeling, for example for modeling of mechatronic systems within automotive, aerospace and robotics applications. Such systems are composed of mechanical, electrical and hydraulic subsystems, as well as control systems.

General equations are used for modeling of the physical phenomena. The language has been designed to allow tools to generate efficient code automatically. The modeling effort is thus reduced considerably since model components can be reused, and tedious and error-prone manual manipulations are not needed.

Background

Modeling and simulation are becoming more important since engineers need to analyse increasingly complex systems composed of components from different domains. Current tools are generally weak in treating multi-domain models because the general tools are block-oriented and thus demand a huge amount of manual rewriting to get the equations into explicit form. The domain-specific tools, such as circuit simulators or multibody programs, cannot handle components of other domains in a reasonable way.

There is traditionally too large a gap between the user’s problem and the model description that the simulation program understands. Modeling should be much closer to the way an engineer builds a real system, first trying to find standard components like motors, pumps and valves from manufacturers’ catalogues with appropriate specifications and interfaces.

Equations and reuse

Equations facilitate true model reuse.

Equations are used in Modelica for modeling of the physical phenomena. No particular variable needs to be solved for manually because Dymola has enough information to decide that automatically. This is an important property of Dymola to enable handling of large models having more than hundred thousand equations. Modelica supports several formalisms: ordinary differential equations (ODE), differential-algebraic equations (DAE), bond graphs, finite state automata, Petri nets etc.

The language has been designed to allow tools to generate very efficient code. Modelica models are used, for example, in hardware-in-the-loop simulation of automatic gearboxes, which have variable structure models. Such models have so far usually been treated by hand, modeling each mode of operation separately. In Modelica, component models are used for shafts, clutches, brakes, gear wheels etc. and Dymola can find the different modes of operation automatically. The modeling effort is considerably reduced since model components can be reused and tedious and error-prone manual manipulations are not needed.

Modelica history

Reuse is a key issue for handling complexity. There had been several attempts to define object-oriented languages for physical modeling. However, the ability to reuse and exchange models relies on a standardized format. It was thus important to bring this expertise together to unify concepts and notations.

A design group was formed in September 1996 and one year later, the first version of the Modelica language was available (http://www.Modelica.org). Modelica is intended to serve as a standard format so that models arising in different domains can be exchanged between tools and users. It has been designed by a group of more than 25 experts with previous know-how of modeling languages and differential-algebraic equation models. After more than 30 three-days' meetings during a five year period, version 2.0 of the language specification was finished in January, 2002.

GETTING STARTED WITH DYMOLA

Getting started with Dymola

Introduction

This guide will take you through some examples in order to get you started with Dymola. For detailed information about the program, you are referred to the on-line documentation and the user's manuals. The on-line documentation is available in the Help menu after selecting Documentation. The tool tips and the "What's this" feature are fast and convenient ways to access information.

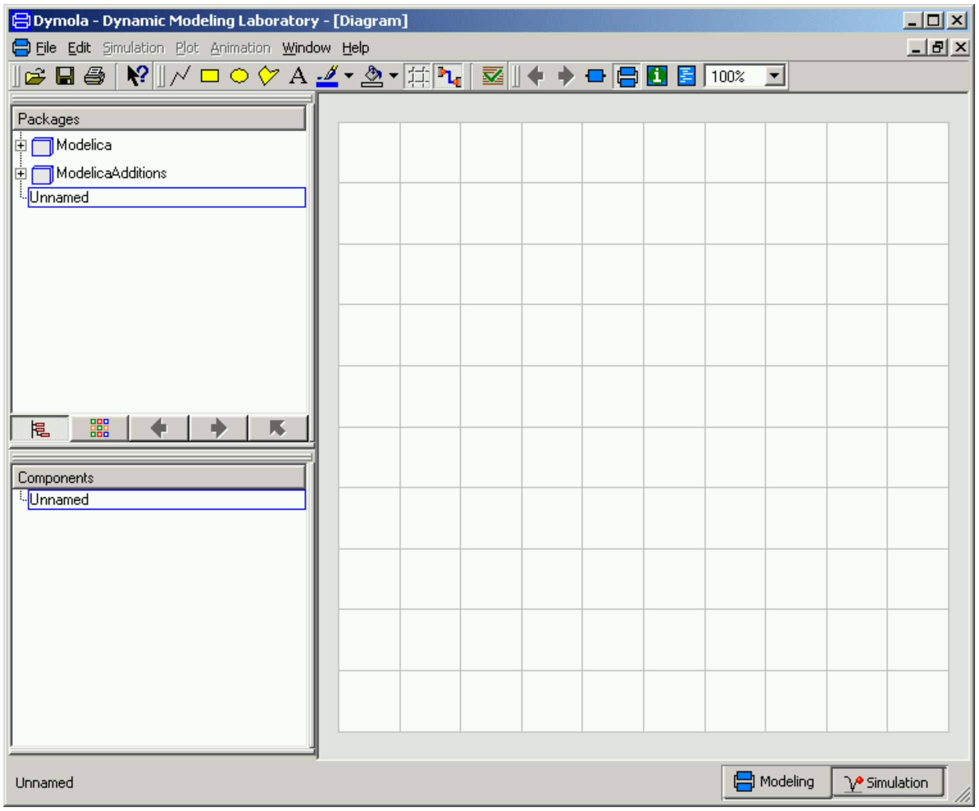
Start Dymola. A Dymola window appears. A Dymola window operates in one of the two modes:

- **Modeling** for finding, browsing and composing models and model components.
- **Simulation** for making experiments on the model, plotting results, and animating behavior.

Dymola starts in Modeling mode. The active mode is selected by clicking on the tabs in the bottom right corner of the Dymola window.

The operations, tool buttons available, and types of sub-windows appearing depends on the mode. Dymola starts with a useful default configuration, but allows customizing.

The Dymola window.

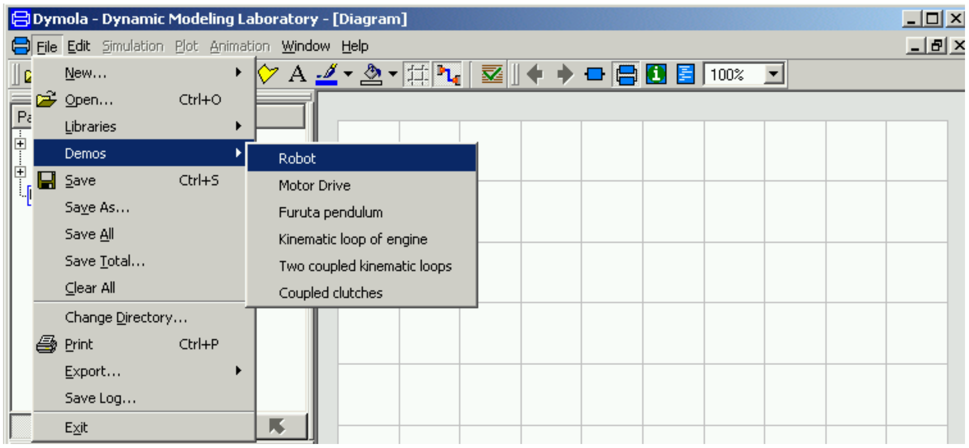

Dymola - Dynamic Modeling Laboratory - [Diagram]
File Edit Simulation Plot Animation Window Help
100%
Packages
Modelica
ModelicaAdditions
Unnamed
Components
Unnamed
Unnamed
Modeling
Simulation

Simulating a model — industrial robot

This first example will show how to browse an existing model, simulate it, and look at the results. If you want to learn the basics first, you can skip to a smaller example in the next section "Solving a non-linear differential equation."

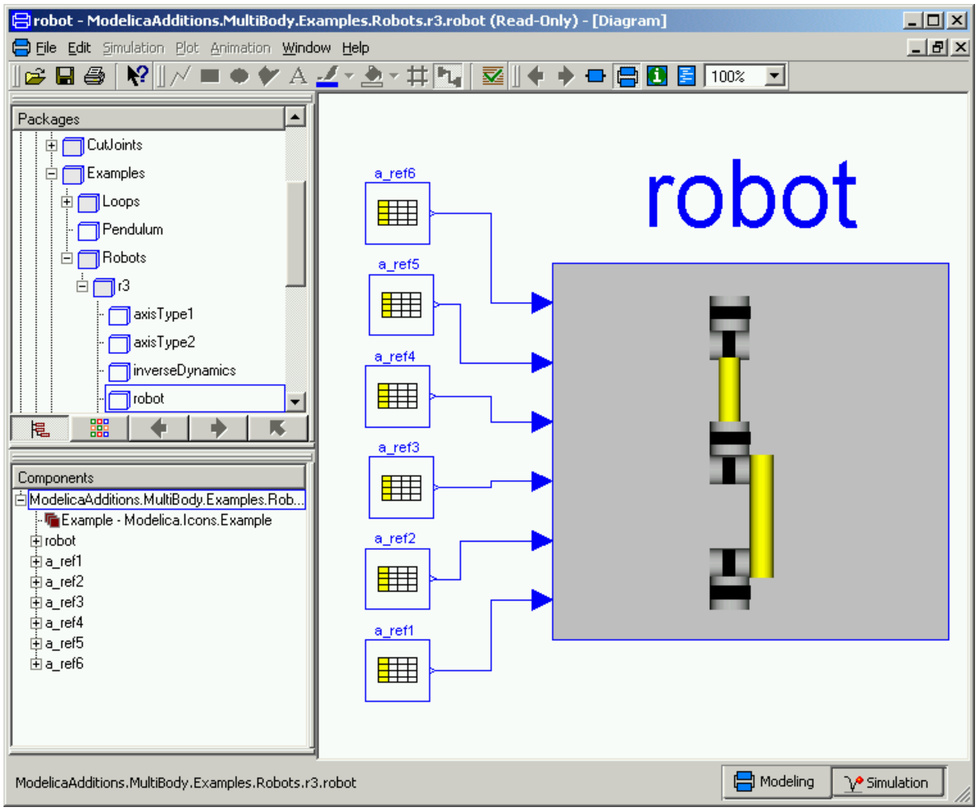
We will study a model of an industrial robot. Start Dymola. The Dymola window appears in Modeling mode as displayed above. To view the industrial robot model, use the File/Demos menu and select Robot.

Opening a demo example.



Dymola starts loading the model libraries needed for the robot model and displays it.

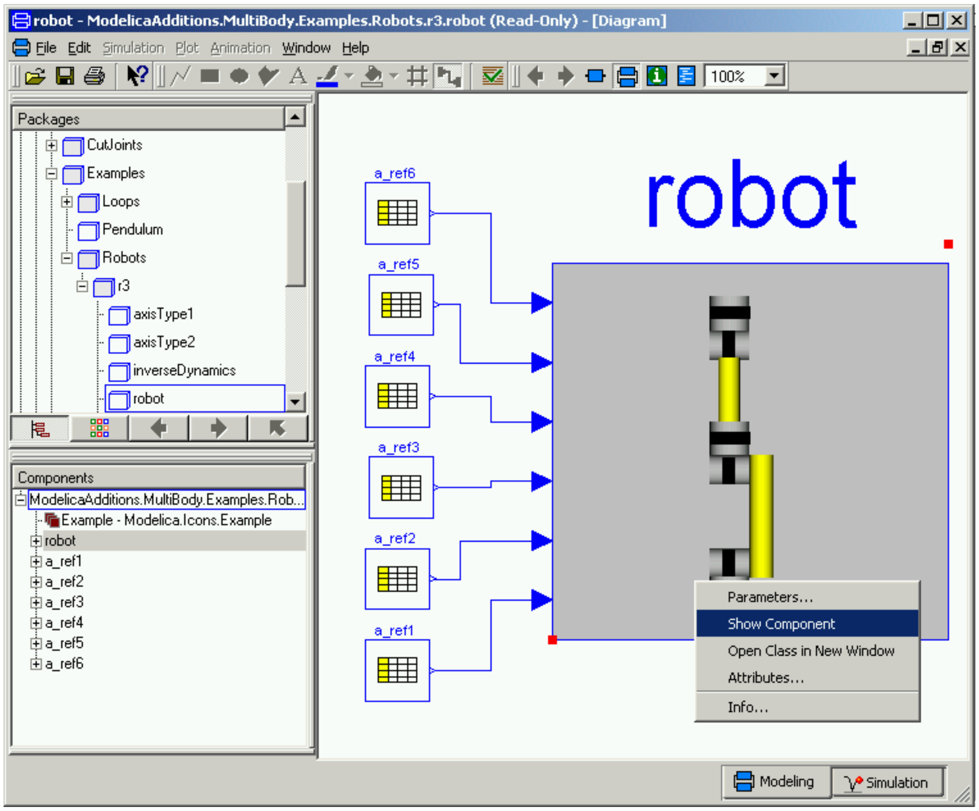
The robot demo.


robot - ModelicaAdditions.MultiBody.Examples.Robots.r3.robot (Read-Only) - [Diagram]
File Edit Simulation Plot Animation Window Help
100%
Packages
CutJoints
Examples
Loops
Pendulum
Robots
r3
axisType1
axisType2
inverseDynamics
robot
Components
ModelicaAdditions.MultiBody.Examples.Rob...
Example - Modelica.Icons.Example
robot
a_ref1
a_ref2
a_ref3
a_ref4
a_ref5
a_ref6
a_ref6
a_ref5
a_ref4
a_ref3
a_ref2
a_ref1
robot
ModelicaAdditions.MultiBody.Examples.Robots.r3.robot
Modeling
Simulation

The package browser in the upper left sub-window displays the package hierarchy and it is now opened up with the robot model selected and highlighted. The model diagram in the right sub-window opens up and shows the top-level structure of the model. The model diagram has an icon for the robot model with reference signal generators connected to the joints. The reference signal generators are implemented as table blocks. The component browser in the lower left sub-window also shows the components of the robot experiment in a tree structured view.

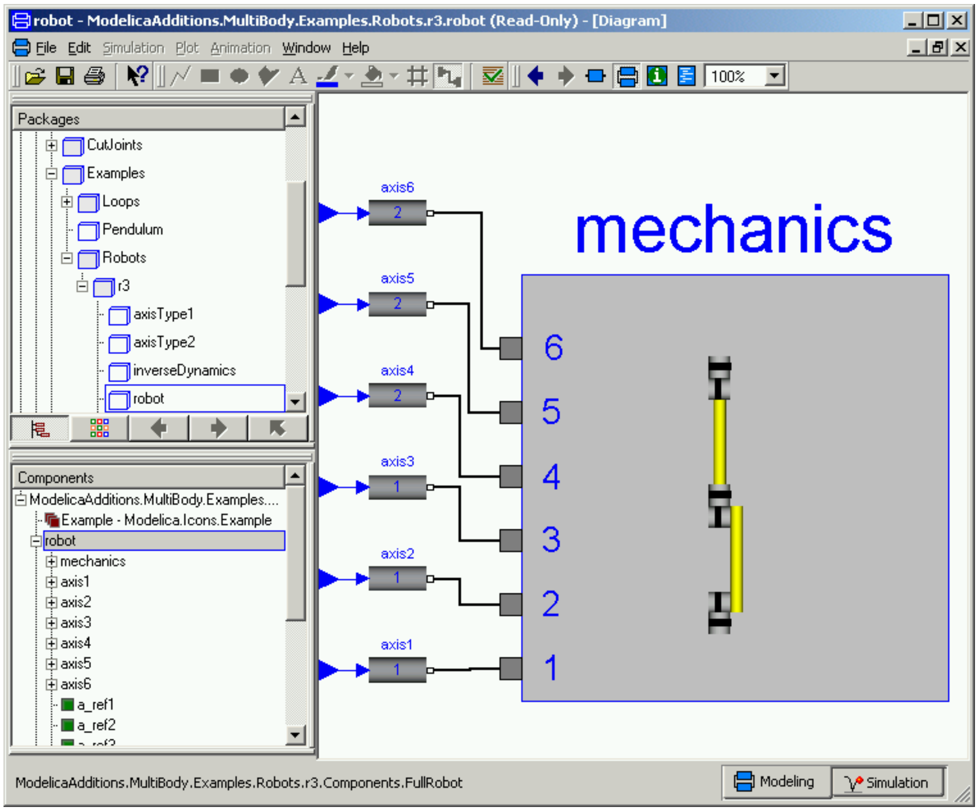
To inspect the robot model, select the icon in the Diagram window (red handles appear, see below) and press the right button on the mouse. From the menu, choose Show Component.

