MapleSim User's Guide

MapleSim User's Guide

Copyright

This document was produced using Maple and DocBook.

Printed in Canada

ISBN 978-1-926902-32-6

Contents

List of Figures

List of Tables

Introduction

MapleSim Overview

MapleSim^{TM} is a modeling environment for creating and simulating complex multidomain physical systems. It allows you to build component diagrams that represent physical systems in a graphical form. Using both symbolic and numeric approaches, MapleSim automatically generates model equations from a component diagram and runs high-fidelity simulations.

Build Complex Multidomain Models

You can use MapleSim to build models that integrate components from various engineering fields into a complete system. MapleSim features a library of over 300 modeling components, including electrical, hydraulic, mechanical, and thermal devices; sensors and sources; and signal blocks. You can also create custom components to suit your modeling and simulation needs.

Advanced Symbolic and Numeric Capabilities

MapleSim uses the advanced symbolic and numeric capabilities of Maple^{TM} to generate the mathematical models that simulate the behavior of a physical system. You can, therefore, apply simplification techniques to equations to create concise and numerically efficient models.

Pre-built Analysis Tools and Templates

MapleSim provides various pre-built templates in the form of Maple worksheets for viewing model equations and performing advanced analysis tasks, such as parameter optimization. To analyze your model and present your simulation results in an interactive format, you can use Maple features such as embedded components, plotting tools, and document creation tools. You can also translate your models into C code and work with them in other applications and tools, including applications that allow you to perform real-time simulation.

Interactive 3-D Visualization Tools

The MapleSim 3-D visualization environment allows you to build and animate 3-D graphical representations of your multibody mechanical system models. You can use this environment to validate the 3-D configuration of your model and visually analyze the behavior of your system under different conditions and at different simulation start times.

Related Products

MapleSim 6 requires Maple 16.

Maplesoft^{TM} also offers a suite of toolboxes, add-ons, and other applications that extend the capabilities of Maple and MapleSim for engineering design projects. For a complete list of products, visit **http://www.maplesoft.com/products**.

Related Resources

Resource	Description
MapleSim Installation Guide	System requirements and installation instructions for MapleSim. The **MapleSim Installation Guide** is available in the **Install.html** file on your MapleSim installation DVD.
MapleSim Help System	Provides the following information: • **MapleSim User's Guide**: conceptual information about MapleSim, an overview of MapleSim features, and tutorials to help you get started. • **Using MapleSim**: help topics for model building, simulation, and analysis tasks. • **MapleSim Component Library**: descriptions of the modeling components available in MapleSim.
MapleSim Examples	Model examples from various engineering domains. These models are available in the **Examples** palette in the **Libraries** tab on the left side of the MapleSim window.
MapleSim User's Guide Examples	Model and Tutorial examples used in the User's Guide. These models are available in the **Examples → User's Guide Examples** palette in the **Libraries** tab on the left side of the MapleSim window and are listed by chapter, in the order that they appear in the User's Guide.
MapleSim Online Resources	Training webinars, product demonstrations, videos, sample applications, and more. For more information, visit **http://www.maplesoft.com/products/maplesim**.
MapleSim Application Center	A collection of sample models, custom components, and analysis templates that you can download and use in your MapleSim projects. For more information, visit **http://www.maplesoft.com/applications/maplesim**.

For additional resources, visit **http://www.maplesoft.com/site_resources**.

Getting Help

To request customer support or technical support, visit **http://www.maplesoft.com/support**.

Customer Feedback

Maplesoft welcomes your feedback. For comments related to the MapleSim product documentation, contact doc@maplesoft.com.

1 Getting Started with MapleSim

In this chapter:

- *Physical Modeling in MapleSim (page 1)*
- *The MapleSim Window (page 6)*
- *Basic Tutorial: Modeling an RLC Circuit and DC Motor (page 8)*

1.1 Physical Modeling in MapleSim

Physical modeling, or physics-based modeling, incorporates mathematics and physical laws to describe the behavior of an engineering component or a system of interconnected components. Since most engineering systems have associated dynamics, the behavior is typically defined with ordinary differential equations (ODEs).

To help you develop models quickly and easily, MapleSim provides the following features:

Topological or “Acausal” System Representation

The signal-flow approach used by traditional modeling tools requires system inputs and outputs to be defined explicitly. In contrast, MapleSim allows you to use a topological representation to connect interrelated components without having to consider how signals flow between them.

Mathematical Model Formulation and Simplification

A topological representation maps readily to its mathematical representation and the symbolic capability of MapleSim automates the generation of system equations.

When MapleSim formulates the system equations, several mathematical simplification tools are applied to remove any redundant equations and multiplication by zero or one. The simplification tools then combine and reduce the expressions to get a minimal set of equations required to represent a system without losing fidelity.

Advanced Differential Algebraic Equation Solvers

Algebraic constraints are introduced in the topological approach to model definition. Problems that combine ODEs with these algebraic constraints are called Differential Algebraic Equations (DAEs). Depending on the nature of these constraints, the complexity of the DAE problem can vary. An index of the DAEs provides a measure of the complexity of the problem. Complexity increases with the index of the DAEs.

The development of generalized solvers for complex DAEs is the subject of much research in the symbolic computation field. With Maple as its computation engine, MapleSim uses advanced DAE solvers that incorporate leading-edge symbolic and numeric techniques for solving high-index DAEs.

Acausal and Causal Modeling

Real engineered assemblies, such as motors and powertrains, consist of a network of interacting physical components. They are commonly modeled in software by block diagrams. The lines connecting two blocks indicate that they are coupled by physical laws. When simulated by software, block diagrams can either be causal or acausal.

Causal Modeling

Many simulation tools are restricted to causal (or signal-flow) modeling. In these tools, a unidirectional signal, which is essentially a time-varying number, flows into a block. The block then performs a well-defined mathematical operation on the signal and the result flows out of the other side. This approach is useful for modeling systems that are defined purely by signals that flow in a single direction, such as control systems and digital filters.

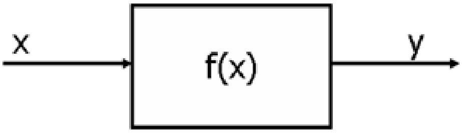


Figure 1.1: Causal Model Block Diagram

This approach is analogous to an assignment, where a calculation is performed on a known variable or set of variables on the right hand side and the result is assigned to another variable on the left:

$$y := f(x)$$

Acausal Modeling

Modeling how real physical components interact requires a different approach. In acausal modeling, a signal from two connected blocks travels in both directions. The programming analogy would be a simple equality statement:

$$y = f(x)$$

The signal includes information about which physical quantities (for example, energy, current, torque, heat and mass flows) must be conserved. The blocks contain information about which physical laws (represented by equations) they must obey and, hence, which physical quantities must be conserved.

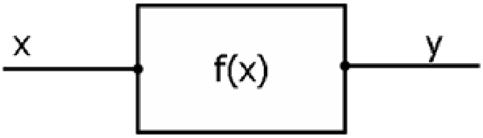


Figure 1.2: Acausal Model Block Diagram

MapleSim allows you to use both approaches. You can simulate a physical system (with acausal modeling) together with the associated logic or control loop (with causal modeling) in a manner that suits either task best.

Through and Across Variables

When using the acausal modeling approach, it is useful to identify the through and across variables of the component you are modeling. In general terms, an across variable represents the driving force in a system and a through variable represents the flow of a conserved quantity. The through variable also establishes the flow direction for the sign convention of the conserved quantity.

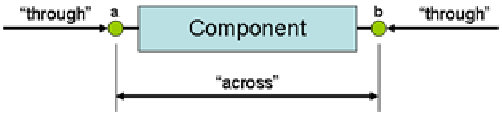


Figure 1.3: Simple Through and Across Variable Model

For an example of sign convention and how arrow direction represents a force acting on the model, under the **Libraries** tab, browse to the **Examples → User's Guide Examples** menu, and then open the **Constant Acceleration, Sign Convention** and **Arrow Convention** examples.

In the following example, in an electrical circuit, the through variable, *i*, is the current and the across variable, *V*, is the voltage drop:

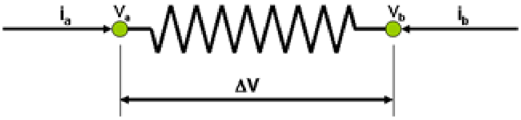


Figure 1.4: Simple Through and Across Variable Electrical Model

The following table lists some examples of through and across variables for other domains:

Table 1.1: Through and Across Variable Domain Types

Domain	Through	Across
Electrical	Current (*A*)	Voltage (*V*)
Magnetic	Magnetic Flux (*Wb*)	MMF (*A*)
Mechanical (translational)	Force (*N*)	Velocity $\left(\frac{m}{s}\right)$
Mechanical (rotational)	Torque (*N.m*)	Angular Velocity $\left(\frac{rad}{s}\right)$
Hydraulic	Flow $\left(\frac{m^3}{s}\right)$	Pressure $\left(\frac{N}{m^2}\right)$
Heat flow	Heat flow (*W*)	Temperature (*K*)

As a simple example, the form of the governing equation for a resistor is

$$V = R \cdot i$$

This equation, in conjunction with Kirchhoff's conservation of current law, allows a complete representation of a circuit.

$$R \cdot i_b = V_b - V_a \text{ and } i_b + i_a = 0$$

To extend this example, the following schematic diagram describes an RLC circuit, an electrical circuit consisting of a resistor, inductor, and a capacitor connected in series:

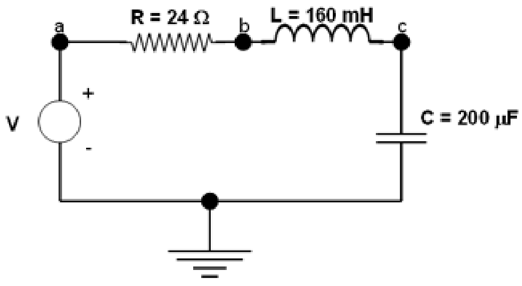


Figure 1.5: RLC Circuit

If you wanted to model this circuit manually, it can be represented with the following characteristic equations for the resistor, inductor, and capacitor respectively:

$$R \cdot i_R = V_a - V_b$$

$$L\frac{d}{dt}i_L = \left(V_b - V_c\right)$$

$$i_c = C\cdot\frac{d}{dt}V_c$$

By applying Kirchhoff's current law, the following conservation equations are at points *a*, *b*, and *c*:

$$i_V - i_R = 0$$

$$i_R - i_L = 0$$

$$i_L - i_C = 0$$

These equations, along with a definition of the input voltage (defined as a transient going from 0 to 1 volt, 1 second after the simulation starts)

$$V_a = \begin{cases} 0.0 & 0.0 \le t < 1.0 \\ 1.0 & \mathrm{t} \ge 1.0 \end{cases}$$

provide enough information to define the model and solve for the voltages and currents through the circuit.

In MapleSim, all of these calculations are performed automatically; you only need to draw the circuit and provide the component parameters. These principles can be applied equally to all engineering domains in MapleSim and allow you to connect components in one domain with components in others easily.

In the *Basic Tutorial: Modeling an RLC Circuit and DC Motor (page 8)* section of this chapter, you will model the RLC circuit described above and explore the capabilities of MapleSim to mix causal models with acausal models. The following figure shows how the RLC circuit diagram appears when it is built in MapleSim.

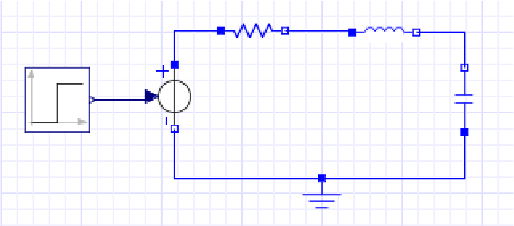
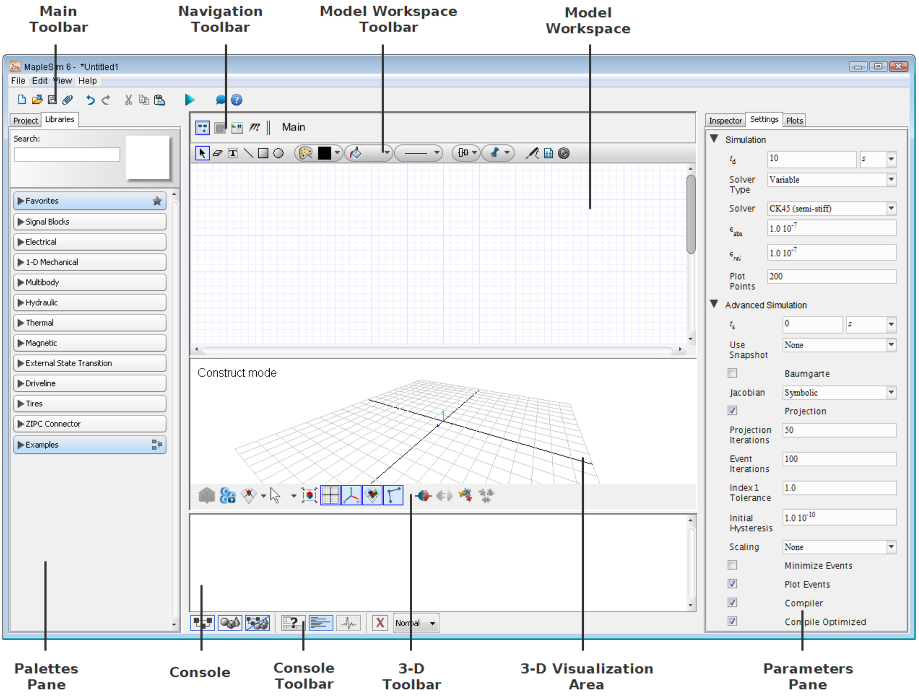


Figure 1.6: RLC MapleSim Circuit

For an another example of how a model can be represented using causal and acausal components, under the **Libraries** tab, browse to the **Examples** → **User's Guide Examples** menu, and then open the **Double Mass Spring Damper** example.

1.2 The MapleSim Window

The MapleSim window contains the following panes and components:



▼ Simulation

t_d 10 s

Solver Type Variable

Solver CK45 (semi-stiff)

ϵ_{abs} $1.0\ 10^{-7}$

ϵ_{rel} $1.0\ 10^{-7}$

Plot Points 200

▼ Advanced Simulation

t_s 0 s

Use Snapshot None

□ Baumgarte

Jacobian Symbolic

☑ Projection

Projection Iterations 50

Event Iterations 100

Index1 Tolerance 1.0

Initial Hysteresis $1.0\ 10^{-10}$

Scaling None

□ Minimize Events

☑ Plot Events

☑ Compiler

☑ Compile Optimized

Figure 1.7: MapleSim Window

Table 1.2: MapleSim Window Components

Component	Description
Main Toolbar	Contains tools for running a simulation, attaching MapleSim analysis templates to your model, and performing other common tasks.
Navigation Toolbar	Contains tools for browsing your model and subsystems hierarchically,changing the model view, and viewing corresponding Modelica code.

Component	Description
Model Workspace Toolbar	Contains tools for laying out and selecting objects, and adding elements such as annotations and probes.
Model Workspace	The area in which you build and edit a model in a block diagram view.
Palettes Pane	Contains expandable menus with tools that you can use to build a model and manage your MapleSim project. This pane contains two tabs: • **Libraries**: contains palettes with sample models and domain-specific components that you can add to models. • **Project**: contains palettes with tools to help you browse and build a model, and manage simulation results, probes, and documents that you attach to a model.
Console	Use the console buttons (? ≡ ∿) to display the following panes: (?) **Help**: displays the Help topic associated with a modeling component. (≡) **Message Console**: displays progress messages indicating the status of the MapleSim engine during a simulation and allows you to clear the console using **Clear Console** (X). (∿) **Debugging**: displays diagnostic messages as you build your model identifying the subsystem in which the errors are located.
Console Toolbar	Contains controls for selecting and controlling the types of messages shown in the console (? ≡ ∿) and the type of view you want work in (⊞ ● ⊠).
3-D Toolbar	Contains tools for building your multibody models and playing back the 3-D simulations.
3-D Visualization Area - Construct Mode (3-D Workspace)	In the Construct mode, the **3-D Visualization Area** allows you to build and analyze 3-D graphical representations of multibody systems.
3-D Visualization Area - Playback Mode	In the Playback mode, the **3-D Visualization Area** allows you to playback 3-D graphical representations of multibody systems. Changes to the model made in the **Model Workspace** are automatically reflected in the **3-D Visualization Area**.

Component	Description
Parameters Pane	Contains the following tabs: • **Inspector**: allows you to view and edit modeling component properties, such as names and parameter values, and specify simulation options and probe values. • **Settings**: allows you to specify simulation options, such as, the duration of the simulation and optional parameter values for the solver, simulation engine, and **3-D Workspace**. • **Plots**: allows you to define custom layouts for simulation graphs and plot windows. The contents of this pane change depending on your selection in the **Model Workspace**.

1.3 Basic Tutorial: Modeling an RLC Circuit and DC Motor

This tutorial introduces you to the modeling components and basic tools in MapleSim. It illustrates the ability to mix causal models with acausal models.

In this tutorial, you will perform the following tasks:

1. Build an RLC circuit model.
2. Set parameter values to specify component properties.
3. Add probes to identify values of interest for the simulation.
4. Simulate the RLC circuit model.
5. Modify the RLC circuit diagram to create a simple DC motor model.
6. Simulate the DC motor model using different parameters.

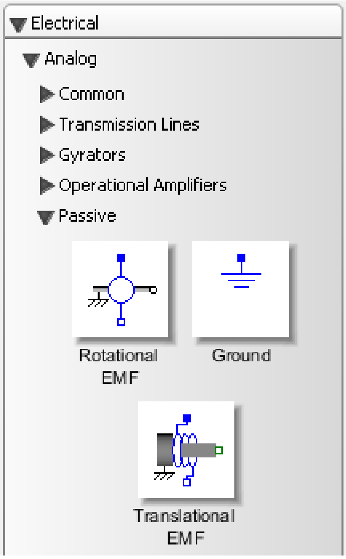
For an example of the **RLC Circuit** model, under the **Libraries** tab, browse to the **Examples → User's Guide Examples** menu, and then open the **RLC Circuit** example. The model you build is identical to the **RLC Circuit** model.

Building an RLC Circuit Model

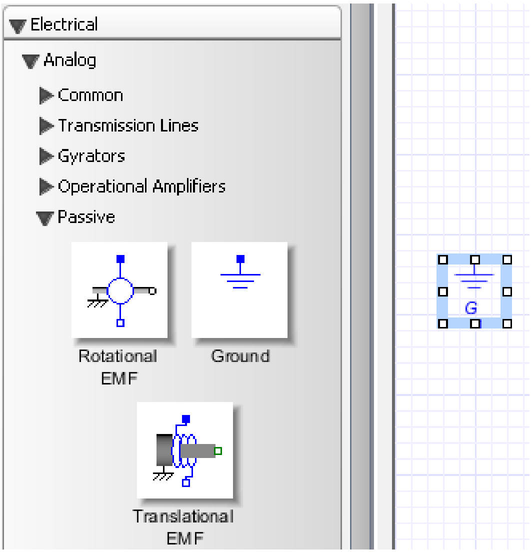
To build the RLC circuit, you add components in the **Model Workspace** and connect them in a system to form a diagram. In this example, the RLC circuit model contains ground, resistor, inductor, capacitor, and signal voltage source components from the Electrical component library. It also contains a step input source, which is a signal generator that drives the input voltage level in the circuit.

To build an RLC circuit

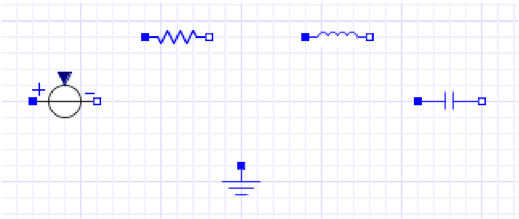
1. In the **Libraries** tab at the left of the **Model Workspace**, click the triangle beside **Electrical** to expand the palette. In the same way, expand the **Analog** menu, and then expand the **Passive** submenu.



2. From the **Electrical** → **Analog** → **Passive** menu, drag the **Ground** component to the **Model Workspace**.



3. Add the remaining electrical components to the **Model Workspace**.
   - From the **Electrical** → **Analog** → **Passive**→ **Resistors** menu, add the **Resistor** component.
   - From the **Electrical** → **Analog** → **Passive** → **Inductors** menu, add the **Inductor** component.
   - From the **Electrical** → **Analog** → **Passive** → **Capacitors** menu, add the **Capacitor** component.
   - From the **Electrical** → **Analog** → **Sources** → **Voltage** menu, add the **Signal Voltage** component.
4. Drag the components in the arrangement shown below.



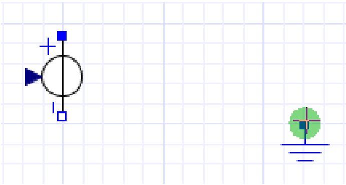
5. To rotate the **Signal Voltage** component clockwise, right-click (**Control**-click for Macintosh®) the **Signal Voltage** component in the **Model Workspace** and select **Rotate Clockwise**.

6. To flip the **Signal Voltage** component horizontally, right-click (**Control**-click for Macintosh) the component again and select **Flip Horizontal**. Make sure that the positive (blue) port is at the top.

7. To rotate the **Capacitor** component clockwise, right-click (**Control**-click for Macintosh) the **Capacitor** icon in the **Model Workspace** and select **Rotate Clockwise**.

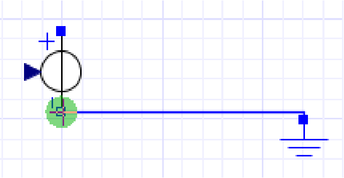
You can now connect the modeling components to define interactions in your system.