About to view the robot model.



It is not necessary to select the robot component explicitly by pressing the left button on the mouse to access its menu. It is sufficient to just have the cursor on its icon in the diagram window and press the right button on the mouse. The component browser also gives easy access to the robot component. Just position the cursor over “robot”. The component browser provides a tree representation of the component structure. The diagram window and the component browser are synchronized to give a consistent view. When you select a component in the diagram window, it is also highlighted in the component browser and vice versa. The diagram window gives the component structure of one component, while the component browser gives a more global view; useful to avoid getting lost in the hierarchical component structure.

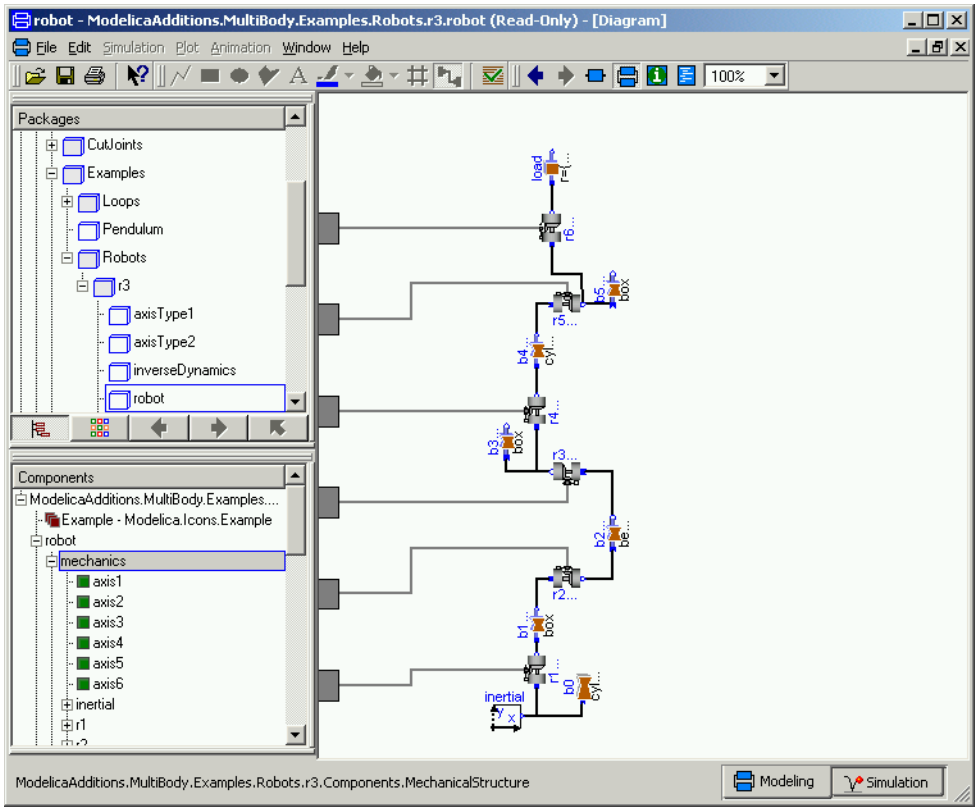
The robot model.


robot - ModelicaAdditions.MultiBody.Examples.Robots.r3.robot (Read-Only) - [Diagram]
File Edit Simulation Plot Animation Window Help
100%
Packages
CutJoints
Examples
Loops
Pendulum
Robots
r3
axisType1
axisType2
inverseDynamics
robot
Components
ModelicaAdditions.MultiBody.Examples....
Example - Modelica.Icons.Example
robot
mechanics
axis1
axis2
axis3
axis4
axis5
axis6
a_ref1
a_ref2
axis6
axis5
axis4
axis3
axis2
axis1
mechanics
6
5
4
3
2
1
ModelicaAdditions.MultiBody.Examples.Robots.r3.Components.FullRobot
Modeling
Simulation

The Diagram window now displays the internals of the robot model. There is a large icon for the mechanics connected to drive lines for each joint. The component browser is opened up to also show the internals of the robot model.

To view the mechanical structure, select the icon and choose Show Component as above. The diagram window as shown below shows how the robot model consists of connected joints and masses.

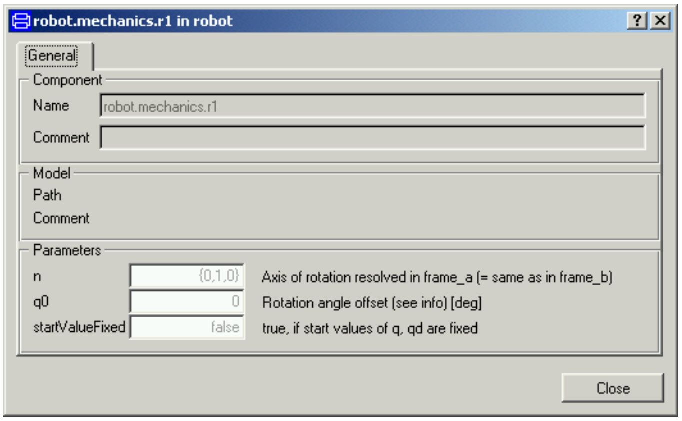
The mechanical structure of the robot.



ModelicaAdditions.MultiBody.Examples.Robots.r3.Components.MechanicalStructure

Double click on, for example, r1 at the bottom of the diagram window. This is a revolute joint. The Parameter dialogue appears. The Parameter dialogue can also be accessed from the right button menu. Double clicking the left button selects the first alternative from the right button menu.

Parameter dialogue.



robot.mechanics.r1 in robot

General

Component

Name	robot.mechanics.r1
Comment	

Model

Path

Comment

Parameters

n	{0,1,0}	Axis of rotation resolved in frame_a (= same as in frame_b)
q0	0	Rotation angle offset (see info) [deg]
startValueFixed	false	true, if start values of q, qd are fixed

Close

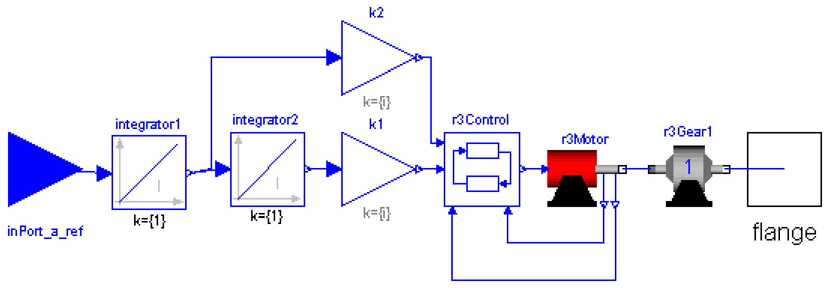
The parameter dialogue allows the user to inspect actual parameter values. In this case the parameter values are write protected to avoid unintentional changes of the demo example. Thus the dialogue just has a Close button. When the parameter values can be changed there is one OK button and one Cancel button to choose between. The values are dimmed to indicate they are not given at the top-level model, but somewhere down in the component hierarchy.

The parameter n defines the axis of rotation. The value for this revolute joint is {0, 1, 0}, i.e. the axis of rotation is vertical. Press Close.

To learn more about this component, press the right mouse button and select Info. An HTML browser is opened to show the documentation of the Revolute joint. By double clicking on the other components, you can, for example, see what parameters a mass has.

Let us now inspect the drive train model. There are several possible ways to get it displayed. Press the Previous button (the toolbar button with the bold left arrow) once to go to the robot model and then use the right button menu on one of the axis icons. Please, note that robot.mechanics also has components axis1, ..., axis6, but those are just connectors. You shall inspect for example robot.axis1. Another convenient way is to use the component browser and use the right button menu and select Show Component. Since this is the first menu option, double clicking will open up the component in the Diagram window. Please, recall that double clicking on a component in the diagram window pops up the parameter dialogue.

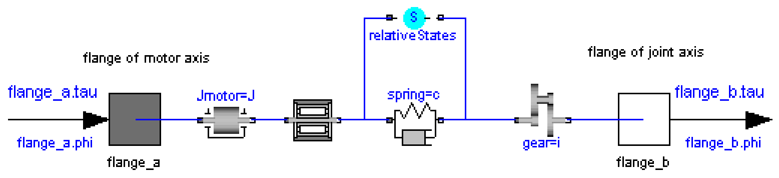
The robot drive train.



The controller and drive train have angular accelerations as reference signals of the joint. Two integrators are used to form angular velocity and angle. These signals are connected to the control system that generates the current references for the motor which drives the gearbox.

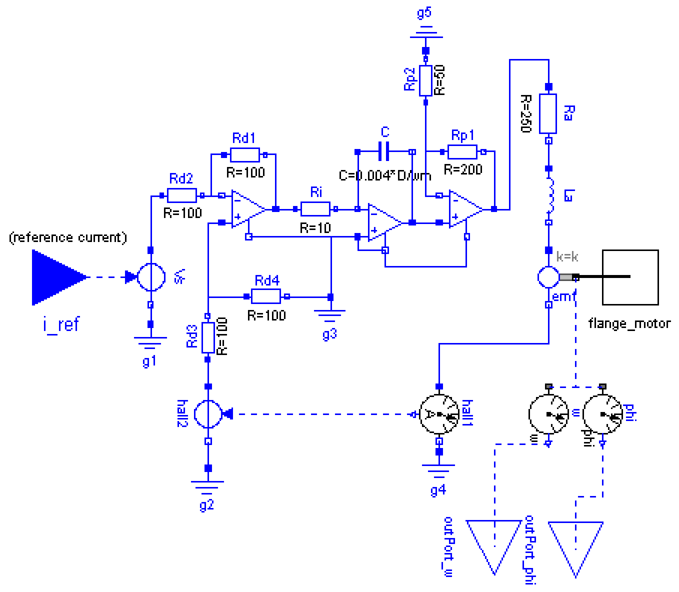
View the component r3Gear1 in a similar way as for the other components. It shows the gearbox and the model of friction of the bearings and elasticity of the shafts.

The robot gearbox.



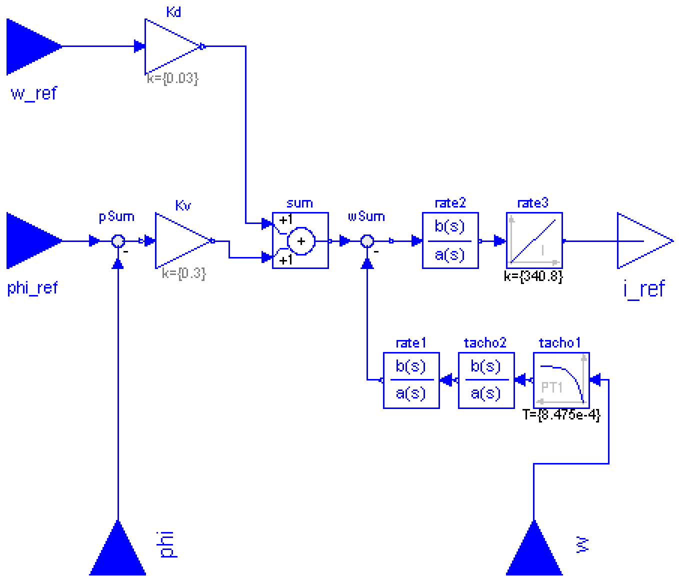
The motor model consists of the electromotorical force, three operational amplifiers, resistors, inductors, and sensors for the feedback loop.

The robot motor.



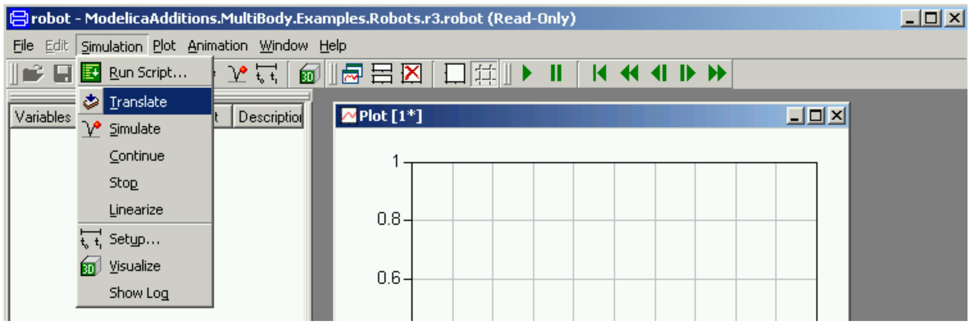
The control system takes measurements of angle and angular velocity from the motor and corresponding reference signals and produces the reference current for the motor.

The robot control system



Simulation

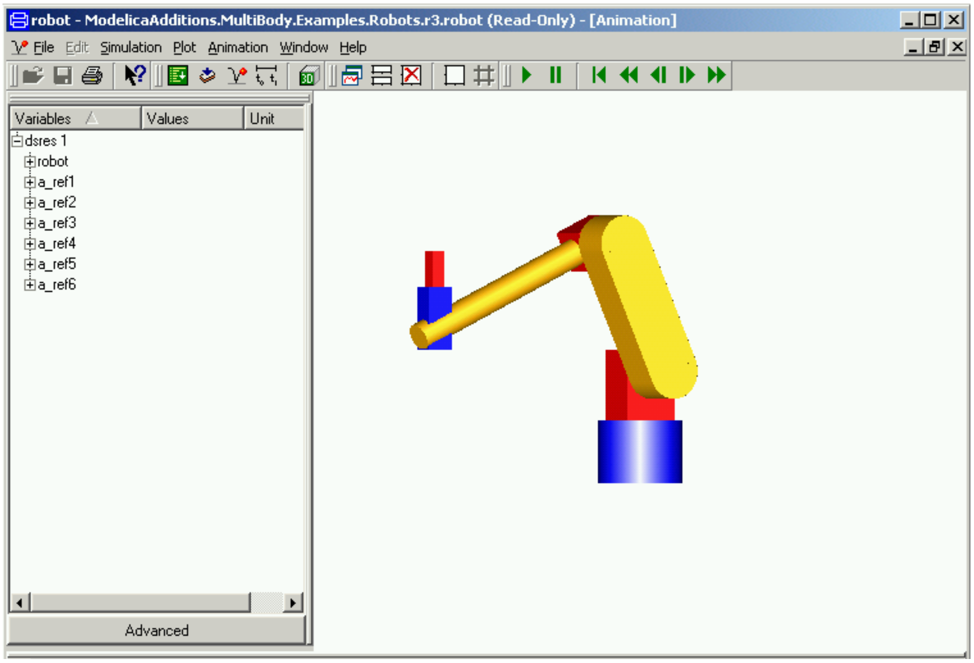
Let us simulate the robot model. To enter the Simulation mode, click on the tab at the bottom right. The simulation menu is now activated and new tool bar buttons appear.


robot - ModelicaAdditions.MultiBody.Examples.Robots.r3.robot (Read-Only)
File Edit Simulation Plot Animation Window Help
Run Script...
Translate
Simulate
Continue
Stop
Linearize
Setup...
Visualize
Show Log
Variables
Description
Plot [1*]
1
0.8
0.6

The simulation menu includes commands to setup and run simulations. The Setup menu item opens a dialogue allowing the setting of start and stop times. The icon indicates which toolbar button to use for quick access to the set up dialogue. The Simulate menu item starts a simulation.

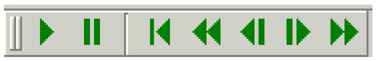
In this case, however, a command script has been prepared. To run the script, select Simulation/Run Script or push the toolbar button and select r3.mos. Dymola then translates the model and simulates automatically. The animation window is opened automatically, because the simulation result contains animation date.

Animated 3D view of the robot.


robot - ModelicaAdditions.MultiBody.Examples.Robots.r3.robot (Read-Only) - [Animation]
File Edit Simulation Plot Animation Window Help
Variables
Values
Unit
dsres 1
robot
a_ref1
a_ref2
a_ref3
a_ref4
a_ref5
a_ref6
Advanced

Start the animation by selecting Animation/Run or clicking the run button on the tool-bar,

Animation toolbar.