8. Hover your mouse pointer over the **Ground** component port. The port is highlighted in green.

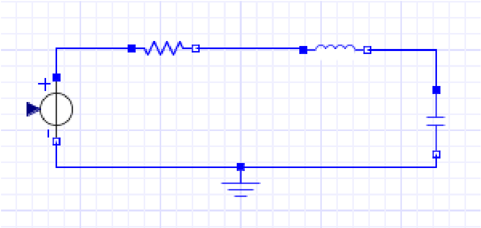


9. Click the **Ground** input port to start the connection line.

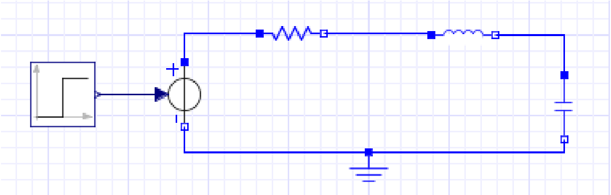
10. Hover your mouse pointer over the negative port of the **Signal Voltage** component.



11. Click the port once. The **Ground** component is connected to the **Signal Voltage** component.
12. Connect the remaining components in the arrangement shown below.



13. You can now add a source to your model. Expand the **Signal Blocks** palette, expand the **Sources** menu and then expand the **Real** submenu.
14. From the palette, drag the **Step** source and place it to the left of the **Signal Voltage** component in the **Model Workspace**. The step source has a specific signal flow, represented by the arrows on the connections. This flow causes the circuit to respond to the input signal.
15. Connect the **Step** source to the **Signal Voltage** component. The complete RLC circuit model is shown below.



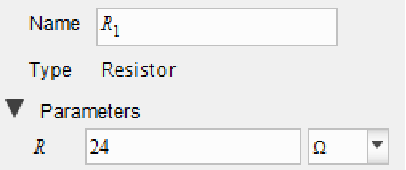
Specifying Component Properties

To specify component properties, you can set parameter values for components in your model.

To specifying component properties

1. In the **Model Workspace**, click the **Resistor** component. The **Inspector** tab at the right of the **Model Workspace** displays the name and parameter values of the resistor.

2. In the **R** field, enter 24, and press **Enter**. The resistance changes to *24Ω*..



Name R_1

Type Resistor

▼ Parameters

R 24 Ω

3. Specify the following parameter values for the other components. You can specify units for a parameter by selecting a value from the drop-down menu found beside the parameter value field.

- For the **Inductor**, specify an inductance of 160*mH*.

- For the **Capacitor**, specify a capacitance of 200µ*F*.

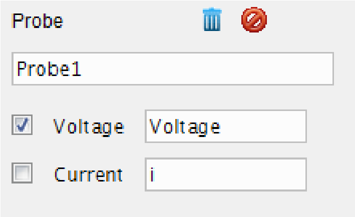
- For the **Step** source, specify a $\mathbf{T_0}$ value of 0.1*s*.

Adding a Probe

To specify data values for a simulation, you must attach probes to lines or ports to the model. In this example, you will measure the voltage of the RLC circuit.

To add a probe

1. In the **Model Workspace Toolbar**, click **Attach Probe** ( ).
2. Hover your mouse pointer over the line that connects the **Inductor** and **Capacitor** components. The line is highlighted.
3. Click the line once. The probe appears in the **Model Workspace**.
4. Drag the probe to an empty location on the **Model Workspace**, and then click the workspace to position the probe.
5. Select the probe. The probe properties appear under the **Inspector** tab to the right of the **Model Workspace**.
6. Under the **Inspector** tab, select the **Voltage** check box to include the voltage quantity in the simulation graph.
7. To display a custom name for this quantity in the **Model Workspace**, enter **Voltage** as shown below and press **Enter**.



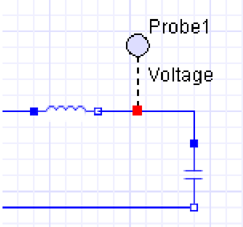
Probe

Probe1

☑ Voltage: Voltage

□ Current: i

The probe with the custom name is added to the connection line.



For another example of how to use a probe value in a simulation, under the **Libraries** tab, browse to the **Examples** → **User's Guide Examples** menu, and then open the **Sensors and Probes** example.

Simulating the RLC Circuit Model

Before simulating your model, you can specify the simulation duration run time.

To simulate the RLC circuit

1. Click the **Settings** tab at the top of the Parameters Pane and in the **Simulation** section, set the simulation duration time (**t_d**) to 0.5s.
2. In the **Advanced Simulation** section, clear the **Compiler** check box.
3. Click **Run Simulation** (▶) in the **Main Toolbar**. MapleSim generates the system equations and simulates the response to the step input.

When the simulation is complete, the voltage response is plotted in a graph.

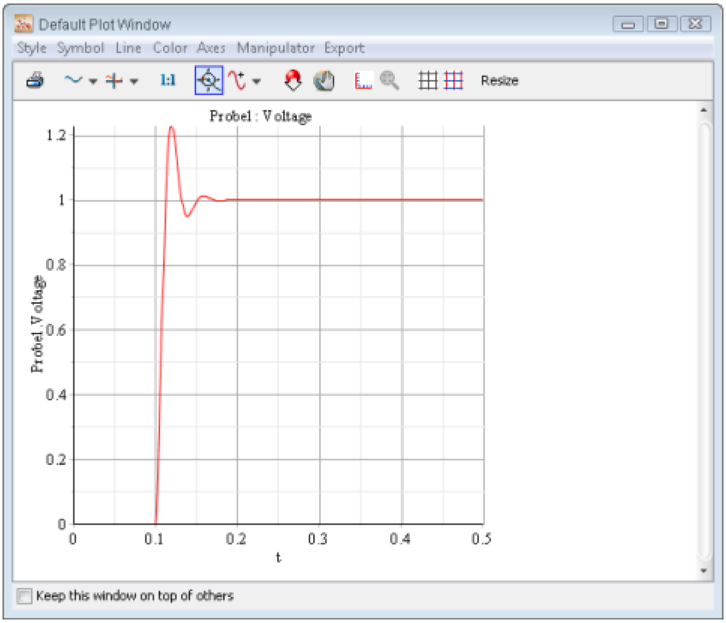


Figure 1.8: Voltage Response Plot

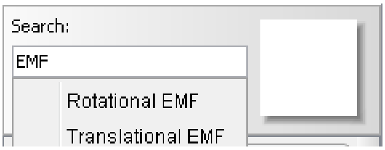
4. Save the model as **RLC_Circuit1.msim**. The probes and modified parameter values are saved as part of the model.

Building a Simple DC Motor Model

You will now add an electromotive force (EMF) component and a mechanical inertia component to the RLC circuit model to create a DC motor model. In this example, you will add components to the RLC circuit model using the search feature.

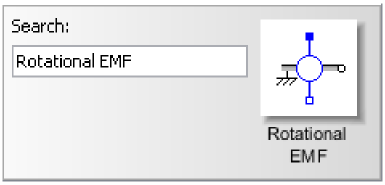
To build a simple DC motor

1. In the **Libraries** tab, type **EMF** in the **Search** field located above the palettes. A drop-down list displays matches for your search results.



Search:
EMF
Rotational EMF
Translational EMF

2. Select **Rotational EMF** from the drop-down list. The **Rotational EMF** component appears in the square beside the search field.



3. Drag the **Rotational EMF** component to the modeling workspace and place it to the right of the **Capacitor** component.
4. In the **Search** field, search for **Inertia**.
5. Add the **Inertia** component to the **Model Workspace** and place it to the right of the **Rotational EMF** component.
6. Connect the components as shown below.

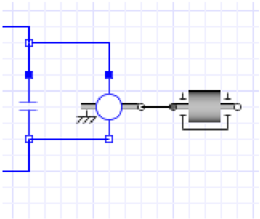


Figure 1.9: Rotational EMF

Note: To connect the positive blue port of the **Rotational EMF** component, click the port once, drag your mouse pointer to the line connecting the capacitor and inductor, and then click the line.

7. In the **Model Workspace**, click the **Rotational EMF** component.
8. Click the **Inspector** tab and in the **Parameters** section, set the value of the transformation coefficient (**k**) to **10** $\frac{N \cdot m}{A}$.
9. Click the **Step** component and change the value of the parameter, $\mathbf{T_0}$, to $1s$.

Simulating the DC Motor Model

To simulate the DC motor

1. In the **Model Workspace**, delete **Probe1**.
2. In the **Model Workspace Toolbar**, click **Attach Probe** (⟋).
3. Hover your mouse pointer over the line that connects the **Rotational EMF** and **Inertia** components.

4. Click the line, and then click on an empty area of the workspace to position the probe.
5. Select the probe, and in the **Inspector** tab, select the **Speed** and **Torque** check boxes and clear the **Angle** check box. The probe, with an arrow indicating the direction of the conserved quantity flow, is added to the model. The direction of the conserved quantity flow (Torque) can be reversed by selecting the probe and then clicking on **Reverse Probe** (↩) in the **Inspector** tab.
6. Click a blank area in the **Model Workspace**.
7. In the **Settings** tab, set the simulation duration time (**t_d**) to 5*s*.
8. Click **Run Simulation** (▶) in the **Main Toolbar**. The following graphs appear.

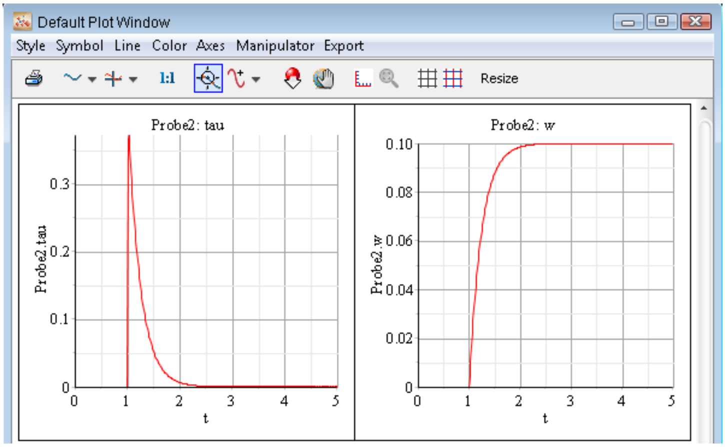


Figure 1.10: Plots of DC Motor Torque (tau) and Speed (w)

9. Save the model as **DC_Motor1.msim**.

2 Building a Model

In this chapter:

2.1 The MapleSim Component Library

The MapleSim component library contains over 500 components that you can use to build models. All of these components are organized in palettes according to their respective domains: electrical, magnetic, hydraulic, 1-D mechanical, multibody, signal blocks, and thermal. Most of these components are based on the Modelica® Standard Library 3.1.

Table 2.1: MapleSim Component Library

Library	Description
Electrical	Components to model electrical analog circuits, single-phase and multiphase systems, and electric machines.
Magnetic	Components to model magnetic circuits.
Hydraulic	Components to model hydraulic systems such as fluid power systems, cylinders, and actuators.
1-D Mechanical	Components to model 1-D translational and rotational systems.
Multibody	Components to model multibody mechanical systems, including force, motion, and joint components.
Signal Blocks	Components to manipulate or generate input and output signals.
Thermal	Components to model heat flow and heat transfer.

The library also contains sample models that you can view and simulate, for example, complete electrical circuits and filters. For more information about the MapleSim library structure and modeling components, see the **MapleSim Component Library** in the MapleSim Help system.

To extend the default library, you can create a custom modeling component from a mathematical model and add it to a custom library. For more information, see *Creating Custom Modeling Components (page 65)*.

Viewing Help Topics for Components

In the Help pane below the **Model Workspace**, you can view the Help topic for each component from the MapleSim Component Library. To display the Help pane, click **Component Help** (?) at the bottom of the MapleSim window. You can then select a component that you have added to the **Model Workspace** to view its Help topic. Alternatively, to view Help topics, you can perform one of the following tasks:

- Right-click (**Control-**click for Macintosh) a modeling component in any of the palettes and select **Help** from the context menu.
- Search for the Help pages for components in the MapleSim Help system.

Updating Models Created in MapleSim 4 or Earlier

In MapleSim 4 or earlier, components from the Modelica Standard Library 2.2.1 were included in the MapleSim Component Library. In MapleSim 4.5, the Modelica Standard Library 2.2.1 was replaced by the Modelica Standard Library 3.1.

If you created a model in MapleSim 4 or earlier, you can open it in MapleSim 4.5 or later; it will then be updated automatically to use equivalent components from the Modelica Standard Library 3.1. For more information, see **Using MapleSim → Updating Models Created in MapleSim 4.5 or Earlier** topic in the MapleSim Help system.

Note: Models that contain components from the Modelica Standard Library 3.1 cannot be used in MapleSim 4 or earlier.

2.2 Browsing a Model

Using the **Model Tree** palette or model navigation controls, you can browse your model to view hierarchical levels of components in the **Model Workspace**. You can browse to the top level for an overall view of your system. The top level is the highest level of your model: it represents the complete system, which can include individual modeling components and subsystem blocks that represent groups of components. You can also browse to sublevels in your model to view the contents of individual subsystems or components.

Model Tree

To browse your model, you can use the **Model Tree** palette in the **Project** tab located on the left side of the MapleSim window. Each node in the model tree represents a modeling component, subsystem, or connection port in your model. For example, the model tree of a DC motor is shown below.

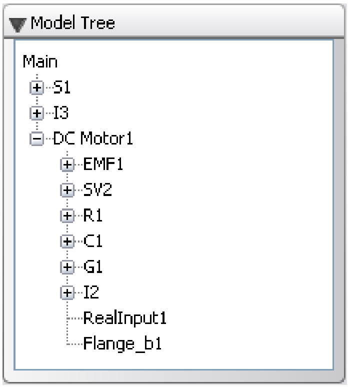


Figure 2.1: Model Tree

To browse your model and view the parameters associated with a component or subsystem, expand and double-click the nodes in the model tree. You can click the **Main** node to view the top level of your model and the child nodes to view the contents of a component or subsystem.

Model Navigation Controls

Alternatively, you can use the model navigation controls located above the **Model Workspace Toolbar** to browse between modeling components, subsystems, and hierarchical levels in a diagram displayed in the **Model Workspace**.

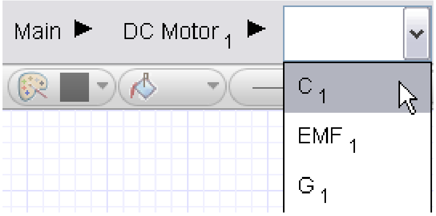
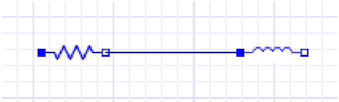


Figure 2.2: Model Navigational Controls

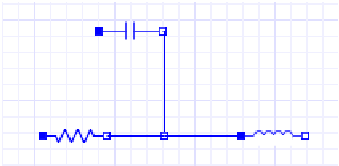
From the drop-down menu, select the name of the subsystem or modeling component that you want to view in the **Model Workspace**. You can click **Main** (Main ►) to browse to the top level of your model. You can also browse directly to subsystems in your model. For example, by clicking **DC Motor$_1$** in the example shown above, you can view the DC motor subsystem contents in the **Model Workspace**.

2.3 Defining How Components Interact in a System

To define interactions between modeling components, you connect them in a system. In the **Model Workspace**, you can draw a connection line between two connection ports.



You can also draw a connection line between a port and another connection line.



MapleSim permits connections between compatible domains only. By default, each line type appears in a domain-specific color.

Table 2.2: Domain Specific Connection Line Colors

Domain	Line Color
Mechanical 1-D rotational	Black
Mechanical 1-D translational	Green
Mechanical multibody	Black
Electrical analog	Blue
Electrical multiphase	Blue
Magnetic	Orange
Digital logic	Purple
Boolean signal	Pink
Causal signal	Navy blue
Integer signal	Orange
Thermal	Red

The connection ports for each domain are also displayed in specific colors and shapes. For more information about connection ports, see the **MapleSim Component Library → Connectors Overview** topic in the MapleSim Help system.

2.4 Specifying Component Properties

To specify component properties, you can set parameter values for components in your model. When you select a component in the **Model Workspace**, the configurable parameter values for that component appear in the **Inspector** tab located on the right side of the MapleSim window.

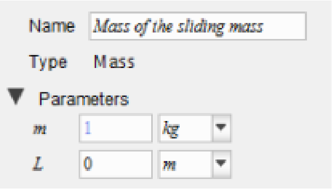
Note: Not all components provide editable parameter values.

You enter parameter values in 2-D math notation, which is a formatting option that allows you to add mathematical text such as superscripts, subscripts, and Greek characters. For more information, see *Entering Text in 2-D Math Notation (page 55)*.

Note: Most parameters in the MapleSim Component Library have default values. However, for some parameters, these default values are simply placeholders that may not represent realistic values for use in a simulation. These placeholder values use a blue font to distinguish them from other parameter values. You should replace these values with values that are more suitable for your simulation. For more information, see **Using MapleSim → Building a Model → Renaming a Modeling Component** in the MapleSim Help system.

Specifying Parameter Units

You can use the drop-down menus beside parameter fields with dimensions to specify units for parameter values. For example, the image below displays the configurable parameter fields for a **Mass** component. You can optionally specify the mass in *kg*, lb_m, *g*, or *slug*, and the length in *m*, *cm*, *mm*, *ft*, or *in*.



Name *Mass of the sliding mass*

Type Mass

▼ Parameters

m 1 *kg* ▾

L 0 *m* ▾

When you simulate a model, MapleSim automatically converts all parameter units to the International System of Units (SI). You can, therefore, select more than one system of units for parameter values throughout a model.

If you want to convert the units of a signal, use the **Unit Conversion Block** component from the **Signal Converters** menu in the **Signal Blocks** palette. This component allows you to perform conversions in dimensions such as time, temperature, velocity, pressure, and volume. In the following example, a **Unit Conversion Block** component is connected between a translational **Position Sensor** and **Feedback** component to convert the units of an output signal.

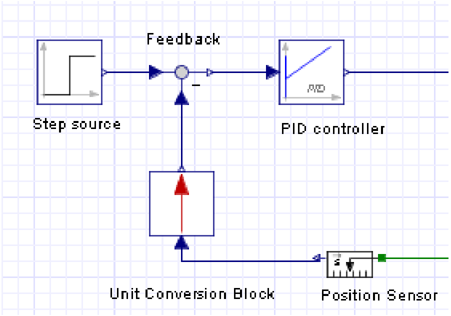


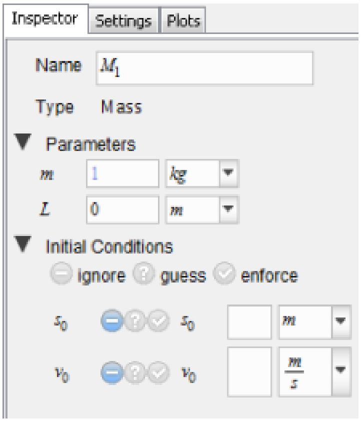
Figure 2.3: Specifying Units using the Unit Conversion Block

If you include an electrical, 1-D mechanical, hydraulic, or thermal sensor in your model, you can also select the units in which to generate an output signal.

Specifying Initial Conditions

You can set parameter values to specify initial conditions for components from all domains in MapleSim. When you select a component that contains state variables in the **Model Workspace**, the available initial condition fields appear in the **Inspector** tab, along with the other configurable parameter values for that component.

For example, the image below displays the initial velocity, position, and acceleration fields that you can set for a **Mass** component.



Name M_1

Type Mass

m 1 kg

L 0 m

s_0 s_0 m

v_0 v_0 $\frac{m}{s}$

Figure 2.4: Initial Conditions

Specifying How Initial Conditions are Enforced

You can determine how the initial conditions that you specified for a particular component are enforced. The options are **ignore** (⊖), **guess** (?), and **enforce** (✓). You can select these options for initial condition parameters individually by clicking the buttons beside the applicable initial condition fields.

If you select the **ignore** option, the parameter value that you enter in the initial condition field is ignored and the solver uses a default value for the initial condition, typically zero. This option is the default setting for all of the initial condition fields.

If you select the **guess** option, the solver treats the parameter value you entered in the initial condition field as a best guess value. In other words, the best guess value is a starting point for determining the initial configuration of the system for which there is a solution to the set of equations that describe the system. The solver initially computes a solution to the system of equations using this best guess value; however, if no solution is found, the solver computes a solution to the system of equations using an initial condition value that is close to the best guess value.

If you select the **enforce** option, the solver uses the parameter value that you enter in the initial condition field as a start value for the simulation. Similar to the **guess** option, the solver searches for a solution to the system of equations using the parameter value you entered in the initial condition field. However, unlike the **guess** option, if there is no solution, no other value is substituted, and an error message appears.

For more information about selecting these options, see *Best Practices: Enforcing Initial Conditions (page 64)*.

For an example of how initial conditions are enforced, under the **Libraries** tab, browse to the **Examples** → **User's Guide Examples** menu, and then open the **Relative Positions** example.

2.5 Creating and Managing Subsystems

A subsystem (or compound component) is a set of modeling components that are grouped in a single block component. A simple DC motor subsystem is shown below.

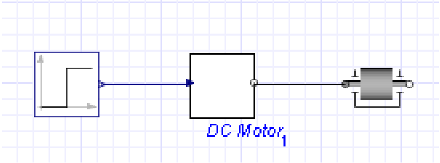
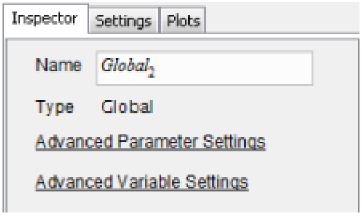


Figure 2.5: Subsystem Group

You can create a subsystem to group components that form a complete system, for example, a tire or DC motor. You can also create a subsystem to improve the layout of a diagram in the **Model Workspace**, add multiple copies of a system to a model, analyze a component group in Maple or to quickly assign parameters and variables. You can organize your model hierarchically by creating subsystems within other subsystems.

Once you create a subsystem you will be able to assign parameters and variables to all components in that subsystem using the **Advanced Parameter Settings** and **Advanced Variable Settings** tool in the **Inspector** tab.



Name *Global$_2$*

Type Global

For best practices on creating subsystems in MapleSim, see *Best Practices: Laying Out and Creating Subsystems (page 58)*.

Example: Creating a Subsystem

In the following example, you will group the electrical components of a DC motor model into a subsystem.

To create a subsystem

1. Under the **Libraries** tab on the left side of the MapleSim window, expand the **Examples** → **User's Guide Examples** menu, and then open the **Simple DC Motor** example.

2. Using the **Selection Tool** ( ) located above the **Model Workspace**, draw a box around the electrical components.

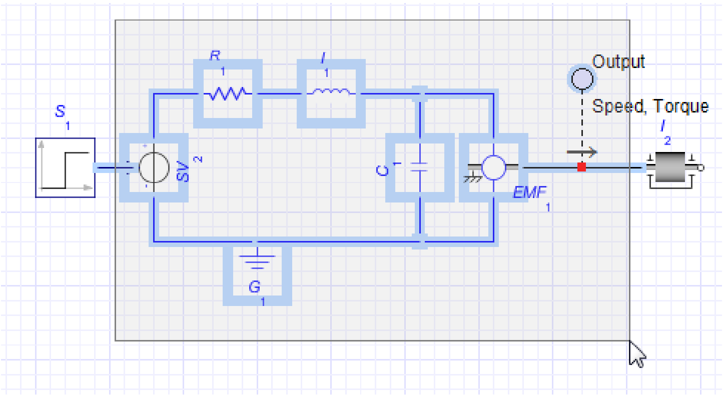
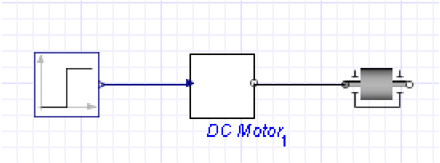


Figure 2.6: Creating a Subsystem

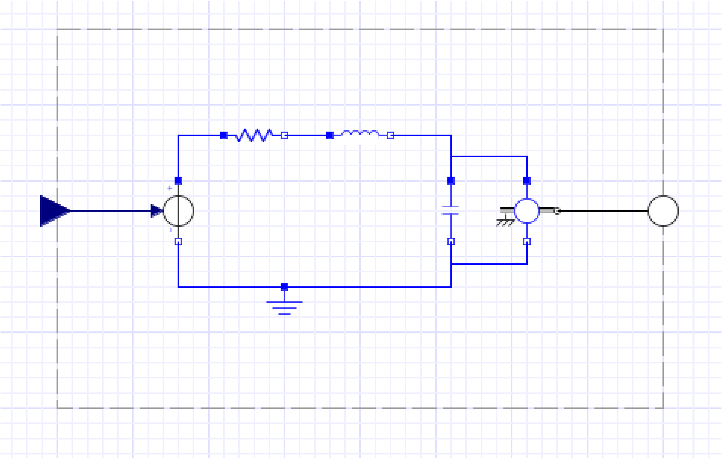
3. From the **Edit** menu, select **Create Subsystem** (or right-click the boxed area and select **Create Subsystem**).
4. In the dialog box, enter **DC Motor**.
5. Click **OK**. A white block, which represents the DC motor, appears in the **Model Workspace**.



In this example, you created a *stand-alone subsystem*, which can be edited and manipulated independently of other subsystems in your model. If you want to add multiple copies of the same subsystem to your model and edit those subsystems as a group, you can create a *subsystem definition*. For more information, see *Adding Multiple Copies of a Subsystem to a Model (page 26)*.

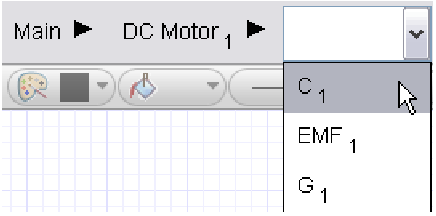
Viewing the Contents of a Subsystem

To view the contents of a subsystem, double-click the subsystem icon in the **Model Workspace**. The detailed view of a subsystem appears.



In this view, a broken line indicates the subsystem boundary. You can edit the connection lines and components within the boundary, add and connect components outside of the boundary, and add subsystem ports to connect the subsystem to other components. If you want to resize the boundary, click the broken line and drag one of the sizing handles displayed around the box.

To browse to the top level of the model or to other subsystems, use the controls in the **Navigation Toolbar**.



Adding Multiple Copies of a Subsystem to a Model

If you plan to add multiple copies of a subsystem to a model and want all of the copies to have the same configuration, you can create a *subsystem definition*. A subsystem definition is the base subsystem that defines the attributes and configuration that you want a series of subsystems to share.

For example, if you want to add three DC motor subsystems that all have identical components and resistance values in your model, you would perform the following tasks:

1. Build a DC motor subsystem with the desired configuration in the **Model Workspace**.
2. Use that subsystem configuration to create a subsystem definition and add it to the **Definitions** palette.
3. Add copies of the DC motor subsystem to your model using the subsystem definition as a source.

To add copies of the DC motor subsystem to your model, you can drag the DC Motor subsystem definition icon from the **Definitions** palette and place it in the **Model Workspace**. The copies that you add to the **Model Workspace** will then share a configuration that is identical to the subsystem definition in the **Definitions** palette; the copies in the **Model Workspace** are called *shared subsystems* because they share and refer to the configuration specified in their corresponding subsystem definition.

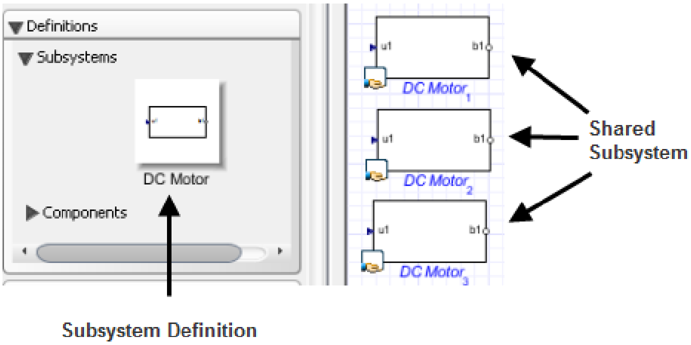


Figure 2.7: Creating Multiple Subsystems

Shared subsystems that are copied from the same subsystem definition are *linked*, which means that changes you make to one shared subsystem will be reflected in all of the other shared subsystems that were created from the same subsystem definition. The changes are also reflected in the subsystem definition entry in the **Definitions** palette.

Using the example shown above, if you change the resistance parameter of the **Resistor** component in the **DC Motor$_2$** shared subsystem from 24Ω to 10Ω, the resistance value of the **Resistor** component in the **DC Motor$_1$** and **DC Motor$_3$** shared subsystems and the **DC Motor** subsystem definition in the **Definitions** palette will also be changed to 10Ω..

For more information, see *Editing Subsystem Definitions and Shared Subsystems (page 29)*.

Example: Adding Subsystem Definitions and Shared Subsystems to a Model

In the following example, you will create a **DC Motor** subsystem definition and add multiple shared subsystems to your model.

Adding a Subsystem Definition to the Definitions Palette

To add a subsystem definition

1. In the **Model Workspace**, right-click (**Control**-click for Macintosh) the stand-alone DC motor subsystem that you created in *Example: Creating a Subsystem (page 24)*.
2. From the context menu, select **Convert to Shared Subsystem**.
3. Enter **DC Motor** as the name for the subsystem definition and click **OK**.
4. Under the **Project** tab on the left side of the **Model Workspace**, expand the **Definitions** palette and then expand the **Subsystems** menu.

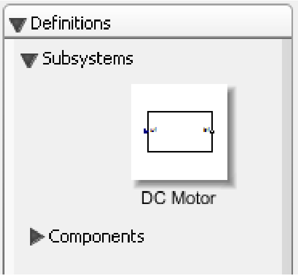


Figure 2.8: Subsystem Definition

The subsystem definition is added to the **Definitions** palette and the subsystem in the **Model Workspace** is converted into a shared subsystem called **DC Motor$_1$**. This shared subsystem is linked to the **DC Motor** subsystem definition.

5. Save this model as **DCMotorSubsystem.msim**. You will be building on this model in *Example: Editing Shared Subsystems that are Linked to the Same Subsystem Definition (page 29)*.

You can now use this subsystem definition to add multiple DC motor shared subsystems to your MapleSim model.

Tip: If you want to use a subsystem definition in another model, add the subsystem definition to a custom library. For more information, see *Creating and Managing Custom Libraries (page 50)*.

Adding Multiple DC Motor Shared Subsystems to a Model

To add multiple **DC Motor** shared subsystems to a model, drag the **DC Motor** subsystem definition icon from the **Definitions** palette and place it in the **Model Workspace**.

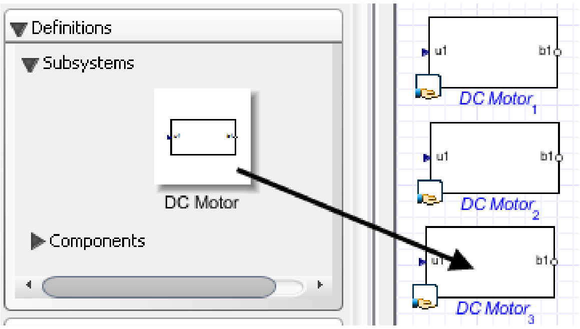


Figure 2.9: Adding Multiple Subsystems to a Model

When you create a new stand-alone subsystem or add shared subsystems to a model, a unique subscript number is appended to the subsystem name displayed in the **Model Workspace**. As shown in the image above, subscript numbers are appended to the names of each DC Motor shared subsystem. These numbers can help you to identify multiple subsystem copies in your model.

Editing Subsystem Definitions and Shared Subsystems

If you edit a shared subsystem in the **Model Workspace**, your changes will be reflected in the subsystem definition that is linked to the shared subsystem, as well as other shared subsystems that were copied from the same subsystem definition.

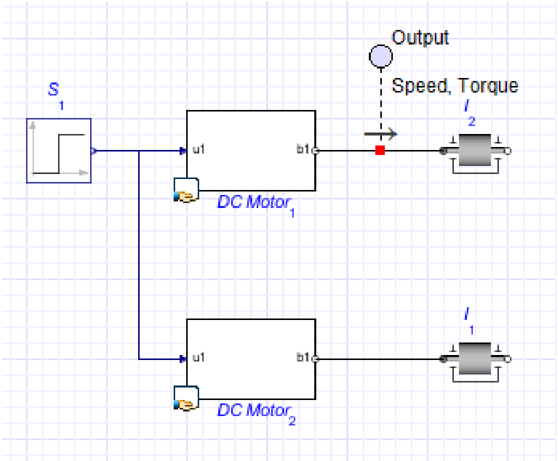
Example: Editing Shared Subsystems that are Linked to the Same Subsystem Definition

In this example, you will create a model that contains two **DC Motor** shared subsystems, and then edit the resistance values and icons for the shared subsystems. These shared subsystems are linked to the **DC Motor** shared subsystem definition that was created in *Example: Adding Subsystem Definitions and Shared Subsystems to a Model (page 28)*. You will verify that when you change one of the component values and the icon for one **DC Motor** shared subsystem, the other **DC Motor** shared subsystems in your model--as well as any new **DC Motor** shared subsystems that you add in the future--will contain the changes.

Note: Before doing this example, you should have already gone through and stored the results from *Example: Adding Subsystem Definitions and Shared Subsystems to a Model (page 28)*.

To use shared subsystems

1. In MapleSim, open the **DCMotorSubsystem.msim** file that you created in *Example: Adding Subsystem Definitions and Shared Subsystems to a Model (page 28)*.
2. Under the **Project** tab, expand the **Definitions** → **Subsystems** menu, and then drag a second **DC Motor** shared subsystem on to the workspace, placing it below the existing **DC Motor** shared subsystem.
3. Under the **Libraries** tab, expand the **1-D Mechanical** → **Rotational** → **Common** menu, and then drag a second **Inertia** component on to the workspace, placing it below the existing **Inertia** component.
4. Make the following connections between the newly added components and the existing components in the model.



5. In the **Model Workspace**, double-click the **DC Motor$_1$** shared subsystem. The detailed view of the shared subsystem appears.

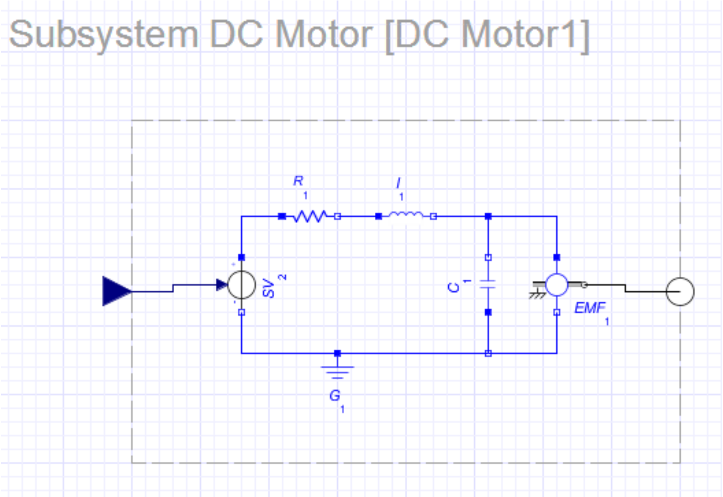
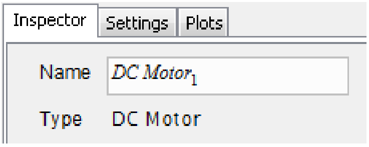


Figure 2.10: DC Motor Subsystem

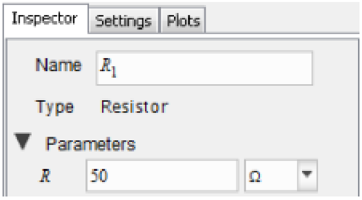
Note that a heading with the subsystem definition name (**DC Motor**) followed by the shared subsystem name (**DC Motor$_1$**) appears at the top of the **Model Workspace**. In the detailed view of all shared subsystems, this heading also appears to help you identify multiple subsystem copies in your model. Also, when you select a shared subsystem, its subsystem definition name appears in the **Type** field in the **Inspector** tab.



Name *DC Motor*$_1$

Type DC Motor

6. Select the **Resistor** component (**R$_1$**) and, in the **Inspector** tab, click **Parameters**. Change the resistance value to **50Ω**.



Name R_1

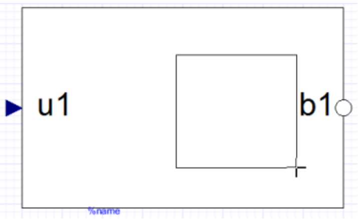
Type Resistor

▼ Parameters

R 50 Ω

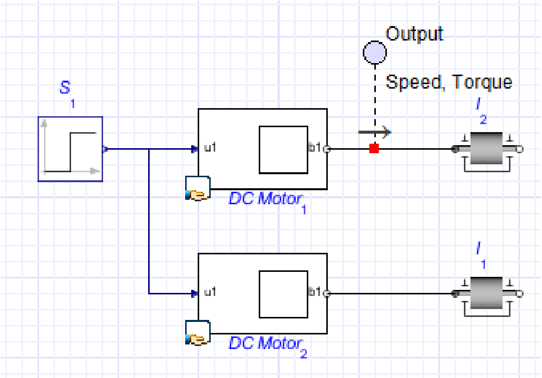
7. In the **Navigation Toolbar**, click **Icon** (✎).

8. Using the **Rectangle Tool** (□) in the **Model Workspace Toolbar**, click and drag your mouse pointer to draw a shape in the box.

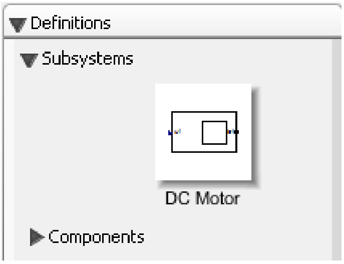


9. In the **Navigation Toolbar**, click **Diagram** ().

10. Click **Main** in the **Navigation Toolbar** to browse to the top level of the model. Both of the **DC Motor** shared subsystems now display the square that you drew.



11. In the **Project** tab on the left side of the MapleSim window, expand the **Definitions** palette, and then expand the **Subsystems** menu. As shown in the image below, your changes are also reflected in the **DC Motor** entry in this palette.



If you double-click the **DC Motor** subsystems in the **Model Workspace** and select their **Resistor** components, you will see that both of the shared subsystems now have a resistance value of 50Ω.

12. From the **Definitions** palette, drag a new copy of the **DC Motor** subsystem and place it anywhere in the **Model Workspace**. Verify that the new copy displays the square that you drew and its resistance value is also 50*Ω*, and then delete it from the workspace.
13. Save this model as **DCMotorSharedSubsystem.msim**. You will be building on this model in *Example: Removing the Link Between a Shared Subsystem and its Subsystem Definition (page 33)*.

Example: Removing the Link Between a Shared Subsystem and its Subsystem Definition

If your model contains multiple shared subsystems that are linked and you want to edit one copy only, you can remove the link between a shared subsystem and its subsystem definition, and edit that subsystem without affecting others in the **Model Workspace**.

Note: Before doing this example, you should have already gone through and stored the results from *Example: Editing Shared Subsystems that are Linked to the Same Subsystem Definition (page 29)* and saved the results from that example.

To remove shared subsystem link

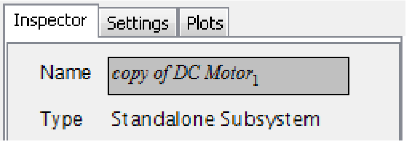
1. Open the **DCMotorSharedSubsystem.msim** model that you created in *Example: Editing Shared Subsystems that are Linked to the Same Subsystem Definition (page 29)*.
2. In the **Model Workspace**, right-click (**Control**-click for Macintosh) the **DC Motor$_2$** shared subsystem.
3. Select **Convert to Stand-alone Subsystem**. The **DC Motor$_2$** subsystem is no longer linked to the **DC Motor** subsystem definition in the **Definitions** palette; it is now called **copy of DC Motor$_1$**.
4. Double-click the **DC Motor$_1$** shared subsystem.
5. Click **Icon** (✎).
6. Using the **Rectangle Tool** (□), click and drag your mouse pointer to draw a shape in the box in the **Model Workspace**.
7. Click **Diagram** (⊞), and then click **Main** to browse to the top level of the model. Your change is shown in the **DC Motor$_1$** shared subsystem in the **Model Workspace** and the **DC Motor** subsystem definition in the **Definitions** palette. Note that your change is not shown in the **copy of DC Motor$_1$** subsystem that is no longer linked to the **DC Motor** subsystem definition.

Tip: When you convert a shared subsystem to a stand-alone subsystem, it is a good practice to assign the stand-alone subsystem a meaningful name that clearly distinguishes it from existing shared subsystems and subsystem definitions.

Working with Stand-alone Subsystems

Stand-alone subsystems are subsystems that are not linked to a subsystem definition. You can create a stand-alone subsystem in two ways: by creating a new subsystem as shown in *Example: Creating a Subsystem (page 24)* or by converting a shared subsystem to a stand-alone subsystem as shown in *Example: Removing the Link Between a Shared Subsystem and its Subsystem Definition (page 33)*. Stand-alone subsystems can be edited independently without affecting other subsystems in the **Model Workspace**.

To identify a subsystem as a stand-alone subsystem, select a subsystem in the **Model Workspace** and examine the **Inspector** tab. If that subsystem is a stand-alone subsystem, the **Type** field reads **Standalone Subsystem**.



Name *copy of DC Motor$_1$*

Type Standalone Subsystem

Also, if you double-click a stand-alone subsystem to browse to its detailed view, no heading is shown for the subsystem in the **Model Workspace**.

When you copy and paste a stand-alone subsystem in the **Model Workspace**, you can optionally convert that subsystem into a shared subsystem and create a new subsystem definition. For more information, see *Example: Copying and Pasting a Stand-alone Subsystem (page 35)*.

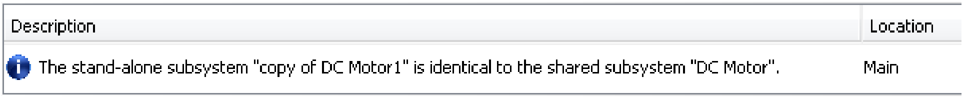
Example: Resolving Warning Messages in the Debugging Console

When you convert a shared subsystem into a stand-alone subsystem, the subsystem is highlighted in the **Model Workspace** and a warning message appears, informing you that the link to the subsystem definition has been removed.

Note: This example is an extension of *Example: Removing the Link Between a Shared Subsystem and its Subsystem Definition (page 33)*.

To resolve a warning message

1. Click **Diagnostic Information** () at the bottom of the MapleSim window to display the debugging console. The following warning message appears in the console.



Description	Location
ℹ The stand-alone subsystem "copy of DC Motor1" is identical to the shared subsystem "DC Motor".	Main

2. To work with the **copy of DC Motor$_1$** subsystem as a stand-alone subsystem, right-click (**Control**-click for Macintosh) the warning message and select **Ignore duplication**

warnings for "copy for DC Motor1" to hide the warning message from the debugging console.

Tip: If you want to view warning messages that you hid from the debugging console, click **Reset Ignored Warnings** (G) below the console. All of the warning messages that you previously hid will appear in the debugging console again.