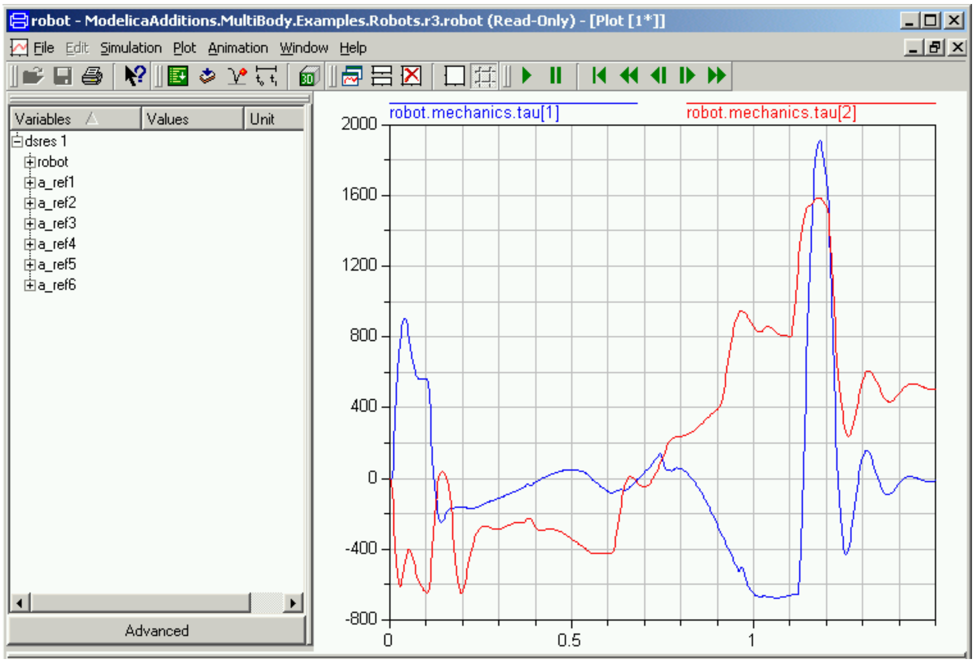


which contains the usual buttons of pausing rewinding, stepping forward and backward, respectively.

It is possible to change the viewing position by opening a view controller, Animation/3D View Control.

The command script also specifies plotting of the applied torque at the first and second joints.

Plotting the torque.



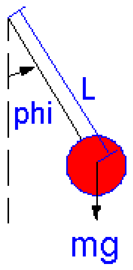
Other demo examples

Other demo examples can be found under the File/Demos menu. After selecting an example, it can be simulated by running a corresponding script file as was done for the robot example.

Solving a non-linear differential equation

This example will show how to define a simple model given by an ordinary differential equation. We will simulate a planar mathematical pendulum as shown in the figure.

A pendulum.



The variable m is the mass and L is the distance from the support to the center of mass. Let us assume the string is inextensible and massless, and further, let us neglect the resistance of the air and assume the gravitational field to be constant with g as the acceleration of gravity. The equation of motion for the pendulum is given by the torque balance around the origin as

```
J*der(w) = -m*g*sin(phi)
```

where J is the moment of inertia with respect to the origin. Assuming a point mass gives

```
J = m*L^2
```

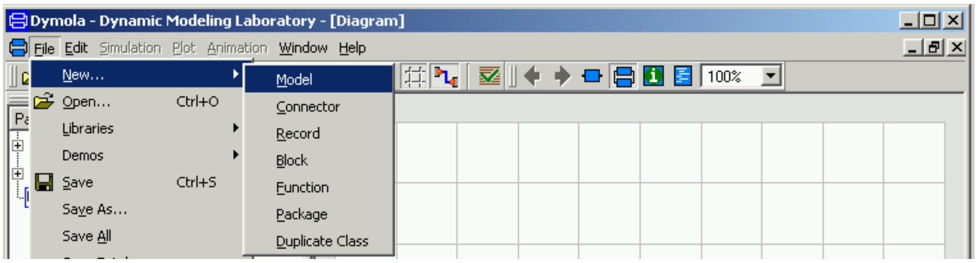
The variable w is the angular velocity and der(w) denotes the time derivative of w, i.e., the angular acceleration. For the angular position we have

```
der(phi) = w
```

Start Dymola or if it is already started then give the command File/Clear All in the Dymola window.

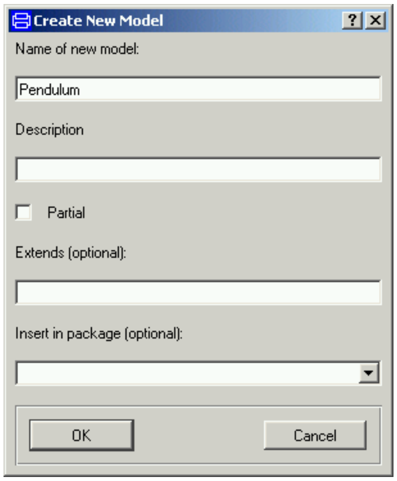
Click on the tab for Modeling at the bottom right. Select File/New Model.

The first step to create a new model.

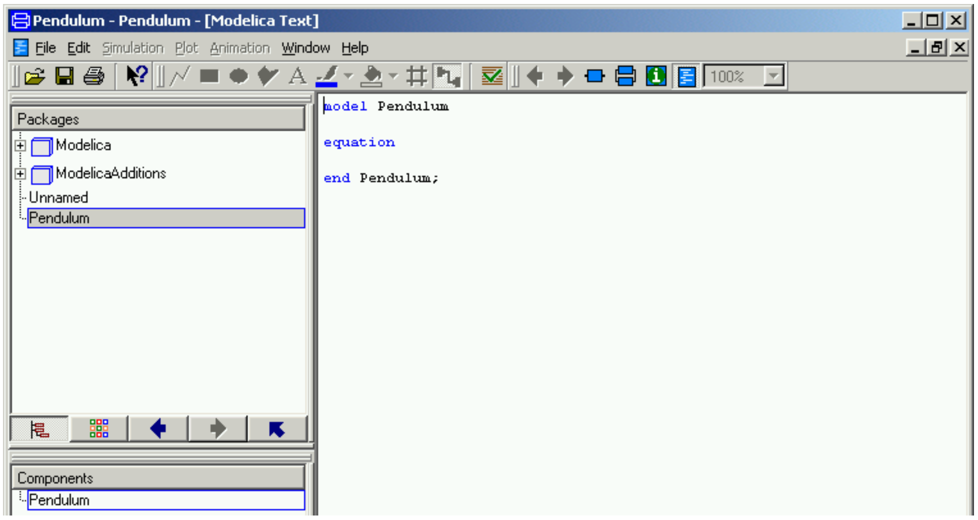


A dialog window opens. Enter “Pendulum” as the name of the model.

The dialogue to name a new model component.



Click OK.



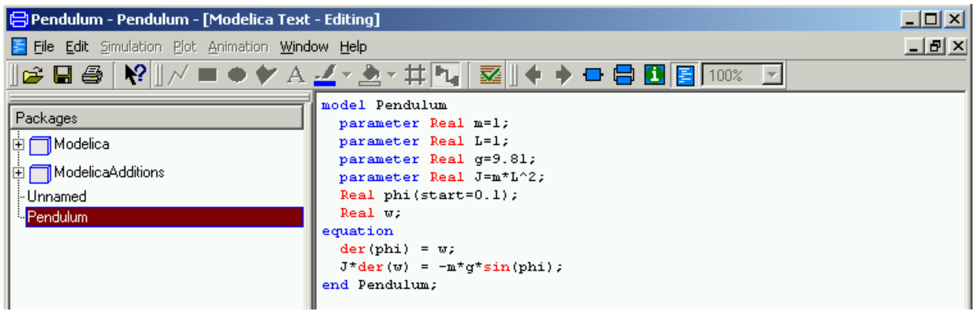
```
model Pendulum

equation

end Pendulum;
```

A model can be inspected and edited in different views. When specifying a behavior directly in terms of equations, it is most convenient to work with the model as the Modelica Text. Press the Modelica Text toolbar button (the rightmost tool button). The right sub-window can now be used as a text editor.

To declare the parameters and the variables, enter as shown the declarations for the parameters m, L and g, which also are given default values. The parameter J is bound in terms of other parameters. Finally, the time varying variables phi and w are declared. A start value is given for phi, while w is implicitly given a start value of zero.

To get the color coding of keywords and types, press the right mouse button and select Highlight Syntax.

Declaration of parameters, variables and equations.


Pendulum - Pendulum - [Modelica Text - Editing]
File Edit Simulation Plot Animation Window Help
100%
Packages
Modelica
ModelicaAdditions
Unnamed
Pendulum

```
model Pendulum
  parameter Real m=1;
  parameter Real L=1;
  parameter Real g=9.81;
  parameter Real J=m*L^2;
  Real phi(start=0.1);
  Real w;
equation
  der(phi) = w;
  J*der(w) = -m*g*sin(phi);
end Pendulum;
```

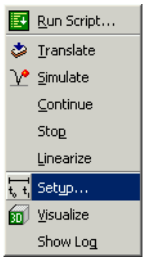
The model is ready to be saved. Select File/Save. Call the file pendulum.mo and place it in a working directory.

Simulation

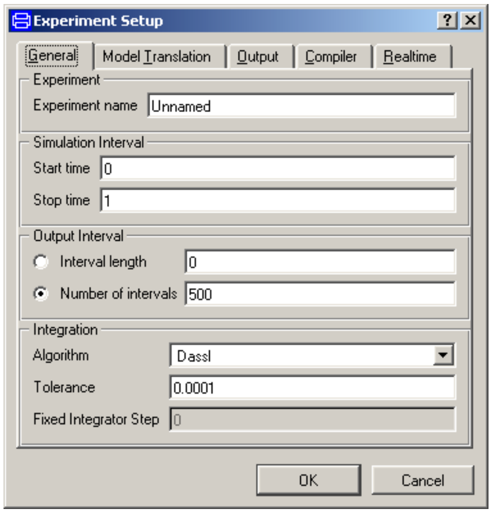
Now it is time to simulate. To enter the Simulation mode, click on the tab at the bottom right. The simulation menu is now activated and new tool bar buttons appear.

To set up the simulation select Simulation/Setup or click directly on the Setup toolbar button.

Selecting Setup in the Simulation menu.



The Simulation Setup menu.



Experiment
Experiment name: Unnamed

Simulation Interval
Start time: 0
Stop time: 1

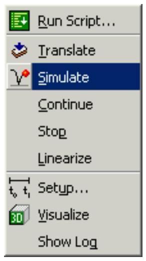
Output Interval
Interval length: 0
[✓ Number of intervals]: 500

Integration
Algorithm: Dassl
Tolerance: 0.0001
Fixed Integrator Step: 0

Set the stop time to 10 seconds. Click OK.

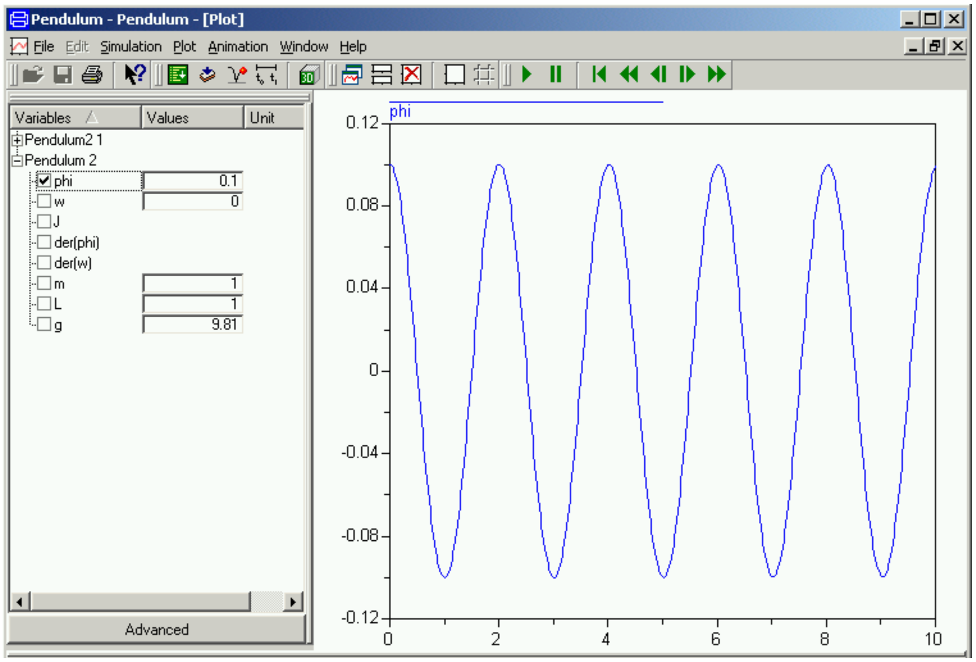
To run the simulation select Simulation/Simulate or click directly on the Simulate toolbar button.

Selecting Simulate in the Simulation menu.



Dymola first translates and manipulates the model and model equations to a form suitable for efficient simulation and then runs the simulation. You may explicitly invoke translation yourself by selecting Simulation/Translate or click on the Translate toolbar button.

Plotting the angle.



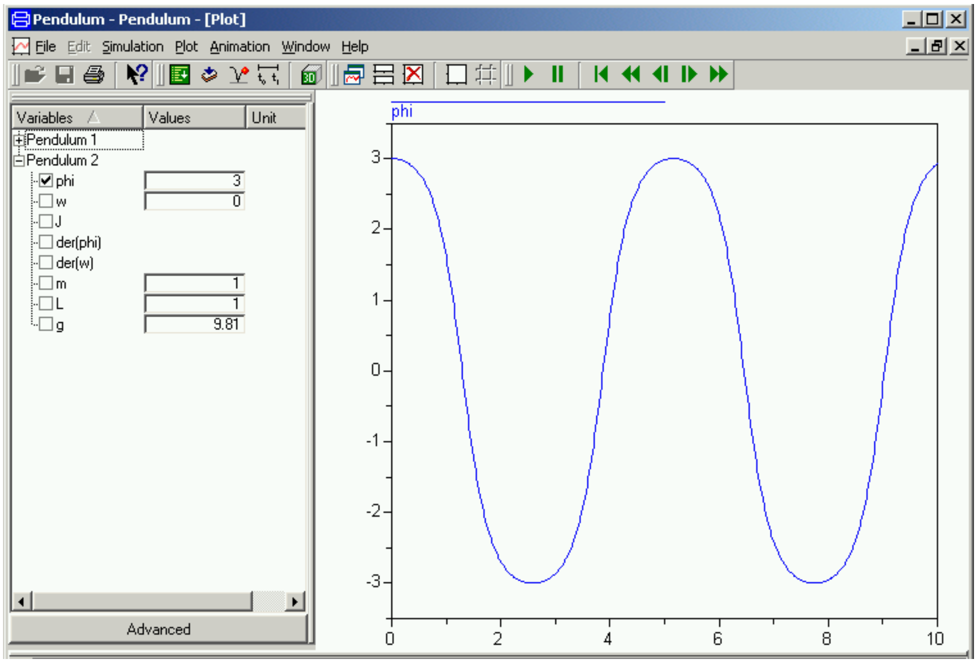
Pendulum - Pendulum - [Plot]
File Edit Simulation Plot Animation Window Help
Advanced
phi
0.12
0.08
0.04
0
-0.04
-0.08
-0.12
0
2
4
6
8
10

Variables	Values	Unit
Pendulum2 1		
Pendulum 2		
☑ phi	0.1	
☐ w	0	
☐ J		
☐ der(phi)		
☐ der(w)		
☐ m	1	
☐ L	1	
☐ g	9.81	

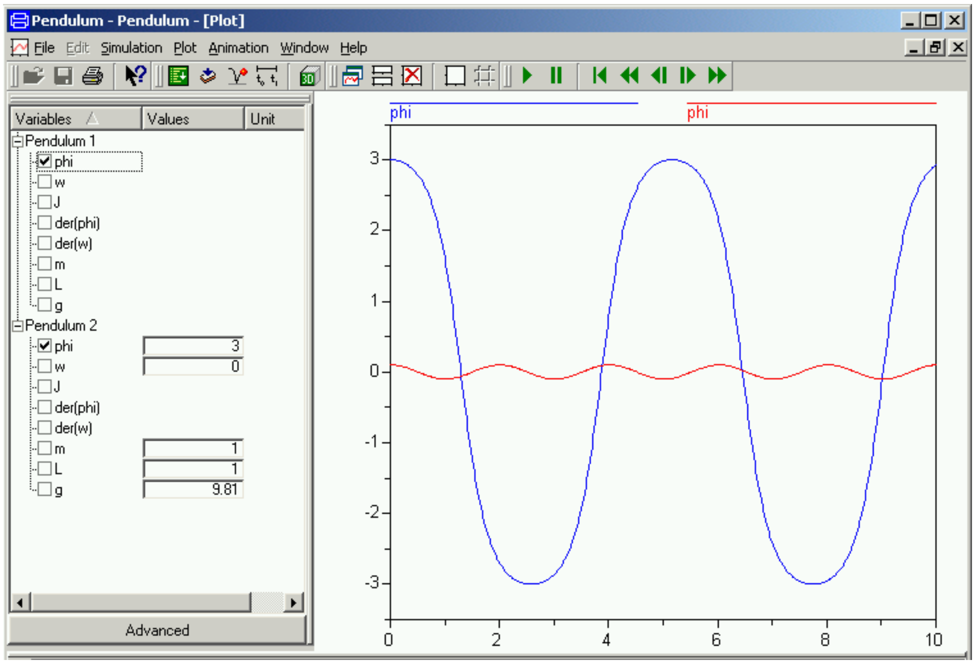
When the simulation is finished, the component browser displays variables to plot. Click in the square box in front of phi to get the angle plotted as shown above.