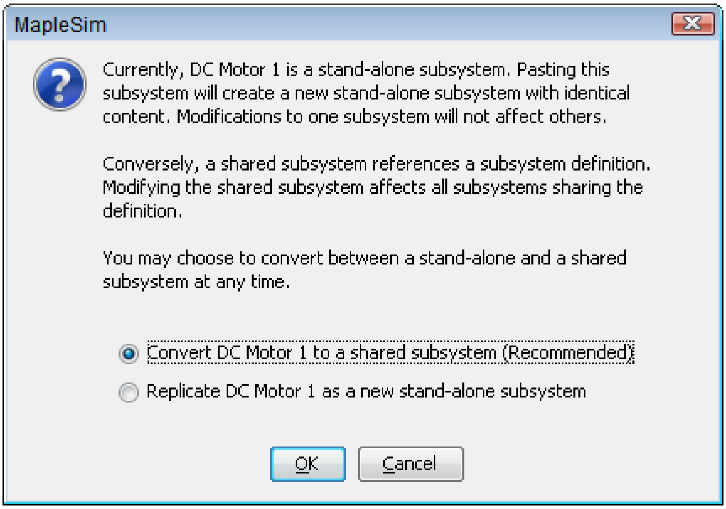
Alternatively, if you want to link the **copy of DC Motor**$_1$ stand-alone subsystem to the **DC Motor** subsystem definition again, you can right-click (**Control**-click for Macintosh) the warning message and select **Update "copy of DC Motor1" to use the shared subsystem "DC Motor"**.

Example: Copying and Pasting a Stand-alone Subsystem

Note: This example is an extension of *Example: Removing the Link Between a Shared Subsystem and its Subsystem Definition (page 33)*.

To copy and past a stand-alone subsystem

1. In the **Model Workspace**, copy and paste the **copy of DC Motor**$_1$ stand-alone subsystem. The following dialog box appears:
2. Select **Convert DC Motor 1 to a shared subsystem (Recommended)**. A new subsystem definition called **DC Motor 1** is added to the **Definitions** palette.



Currently, DC Motor 1 is a stand-alone subsystem. Pasting this subsystem will create a new stand-alone subsystem with identical content. Modifications to one subsystem will not affect others.

Conversely, a shared subsystem references a subsystem definition. Modifying the shared subsystem affects all subsystems sharing the definition.

You may choose to convert between a stand-alone and a shared subsystem at any time.

● Convert DC Motor 1 to a shared subsystem (Recommended)

○ Replicate DC Motor 1 as a new stand-alone subsystem

In the **Model Workspace**, the **copy of DC Motor**$_1$ stand-alone subsystem has been converted to a shared subsystem called **copy of DC Motor**$_1$ and another copy of that shared subsystem called **copy of DC Motor**$_2$ has been added to the **Model Workspace**. Both the **copy of DC**

Motor$_1$ and **copy of DC Motor$_2$** shared subsystems are linked to the new **DC Motor 1** subsystem definition. Therefore, if you edit either **copy of DC Motor$_1$** or **copy of DC Motor$_2$** in the **Model Workspace**, your changes will not be reflected in subsystems that are linked to the original **DC Motor** subsystem definition.

Note: Alternatively, you can select **Replicate DC Motor 1 as a new stand-alone subsystem** to add another stand-alone subsystem that can be edited independently without affecting the other subsystems in the **Model Workspace**.

2.6 Global and Subsystem Parameters

MapleSim lets you define global and subsystem parameter values, and assign them to components using the **Add or Change Parameters** editor, parameter blocks, parameter sets and the **Advanced Parameter Settings** and **Advanced Variable Settings** in the **Inspector** tab.

Global Parameters

If your model contains multiple components that share a common parameter value, you can create a global parameter. A global parameter allows you to define a common parameter value in one location and then assign that common value to multiple components in your model.

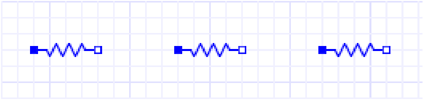
The following example describes how to define and assign a global parameter. To view a more detailed example, see *Tutorial 1: Modeling a DC Motor with a Gearbox (page 145)* in Chapter 6 of this guide.

Example: Defining and Assigning a Global Parameter

If your model contains multiple **Resistor** components that have a common resistance value, you can define a global parameter for the resistance value in the parameter editor view.

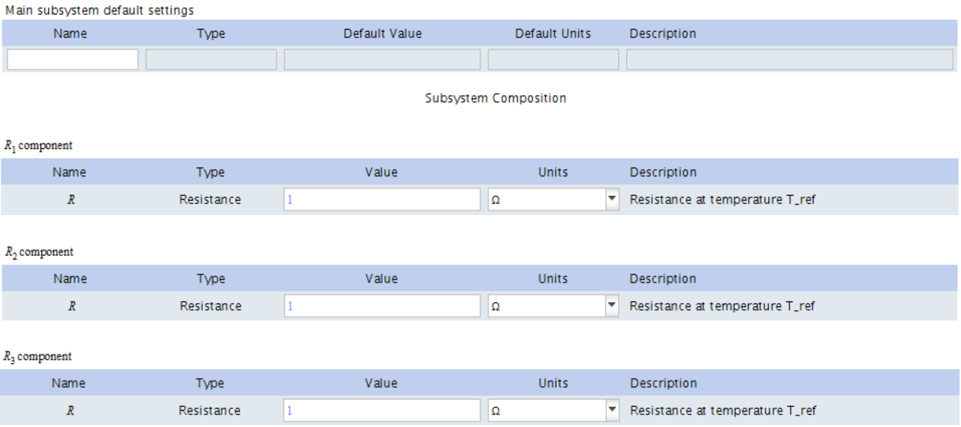
To define and assign a global parameter

1. In the **Libraries** tab, expand the **Electrical** palette, expand the **Analog** menu, expand the **Passive** menu, and then expand the **Resistors** menu.
2. From the palette, drag three copies of the **Resistor** component into the **Model Workspace**.



3. In the **Navigation Toolbar**, click **Parameters** (⊞), or click the workspace and from the **Inspector** tab, click **Add or Change Parameters**. The **Main subsystem default settings**

screen appears. You will use this screen to define the global parameter and assign it to the **Resistor** components in your model.



Main subsystem default settings

Name	Type	Default Value	Default Units	Description

Subsystem Composition

R_1 component

Name	Type	Value	Units	Description
R	Resistance	1	Ω	Resistance at temperature T_ref

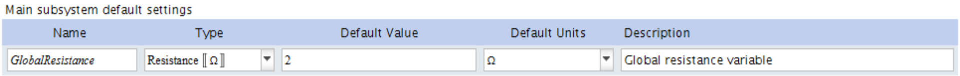
R_2 component

Name	Type	Value	Units	Description
R	Resistance	1	Ω	Resistance at temperature T_ref

R_3 component

Name	Type	Value	Units	Description
R	Resistance	1	Ω	Resistance at temperature T_ref

4. Click the first field under the **Name** column in the **Main subsystem default settings** table.
5. Enter **GlobalResistance** as the global parameter name and press **Enter**.
6. Select Resistance[[Ω]] and specify a default value of 2.
7. Enter **Global resistance variable** as the description and press **Enter**.

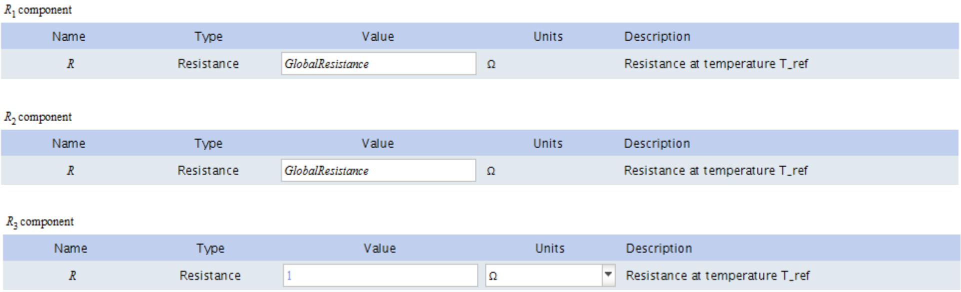


Main subsystem default settings

Name	Type	Default Value	Default Units	Description
GlobalResistance	Resistance [[Ω]]	2	Ω	Global resistance variable

The global parameter for the resistance value is now defined. You can now assign the common **GlobalResistance** parameter value to the individual **Resistor** components that you added to the **Model Workspace**.

8. In the $\mathbf{R_1}$ **component** table and $\mathbf{R_2}$ **component** table, enter **GlobalResistance** as the resistance value.



R_1 component

Name	Type	Value	Units	Description
R	Resistance	*GlobalResistance*	Ω	Resistance at temperature T_ref

R_2 component

Name	Type	Value	Units	Description
R	Resistance	*GlobalResistance*	Ω	Resistance at temperature T_ref

R_3 component

Name	Type	Value	Units	Description
R	Resistance	1	Ω	Resistance at temperature T_ref

The resistance value of the parameter **GlobalResistance** (**2**, as defined in the **Main subsystem default settings** table) has now been assigned to the resistance parameters of the **R$_1$** and **R$_2$** components.

The **R$_1$** and **R$_2$** components will now inherit any changes made to the **GlobalResistance** parameter value in the **Main subsystem default settings** table. For example, if you change the default value of the **GlobalResistance** parameter to **5** in the **Main subsystem default settings** table, the resistance parameters of the **R$_1$** and **R$_2$** components will also be changed to **5**. Any change to the **GlobalResistance** parameter value will not apply to the **R$_3$** component because it has not been assigned **GlobalResistance** as a parameter value.

Subsystem Parameters

You can create a subsystem parameter if you want to create a common parameter value to share with multiple components in a subsystem. Similar to global parameters, a subsystem parameter is a common value that you define in the parameter editor view and assign to components.

There are two ways to assign subsystem parameters; one is by clicking **Parameters** (#) and the other is by using the **Advanced Parameter Settings** tool in the **Inspector** tab. Parameters can only be assigned to components in the subsystem in which they are defined. If you click a subsystem in the **Model Workspace**, click **Parameters** (#) or **Advanced Parameter Settings**, and define a parameter in the parameter editor view, the parameter that you define is assigned to components in the subsystem that you selected and any nested subsystems.

To view an example, see *Tutorial 3: Modeling a Nonlinear Damper (page 156)* in Chapter 6 of this guide.

Note: If you create a parameter within a subsystem and assign its value to a component at the top level, the component at the top level will not inherit the parameter value.

Example: Assigning a Subsystem Parameter to a Shared Subsystem

If you assign a subsystem parameter to a shared subsystem in your model, the default subsystem parameter will also be assigned to other shared subsystems that are linked to it. However, after the default subsystem parameter is assigned, you can edit the subsystem parameter value for each shared subsystem separately without affecting other parameter values in the model.

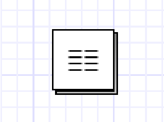
To assign a subsystem parameter to a shared subsystem

1. In the **Examples** palette, expand the **Physical Domains** → **Multibody** menu, and then open the **Double Pendulum** model. This model contains two shared subsystems, $\mathbf{L_1}$ and $\mathbf{L_2}$, which are linked to a subsystem definition called **L**.
2. Double-click the $\mathbf{L_1}$ shared subsystem.
3. Click **Parameters** (▦).
4. In the **L subsystem default settings table**, click the empty field at the bottom of the table.
5. Type **c** as the parameter name, keep the default value as **1**, and press **Enter**.
6. Click **Diagram** (▦). The new subsystem parameter, **c**, appears in the **Inspector** tab for the $\mathbf{L_1}$ shared subsystem.
7. In the top view of the model, select the $\mathbf{L_2}$ subsystem and examine the **Inspector** tab. The new subsystem parameter is also displayed for the $\mathbf{L_2}$ shared subsystem.
8. In the **Inspector** tab, change the value of **c** to **50**.
9. Click the $\mathbf{L_1}$ shared subsystem in the **Model Workspace** and examine the **Inspector** tab. Note that the value of its parameter, **c**, remains the same.

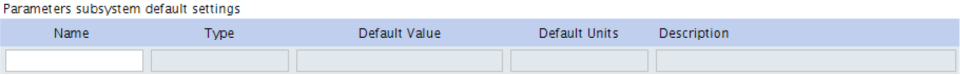
Creating Parameter Blocks

As an alternative to defining subsystem parameters using the methods described above, you can create a parameter block to define a set of subsystem parameters and assign them to components in your model. Parameter blocks allow you to apply parameters in multiple models at the top level of the **Model Workspace**.

The following image shows a parameter block that has been added to the **Model Workspace**.



When you double-click this block, the parameter editor view appears. This view allows you to define parameter values for the block.



Parameters subsystem default settings

Name	Type	Default Value	Default Units	Description

After defining parameter values, you can assign those values to the component parameters in your model.

To use parameter values in another model, you can add a parameter block to a custom library. For more information about custom libraries, see *Creating and Managing Custom Libraries (page 50)*.

Notes:

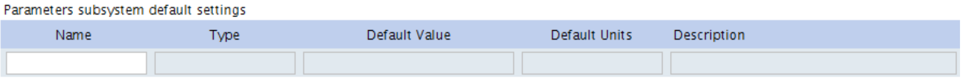
- Parameter blocks must be placed in the same subsystem as the components to which you want to assign the parameter value.
- Parameter blocks at the same hierarchical level in a model cannot have the same parameter names. For example, two separate parameter blocks in the same subsystem cannot each contain a parameter called **mass**.

Example: Creating and Using a Parameter Block

In this example, you will create a set of parameters that can be shared by multiple components in your model. By creating a parameter block, you only need to edit parameter values in one location to compare results when you run multiple simulations.

To create and use a parameter block

1. In the **Libraries** tab on the left side of the MapleSim window, expand the **Examples** palette, expand the **Physical Domains** menu, expand the **1-D Mechanical** menu, and then open the **PreLoad** example.
2. Under the **Settings** tab, enter **0.012** seconds for $\mathbf{t_d}$, the simulation duration time.
3. Click the $\mathbf{SM_1}$ Mass component on the workspace, and then click the **Inspector** tab. You will be using a parameter block to set values for the following parameters: m, L, s_0, and v_0.
4. From the **Model Workspace Toolbar**, click **Add a parameter block** (▤), and then click on a blank area in the **Model Workspace**.
5. Click the **Inspector** tab and enter the name **SlidingMassParams** for the parameter block.
6. Double-click the **SlidingMassParams** parameter block in the **Model Workspace**. The parameter editor view appears.



Parameters subsystem default settings

Name	Type	Default Value	Default Units	Description

7. Click the first field in the table and define a new symbolic parameter called **MASS**.
8. Press **Enter**. The remaining fields for this row are activated.
9. From the **Type** drop-down menu, select **Mass [[kg]]**.
10. Enter a default value of **5**.

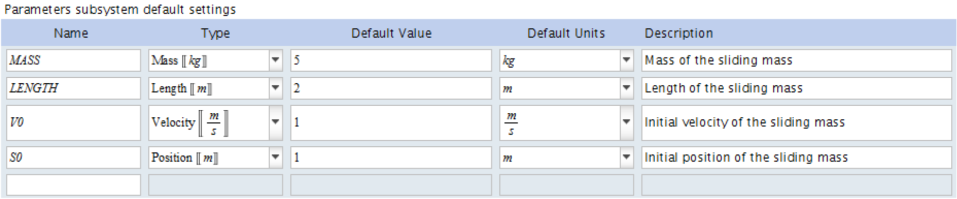
11. From the **Default Units** drop-down menu, select **kg**.

12. Enter **Mass of the sliding mass** for the **Description** field.

13. In the same way, define the following parameters and values in the **Parameters subsystem default settings** table.

Name	Type	Default Value	Default Units	Description
LENGTH	Length $[[m]]$	2	m	Length of the sliding mass
V0	Velocity $\left[\left[\frac{m}{s}\right]\right]$	1	$\frac{m}{s}$	Initial velocity of the sliding mass
S0	Position $[[m]]$	1	m	Initial position of the sliding mass

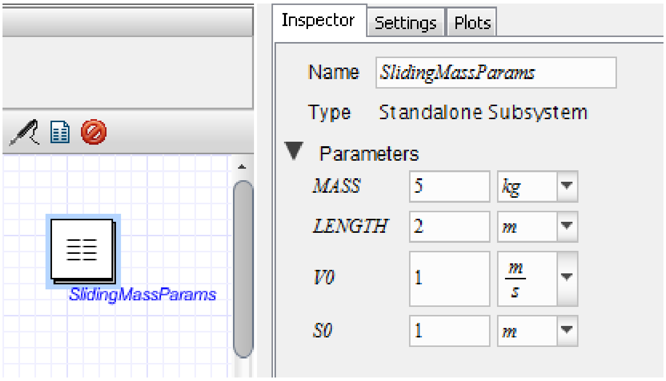
The parameter editor view appears as follows when the values are defined.



Parameters subsystem default settings

Name	Type	Default Value	Default Units	Description
$MASS$	Mass $[[kg]]$	5	kg	Mass of the sliding mass
$LENGTH$	Length $[[m]]$	2	m	Length of the sliding mass
$V0$	Velocity $\left[\left[\frac{m}{s}\right]\right]$	1	$\frac{m}{s}$	Initial velocity of the sliding mass
$S0$	Position $[[m]]$	1	m	Initial position of the sliding mass

14. Click **Diagram** ( ) and then click **Main** (Main ▶) in the **Navigation Toolbar**. When you select the parameter block in the **Model Workspace**, the defined parameters appear in the **Inspector** tab on the right side of the MapleSim window.



Name $SlidingMassParams$

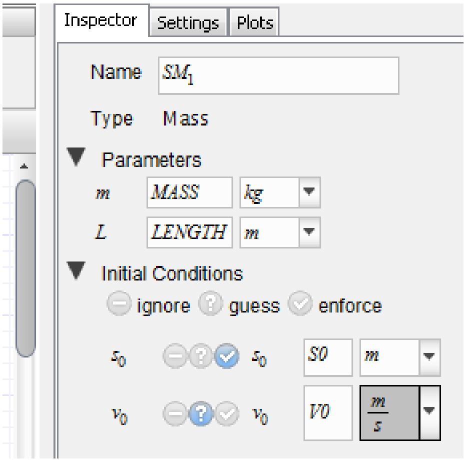
Type Standalone Subsystem

Parameters

$MASS$	5	kg
$LENGTH$	2	m
$V0$	1	$\frac{m}{s}$
$S0$	1	m

15. In the **Model Workspace**, select the $\mathbf{SM_1}$ mass component in the diagram.

16. In the **Inspector** tab, assign the following values and press **Enter**.



Inspector | Settings | Plots

Name SM_1

Type Mass

▼ Parameters

m $MASS$ kg

L $LENGTH$ m

▼ Initial Conditions

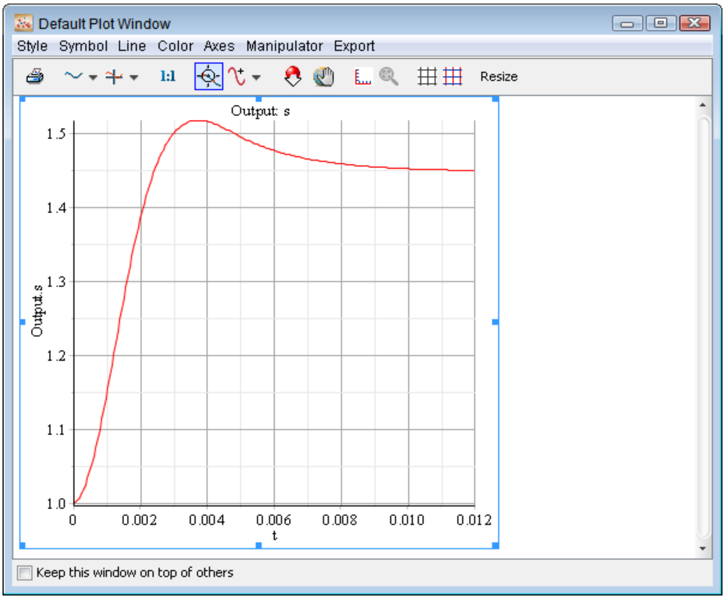
○ ignore ? guess ✓ enforce

s_0 ✓ s_0 $S0$ m

v_0 ? v_0 $V0$ $\frac{m}{s}$

The parameters of this **Mass** component now inherit the numeric values that you defined in the parameter block.

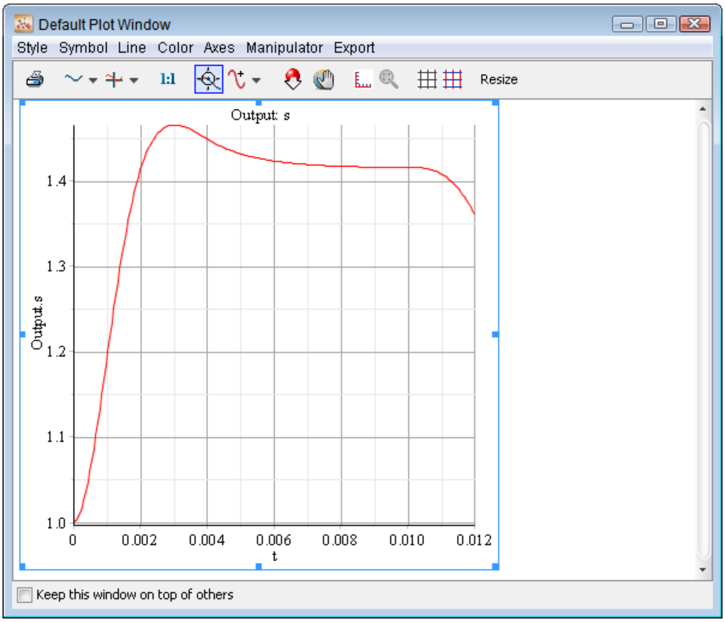
17. In the same way, assign the same values to the parameters of the **SM$_2$** and **SM$_3$** mass components in the model.
18. In the **Model Workspace**, delete the probe labeled **Input**.
19. Select the probe labeled **Output**.
20. In the **Inspector** tab, clear the check box beside **Velocity**.
21. To simulate the model, click **Run Simulation** (▶) in the **Main Toolbar**. The following graph appears.



22. In the **Model Workspace**, click the parameter block.

23. In the **Inspector** tab, change the mass to **3.5** and the initial velocity to **5**. Press **Enter**. These changes apply to all of the **Mass** components to which you assigned the symbolic parameter values.