Let us study a swing pendulum with larger amplitude and let it start in almost the top position with phi = 3. It is easy to change initial conditions. Just enter 3 in the value box for phi. Click on the Simulate tool button.

Pendulum angle when starting in almost the top position.



The results of previous simulations are available as the experiment Pendulum 1 and we can open it up and have phi of the two runs plotted in the same diagram.

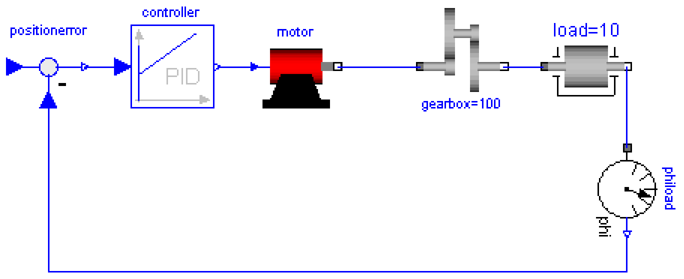


Values of parameters are changed in a similar way. To change the length of the pendulum, just enter a new value for L.

Using the Modelica Standard Library

In this example, we will show how a model is built up using components from the Modelica Standard Library. The task is to model a motor drive with an electric DC motor, gearbox, load, and controller.

Motor drive built with standard components.



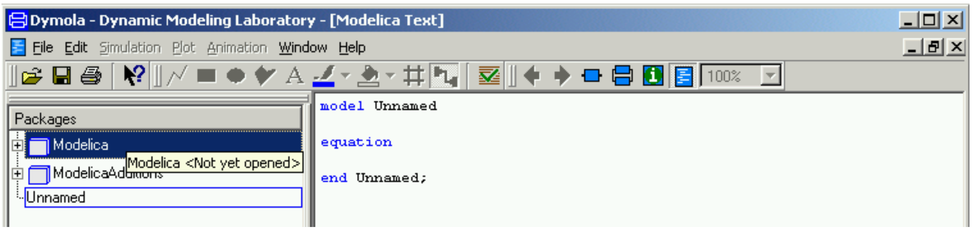
As when building a real system, there are several approaches. One extreme approach is to build the system from scratch. However, it is often a difficult and time-consuming task. Another approach is to investigate if the system already is available on the market or if there is some product that easily can be adapted or modified. If not, build the system from components available when possible and develop only when necessary.

The idea of object-oriented modeling is to support easy and flexible reuse of model knowledge. Modelica has been designed to support reuse of model components as parts in different models and to support easy adaptation of model components to make them describe similar physical components. The design of Modelica has also been accompanied by the development of model libraries.

The Modelica Standard Library

We will now have a look at the Modelica Standard Library to see what is available and how we access the model components and their documentation. To open the library, double click on Modelica in the Package browser.

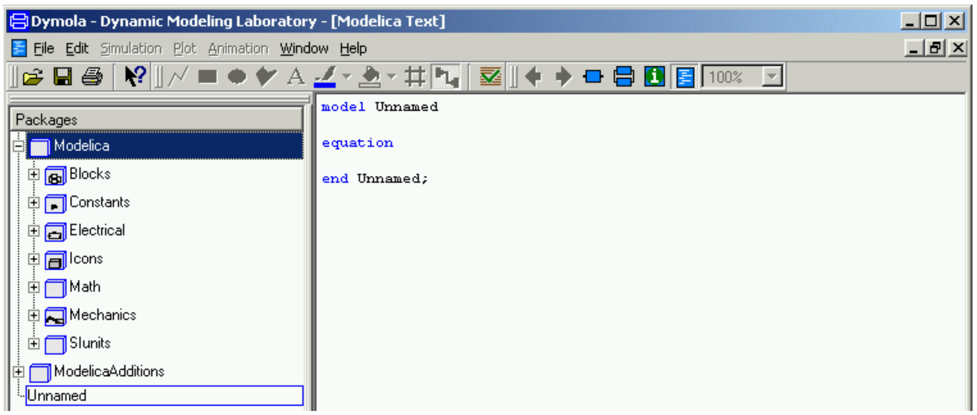
Opening the Modelica Standard Library.


Dymola - Dynamic Modeling Laboratory - [Modelica Text]
File Edit Simulation Plot Animation Window Help
100%
Packages
Modelica
ModelicaAdditions
Modelica <Not yet opened>
Unnamed

```
model Unnamed

equation

end Unnamed;
```

Dymola reads in the library. The Modelica Standard Library is hierarchically structured into sub-libraries.

The sub-libraries of the Modelica Standard Library.


Dymola - Dynamic Modeling Laboratory - [Modelica Text]
File Edit Simulation Plot Animation Window Help
100%
Packages
Modelica
Blocks
Constants
Electrical
Icons
Math
Mechanics
SIunits
ModelicaAdditions
Unnamed

```
model Unnamed

equation

end Unnamed;
```

As shown by the package browser, the Modelica Standard Library includes

1. **Blocks** with continuous and discrete input/output blocks such as transfer functions, filters, and sources.
2. **Constants** provides constants from mathematics, machine dependent constants and constants from nature.
3. **Electrical** provides electric and electronic components such as resistor, diode, MOS and BJT transistor.
4. **Math** gives access to mathematical functions such as sin, cos and log.
5. **Mechanics** includes a one-dimensional translational and rotational components such as inertia, gearbox, planetary gear, bearing friction and clutch.
6. **SIunits** with about 450 type definitions with units, such as Angle Voltage and Inertia.

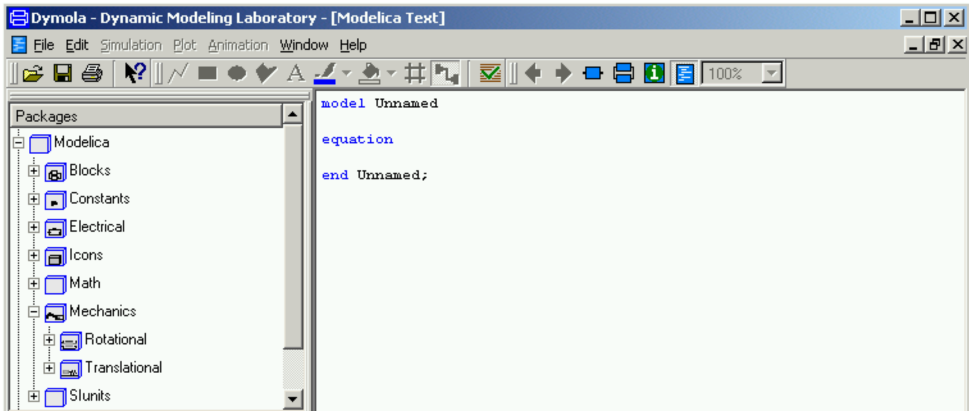
To get documentation for the entire Modelica Standard Library, place the cursor on Modelica, press the right mouse button and select Info. Your web browser is directed to an html file containing documentation for Modelica. This documentation has been generated automatically from the Modelica description of the library. There is basic information such as the content of the library, conventions and conditions for use.

Dymola comes also with other free model libraries. To have a list select Files / Libraries.

The package menu gives direct access to the sub-libraries. We will need components from various sub-libraries. We will need rotational mechanical components as well as electrical components for the motor.

To open the Modelica.Mechanics, double click on Mechanics.

Opening Modelica.Mechanics.

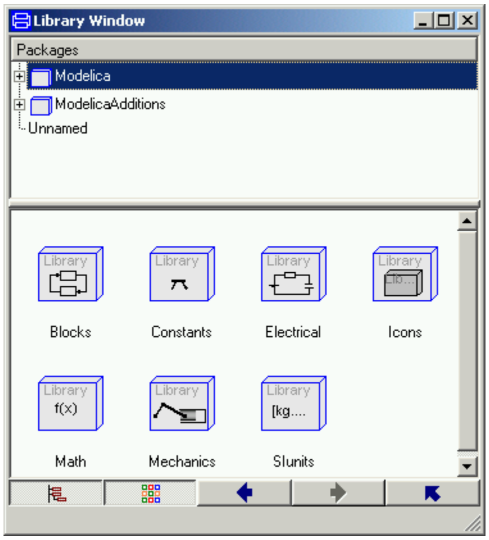

Dymola - Dynamic Modeling Laboratory - [Modelica Text]
File Edit Simulation Plot Animation Window Help
100%
Packages
Modelica
Blocks
Constants
Electrical
Icons
Math
Mechanics
Rotational
Translational
SIunits

```
model Unnamed

equation

end Unnamed;
```

To get documentation on Modelica.Mechanics, as previously demonstrated, place the cursor on Mechanics, press the right mouse button and select Info.

Besides using the package browser of the Dymola window, it is also possible to open a library browser. It is done by either selecting Window/New Library Browser or using the right mouse button and selecting Show Library Window.

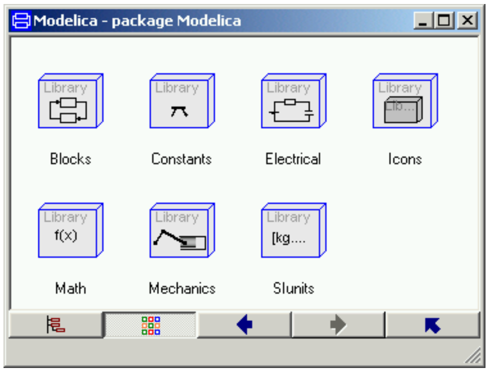
A library window displaying the Modelica Standard Library.



A Library window includes a package browser, where the components of the selected sublibrary are displayed in a special window.

It is possible to close the package browser by toggling the button to the bottom left.

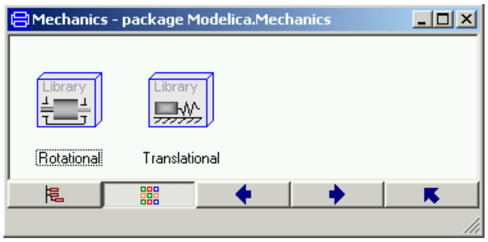
A library window displaying the components of the Modelica Standard Library.



By using the right button at the bottom it is possible to go up in the package hierarchy and by double clicking on the icons in the window it is possible to go down in the hierarchy. The left and right arrow buttons allow going back and forth as in an ordinary web browser.

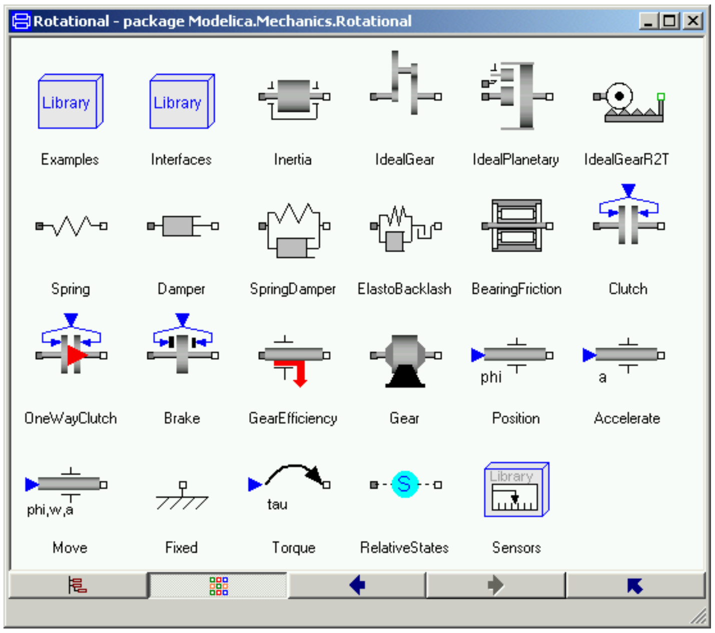
Open Modelica.Mechanics.Rotational in the library window by first double clicking on the icon for Mechanics

Modelica.Mechanics.



and then on the icon for Rotational. It contains components for rotating elements and gearboxes, which are useful for our modeling of the electrical motor drive.

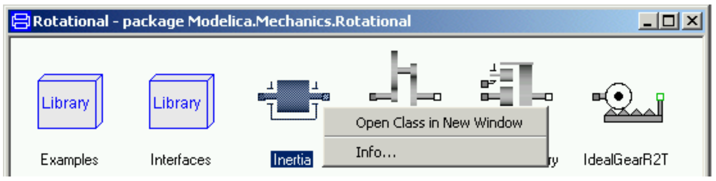
The rotational mechanics library window.



The Info for Modelica.Mechanics.Rotational contains important information on design principles behind the library and a list of components. A quick scan of the list indicates that the model Inertia may be of interest for us.

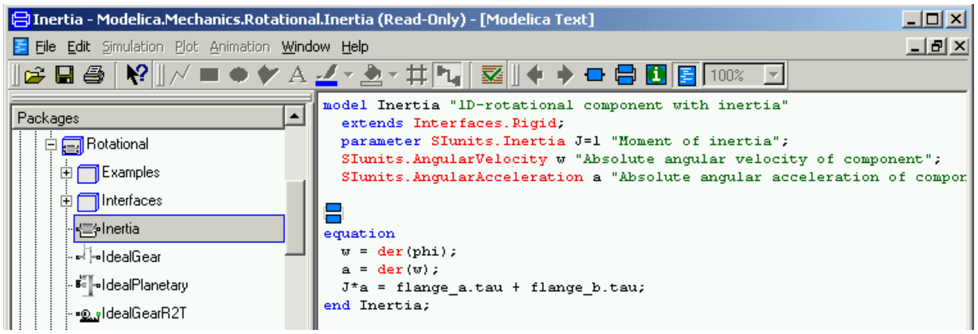
Select Inertia and press the right mouse button for a context menu. Select Info to get documentation for the model.

The context menu for a component.



To get a model window for Inertia, select its icon in the library window and once again press the right mouse button, select Show in New Window and a window for the model Inertia is created.

Mathematical definition of a rotational inertia.


Inertia - Modelica.Mechanics.Rotational.Inertia (Read-Only) - [Modelica Text]
File Edit Simulation Plot Animation Window Help
100%
Packages
Rotational
Examples
Interfaces
Inertia
IdealGear
IdealPlanetary
IdealGearR2T

```
model Inertia "1D-rotational component with inertia"
  extends Interfaces.Rigid;
  parameter SIunits.Inertia J=1 "Moment of inertia";
  SIunits.AngularVelocity w "Absolute angular velocity of component";
  SIunits.AngularAcceleration a "Absolute angular acceleration of compor

equation
  w = der(phi);
  a = der(w);
  J*a = flange_a.tau + flange_b.tau;
end Inertia;
```

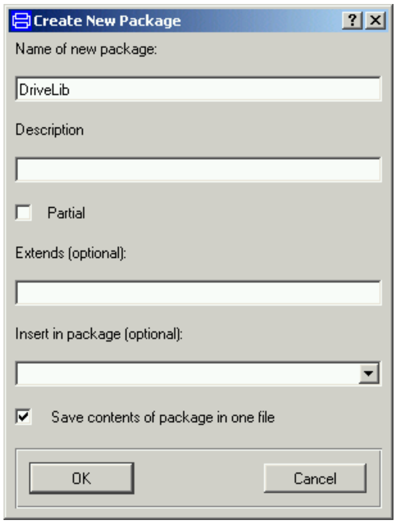
Switch to the Modelica Text representation, where you find Euler's equation as the last equation.