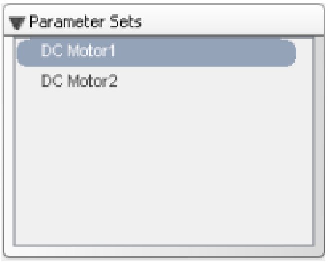
24. Simulate the model again. Another simulation graph appears, which you can compare to your first graph.



Creating Parameter Sets

The parameters you create for your model can be stored as reusable Parameter Sets. Parameter Sets let you save, reuse, and compare different sets of parameters for the same model displayed in the workspace. At any time you may easily apply and run different simulations, saving new values for each model. A Parameter Set provides a snapshot of all the parameters in the **Model Workspace**.

Parameter Sets for your model are listed in the **Project** tab, under Parameter Sets as shown in the following figure.



You can use, save, reuse, and compare different sets of parameters for the same model by right-clicking (**Control**-click for Macintosh) on a Parameter Set. For more information, see the **Using MapleSim → Building a Model → Storing and Applying Parameter Sets** section in the MapleSim Help system.

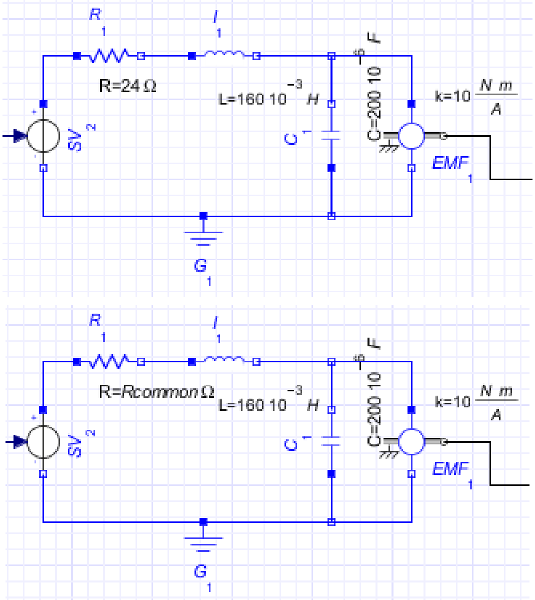
Using Advanced Parameter and Variable Settings

At the top level of your model, in the **Main subsystem default settings** window, you define the subsystem by adding parameters and setting their default values. An alternative is to directly assign subsystem parameters, variables, and initial conditions to components in your subsystem by using the **Advanced Parameter Settings** and **Advanced Variable Settings** tool in the **Inspector** tab. Advanced Settings lets you override one or more default values.

Advanced Parameter Settings

Advanced Parameter Settings lets you override the default values for selected subsystem components. If desired, you can parametrize the override using the parametrization feature (▦). A component override in one subsystem can be converted to a parameter visible in all the other subsystems.

In the following model an override was applied to the initial value of R and changed to the parameter **Rcommon**.



Advanced Variable Settings

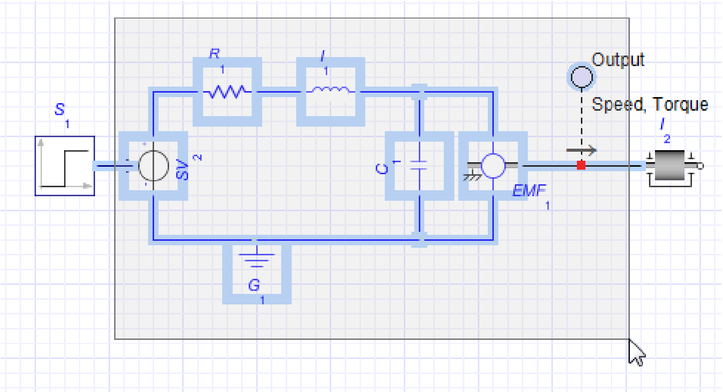
Advanced Variable Settings lets you specify initial conditions for subsystem components. When you select **Advanced Variable Settings** the initial condition fields appear for all configurable components for that subsystem.

Example: Creating a Parameter Override

To create a parameter override

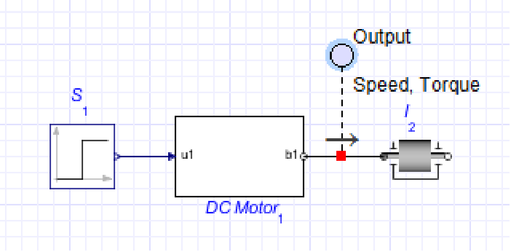
1. Under the **Libraries** tab on the left side of the MapleSim window, expand the **Examples** palette, expand the **User's Guide Examples** menu, and then open the **Simple DC Motor** example.

2. Using the **Selection Tool** (↖) located above the **Model Workspace**, draw a box around the electrical components.



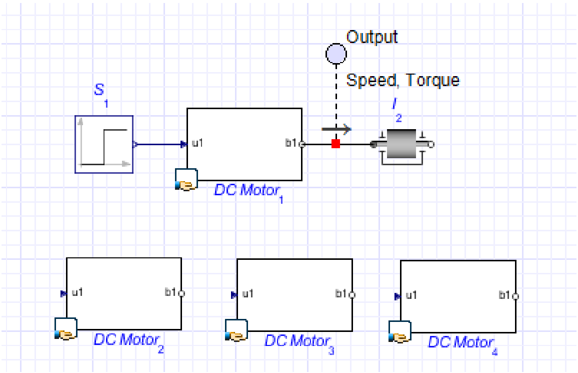
3. From the **Edit** menu, select **Create Subsystem** or right-click (**Control**-click for Macintosh) the boxed area and select **Create Subsystem**.

4. In the dialog box, enter **DC Motor**, and then click **OK**. The DC Motor subsystem appears.

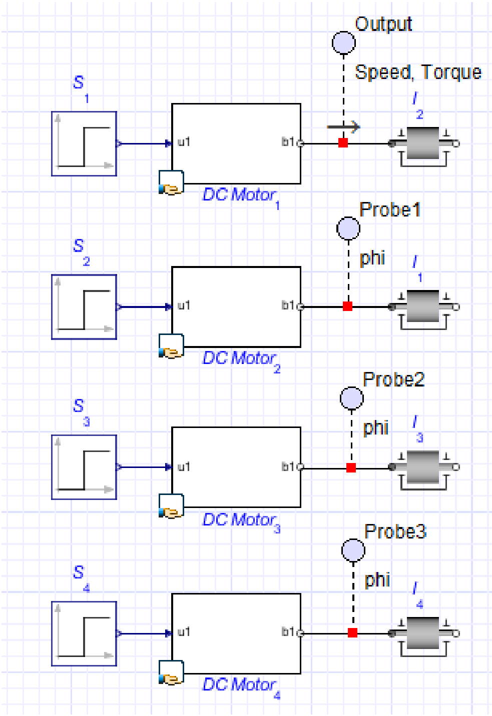


5. Use the DC Motor subsystem to create the shared subsystem definition and add it to the **Definitions** palette.

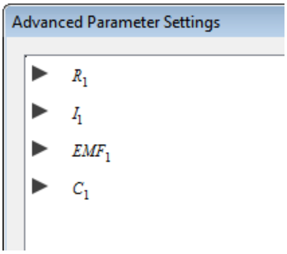
6. Add three copies of the DC motor subsystem to your **Model Workspace** by dragging the DC Motor subsystem definition icon from the **Definitions** palette and placing it in the **Model Workspace**.



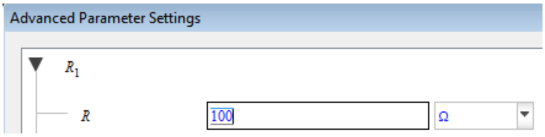
7. Create three additional DC Motor subsystems as shown below.



8. Click the **DC Motor$_4$** subsystem, and then click **Advanced Parameter Settings** under the **Inspector** tab. The **Advanced Parameter Settings** window appears, showing all of the subsystem components.

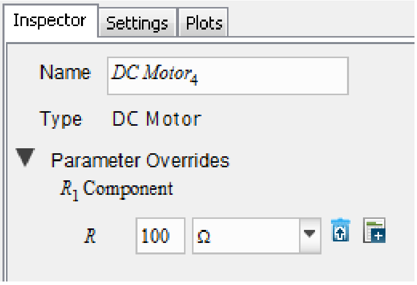


9. Expand R1 and enter a value of 100 for the Resistance parameter (R).



R 100 Ω

10. Click **OK**. The new parameter appears in the **Inspector** tab as an override.



Name $DC\ Motor_4$

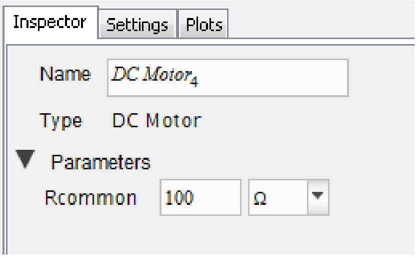
Type DC Motor

▼ Parameter Overrides

R_1 Component

R 100 Ω

11. To change this override to make it a reusable parameter, click **Parametrize** (⊞), enter **Rcommon** as the new parameter name, and then click **OK**. **Rcommon** appears in the **Inspector** tab as a parameter that can be can now be reused in the other subsystems. Note that it is no longer an override.



Name $DC\ Motor_4$

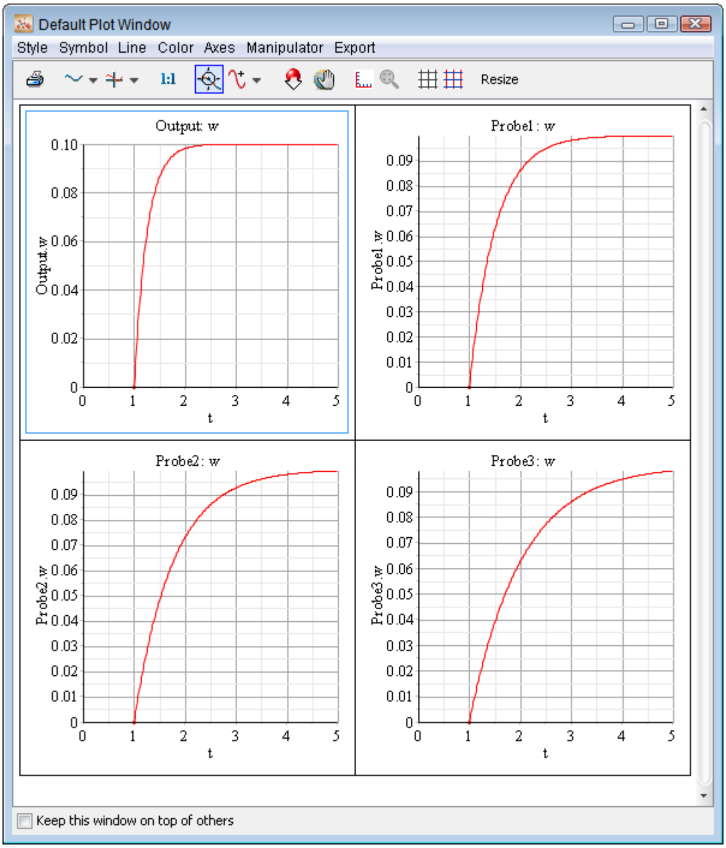
Type DC Motor

▼ Parameters

Rcommon 100 Ω

12. For each of the other subsystems, click the subsystem and enter values of 75*Ω*, 50*Ω*, and 25*Ω* for **Rcommon** in DC Motor subsystems 3, 2,and 1 respectively.

13. For each of the subsystems, select the probe, and in the Inspector tab select the **Speed** check box and clear all of the other check boxes.
14. Click **Run Simulation** (▶) in the **Main Toolbar**. The following graphs appear for each of the subsystems.



Specifying Initial Condition Overrides

You can set initial condition values to override existing initial conditions for specific subsystem components. When you select a component, the available initial condition fields and any existing overrides appear in the **Inspector** tab, along with the other configurable parameter values for that component.

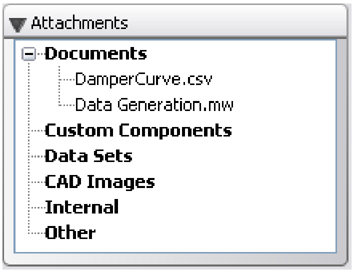
When you select a subsystem and then click **Advanced Variable Settings**, all subsystem components appear. You can select a component and specify the initial conditions for that component. This feature is especially useful for models that contain multiple shared subsystems.

2.7 Attaching Files to a Model

You can use the **Attachments** palette in the **Project** tab to attach files of any format to a model (for example, spreadsheets or design documents created in external applications). You can save files attached in the **Attachments** palette as part of the current model and refer to them when you work with that model in a future MapleSim session. To save a file, right-click (**Control**-click for Macintosh) the category in which you want to save the attachment in the palette and select **Attach File**.

You can also attach a file to a model from the menu bar by selecting **File > Attach File...** . Using this method, by default, the file attaches to the **Documents** category. If you want to move this attachment, you can click and drag the entry to another category.

The following image shows an **Attachments** palette that contains files called **DamperCurve.csv** and **Data Generation.mw**.



Attachments

Documents

DamperCurve.csv

Data Generation.mw

Custom Components

Data Sets

CAD Images

Internal

Other

Figure 2.11: Attachments

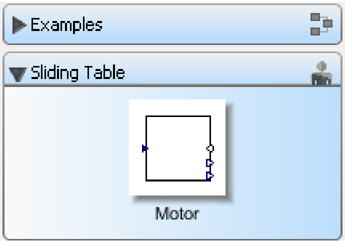
You can also use the **Attachments** palette to open MapleSim templates to perform analysis tasks in Maple, create custom modeling components, and generate data sets for a model. For more information about performing analysis tasks, see *Analyzing and Manipulating a Model (page 125)* in this guide.

2.8 Creating and Managing Custom Libraries

You can create a custom library to save a collection of subsystems, custom modeling components, or attachments that you plan to reuse in multiple files or MapleSim sessions. Custom libraries that you create appear in custom palettes below the **Examples** palette, under the **Libraries** tab, on the left side of the MapleSim window and are saved as .msimlib files on your computer. These custom palettes will appear in the MapleSim window in future MapleSim sessions.

You can add any subsystems or custom modeling components saved in your custom library to models created in future MapleSim sessions; you can also save attachments that you want to keep with the custom library (for example, design documents for the subsystems or custom components in the custom library or other documents that you want to refer to in future MapleSim sessions).

A sample custom palette with a subsystem is shown below.



If you used a third-party tool to create models or model libraries based on the Modelica 3.1 programming language, you can import the .mo files for the models or model libraries into MapleSim as custom libraries. You can then use the imported models and libraries in your MapleSim models as you would use any other modeling components. For more information, see **Using MapleSim → Building a Model → Creating and Managing Custom Libraries → Importing Modelica Models and Libraries** in the MapleSim Help system.

Example: Adding Subsystems and Attachments to a Custom Library

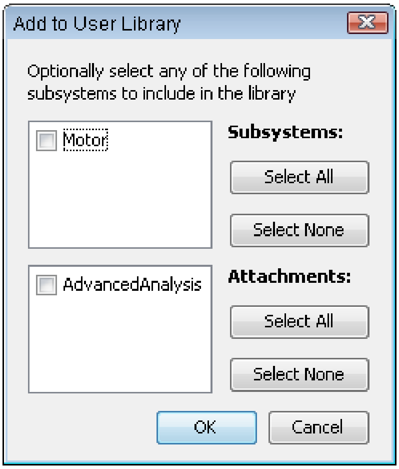
In this example, you will add a subsystem and an .mw attachment to a custom library to make them available in a future MapleSim session.

To add a subsystem or an attachment to a custom library

1. Under the **Libraries** tab, expand the **Examples** palette, expand the **User's Guide** menu, and then open the **Sliding Table** example.
2. From the **File** menu, select **Create Library....**
3. Select a path and specify the file name **Sliding Table.msimlib**.

Note: This file will store the custom library and the file name that you specify will appear as the custom palette name in the MapleSim interface.

4. Click **Save**. The **Add to User Library** dialog box displays all of the subsystems in your model and files attached in the **Attachments** palette.



Add to User Library

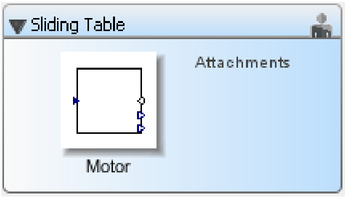
Optionally select any of the following subsystems to include in the library

□ Motor

□ AdvancedAnalysis

Figure 2.12: Add to User Library

5. Select the check box beside **Motor** to add the subsystem to the custom library.
6. Select the check box beside **AdvancedAnalysis.mw** to add the attachment to the custom library.
7. Click **OK**. A new custom library palette is added in the **Libraries** tab on the left side of the MapleSim window.



This palette and its contents appear in the **Libraries** tab. They can be used in a model the next time you start MapleSim.

8. In the **Sliding Table** palette, click **Attachments**. The **Library Attachments** dialog box appears. This dialog box lists all of the attachments added to the custom library.

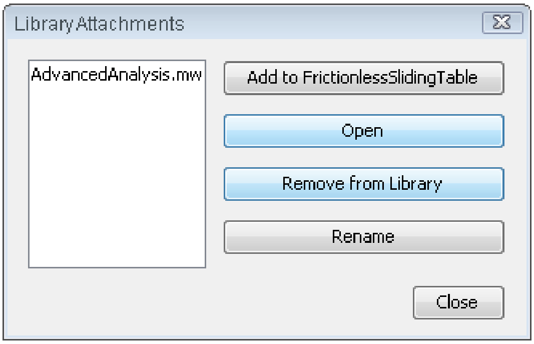


Figure 2.13: Library Attachments

Opening a MapleSim Model in a Maple Worksheet Attachment

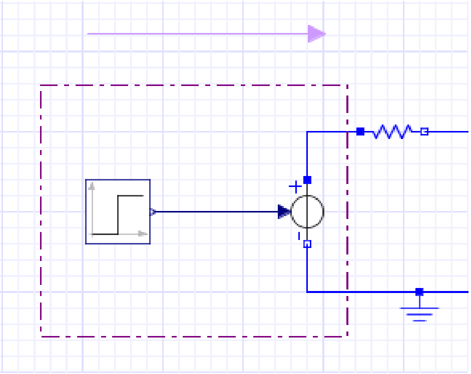
You can also use this dialog box to add attachments to the **Attachments** palette of another model and open attachments in their associated applications. For further instructions on opening your attachments, see **Using MapleSim → Analyzing a Model → Working with Attachments → Viewing an Attachment in its Associated Application** in the MapleSim Help system.

If the attachment you opened is a Maple worksheet (such as the AdvancedAnalysis.mw template in the previous example), you can open saved MapleSim models in the worksheet, giving you programmatic access to the model (see **Using MapleSim → MapleSim Application Programming Interface → API Commands → MapleSim,LinkModel**) in the MapleSim Help system. Opening a MapleSim model in a Maple worksheet requires that your worksheet contain a MapleSim Model embedded component.

For instructions on opening a MapleSim model in a MapleSim embedded component, see **Using MapleSim → Analyzing a Model → Working with Attachments → Opening a MapleSim Model in a MapleSim Model Embedded Component** in the MapleSim Help system.

2.9 Annotating a Model

You can use the tools in the **Model Workspace Toolbar** to draw lines, arrows, and shapes. MapleSim also provides many tools for customizing the colors, line styles, and shape fills.



You can use the text tool (T) in the **Model Workspace Toolbar** to add text annotations to your model. In text annotations, you can enter mathematical text in 2-D math notation and modify the style, color, and font of the text. For more information about 2-D math notation, see *Entering Text in 2-D Math Notation (page 55)*.

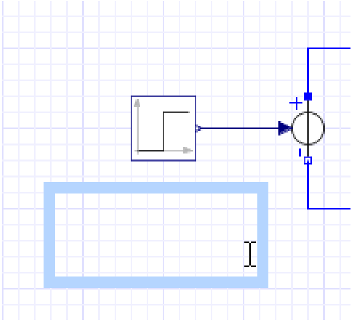
Example: Adding Text Annotation to a Model

To add text annotation to a model

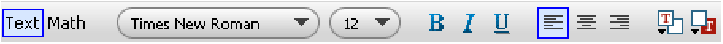
1. Under the **Libraries** tab, expand the **Examples** palette, expand the **User's Guide Examples**, and then open the **Simple DC Motor** example.

2. From the **Model Workspace Toolbar**, click **Text Tool** (T).

3. In the **Model Workspace**, draw a text box for an annotation below the **Step** component.



When you release your left mouse button, the toolbar above the **Model Workspace** switches to the text formatting toolbar.

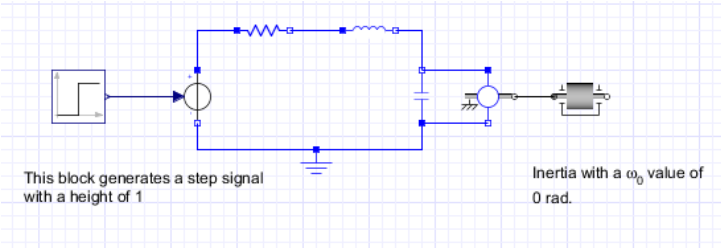


4. Enter the following text: **This block generates a step signal with a height of 1.**

5. Select the text that you entered and change the font to **Arial**.
6. Click anywhere outside of the text box.
7. Draw another text box below the **Inertia** component.
8. Enter the following text: **Inertia with a ω_0 value of 0 rad.**

Tip: To enter the omega character (ω), press **F5** to switch to the 2-D math mode, type **omega**, and then press **Ctrl + Space** (Windows®), **Ctrl + Shift + Space** (Linux®), or **Esc** (Macintosh). To enter the subscript, press **Ctrl + Shift +** the **underscore** key (Windows and Linux) or **Command + Shift +** the **underscore** key (Macintosh) followed by **0**. Press the right arrow key to move the cursor from the subscript position.