After this introduction of how to access model components and documentation of a library, we will continue by actually building a model for an electric DC motor. This task will give us more experience.

Creating a library for components

It is a good idea to insert developed components into a library. It is a good way to keep track of components and it supports also the drag and drop feature when you will use the models as components. Let us collect all developed components in a library called DriveLib. To create it, go to the Dymola window, and select File/New/Package and a dialog is opened.

Creating a new Modelica package.


Create New Package

Name of new package: DriveLib

Description:

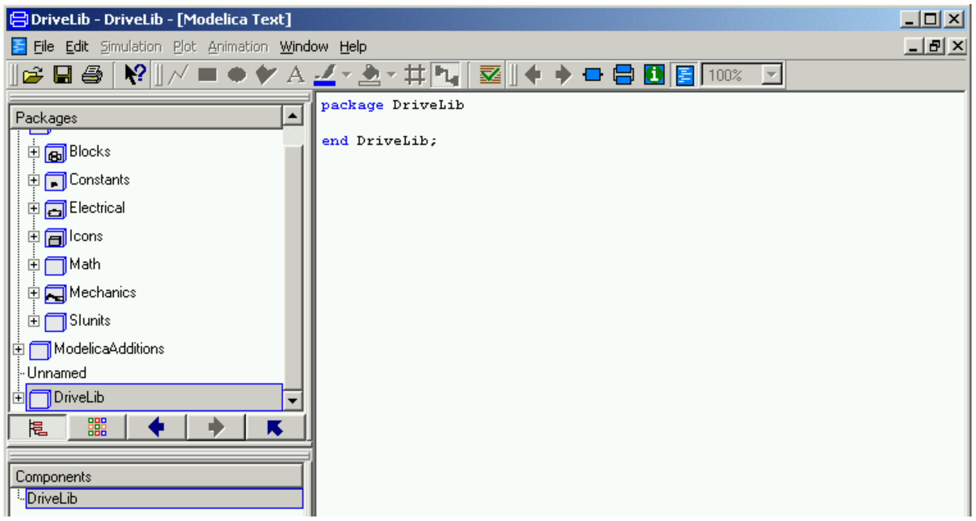
[ ] Partial

Extends (optional):

Insert in package (optional):

[✓] Save contents of package in one file

Enter DriveLib as the new name of the package and click OK.

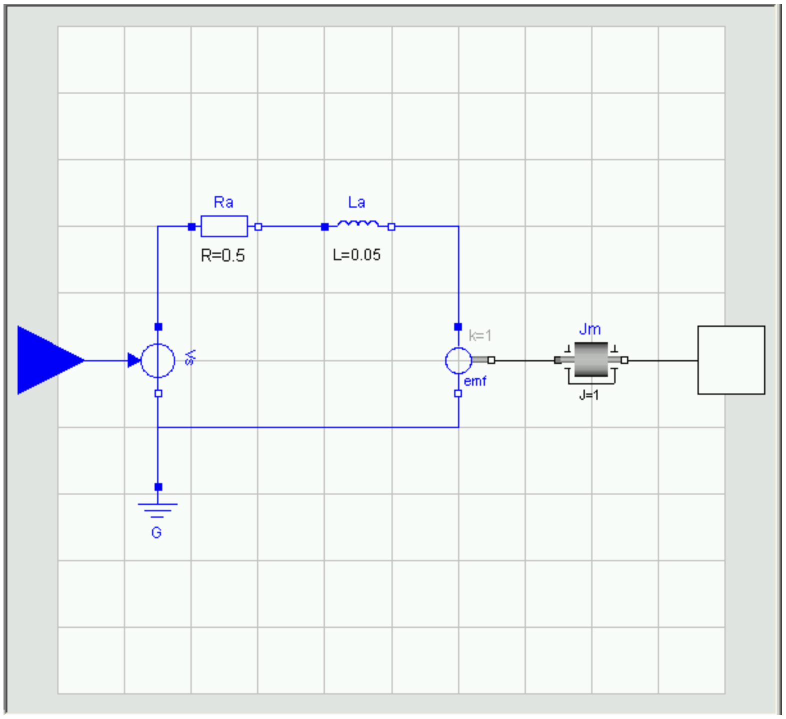


```
package DriveLib

end DriveLib;
```

A package DriveLib is created and made visible in the package browser. Select Modelica text to get the Modelica representation, which at this stage just specifies a package with no contents.

Creating a model for an electric DC motor

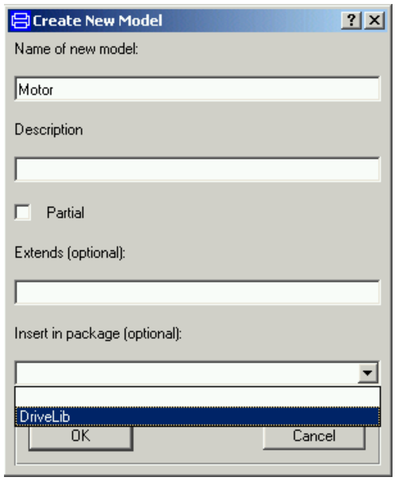
An electrical DC motor.



A model of the complexity indicated above will be developed for the electric DC motor. For simplicity the voltage supply is included in the motor model. The model includes an ideal controlled voltage source. The electric part of the motor model includes armature resistance and armature inductance. The electromotive force, emf, transforms electrical energy into rotational mechanical energy. The mechanical part includes the mechanical inertia of the motor.

Let us start building the motor model. Select in the Dymola window File/New/Model. Enter Motor as name of the new model. To have the Motor model being a part of DriveLib, we need to enter DriveLib for Insert in package. This can be done in several ways. Dymola provides alternatives to be selected from and DriveLib is an available alternative. The are no other alternative because all other open packages are such as Modelica are write protected. It is also possible to use the drag and drop feature and drag DriveLib into the slot. In the package browser, put the cursor on DriveLib and press the left mouse button. While keeping it pressed, drag the cursor to the slot for Insert in package, release the button and the text DriveLib will appear in the slot.

Inserting Motor in DriveLib.



Name of new model: Motor

Description

Partial

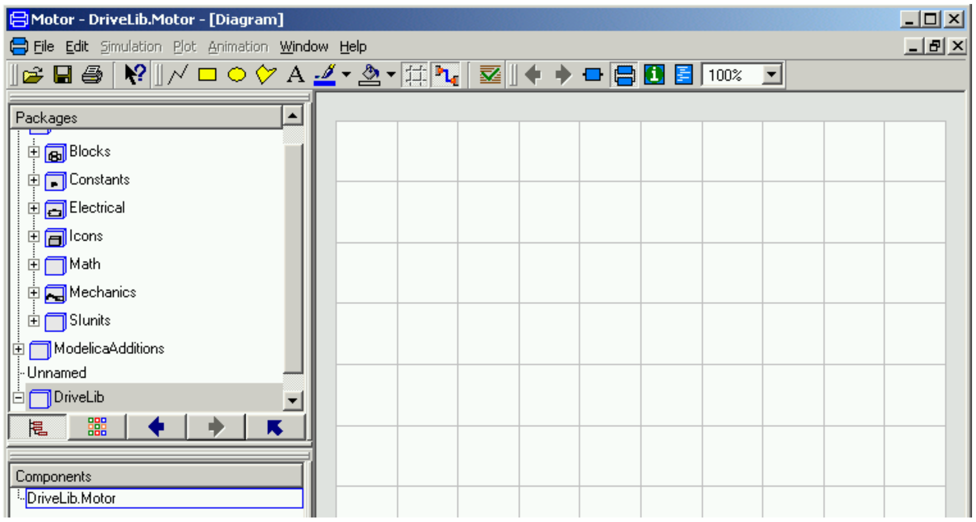
Extends (optional):

Insert in package (optional): DriveLib

Click on OK.

Opening DriveLib in the package browser shows that it has a component Motor as desired.

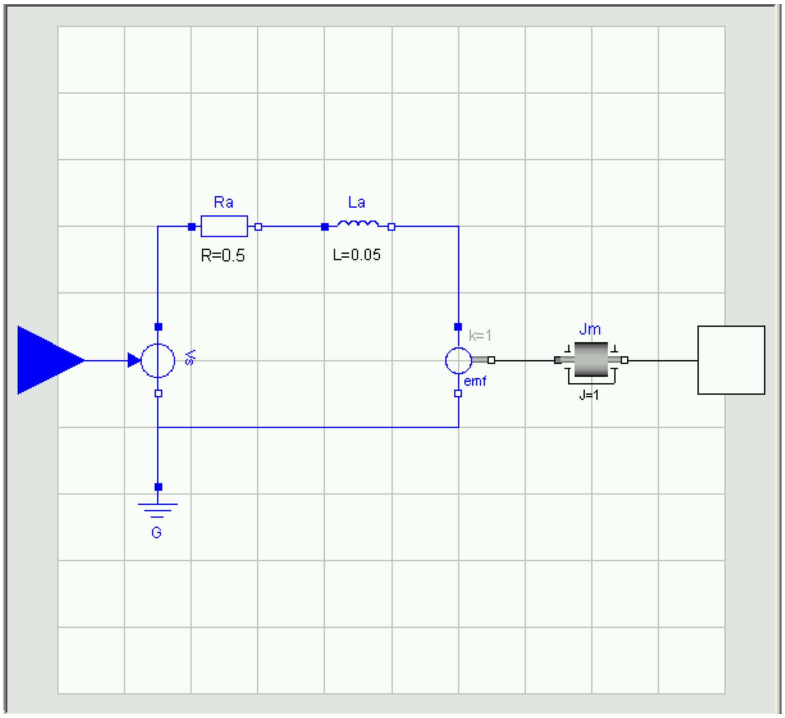
An empty Motor model.



The model window now contains an empty Motor model. The diagram window has a gray frame and grid to indicate that the component is not write protected. It is possible to toggle the grid using the toolbar button.

We will now start building the motor model. To make it easier to follow the instructions, the result is displayed below:

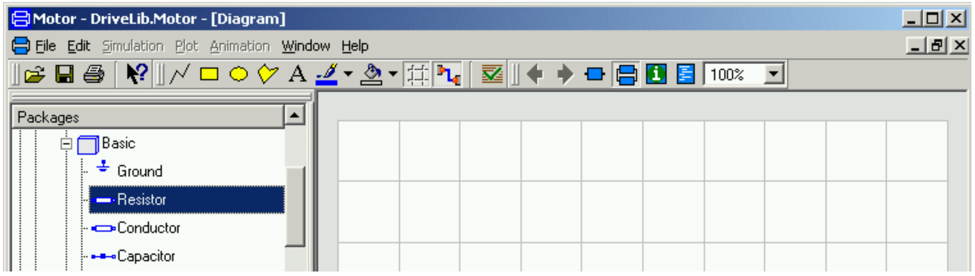
The finished motor model with all components.



We need a model component for a resistor. It can be found in Modelica.Electrical.Analog.Basic. The basic approach is to use drag and drop. You can drag and drop from the package browser or from a library window.

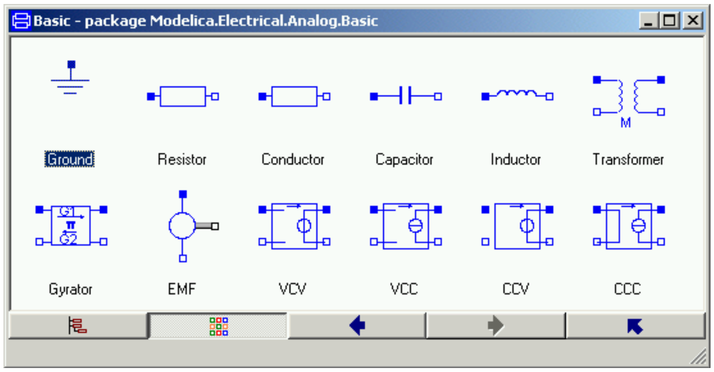
To drag from package browser, open in turn Modelica, Electrical, Analog and Basic. Note that title of the Dymola window is still DriveLib.Motor and also the component browser has DriveLib.Motor as top level to indicate that we are editing the motor model.

About to drag a resistor from the package browser.



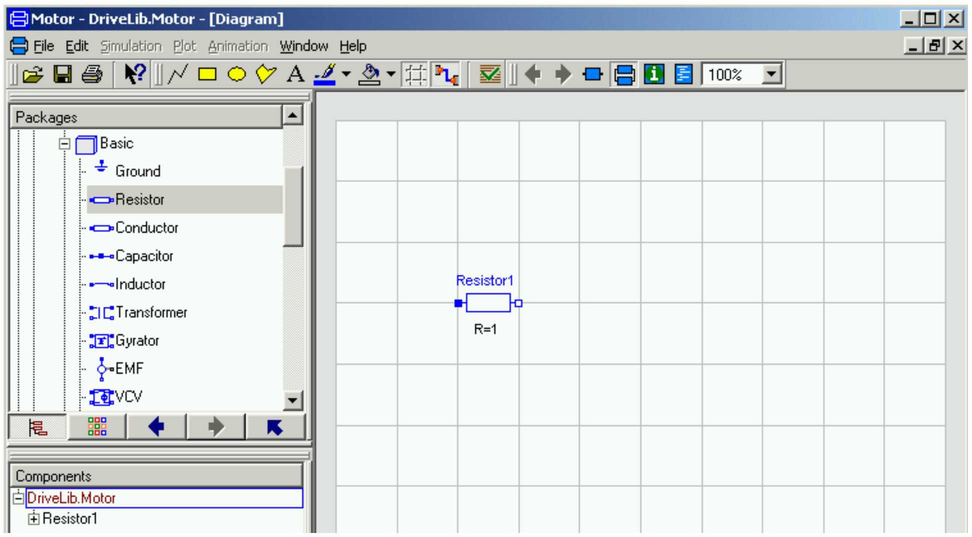
You can also drag from a library window. If you do not have a library window for it then open one. Go to Modelica.Electrical.Analog.Basic.

The basic analog components library window.



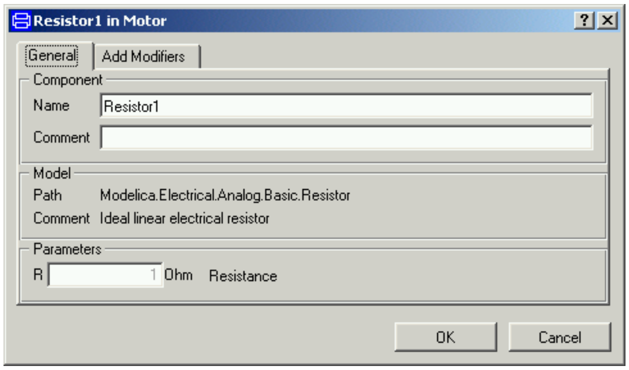
Drag a resistor from Basic to the Motor window and place it as shown above. The component browser displays that Motor has a component Resistor1.

Inserting a resistor component.



When inserting a component is given an automatically generated name. The name may be changed in the parameter window. Double click on the component, to get its parameter window. The parameter window can also be reached by pointing on the component and pressing the right mouse button and select Parameters.

The parameter window of a resistor with default settings.



Component

Name: Resistor1

Comment:

Model

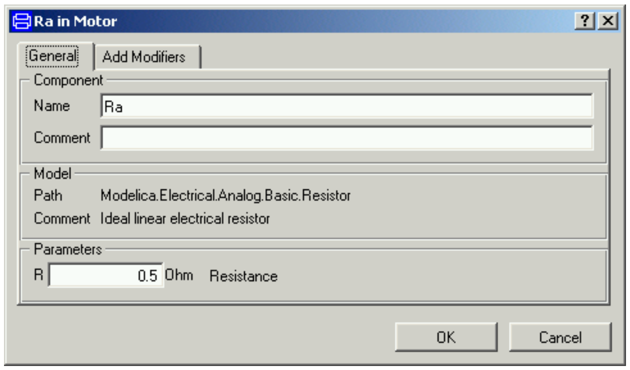
Path: Modelica.Electrical.Analog.Basic.Resistor

Comment: Ideal linear electrical resistor

Parameters

R: 1 Ohm Resistance

Change the component name to Ra. The parameter window allows setting of parameter values. To set the resistance parameter, R, select the value field of parameter R and input 0.5.


The parameter window of a resistor with new settings.

Ra in Motor

General | Add Modifiers

Component

Name: Ra

Comment:

Model

Path: Modelica.Electrical.Analog.Basic.Resistor

Comment: Ideal linear electrical resistor

Parameters

R: 0.5 Ohm Resistance

Click OK.