9. Click anywhere outside of the text box.
10. Select the text that you entered and change the font to **Arial**.
11. Click anywhere outside of the text box to complete the annotation.



This block generates a step signal with a height of 1

Inertia with a ω_0 value of 0 rad.

2.10 Entering Text in 2-D Math Notation

In parameter values and annotations, you can enter text in 2-D math notation, which is a formatting option for adding mathematical elements such as subscripts, superscripts, and Greek characters. As you enter text in 2-D math notation, you can use the command and symbol completion feature to display a list of possible Maple commands or mathematical symbols that you can insert.

The following table lists common key combinations for 2-D math notation:

Table 2.3: 2-D Mat Notation Key Combinations

Task	Key Combination	Example
Switch between text and 2-D math mode (annotations only)	**F5**	-
Command and symbol completion (parameter values and annotations only)	1. Enter the first few characters of a symbol name, Greek character, or Maple command. 2. Enter the key combination for your platform: • **Ctrl** + **Space** (Windows) • **Ctrl** + **Shift** + **Space** (Linux) • **Esc** (Macintosh) 3. From the menu, select the symbol or command that you want to insert.	-
Enter a subscript for a variable	**Ctrl** (or **Command**) **+ Shift + underscore** (_)	x_{a}
Enter a superscript	**caret** (^)	x^2
Enter a square root (annotations only)	Enter **sqrt** and press **Ctrl** (or **Command** for Macintosh) + **Space**.	$\sqrt{x}$
Enter a root (annotations only)	Enter **nthroot** and press **Ctrl** (or **Command**) + **Space**.	$\sqrt[n]{x}$
Enter a fraction	**forward slash** (/)	$\frac{1}{8}$
Enter a piecewise, matrix, or vector row (annotations only)	**Ctrl** (or **Command**) **+ Shift + R**	$\begin{bmatrix} 3 & 8 \end{bmatrix}$
Enter a table column (annotations only)	**Ctrl** (or **Command**) **+ Shift + C**	$\begin{bmatrix} 6 \end{bmatrix}$

For more information, see the **Using MapleSim → Building a Model → Annotating a Model → Key Combinations for 2-D Math Notation** topic in the MapleSim Help system.

2.11 Creating a Data Set for an Interpolation Table Component

You can create a data set to provide values for an interpolation table component in your model. For example, you can provide custom values for input signals and electrical **Current Table** and **Voltage Table** sources. To create a data set, you can either attach a Microsoft®

Excel® spreadsheet (.xls or .xlsx) or comma-separated values (.csv) file that contains the custom values, or you can create a data set in Maple using the Data Generation Template or Random Data Template provided in the MapleSim templates dialog box.

For more information about interpolation table components, see the **MapleSim Component Library → Signal Blocks → Interpolation Tables → Overview** topic in the MapleSim Help system.

Example: Creating a Data Set in Maple

In this example, you will use the Data Generation Template to create a data set for a MapleSim **1D Lookup Table** component. In this template, you can use any Maple commands to create a data set; however, for demonstration purposes, you will create a data set using a computation that has already been defined.

To create a data set in Maple

1. Open a new MapleSim document.
2. In the **Libraries** tab, expand the **Signal Blocks** palette, and then expand the **Interpolation Tables** menu.
3. Add a **1D Lookup Table** component to the **Model Workspace**.
4. In the **Main Toolbar**, click **Templates** (✎).
5. From the templates list, select **Data Generation**.
6. In the **Attachment** field, enter **My First Data Set** and click **Create Attachment**. The Data Generation Template opens in Maple.
7. To execute the entire worksheet, click (!!!) at the top of the Maple window.
8. At the bottom of the template, in the **Data set name** field, enter **TestDataSet**.
9. To make the data set available in MapleSim, click **Attach Data in MapleSim**n.
10. In MapleSim, under the **Project** tab, expand the **Attachments** palette, and then expand the **Data Sets** category. The data set file appears in the list. You can now assign this data set to the interpolation table component in the **Model Workspace**.
11. In the **Model Workspace**, select the **1D Lookup Table** component.
12. In the **Inspector** tab, from the **data** drop-down menu, select the **TestDataSet.mpld** file. The data set is now assigned to the **1D Lookup Table** component.
13. Save the Data Generation Template in Maple and then save your model in MapleSim.

2.12 Best Practices: Building a Model

This section describes best practices to consider when laying out and building a MapleSim model.

Best Practices: Laying Out and Creating Subsystems

To start building your model, drag components from the palettes to the center of the **Model Workspace**. Drag the components into the arrangement that you want in the **Model Workspace** and then, if necessary, change their orientation so that the components are facing in the direction that you want. When you have established the position and orientation of the components, connect them in the **Model Workspace**.

When grouping components into subsystems, make sure that you include logical component groups that fit on one screen at a time. This will allow you to see all of the subsystem components at a certain level without scrolling.

Create Subsystems for Component Groups That You Plan to Reuse

Create subsystems for component groups that you plan to reuse throughout a diagram or in multiple files. For example, if you plan to include multiple planar link models in a pendulum system, you can create a link subsystem so that multiple copies of that component group could be added. If you wanted to add the link subsystem to another pendulum model, you can create a custom library to use the subsystem in another file.

Create Subsystems for Component Groups That You Plan to Analyze

Make sure that you create subsystems for component groups that you plan to analyze in more depth, test, or translate into source code. Several MapleSim templates allow you to analyze and retrieve equations from particular subsystems. The Code Generation Template allows you to generate source code from subsystems only.

For more information about performing analysis tasks, see *Analyzing and Manipulating a Model (page 125)* in this guide.

Use the Debugging Console to Identify Subsystem Copies and Unconnected Lines

You can display the debugging pane by clicking **Debugging** ( ) at the bottom of the MapleSim window.

The debugging pane displays diagnostic messages that can help you troubleshoot potential errors as you build a model. When you click **Run Diagnostics** (✓) below the debugging pane (or from the **File** menu, select **Check Model**), MapleSim verifies whether your model contains unconnected lines or subsystems that have identical content but are not linked to a subsystem definition. When either of these issues are detected, a message that identifies the subsystem in which the issue is located appears in the debugging console. You can right-click (**Control**-click for Macintosh) the message in the debugging pane to display options that can help you to resolve the issue.

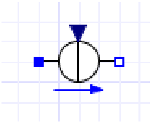
Best Practices: Building Electrical Models

Include a Ground Component in Electrical Circuits

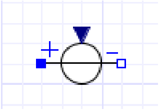
In each electrical circuit model, you must add and connect a **Ground** component to provide a reference for the voltage signals.

Verify the Connections of Current and Voltage Sources

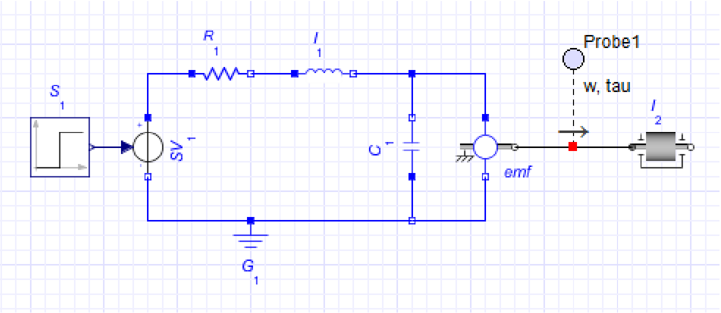
Simulation results can be affected by the way in which a current or voltage source is connected in your model. If you receive unexpected simulation results, verify the connections between electrical sources and other components in your model. All of the current sources in the MapleSim Component Library display an arrow that indicates the direction of the positive current.



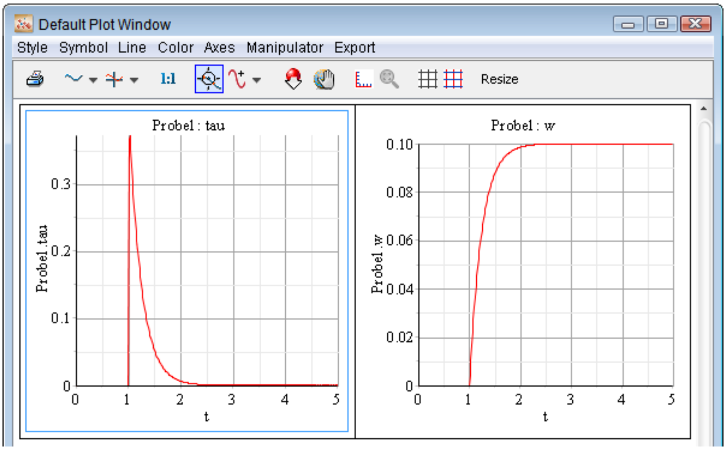
Also, all of the voltage sources display a plus sign indicating the location of the positive voltage and a minus sign indicating the location of the negative voltage.



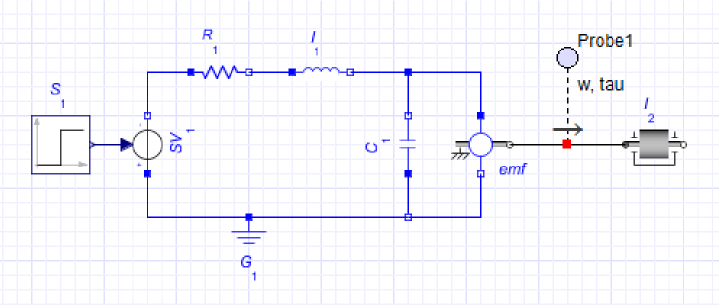
Consider the following **Simple DC Motor** model. Note that the positive port of the **Signal Voltage** source at the left of the diagram is connected to the positive port of the **Resistor** component.



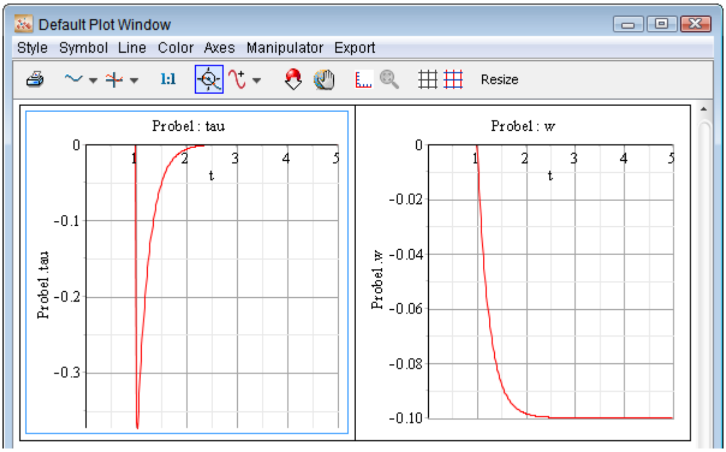
When this model is simulated, MapleSim returns the following results for the torque (tau) and speed (w) quantities.



On the other hand, if the negative port of the **Signal Voltage** source is connected to the positive port of the **Resistor** component, as shown in the following model.



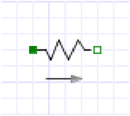
MapleSim returns different results for the speed and torque quantities.



Best Practices: Building 1-D Translational Models

Verify That All Force Arrows Are Pointed in the Same Direction

In MapleSim, all of the 1-D translational mechanical components are defined in a 1-D co-ordinate system with the positive direction defined as the direction of the gray arrow displayed by the component icon.



Any positive forces acting on the model cause the component to move in the direction of the arrow, so make sure that all of the arrows displayed by the 1-D translational mechanical components in your model point in the same direction. As an example, note that all of the force arrows are pointed to the right in the following model.

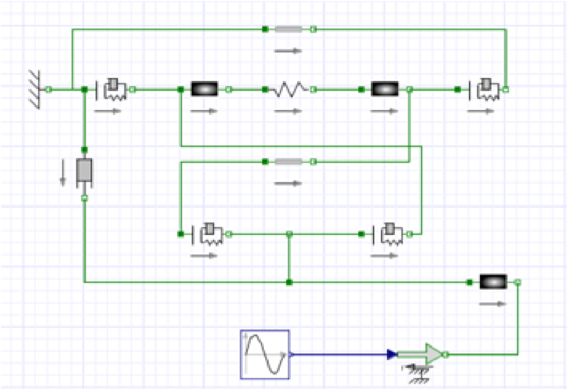


Figure 2.14: Verifying Force Arrows

For an example of sign convention and how arrow direction represents a force acting on the model, under the **Libraries** tab, browse to the **Examples** → **User's Guide Examples** menu, and then open the **Constant Acceleration, Sign Convention** and **Arrow Convention** examples.

Best Practices: Building Multibody Models

Connect the Inboard Port of a Rigid Body Frame to a Center-of-mass Frame

Make sure that you connect the inboard port of any **Rigid Body Frame** components in your model to the center-of-mass frame of a **Rigid Body** component. This ensures that the local reference frame used to describe displacements and rotations for the **Rigid Body Frame** component match with the center-of-mass reference frame defined on the **Rigid Body** component.

In the following planar link example, the **Rigid Body Frame** inboard ports (that is, the ports with the cross-hatched circles) are both connected to a **Rigid Body** component.

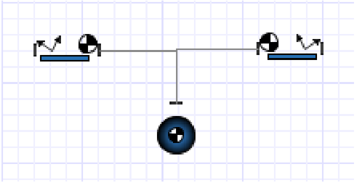


Figure 2.15: Center of Mass Placement Best Practice

Best Practices: Building Hydraulic Models

Define Fluid Properties

When building hydraulic models, you must define the properties of the fluid that will be used by placing the **Hydraulic Fluid Properties** component at the top level of your model or at the same level as a hydraulic subsystem. If you place this component at the top level of your model, all hydraulic components and subsystems in your model will inherit the fluid properties defined by that component instance; if you place the **Hydraulic Fluid Properties** component at the same level as a subsystem, all hydraulic components in that subsystem and all nested subsystems will inherit the properties defined by that component instance.

In the following example, all of the hydraulic components in the model inherit the fluid properties defined by the **Hydraulic Fluid Properties** component at the top-right of the diagram.

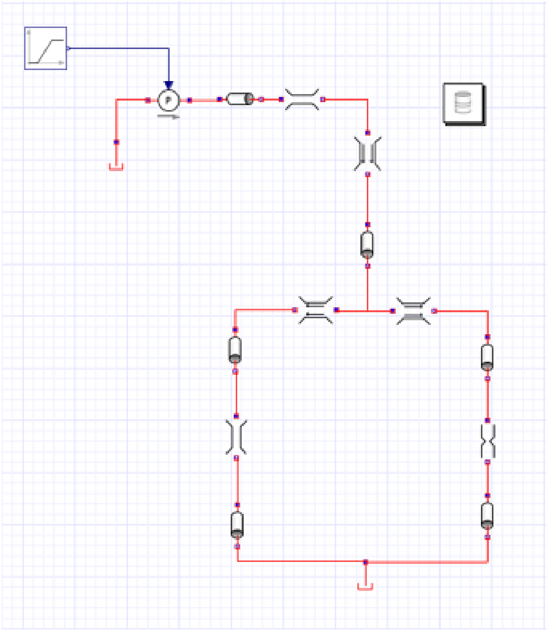


Figure 2.16: Hydraulic Model

For a complete tutorial on how to model hydraulic systems *Tutorial 6: Using the External C Code/DLL Custom Component Template (page 184)* Modeling Hydraulic Systems.

Best Practices: Enforcing Initial Conditions

In complex models, all of the initial conditions might not be independent of each other. In general, use the **enforce** (✓) option to strictly enforce as many initial conditions as you have degrees of freedom in your model. However, you can use the **guess** option (?) for a specified initial condition parameter value to help the solver determine the desired starting configuration for your system faster.

3 Creating Custom Modeling Components

In this chapter:

3.1 Understanding Custom Components

The Custom Component template is an interactive Maple worksheet that lets you quickly create your own specialized MapleSim components without writing code, or having to rearrange your equations for every new model. Creating custom components extends the MapleSim component library, enabling you to create custom modeling components based on the mathematical models that you define. Custom components can use signals, ports with associated physical domains, or a combination of the two. You can also create libraries of custom components and create custom components to contain particular subsystems with specialized functionality.

You can use the **Custom Component Template** to define system parameters and variables, set the level of equation optimization, generate the equations, and then further analyze the resulting equations. The Custom Component templates contain pre-built embedded components that lets you extract, manipulate and analyze the symbolic system equations generated by any MapleSim model. Using various components from the library, you will create models, set initial conditions and component properties, and assign new values to parameters and variables.

For a complete tutorial on how to create domain specific custom components, see *Tutorial 5: Using the Custom Component Template (page 170)* **Using the Custom Component Template**.

There are several different Custom Component templates, each of which can be attached to a model available through the MapleSim Create Attachment templates dialog box, from the **View → Create Attachment** menu.

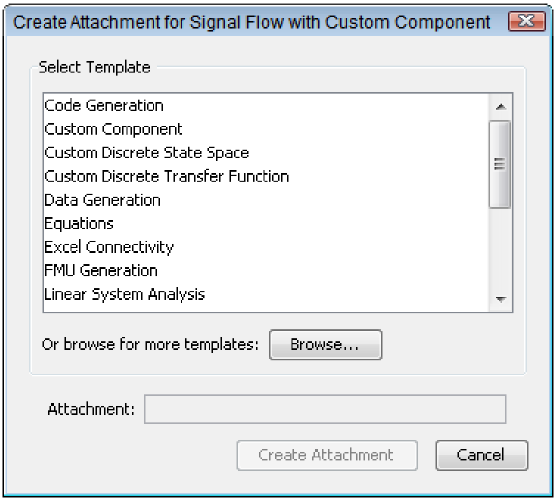


Figure 3.1: MapleSim Templates

Creating a Simple Custom Component

The general process of creating a custom component for a MapleSim model consists of specifying the component equations for the custom component, component parameters and system model, specifying the port types and their values, and generating the component.

To create a custom component

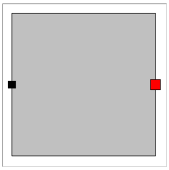
1. Start a new MapleSim model and click **View** → **Create Attachment...**.
2. Select the **Custom Component** template and then press **Create Attachment**. The Maple **Custom Component** template is loaded.
3. Under **Component Description**, change the component name to **Custom**.
4. In the **Component Equations** area, enter the equations, parameters and initial conditions for your custom component. Press **Enter** at each line.

$$eq := \left[s(t) = sa(t) - sb(t), 0 = Fa(t) + Fb(t), Fa(t) = piecewise\left(s(t) < 0, K \cdot s(t) + B \cdot \frac{\mathrm{d}}{\mathrm{d}t} s(t), 0 \right) \right]$$

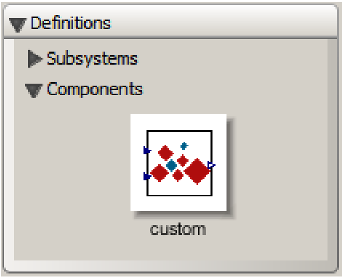
$$params := [\mathrm{K} = 1000, B = 10]$$

$$initialconditions := [\ ]$$

5. Scroll down to **Component Ports** and press **Clear All Ports**. The ports are removed from the component.
6. Add ports to the custom components by pressing **Add Port** as shown below.



7. Provide a name for the component, define the port type and the component parameters.
8. Press **Generate MapleSim Component** to create your component and to bring you back into the MapleSim environment. The custom component now appears in the **Project** → **Definitions** → **Components**.



Typical Uses

The Custom Component template is the most general template and is specifically designed to help you create custom components from algebraic expressions, differential equations or systems of differential-algebraic equations. The Custom Component Template is a collection of pre-built controls and procedures associated specific Maple commands to easily create new MapleSim components. In addition to the Custom Component, the **Custom Discrete State Space** and the **Custom Discrete Transfer Function** templates implement specified transfer functions, and the **Modelica Custom Component** allows for the creation of a custom component via user-provided Modelica code.

Custom Component templates are more than just containers for your equations. You can also access all of Maple's functionality to further develop your equations before you generate your Custom Component for your model. This includes access to Maple's programming language, symbolic algebra functionality, and documentation tools to instantly analyze and verify the behavior of your component.

By using the Custom Component Template you create a custom component in Maple by performing the following tasks:

- Attach a custom component template to your model
- Define and enter your governing equations and properties that determine the behavior of the component (for example, parameters and port variables)
- Specify ports for your component
- Define the associated port variable mappings
- Map variables from your equations to the ports
- Generate the component and make it available in MapleSim
- Test and analyze your mathematical model

The Custom Component Template contains pre-built controls that allow you to perform these tasks with the same validation as for built-in components, preventing invalid connections and parameter values.

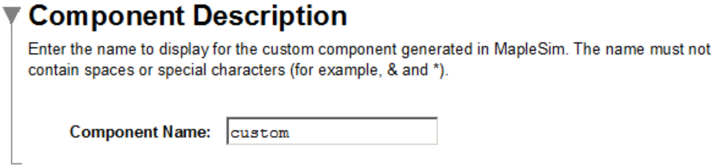
3.2 Using The Custom Component Template

The Custom Component template consists of four main areas, each of which are typically used in the following order when developing a new component.

- Component Description
- Component Equations
- Component Ports
- Component Generation

Component Description Area

Use this section to provide a name for the custom component. No spaces or special characters are allowed.



Component Description

Enter the name to display for the custom component generated in MapleSim. The name must not contain spaces or special characters (for example, & and *).

Component Name: custom

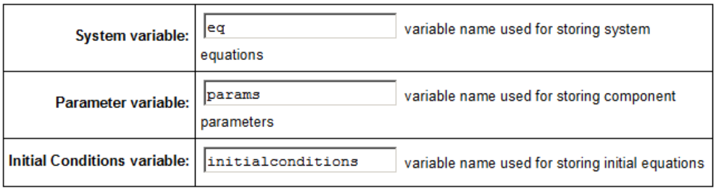
Figure 3.2: Component Description

Component Equations Area

In this section, you provide specific names for the system variables, parameters, initial conditions, and define their equations. Sample equations are presented below the table to let you replace them with your custom equations.