Similarly drag an inductor to the Motor window. Name it La and set the inductance, L, to 0.05.

Drag a ground component into the motor model. Name it G. The ground component is as important as in real electrical circuits. It defines the electrical potential to be zero at its connection point. As in the real world, never forget to ground an electrical circuit.

Drag an electromotive force, EMF, component into the motor model. Name it emf.

A voltage source is to be found in Modelica.Electrical.Analog.Sources. Use a library window or package browser to locate it. Select SignalVoltage and drag it to the model window of Motor. Name it Vs. Let Vs be selected and use Edit/Rotate 90 to turn the signal input, Vs.inPort, from a top position to a left position. SignalVoltage produces between its two electrical pins, p and n, a voltage difference, p.v-n.v, that is equal to the signal input. Get the Info for SignalVoltage displayed in your web browser (point on the icon and use the right mouse button menu). The documentation shows that SignalVoltage extends the model Modelica.Electrical.Analog.Interfaces.OnePort. Click on that link. The documentation shows that the pin p is a filled blue square. To get the proper sign we would like to have pin p in the top position. To flip the component, use Edit/Flip Vertical.

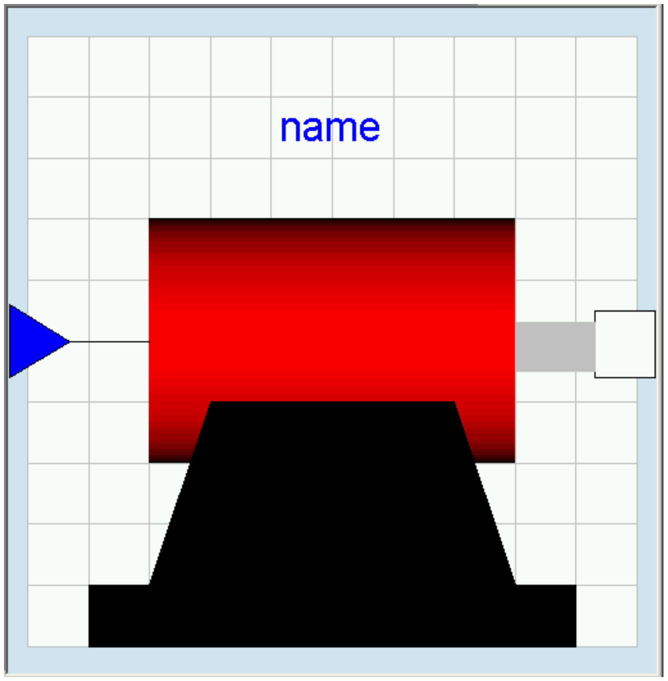
A rotating inertia component is to be found in Modelica.Mechanics.Rotational. Drag and drop such an inertia component. Name it Jm and set the inertia parameter, J, to 0.001.

Now all model components are in place. Components are connected by drawing lines between connectors. To connect the resistor to the inductor, point at the right connector of the resistor (the small white square) and drag to the left connector of the inductor. The resistor and the inductor are now connected and the graphical result is a line between them. When connecting the voltage source and the resistor, break the line by clicking at an intermediate point. There is a possibility to obtain automatic Manhattanize of connections. Select the connection, Edit/Manhattanize. Draw all connections. Note that we could have drawn a connection between two components as soon as we have the components and we have not to wait until all model components are in place.

Finally, we have to introduce a signal connector for the voltage control and a flange connector corresponding to the shaft of the motor so the motor can be connected to an environment. We would like to place the icon of the connectors at the border of the grid of the drawing pane, because the icon of a model component also includes the connectors. The connector inPort must be compatible with the connector of Vs.inPort. There is a simple automatic to get a connector inPort that is a clone of Vs.inPort. Start drawing a connection from Vs.inPort and go to the left until you reach the border of the grid. Then you double click and select Create Connector from the menu popped up. The connector flange_b is created in a similar way. If you would like to adjust the position of a connector it is easy to get into connect mode. This can be avoided by toggling the toolbar button Connect Mode (to the right of the Toggle Grid button).

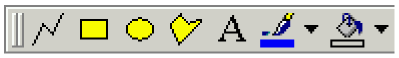
Click on the toolbar button (to left of the button for activating the Diagram view) to find that you also there can see icons for the connectors. Let us draw an icon for the motor model. One design is shown below.

The icon of the electrical DC motor.



To draw it, we will use the toolbar for editing graphics.

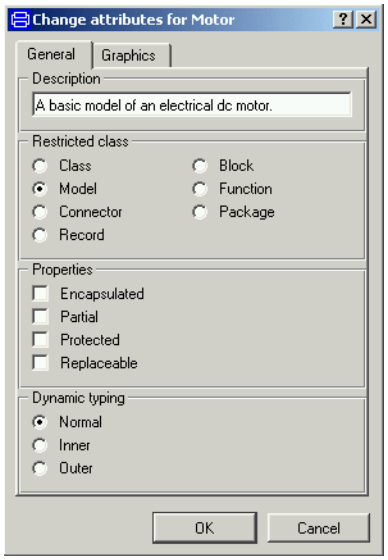
Toolbar for editing.



Start by drawing the big red cylinder (shaded rectangle); Click the Draw rectangle button (yellow rectangle) and lay out a rectangle. Let it be selected. Click on the arrow to the right of the Fill Color button. Select Colors... and then select a red color. Click OK. To select the gradient, click once again on the arrow to the right of the Fill Color button. Select Gradient/ Horizontal. Draw the rest of the parts using draw rectangle or draw closed polygon in an analogous way. To enter the text, click the Text button (the button labeled A) and lay out a rectangle that is as long as the cylinder and two grid squares high. In the window prompt for the string enter %name. The %-sign has the magic function that when the model is used, the actual component name will be displayed.

We have now edited the icon and the diagram. It is also important to document the model. When creating the model, the dialog has a slot Description. It is possible to edit this afterwards. Select Edit/Attributes to open the dialogue.

Model attributes.



Description: A basic model of an electrical dc motor.

Restricted class: ( ) Class [✓ Model] ( ) Connector ( ) Record ( ) Block ( ) Function ( ) Package

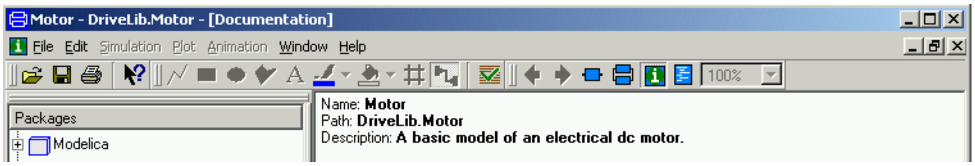
Properties: ☐ Encapsulated ☐ Partial ☐ Protected ☐ Replaceable

Dynamic typing: [✓ Normal] ( ) Inner ( ) Outer

Enter a description and click OK.

A longer documentation can be provided in the following way. Click on the toolbar button for Documentation (the button between the Diagram button and the Modelica Text button).

Documentation View.



Name: Motor
Path: DriveLib.Motor
Description: A basic model of an electrical dc motor.

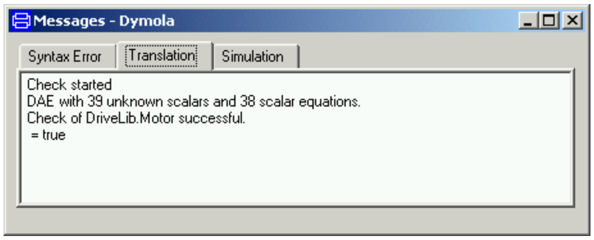
To enter a description, put the cursor in the window. Click right mouse button and select Edit Source. The description is given as html code. When done, click right mouse button and deactivate Edit Source.

We have now created the model. Save it.

Testing the model

It is possible to check the model for syntactic and semantic errors. Select Edit/Check. Hopefully your model will pass the check and you will get the following message:

Checking the model.

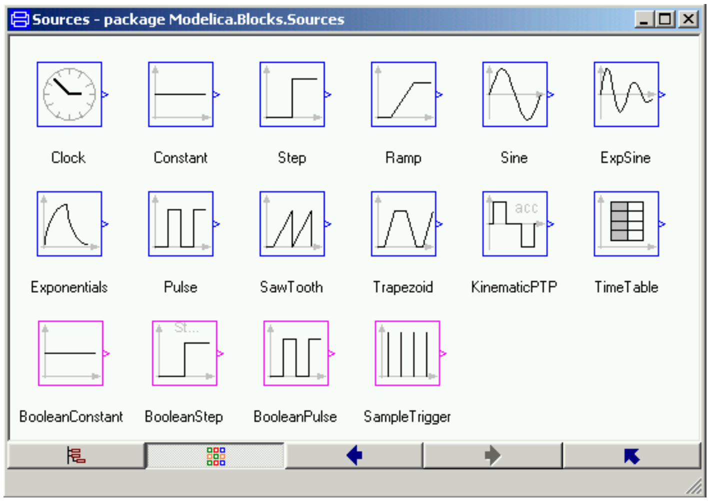


```
Check started
DAE with 39 unknown scalars and 38 scalar equations.
Check of DriveLib.Motor successful.
 = true
```

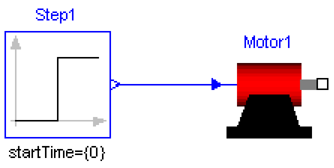
The connector inPort defines the voltage reference and we have not defined that. This freedom explains why there is one equation too few.

It is important to test all models carefully, because that eliminates tedious debugging later on. Let us connect the motor to a voltage source. Create a model called TestMotor (select File/New/Model) and insert it into DriveLib. It is good practice to keep test models. Use the package browser to drag over a Motor component from DriveLib to TestMotor. We need a source for the signal input to the motor. Signal sources are to be found in Modelica.Blocks.Sources.

Signal sources.



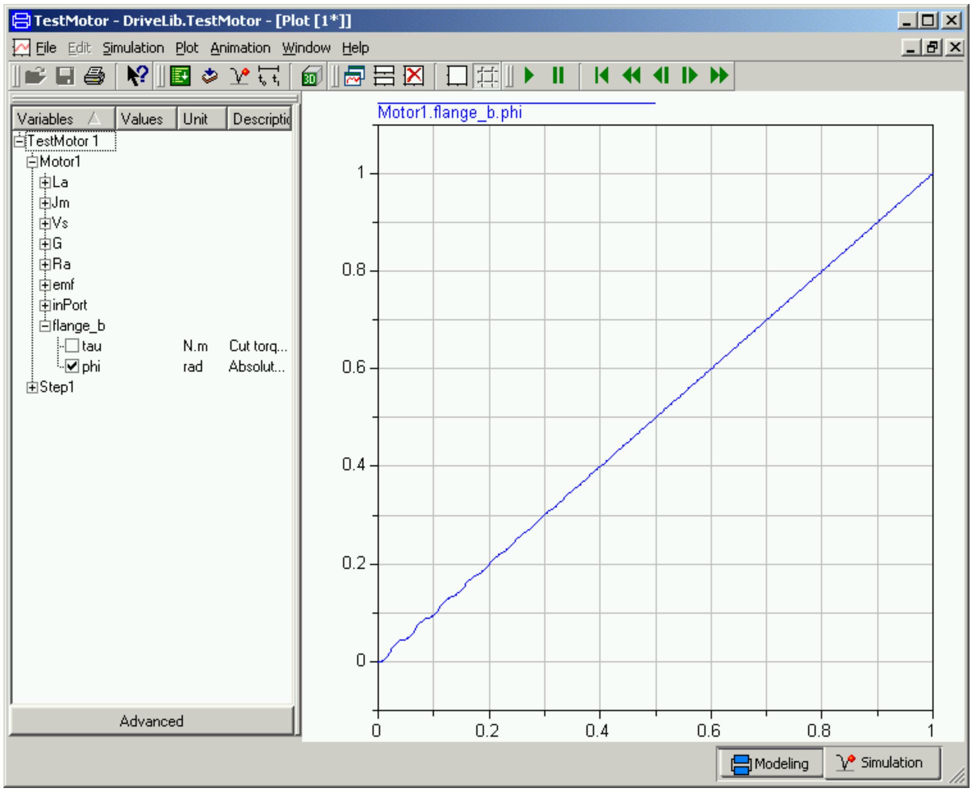
Drag, for example, over Step to the model window and connect it to the motor.



Now it is time to simulate. Click on the tab for Simulate. Click on the toolbar button Simulate to start a simulation.

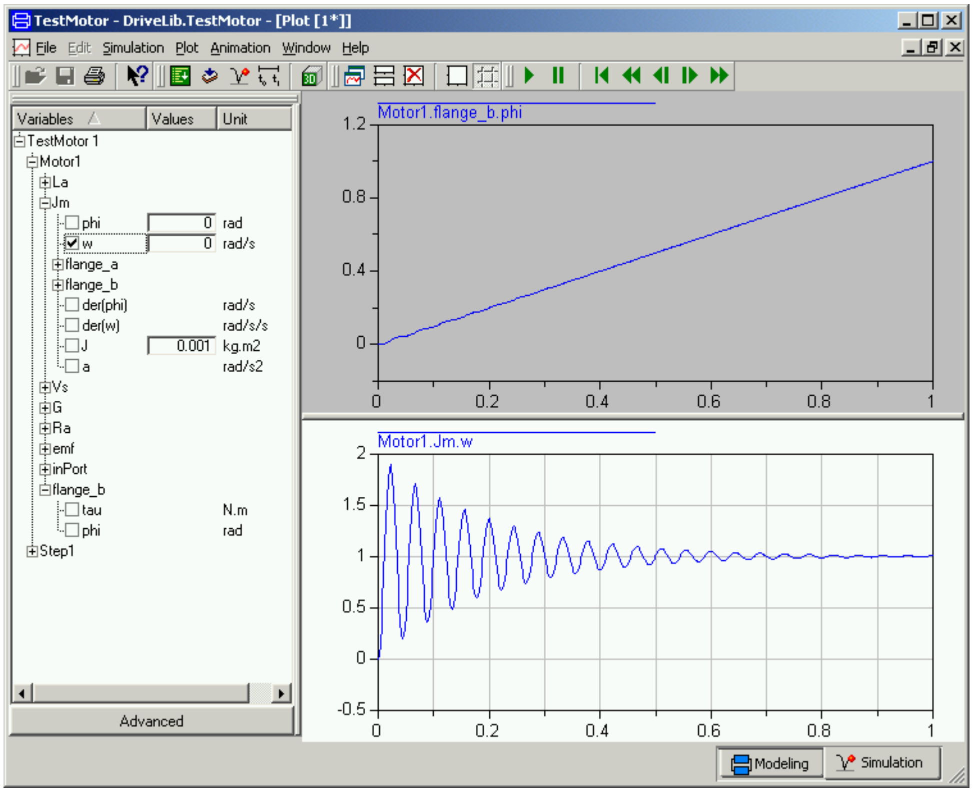
To inspect the result, we will first look at the angular position of the motor, Motor1.flange_b.phi. Open Motor in the plot selector by clicking on the + sign. Open the flange_b and tick phi.

Angular position.



First, we may establish that a positive input signal, makes angular position increase. The plot looks almost like a straight line. However, there are some wriggles in the beginning. Zoom in; use the mouse to stretch a rectangle over that portion of the curve you would like to see. We may also plot the angular velocity Motor1.Jm.w: There is an oscillation which dies out and the velocity becomes constant. There is much to be done to validate the model. However, model validation is out of the scope for this introduction to Dymola.

It is possible to show several curves in the same diagram. It is just to tick the variables to be plotted. A curve is erased by ticking once more. The toolbar button Erase Curves (white rectangle) erases all curves in the active diagram. It also possible to have several diagrams. To get a new diagram, select Plot/New Diagram or click on the toolbar button. The new diagram becomes active. Tick Motor1.Jm.w and the result shown below is obtained. Selecting a diagram makes it active. Selecting Plot/Delete Diagram removes the diagram.



Variables	Values	Unit
TestMotor 1		
Motor1		
La		
Jm		
phi	0	rad
✓ w	0	rad/s
flange_a		
flange_b		
der(phi)		rad/s
der(w)		rad/s/s
J	0.001	kg.m2
a		rad/s2
Vs		
G		
Ra		
emf		
inPort		
flange_b		
tau		N.m
phi		rad
Step1		

Angular velocity.

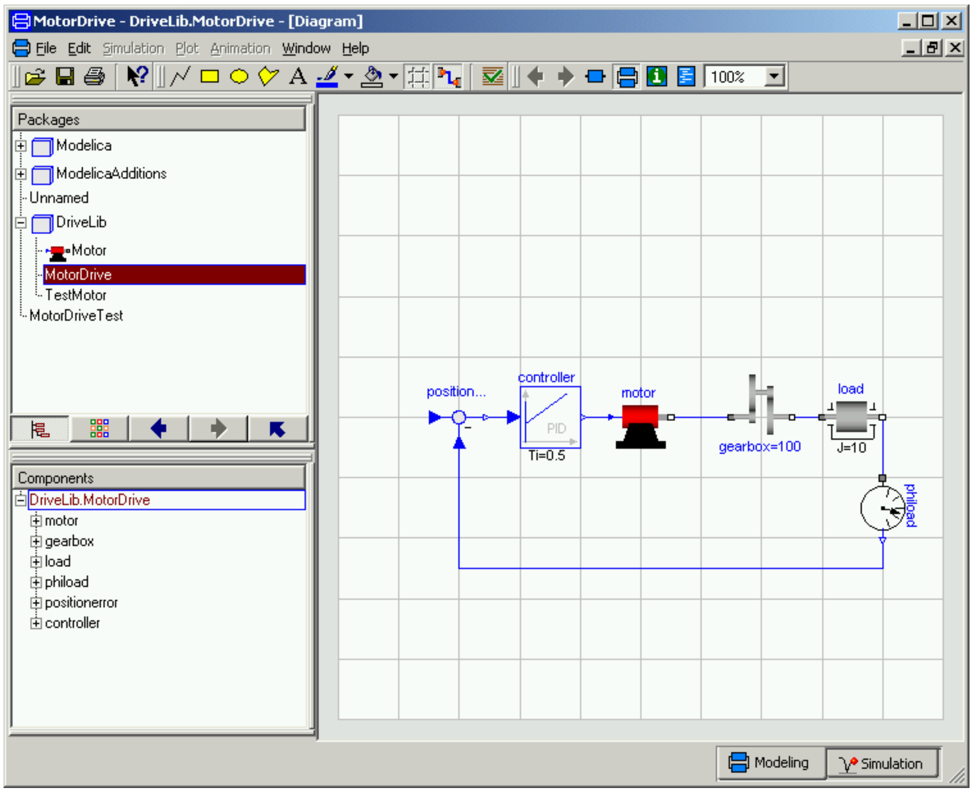
Creating a model for the motor drive

The task was to build a model for a motor drive and it ought now to be a simple task to complete the model. We will just give a few hints.

To generate the position error, you may use the model component Modelica.Blocks.Math.Feedback. For the controller, there is Modelica.Blocks.Continuous.PID.

For the meaning of ratio for the gearbox model, please, consult Info for the model. Set ratio to 100 as indicated. It means that the motor load rotates 100 times slower than the DC motor. The library Modelica.Mechanics.Rotational.Sensors contains a sensor for angles.

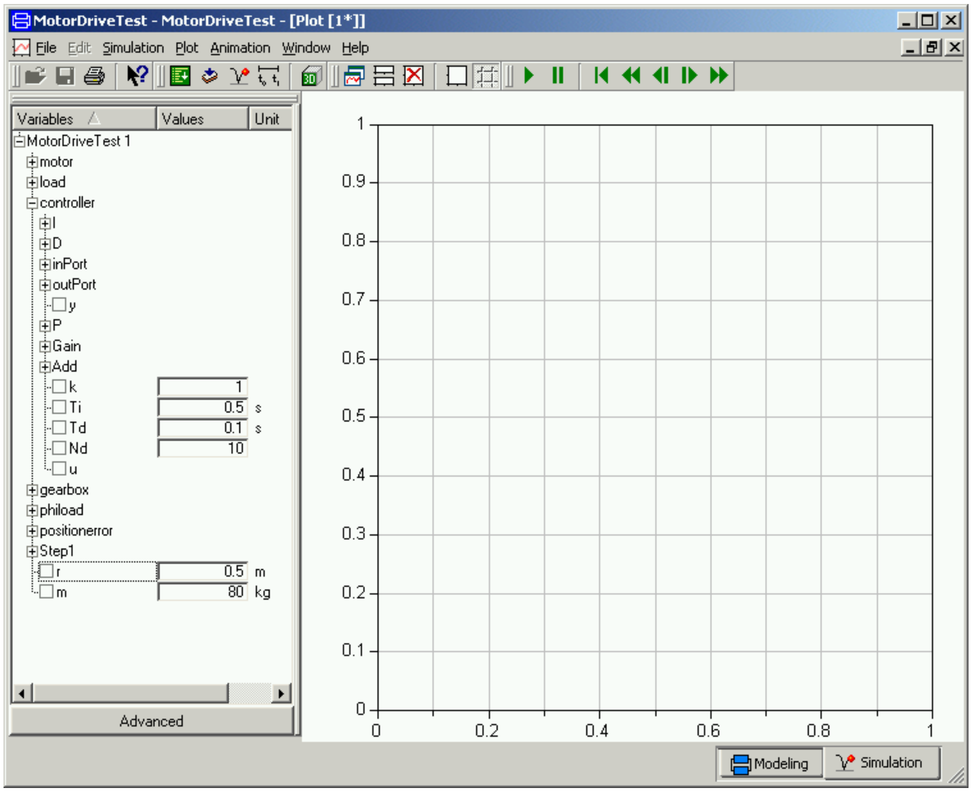
The completed motor drive.



To test the model MotorDrive for normal operation, we need to define a reference for the position. This can be done in different ways. A simple approach is to add a signal source directly to MotorDrive. However, we may like to use MotorDrive also for other purposes. If we would like to use the drive as a component we could add a connector for the reference as we did for the electric DC motor model. However, here we will take the opportunity to show another useful way, namely use of extends. We will develop a new class, say MotorDriveTest, which extends MotorDrive. Go to a model window. Select File/New/Model. Enter MotorDriveTest as the name of the model and drag MotorDrive and drop it in the slot for extends enter MotorDrive. Insert it into DriveLib; select DriveLib for the slot Insert in package. Click OK. The result is a model window, where the diagram layer looks exactly like that of MotorDrive. However, the components cannot be edited. Try to move or delete a component. It has no effect. Drag over a component Step from Modelica.Blocks.Sources and connect it. Save the model.

A model can be used for different tasks. One is tuning the controller manually. Click on the tab for Simulate. Translate the model MotorDriveTest. The PID controller has four parameters: k, Ti, Td and Nd.

Tuning the controller.


MotorDriveTest - MotorDriveTest - [Plot [1*]]
File Edit Simulation Plot Animation Window Help
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
Advanced
Modeling
Simulation

Variables	Values	Unit
MotorDriveTest 1		
motor		
load		
controller		
I		
D		
inPort		
outPort		
y		
P		
Gain		
Add		
k	1	
Ti	0.5	s
Td	0.1	s
Nd	10	
u		
gearbox		
phiload		
positionerror		
Step1		
r	0.5	m
m	80	kg

There are many ways to tune a PID controller. One is to disable the integrator part by setting a large vale for Ti, say 1000 here. Disable also the derivative part by setting Td small, say 0.001. Simulate for 10 seconds. The step response for k = 1 is very slow. Increase it to find out what happens. We leave the problem of tuning the controller to the interested reader.

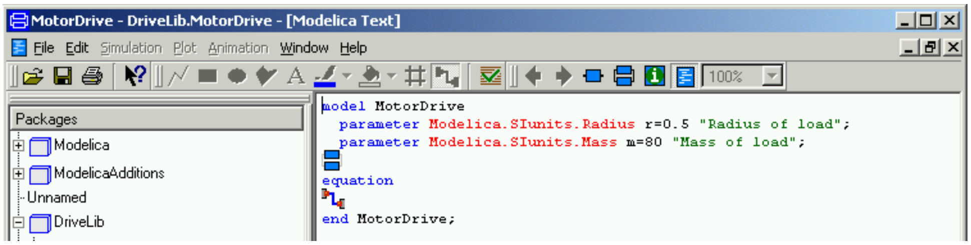
Parameter propagation

Modelica supports parameter propagation, which means that a parameter can be expressed in terms of others. Assume that the load is a homogeneous cylinder rotating long its axis and we would to have its mass, m, and radius, r, as primary parameters on the top level. The inertia is

```
        J = 0.5 * m * r^2
```

We need to declare m and r in MotorDrive. Open MotorDrive. Activate the Modelica Text representation and enter the declarations for the two parameters as shown.

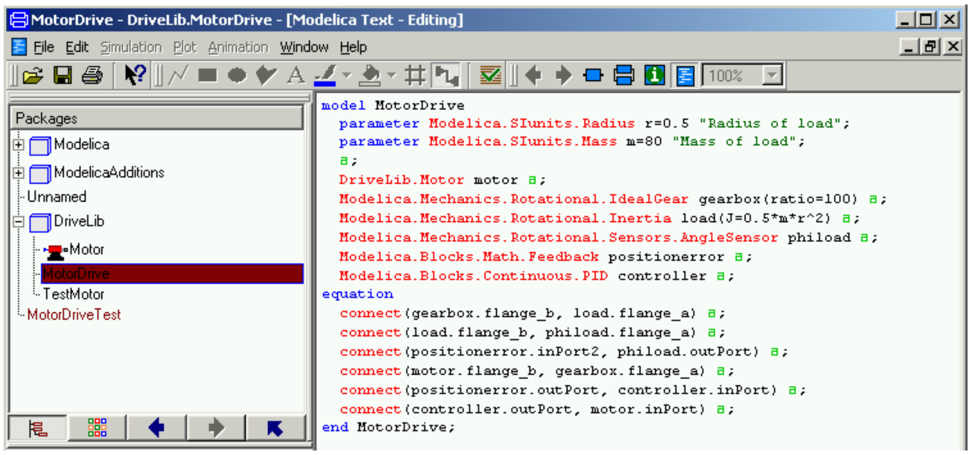
Parameter declarations added to motor drive.


MotorDrive - DriveLib.MotorDrive - [Modelica Text]
File Edit Simulation Plot Animation Window Help
100%
Packages
Modelica
ModelicaAdditions
Unnamed
DriveLib

```
model MotorDrive
  parameter Modelica.SIunits.Radius r=0.5 "Radius of load";
  parameter Modelica.SIunits.Mass m=80 "Mass of load";
equation
end MotorDrive;
```

The parameters are more than real numbers. They are physical quantities. The Modelica standard library provides type declarations for many physical quantities. Open Modelica.SI-units in the package browser and drag over Radius to get the string Modelica.SIunits.

In Modelica Text representation above, the components and connections are indicated by icons. It is possible have them expanded textually. Click the right mouse button and select Expand Components and Connect.

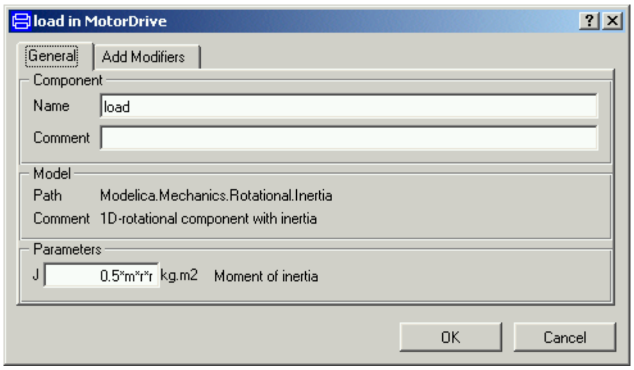

MotorDrive - DriveLib.MotorDrive - [Modelica Text - Editing]
File Edit Simulation Plot Animation Window Help
100%
Packages
Modelica
ModelicaAdditions
Unnamed
DriveLib
Motor
MotorDrive
TestMotor
MotorDriveTest

```
model MotorDrive
  parameter Modelica.SIunits.Radius r=0.5 "Radius of load";
  parameter Modelica.SIunits.Mass m=80 "Mass of load";
  a;
  DriveLib.Motor motor a;
  Modelica.Mechanics.Rotational.IdealGear gearbox(ratio=100) a;
  Modelica.Mechanics.Rotational.Inertia load(J=0.5*m*r^2) a;
  Modelica.Mechanics.Rotational.Sensors.AngleSensor phiload a;
  Modelica.Blocks.Math.Feedback positionerror a;
  Modelica.Blocks.Continuous.PID controller a;
equation
  connect(gearbox.flange_b, load.flange_a) a;
  connect(load.flange_b, phiload.flange_a) a;
  connect(positionerror.inPort2, phiload.outPort) a;
  connect(motor.flange_b, gearbox.flange_a) a;
  connect(positionerror.outPort, controller.inPort) a;
  connect(controller.outPort, motor.inPort) a;
end MotorDrive;
```

It is also possible to expand the annotations such as the graphics for the icon of the model, the positions and sizes of the icons of the components, the path of the connections etc., by clicking the right mouse button and select Expand Annotation. However, we refrain from showing it in this document.

Activate the diagram representation. Double click on the load icon to open the parameter window.

Binding a parameter to an expression.



load in MotorDrive

General | Add Modifiers

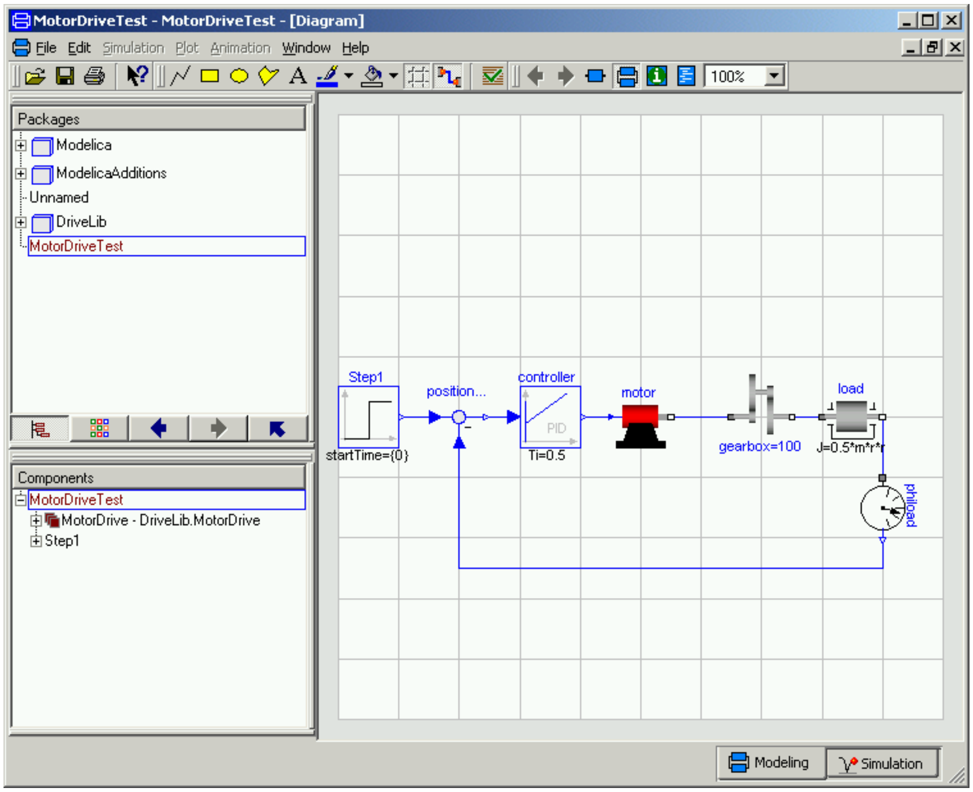
Component
Name: load
Comment:

Model
Path: Modelica.Mechanics.Rotational.Inertia
Comment: 1D-rotational component with inertia

Parameters
J: 0.5*m*r*r kg.m2 Moment of inertia

Click in the value field of J and enter the expression for J. Click OK. The model window now displays the expression for the load inertia.

The component's parameter definition is visible in the model.