Custom Component Variables

Figure 3.3 shows the default variable names for storing component equations and parameters.



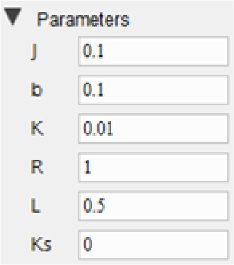
System variable:	eq variable name used for storing system equations
Parameter variable:	params variable name used for storing component parameters
Initial Conditions variable:	initialconditions variable name used for storing initial equations

Figure 3.3: Component Equation Variables

Replace the following default variable names shown in **Figure 3.3** with specific equation names. Use these variable names for defining the system equations in Custom Component Equations Area.

eq is a list of equations. These can be differential equations, algebraic expressions or transfer functions.

params is a list of parameters and their default values. They appear in the Parameter Inspector in MapleSim. For the parameter values specified in **Figure 3.5**, you would see these values in the Parameter Inspector.



▼ Parameters

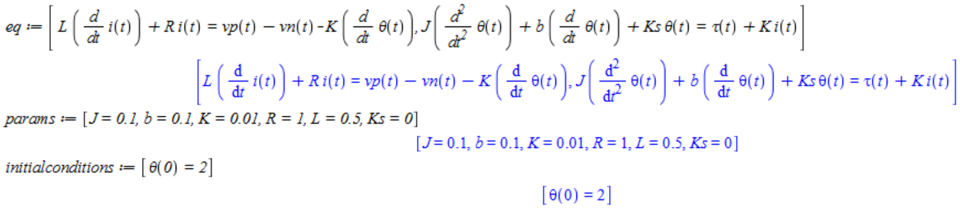
J	0.1
b	0.1
K	0.01
R	1
L	0.5
Ks	0

Figure 3.4: List of Parameters

initialconditions are the initial conditions for any variable in **eq**.

Custom Component Equations Area

In this area you assign equations, define parameters and set initial conditions (see **Figure 3.5**). When replacing the sample equations with specific equations ensure that the equation names match the variable names shown in **Figure 3.3**. Press **Enter** after entering each equation.



$$eq := \left[L\left(\frac{d}{dt}\, i(t)\right) + R\, i(t) = vp(t) - vn(t) - K\left(\frac{d}{dt}\, \theta(t)\right), J\left(\frac{d^2}{dt^2}\, \theta(t)\right) + b\left(\frac{d}{dt}\, \theta(t)\right) + Ks\, \theta(t) = \tau(t) + K\, i(t)\right]$$

$$\left[L\left(\frac{d}{dt}\, i(t)\right) + R\, i(t) = vp(t) - vn(t) - K\left(\frac{d}{dt}\, \theta(t)\right), J\left(\frac{d^2}{dt^2}\, \theta(t)\right) + b\left(\frac{d}{dt}\, \theta(t)\right) + Ks\, \theta(t) = \tau(t) + K\, i(t)\right]$$

$params := [J = 0.1, b = 0.1, K = 0.01, R = 1, L = 0.5, Ks = 0]$

$[J = 0.1, b = 0.1, K = 0.01, R = 1, L = 0.5, Ks = 0]$

$initialconditions := [\theta(0) = 2]$

$[\theta(0) = 2]$

Figure 3.5: Custom Component Equations

Component Ports Area

In this area of the template you customize, define, and assign ports and port components for the defined equations shown in **Figure 3.5**. Since MapleSim supports acausal modeling, you can assign a combination of domain specific input and output variables to the component ports.

In this part of the template, you can:

- Add or remove connector ports, and drag them to the desired position
- Choose the port type (e.g. signal, electrical, hydraulic, etc.)
- Map variables from the equations defined in the Component Equations section to ports

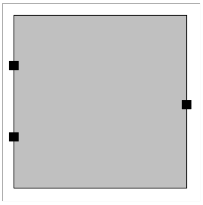


Figure 3.6: Component Ports

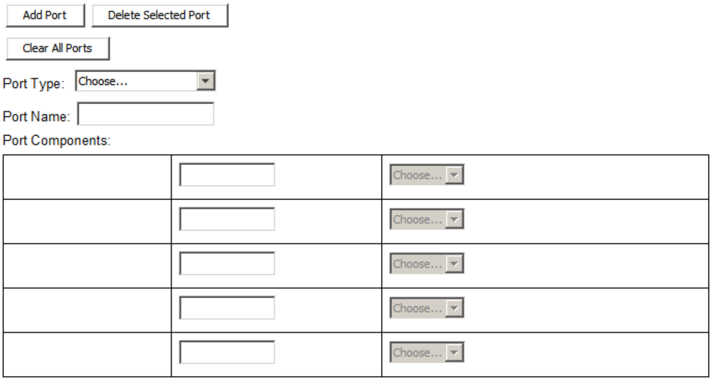


Figure 3.7: Component Port Felds

Use the following template tools shown in **Figure 3.7** to define the custom component and assign values to ports.

Add Port / Delete Selected Port. Add or remove ports in the component diagram.

Clear All Ports. Removes all ports from the component diagram.

Port Type. Defines the domain the port will interact in.

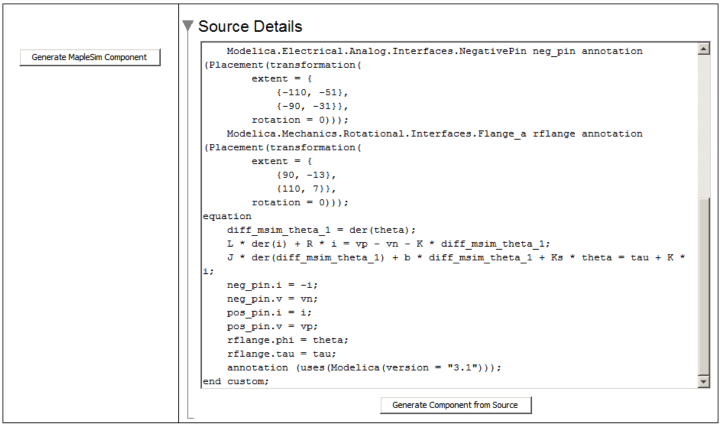
Port Name. Provides a name for the port.

Port Components. Use this menu to assign the input and/or output variables to the port. The drop-down menu lists the input and output variables shown in **Figure 3.3**.

Component Generation Area

The Component Generation area shows equation details.

Generate the MapleSim component by clicking **Generate MapleSim Component** shown in **Figure 3.8**. Expand the Source Details section to view the generated Modelica code. This code may be edited directly, and the component regenerated via the **Generate Component from Source** button.



```
    Modelica.Electrical.Analog.Interfaces.NegativePin neg_pin annotation
(Placement(transformation(
        extent = {
            {-110, -51},
            {-90, -31}},
        rotation = 0)));
    Modelica.Mechanics.Rotational.Interfaces.Flange_a rflange annotation
(Placement(transformation(
        extent = {
            {90, -13},
            {110, 7}},
        rotation = 0)));
equation
    diff_msim_theta_1 = der(theta);
    L * der(i) + R * i = vp - vn - K * diff_msim_theta_1;
    J * der(diff_msim_theta_1) + b * diff_msim_theta_1 + Ks * theta = tau + K *
i;
    neg_pin.i = -i;
    neg_pin.v = vn;
    pos_pin.i = i;
    pos_pin.v = vp;
    rflange.phi = theta;
    rflange.tau = tau;
    annotation (uses(Modelica(version = "3.1")));
end custom;
```

Figure 3.8: Component Generation

In MapleSim, after the component generates, the custom component is available in the **Definitions** palette located under the **Project** tab.

3.3 Creating Custom Components with Signal-Flow Behavior

Custom components simplify model construction by reducing the need to connect many signal-flow components together. This example shows how to create a custom component for a simple signal flow equation.

Creating a Simple Signal-Flow Custom Component

Create a custom component that implements the following equation

$$x(t) = y(t) + z(t)$$

To create a custom component

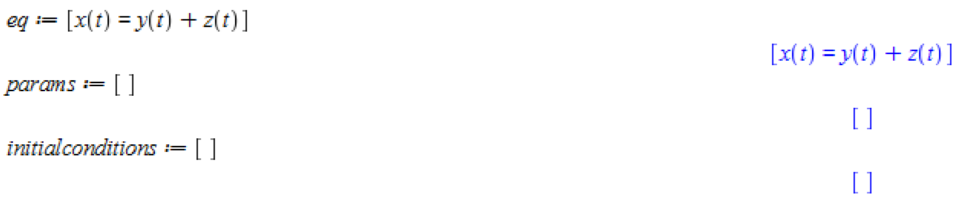
1. Start a new MapleSim model and Click **View** → **Create Attachment...**.
2. Select the **Custom Component** template and click **Create Attachment**.
3. In the **Component Equations** section replace the default equations, parameters and initial conditions that appear with the following values.

$$eq := x(t) = y(t) + z(t)$$

$$params := [\ ]$$

$$initialconditions := [\ ]$$

Pressing **Enter** on each line registers the changes.



$eq := [x(t) = y(t) + z(t)]$

$$[x(t) = y(t) + z(t)]$$

$params := [\,]$

$$[\,]$$

$initialconditions := [\,]$

$$[\,]$$

Figure 3.9: New Equations, Parameters and Initial Conditions

Tip: The equations in the custom component do not have to be rearranged into an explicit form. For example, you could replace the equation with

$$eq := x(t) + \log\left(x(t)^2\right) = y(t) + z(t)$$

for which there is no explicit solution for the output $x(t)$. MapleSim solves for $x(t)$ automatically.

4. In the **Component Ports** section, click **Clear All Ports**.
5. Add three new ports by clicking **Add Port** three times and dragging them into following positions.

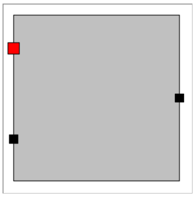
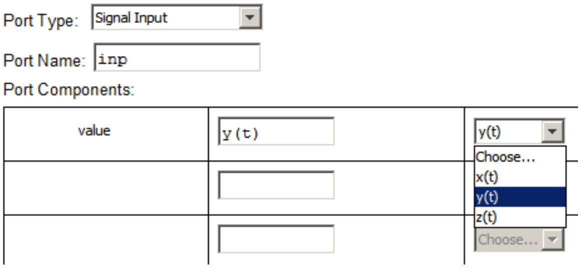


Figure 3.10: Port Mappings

6. Click on the top left port on the left hand side. The black square turns red.
7. From the **Port Type** drop down list, select **Signal Input**. The Port Name is assigned **inp** and **value** appears in the **Port Components** list.



Port Type: Signal Input

Port Name: inp

Port Components:

value | y(t) | y(t)

Figure 3.11: Variable to Port Mapping

8. Select **y(t)** from the **value** dropdown list as shown in **Figure 3.11**.
9. Assign the remaining port mappings using the settings in **Table 3.1**.

Table 3.1: Port Map

Port Name	Port Type	Port Component
inp	Signal Input Value	$y(t)$
inpO	Signal Input Value	$z(t)$
out	Signal Output Value	$x(t)$

10. Under the **Component Generation** section, click **Generate MapleSim Component**. The custom component equations are generated and assigned to the model. The custom component icon appears in MapleSim under the **Project** tab, in the **Definitions** → **Components** palette.

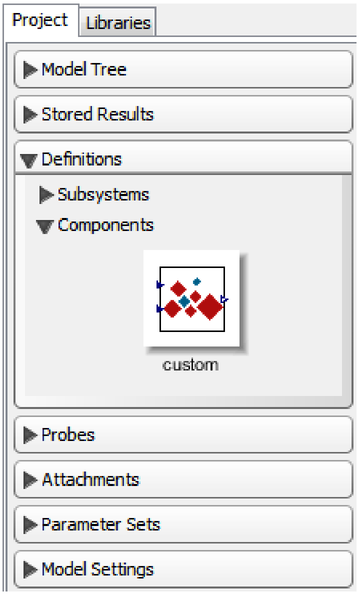


Figure 3.12: Complete Custom Component

11. In the following model, drag the custom component into the workspace from the **Components** palette, connect two signal sources to the input ports on the left, and then place a probe on the right-most port (right-click and select **Add Probe**) using the components from **Table 3.2**.

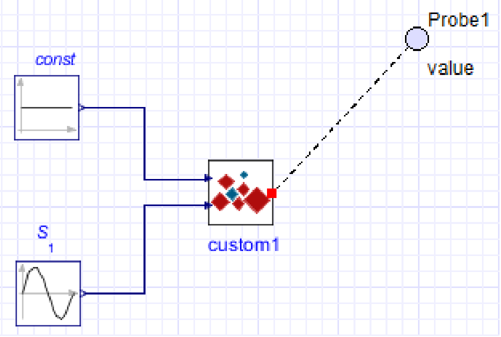
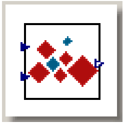
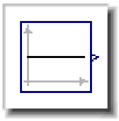
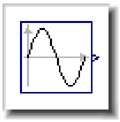
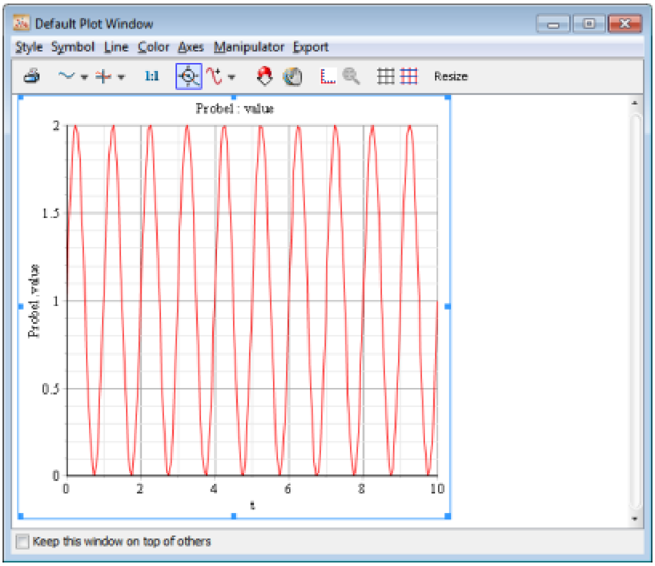


Figure 3.13: Completed Custom Component Model

Table 3.2: Signal Flow Components

Component	Number of Components	Symbol	Library Location	Required Settings
Custom Component	1		Project → Definitions → Components	Custom settings
Constant	1		Signal Blocks → Sources → Real	Use default settings
Sine Source	1		Signal Blocks → Sources → Real	Use default settings

12. Click **Run Simulation** (▶) in the **Main Toolbar**. When the simulation is complete, the following graph appears.


Default Plot Window
Style Symbol Line Color Axes Manipulator Export
1:1
Resize
Probe1 : value
2
1.5
1
0.5
0
Probe1.value
0
2
4
6
8
10
t
Keep this window on top of others

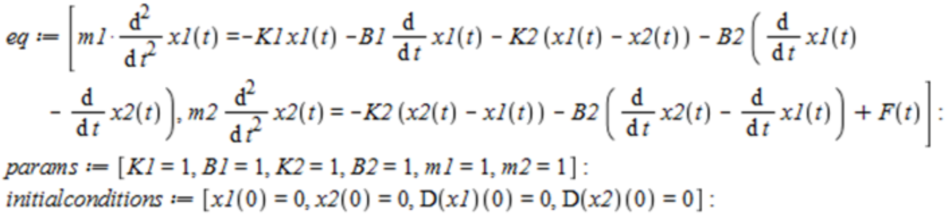
Advantages of Acausal Mapping

Even though the custom component ports are specified as Signal Inputs and Signal Outputs, MapleSim is truly acausal; signals can be inputs or outputs regardless of the pin type. For example, if $x(t)$ and $z(t)$ were specified, and a probe was placed on $y(t)$, MapleSim would automatically rearrange the specified equation into $y(t) = x(t) + z(t)$.

The concept of signal inputs and outputs are necessary for the code generation capabilities of MapleSim, since the code is 'causalized', MapleSim expects inputs and provides outputs.

Using Differential Equations in Custom Components

Instead of using library components for you model, you can also use differential equations to define your custom component. For example, **Figure 3.14** shows the equations that describe the motion of two coupled mass-spring-dampers subjected to a driving force.



$$eq := \left[m1 \cdot \frac{d^2}{dt^2} x1(t) = -K1\, x1(t) - B1 \frac{d}{dt} x1(t) - K2\,(x1(t) - x2(t)) - B2 \left(\frac{d}{dt} x1(t) - \frac{d}{dt} x2(t) \right), m2 \frac{d^2}{dt^2} x2(t) = -K2\,(x2(t) - x1(t)) - B2 \left(\frac{d}{dt} x2(t) - \frac{d}{dt} x1(t) \right) + F(t) \right]:$$

$$params := [K1 = 1, B1 = 1, K2 = 1, B2 = 1, m1 = 1, m2 = 1]:$$

$$initialconditions := [x1(0) = 0, x2(0) = 0, \mathrm{D}(x1)(0) = 0, \mathrm{D}(x2)(0) = 0]:$$

Figure 3.14: Double Mass-Spring-Damper Equations

Figure 3.15 shows how the parameters are mapped to component ports:

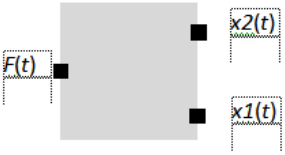


Figure 3.15: Port Mapping for Double Mass-Spring-Damper

3.4 Creating Custom Components with Physical Connections

When you create custom components based on physical connections, each connection port has two variables associated with it: the *across variable* and the *through variable*. The through variable represents a flow of a conserved quantity (such as heat, mass, current, force or torque) and the across variable represents the driving force in a system (temperature difference, pressure difference, voltage drop, velocity or relative angular velocity).

Table 3.3: Characteristics of Through and Across Variables

Characteristics of Through Variables	Characteristics of Across Variables
Conserved quantity (like heat or mass) Has a direction of flow Satisfies the relationship input = output + accumulation Uniform across a domain	Drives the flow of the conserved quantity Is a scalar Defined as the difference between two points within a physical domain

Table 3.4 shows the mathematical relationships defining the connection between various across and through variables.

Table 3.4: Through and Across Variable Mathematical Relationship

Domain	Governing Equation	Through Variable	Across Variable
Ohm's Law Electrical Domain	$I = \frac{V}{R}$	I	V
Hagen–Poiseuille equation Hydraulic Domain	$\frac{dm}{dt} = \frac{\pi D^4 \rho}{128 L \mu} \cdot P$	$\frac{dm}{dt}$	P
Fourier's law Thermal Domain	$\frac{dQ}{dt} = h A T$	$\frac{dQ}{dt}$	T

Deriving the System Equations for a Resistor

Table 3.5 shows a model of a simple resistor with several variables and one parameter.

Table 3.5: Resistor Variables and Parameters

Variable	Parameter	Description
i(t)		Current
v(t)		Voltage difference
vLeft(t)		Voltage on the left port
vRight(t)		Voltage on the right port
	R	Resistance

Ohm's Law defines the relationship between the voltage and the current as

$$v(t) = V_{right(t)} - V_{left(t)}$$

$$v(t) = i(t) \cdot R$$

Figure 3.16 shows the equations mapped to the custom component ports.

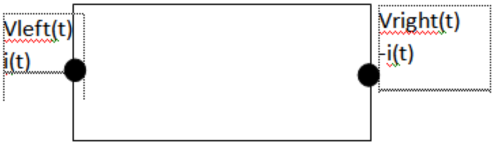


Figure 3.16: Resistor Port Mapping

The current i(t) on the right port has a negative sign, representing flow out of the resistor. The current on the left port is positive, representing flow into the resistor. The resistance (**R**) is defined as a parameter available in the Parameter Inspector.

3.5 Working with Custom Components in MapleSim

In MapleSim, you can work with a custom component in the same way as you would work with a subsystem. You can perform the following tasks:

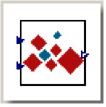
- Add Text and Pictures to a Custom Component
- Save a Custom Component as Part of the Current Model
- Add a Custom Component to a Custom Library
- Edit a Custom Component
- Open Custom Component Examples

Add Text and Pictures to a Custom Component

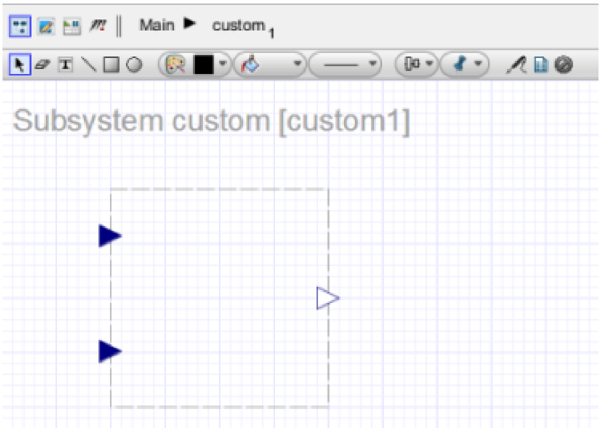
To customize the appearance of a custom component, you can change the default custom component icon. Select the custom component in the **Model Workspace**, click **Icon** (✎) in the **Navigation Toolbar**, and use the drawing and annotation tools to add text and illustrations, and to import a custom picture.