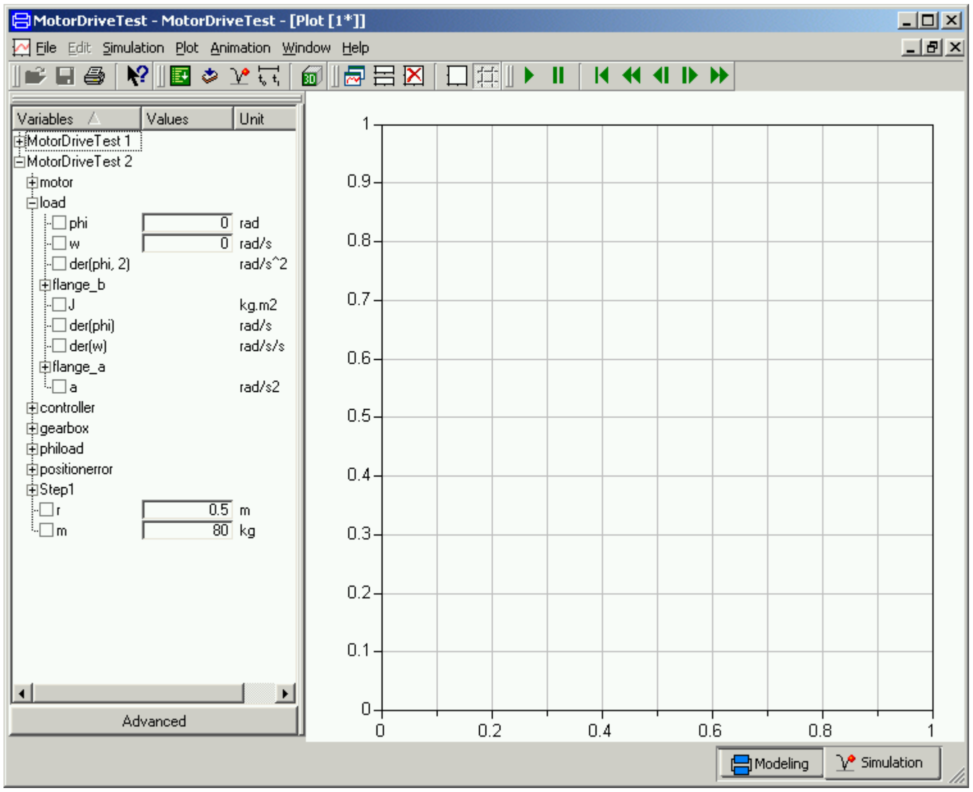


MotorDriveTest - MotorDriveTest - [Diagram]

Switch to Simulation mode.

Translate MotorDriveTest.

A bound parameter cannot be changed interactively.


MotorDriveTest - MotorDriveTest - [Plot [1*]]
File Edit Simulation Plot Animation Window Help
Advanced
Modeling
Simulation

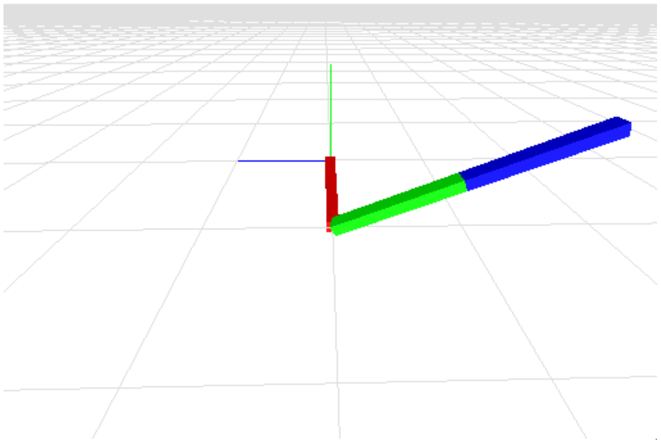
Variables	Values	Unit
MotorDriveTest 1		
MotorDriveTest 2		
motor		
load		
phi	0	rad
w	0	rad/s
der(phi, 2)		rad/s^2
flange_b		
J		kg.m2
der(phi)		rad/s
der(w)		rad/s/s
flange_a		
a		rad/s2
controller		
gearbox		
phiload		
positionerror		
Step1		
r	0.5	m
m	80	kg

The parameters r and m can be set interactively between simulation runs, but not load.J, because it is no longer a free parameter, because there is an expression binding it to r and m.

Building a mechanical model

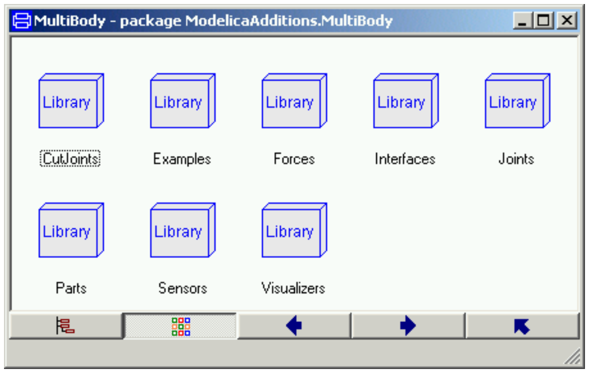
We will now develop a more complex model, a 3D mechanical model of a pendulum called a Furuta pendulum. It has an arm rotating in a horizontal plane with a single or double pendulum attached to it, see below.

The Furuta pendulum.



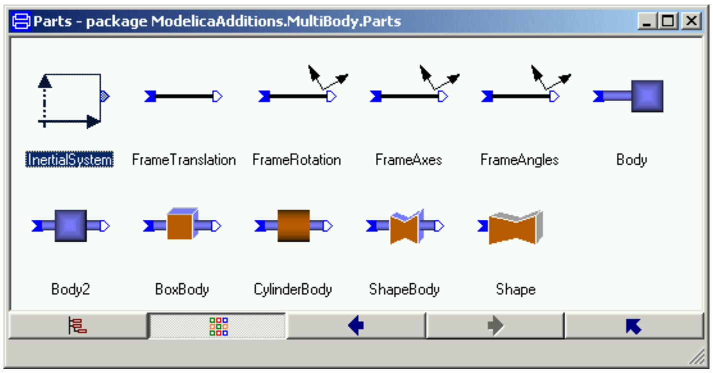
Start Dymola. The package ModelicaAdditions contains a sub-library Multibody with 3D mechanical components such as joints and bodies, which can be used to build a model of the Furuta pendulum. Use the package browser to open ModilicaAdditions and Multibody. You may also create a new library window. Select MultiBody in the package browser and click the right mouse button and select Open Library Window.

The MBS library window.

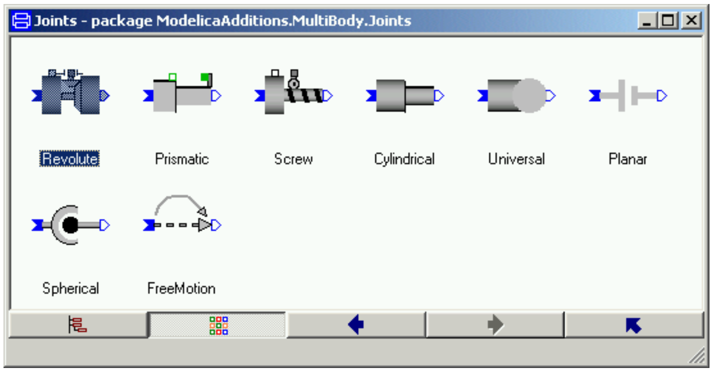


To build the Furuta pendulum, you will need to use the Parts and Joints sub-libraries. Open them by double-clicking.

The MBS parts library window.



The MBS joints library window.



Select File/New/Model and give the name Furuta.

The first step in building an MBS model is to define an inertial system. Drag the Parts.InertialSystem icon onto the Furuta window. The default parameters need not be changed.

We then need a revolute joint. Drag the model Joints.Revolute onto the Furuta window. You can either drag from the library window, but it also convenient to drag from the package browser. Select Edit/Rotate 90. This just changes the appearance of the icon. Double click on the icon. Change the name to R1. The axis of rotation is set as the parameter n. Set n={0, 1, 0} which means a vertical axis of rotation. Click OK. Connect the connector of the inertial frame to the bottom connector of the revolute joint.

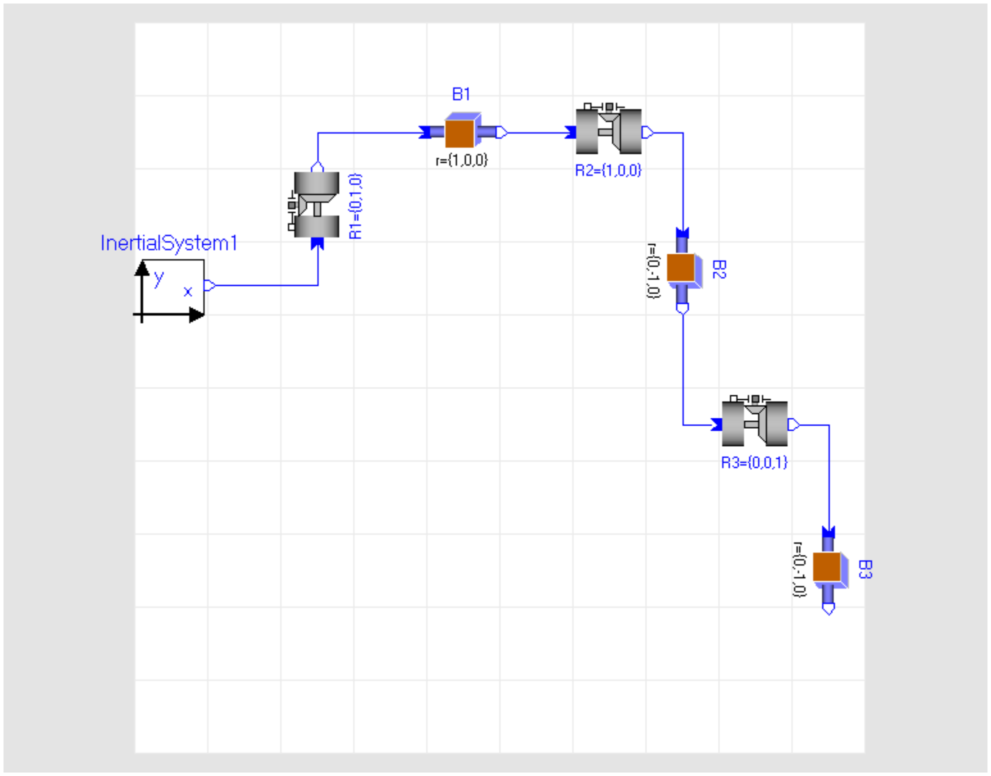
A bar is then connected to the revolute joint. There is one bar which has a the visual appearance of a box during animation, called BoxBody in the Parts library. Drag over a component. Name it B1 and set the vector r between the two ends of the body to {1, 0, 0}. The width and height will be 0.1 by default.

From this bar, we connect another revolute joint, R2, having a horizontal axis of rotation, r={1, 0, 0} and a BoxBody, B2, (rotated –90), with r={0, –1, 0}.

To get a double pendulum, create another similar joint and another BoxBody and connect them. This is accomplished easily by selecting the two already present and choosing Edit/

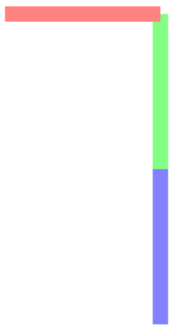
Duplicate. To get nicer animation, you can set different colors for the bars, for example Material = {0, 1, 0, 0.5} and {0, 0, 1, 0.5} respectively for a green bar B2 and a blue B3. You should now have arrived at a model that is similar to the following.

The diagram of the Furuta pendulum.



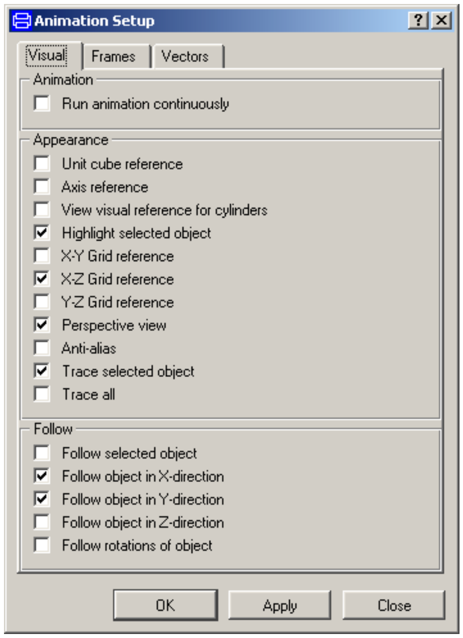
Now it is time to simulate. To enter the Simulation mode, click on the tab at the bottom right. The simulation menu is now activated and new tool bar buttons appear.

When building 3D mechanical models, it is possible to get visual feedback to check that the mechanical system is put together as intended. Click on the toolbar button Visualize. The animation window shows the initial configuration of the system.



To make it easier to interpret the picture, we will introduce a horizontal grid. Select Animation/Setup.

Animation setup.



Animation Setup

Visual | Frames | Vectors

Animation
- [ ] Run animation continuously

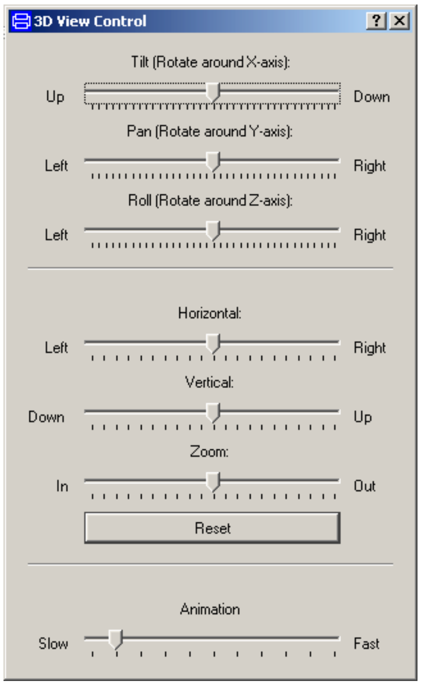
Appearance
- [ ] Unit cube reference
- [ ] Axis reference
- [ ] View visual reference for cylinders
- [✓] Highlight selected object
- [ ] X-Y Grid reference
- [✓] X-Z Grid reference
- [ ] Y-Z Grid reference
- [✓] Perspective view
- [ ] Anti-alias
- [✓] Trace selected object
- [ ] Trace all

Follow
- [ ] Follow selected object
- [✓] Follow object in X-direction
- [✓] Follow object in Y-direction
- [ ] Follow object in Z-direction
- [ ] Follow rotations of object

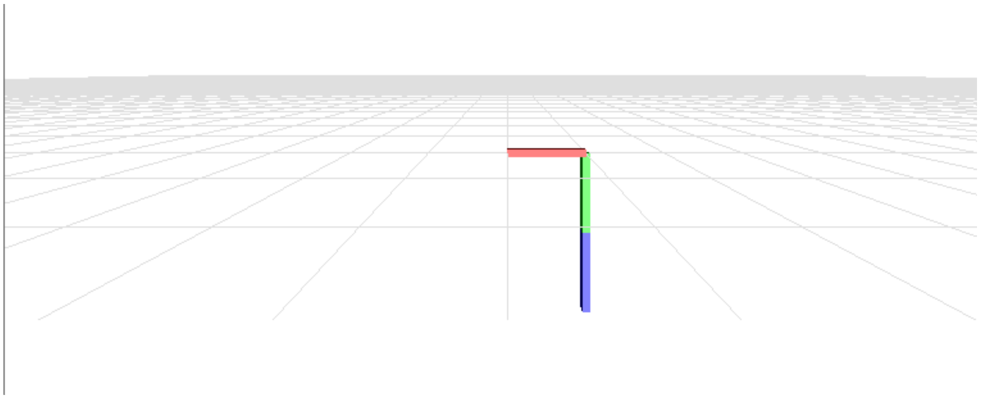
OK | Apply | Close

Tick X-Z reference to get a horizontal grid (the Y-axis is vertical). Tick Perspective view. Click OK. Select Animation/ 3D View Control.

3D View Control.



Move the vertical slider towards down. The animation window will then look as below, giving as a good picture of the pendulum.



Try the other sliders to find out how the works.

Translate the model. In the plot dialogue, open R2 and enter a start value for the angle, q, say 3, and simulate.

View the pendulum in the animation window; you may want to adjust the perspective by using Setup/3D View Control. Change parameters and study the different behavior.

Try to control the pendulum in such a way as to get a stable upright position. (A trick is to use a “mechanical PD controller”, i.e. a spring and damper attached to the tip of the second bar and to a high position.