To adding a picture to a Custom Component

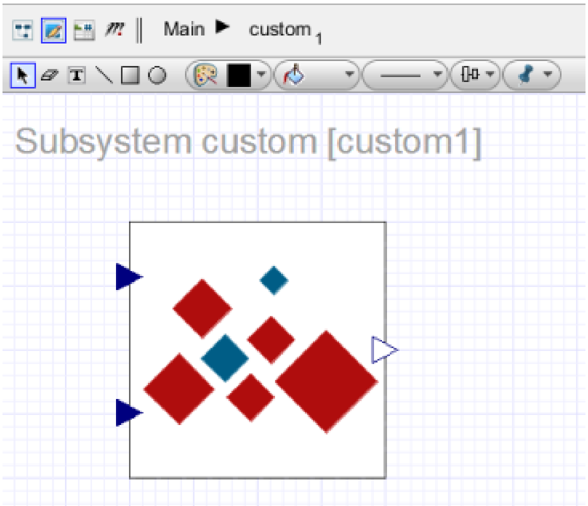
1. Create a custom component and drag it onto the MapleSim workspace.



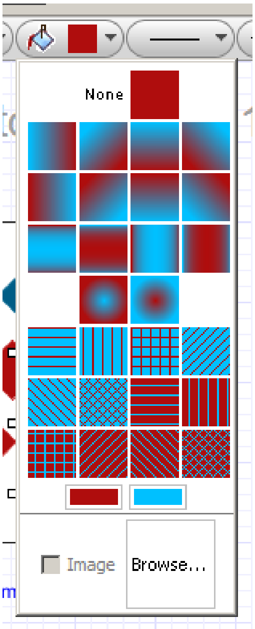
2. Right-click on the custom component and select **Open Component**. The component opens in the workspace showing the port connections.



3. In the navigation toolbar, click **Icon** (). The default image appears inside the component boundary.



4. Click on the image and then click on the **Drawing Fill** drop-down menu (). A list of images appears displaying the Image Browse selection at the bottom.



5. Click **Browse...**. The **Load Image** window appears.
6. Browse to and select an icon image and then click **OK**.
7. In the navigation toolbar, click **Main** to browse to the top level of your model. Your custom component (and all other instances of the same component) will reflect the new icon.

Save a Custom Component as Part of the Current Model

To save a custom component as a part of the current model, add the component by dragging it into the **Model Workspace** and then save the model. The next time you open the file, the custom component appears in the **Model Workspace** and **Definitions** palette.

Add a Custom Component to a Custom Library

If you want to use a custom component in a file other than the current model, add the component to a custom library. For more information, see *Creating and Managing Custom Libraries (page 50)*.

Edit a Custom Component

If you want to edit a custom component that you have generated, make your changes in the corresponding Maple worksheet and regenerate the component.

To edit a custom component

1. In the MapleSim **Model Workspace**, double-click the custom component that you want to edit. The corresponding Custom Component Template opens in Maple.
2. In the Maple worksheet, edit the equations, properties, or port values.
3. At the bottom of the worksheet, click **Generate MapleSim Component**. Your changes are generated in the custom component displayed in MapleSim.
4. Save your changes in the .mw file and the .msim file to which you added the custom component.

Open Custom Component in Examples Palette

The following custom component examples are available with your MapleSim installation in the Examples palette:

- Custom component defined with an algebraic equation
- A simple DC motor component defined with a differential equation
- A simple nonlinear spring-damper component
- Custom component defined with a transfer function

To open an example

1. In MapleSim, click **Templates** (✎) in the **Main Toolbar**.
2. Click **Browse...**.
3. In the dialog box, open the **Component Templates** folder.
4. Select the example that you want to open, and click **Use Template**.
5. (Optional) In the **Attachment** field, enter a name for the template.
6. Click **Create Attachment**. The sample Custom Component Template opens in Maple.

3.6 Example: Creating a Nonlinear Spring-Damper Custom Component

In this example, you will use the Custom Component Template to create a nonlinear spring-damper custom component. The equations defined in this example are based on the **Translational Spring Damper** component in MapleSim. In this case, the stiffness and damping coefficients are replaced with input functions to the component.

To obtain the governing relationships, you can start with a free-body diagram. The diagram for the spring-damper system is shown in the following figure.

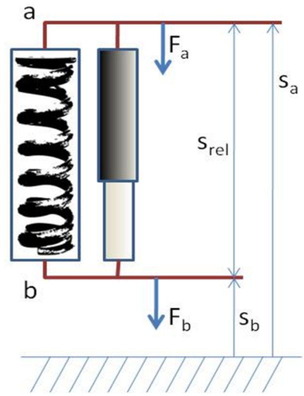


Figure 3.17: Nonlinear Spring-Damper Custom Component

The end points, *a* and *b*, can be defined as the ports for the component; the equations are derived relative to these ports. Therefore, the general equation of motion,

$$d\cdot \frac{\mathrm{d}}{\mathrm{d}t}\, s_{rel}(t) + c\cdot s_{rel}(t) = F(t)$$

where *d* is the damping coefficient, *c* is the stiffness of the spring, and s_{rel} is the relative displacement between the two ports s_a and s_b, can be written as

$$s_{rel}(t) = s_b(t) - s_a(t)$$

Also, an examination of the net force on the system shows that $F(t) = F_b(t)$, where

$$F_a(t) + F_b(t) = 0$$

All of the above relationships are required to define the system behavior.

Opening the Custom Component Template

The Custom Component template is part of the MapleSim templates accessed fromthe **Main Toolbar**.

To open the Custom Component Template

1. In MapleSim, open the model to which you want to add the custom component.
2. Click **Templates** ( ) in the **Main Toolbar**.
3. In the **Select Template** list, select **Custom Component**.

4. In the **Attachment** field, enter **Nonlinear Spring-Damper** as the name for the template and click **Create Attachment**. The Custom Component Template opens in Maple.

Defining the Component Name and Equations

You can now specify the name that will appear for the component in the MapleSim interface and define the equations.

To define the custom component

1. In the Component Description section of the template, specify a component name called **NonLinearSpringDamper**.
2. In the **Component Equations** section, delete the default equations below the table that defines the variables.
3. To define the nonlinear system, enter the following equations.

```
> eq := [d(t)*(diff(s[rel](t),t)) + c(t)*s[rel](t) = F(t), s[rel](t)
      = s[b](t) - s[a](t), v[rel](t) = diff(s[rel](t),t), F(t) = F[b](t),
       F[a](t) + F[b](t) = 0];

> params := [ ]:

> initialconditions := [ ]:
```

Note that the equations are entered in a Maple list. The constants, *d* (damping) and *c* (stiffness) are replaced by the functions *d*(*t*) and *c*(*t*) to define them as input states to the system.

4. Click !!! at the top of the window to execute the entire worksheet.

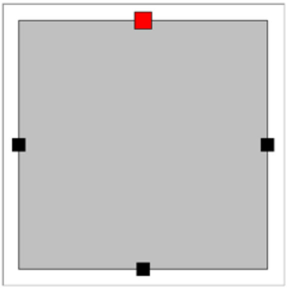
You can now assign these input and output variables to ports that you will include in your generated custom component.

Defining Component Ports

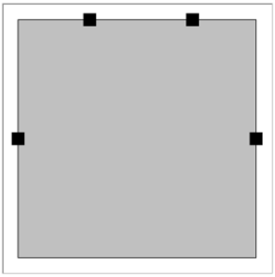
In the **Component Ports** section of the template, you assign input and output variables to ports that will appear in the generated component, and specify the layout of these ports.

To define the component port

1. To remove the sample ports from the diagram, click **Clear All Ports**.
2. Click **Add Port** four times. Four squares, which represent the ports that you will lay out and define, appear in the diagram.



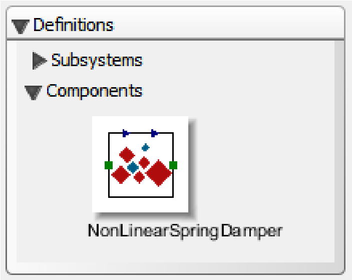
3. Select the port on the left side of the diagram.
4. From the **Port Type** drop-down menu below the diagram, select **Translational Flange**.
5. In the **Port Components** table, in the **Position** row, select **s[b](t)** from the drop-down menu and, in the **Force** row, select **F[b](t)** from the drop-down menu. The left port is now defined as a translational flange and associated with the position variable **s[b](t)** and force variable **F[b](t)**.
6. Select the port on the right side of the diagram.
7. From the **Port Type** drop-down menu, select **Translational Flange**.
8. In the **Position** row, select **s[a](t)** from the drop-down menu and, in the **Force** row, select **F[a](t)** from the drop-down menu. The right port is now defined as a translational flange and associated with the position variable **s[a](t)** and force variable **F[a](t)**.
9. Select the port at the top of the diagram.
10. From the **Port Type** drop-down menu, select **Signal Input**.
11. In the **Value** row, select **c(t)** from the drop-down menu. This port is now defined as a signal input and associated with the stiffness variable **c(t)**.
12. Select the port at the bottom of the diagram.
13. From the **Port Type** drop-down menu, select **Signal Input**.
14. In the **Value** row, select **d(t)** from the drop-down menu. This port is now defined as a signal input and associated with the damping variable **d(t)**.
15. Drag the port that you defined in step 14 and place it at the top right of the diagram. You can also drag the other port to position it.



The ports will be displayed in this arrangement when you generate the custom component in MapleSim.

Generating the Custom Component

To generate the custom component, click **Generate MapleSim Component** at the bottom of the template. To find the generated custom component in MapleSim, select the **Project** tab, expand the **Definitions** palette, and then expand the **Components** submenu.



You can add the custom component to a model by dragging it into the **Model Workspace**.

4 Simulating and Visualizing a Model

In this chapter:

4.1 How MapleSim Simulates a Model

Modelica Description

The equations for many components in the MapleSim library are described using the Modelica physical modeling language. On the other hand, the equations for multibody components are generated by a special-purpose engine, which uses advanced mathematical techniques to ensure that the equations are as concise and efficient as possible, and then converted to Modelica.

For more information about Modelica, visit **http://www.modelica.org**

Model Description

Each component in your model contains a system of equations that describes its behavior; these systems of equations can consist of purely algebraic equations or differential equations. Also, a component may define any number of events, which can change the component behavior during a simulation by enabling or disabling part of the equations in the system or changing state values. Connections between two or more components generate additional equations that describe how these components interact.

System Equations

The topology equations (how the components interact) as well as the terminal (component) equations are then collected into one large system and parameter values are also substituted in. Now, the MapleSim simulation engine has a potentially large system of hybrid differential algebraic equations. This means that the system has differential equations with algebraic constraints, as well as discrete events.

Simplified Equations

A process called *index reduction* reduces the algebraic constraints as much as possible. At the core of this phase is an algorithm that constructs an index-1 DAE system, modified with other symbolic simplification techniques, to reduce the number of equations and variables. Many of these techniques deal with handling of hybrid systems.

You can set initial values for some of the variables by specifying parameter values for certain components in the **Inspector** tab on the right side of the MapleSim window. If the specified initial conditions are not consistent, an error will be detected during the simulation.

Integration and Event Handling

When all of these preprocessing steps are complete, the integration and event handling process begins. Based on the solver type you chose, a sophisticated DAE solver numerically integrates the system of equations. For the variable solver, algebraic constraints are constantly monitored to avoid constraint drift, which would otherwise affect the solution accuracy. For fixed solver type. algebraic constraints are monitored at each fixed time step.

During integration, inequality conditions that are part of the model are monitored and an event is triggered when one or more of these conditions change. Whenever such an event is encountered, the numeric solver stops and the simulation engine computes a new configuration of the system of equations based on the event conditions. This step also involves recomputing initial conditions for the new system configuration. The solver is then restarted and continues to numerically solve the system until another event is triggered or the simulation end time is reached.

Note: Event handling occurs for both variable and fixed step solvers. The difference is that for fixed step solvers, events are only processed at fixed time steps, whereas with a variable solver, the solver will adjust the time step so that events are processed at exactly the time they occur during the integration.

Simulation Results

In the last step of the simulation process, the results are generated and displayed using graphs showing the quantities of interest and, optionally for multibody mechanical systems, a 3-D animation.

The simulation process is summarized in the following chart:

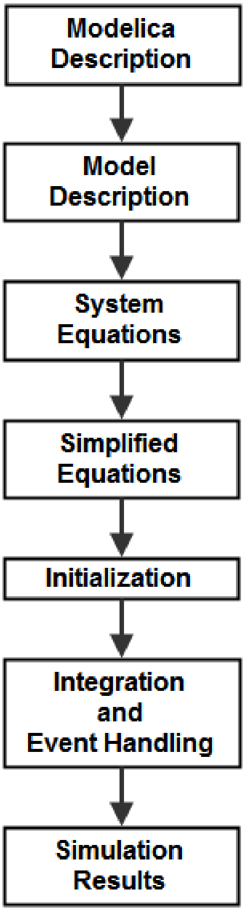


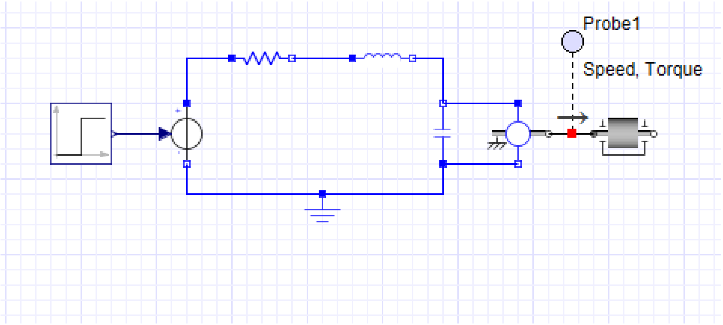
Figure 4.1: Simulation Process

Note that the information in this section is a simplified description of the simulation process. For more information on the DAE solvers used by the simulation engine, see the **dsolve,numeric** topic in the Maple Help system.

4.2 Simulating a Model

To view the behavior or response of physical properties (for example, current or voltage), add probes to connection lines, ports, or components in your 2-D or 3-D model. In MapleSim, probes allow you to identify the variables of interest that are associated with connection ports.

If you add a probe to measure a through variable, an arrow appears to indicate the direction of the positive flow in the **Model Workspace**.



You can specify the simulation duration, the type of solver to use, and other parameter values for the solver, simulation engine, and **3-D Workspace**. After running a simulation, a graph appears for each specified quantity.

You can change the original probe or parameter values and run another simulation to compare the results.

For an example of sign convention and how arrow direction represents a force acting on the model, under the **Libraries** tab, browse to the **Examples → User's Guide Examples** menu, and then open the **Constant Acceleration, Sign Convention** and **Arrow Convention** examples.

Simulation and Advanced Simulation Settings

The parameters used for your simulation are found under the **Settings** tab in the Parameters pane. From here you can access the **Simulation** and **Advanced Simulation** settings sections.

For a description of the Multibody and 3-D Visualization settings see *3-D Visualization and Multibody Settings (page 103)*.

Simulation Settings

In the **Simulation** section you can specify the simulation duration time, the number of plot points, the solver, and other parameters specific to the solver. See **Table 4.1** for a listing and description of the parameters available in the **Simulation** settings section.

Table 4.1: Simulation Settings

Parameter	Default	Description
t_d	$10s$	The duration time of the simulation. You can specify any positive value, including floating-point values. **Note:** The duration time is not the same as the end time of your simulation. The end time for the simulation is given by $t_d + t_s$, where t_s is the start time for the simulation (see **Table 4.2**).
Solver Type	Variable	The type of solver to use for the simulation. • **Variable**: use a variable time step to maintain error tolerances. • **Fixed**: use a fixed time step and disregard integration error. **Note:** The fixed step solvers are identical to those used by MapleSim's exported code.
Solver	Variable: CK45 (non-stiff) Fixed: Euler	DAE solver used during the simulation. The following choices are available when **Solver Type** is set to **Variable**. • **CK45 (semi-stiff):** use a semi-stiff DAE solver (ck45 method). • **RKF45 (non-stiff):** use a non-stiff DAE solver (rkf45 method). • **Rosenbrock (stiff):** use a stiff DAE solver (Rosenbrock method). If your model is complex, you may want to use a stiff DAE solver to reduce the time required to simulate a model. The following choices are available when **Solver Type** is set to **Fixed**. • **Euler**: use a forward Euler solver. • **Implicit Euler**: use an implicit Euler solver (suitable for stiff systems). • **RK2**: use a second-order Runge-Kutta solver. • **RK3**: use a third-order Runge-Kutta solver. • **RK4**: use a fourth-order Runge-Kutta solver.
ϵ_{abs}	$1 \cdot 10^{-7}$	The limit on the absolute error tolerance for a successful integration step if you are using a variable solver to run the simulation. You can specify a floating-point value for this option.

Parameter	Default	Description
ϵ_{rel}	$1 \cdot 10^{-7}$	The limit on the relative error tolerance for a successful integration step if you are using a variable solver to run the simulation. You can specify a floating-point value for this option.
Step size	0.0010	Uniform size of the sampling periods if you are using a fixed-step solver to run the simulation. You can specify a floating-point value for this option.
Plot Points	1000	Minimum number of points to be plotted in a simulation. The data points are distributed evenly according to the simulation duration value. You can specify a positive integer. Additional points can be added for events (see Plot Events in **Table 4.2**). The simulation engine uses a 'Plot points' value which is a maximum of: • the 'Plot points' setting or • '3D Sample Rate' [fps] * simulation duration (if ‘3-D Animation’ is enabled), or • '3D Sample Rate' [fps] * '3-D Playback Time' (if ‘3-D Animation’ is enabled and '3-D Playback Time' is specified) **Note:** This option allows you to specify the number of points for display purposes only. The actual number of points used during the simulation may differ from the number of points in the simulation graph.

Advanced Simulation Settings

In the **Advanced Simulation** section you can specify the simulation start time, a state snapshot to use, compilation options, and other settings. Some of these settings are specific to the solver type (variable or fixed) selected in the **Simulation** settings. See **Table 4.2** for a listing and description of the parameters available in the **Advanced Simulation** settings.

Table 4.2: Advanced Simulation Settings

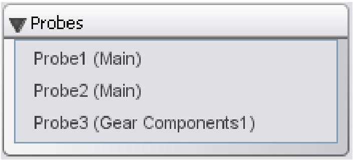
Parameter	Default	Solver Type	Description
t_s	0	All	The simulation start time. You can specify any floating point value, including negative values. **Note:** The simulation start time affects the end time of your simulation, but not the duration time for the simulation, t_d. The end time for the simulation is given by $t_d + t_s$.
Use Snapshot	None	All	A snapshot captures the state of your simulation at a specific time. If you use a snapshot in your simulation, you can override the initial conditions used in your model and replace them with the state your model was in at the time of the snapshot. See *Managing Simulation Results and Snapshots (page 96)* for more information on snapshots.
Baumgarte	☐	All	Apply Baumgarte constraint stabilization to your model. Select to enter values for the derivative gain (alpha) and the proportional gain (beta) that are appropriate for your model.
Jacobian	Symbolic	All	Choose between a symbolic or numeric approximation to the system Jacobian. A symbolic formulation results in faster and more accurate simulations but can take longer to formulate. **Note:** A numeric formulation can only used with stiff solvers (Rosenbrock or Implicit Euler).
Projection	☑	All	Apply constraint projection to your model. Select to project back the solution found at each step of the simulation to the constraint manifold. The projection ends when either the maximum number of Projection Iterations is reached or the defection falls below the Projection Tolerance.
Projection Iterations	50	All	The maximum number of constraint projection iterations. **Note**: This parameter is only available when **Projection** is selected.
Projection Tolerance	0.000010	Fixed	The tolerance value at which the projection iterations are terminated. You can specify any positive floating point value.

Parameter	**Default**	**Solver Type**	**Description**
Event Projection	☑	Fixed	Select to have constraint projection occur during event iterations, but with slower integration results. Not selecting, may cause the simulation to fail if the event changes the solution to the point of not allowing the application of constraint projection at the next step.
Event Iterations	100	All	The maximum number of event iterations allowed before the integrator throws an error. You can specify any positive integer value.
Event Hysteresis	$1.0\ 10^{-7}$	Fixed	The width of the event hysteresis band. You can specify a floating point value greater than or equal to zero. If set to zero, this parameter is disabled.
Index 1 Tolerance	1.0	Variable	Controls the relative error on algebraic variables compared to differential variables. For example, a value of 10 means that algebraic variables can have 10 times the error of differential variables.
Initial Hysteresis	$1.0\ 10^{-10}$	Variable	The width of the event hysteresis for all event triggers at the start of the simulation. You can specify a floating point value greater than or equal to zero.
Scaling	None	Variable	Specifies the method of variable scaling to apply to the system. The available choices are: • None: do not apply scaling • Minimum: use the minimum nominal value • Maximum: use the maximum nominal value • Geometric: use the geometric mean of the nominal values
Minimize Events	□	Variable	This option specifies whether or not to use heuristics to reduce the number of events encountered during your simulation. When selected, the mapping of piecewise transitions into events does not occur.
Plot Events	☑	Variable	Specifies whether or not to include extra plot points at event points during the simulation.
Compiler	☑	Variable	Specifies whether a native C compiler is used during the simulation. When this option is selected, Maple procedures generated by the simulation engine are translated to C code, which is compiled by an external C compiler. If your model is complex, you may want to select this option to reduce the time required to run a simulation.

Parameter	Default	Solver Type	Description
Compile Optimized	☑	All	Optimize the code during compilation. If this parameter is not selected, compile time will be reduced but your simulation will take longer to run.

Editing Probe Values

In the **Project** tab, the **Probes** palette lists all of the probes that you have added to the current MapleSim model.



Probes

Probe1 (Main)

Probe2 (Main)

Probe3 (Gear Components1)

If a probe is attached at the top level of your model, **Main** appears in parentheses beside the probe name; otherwise, the subsystem for the attached probe appears beside the probe name. In the image shown above, three probes have been attached to a model: **Probe1** and **Probe2** at the top level of the model and **Probe3** in a subsystem called **Gear Components1**.

You can click the entries in this palette to browse to a probe in the **Model Workspace**, and view and edit the probe values in the **Inspector** tab. You can also right-click (**Control**-click for Macintosh) entries in this palette and manipulate probes using context menus.

For more information, see **Using MapleSim → Simulating a Model → Using Probes → Editing Probe Values** in the MapleSim Help system.

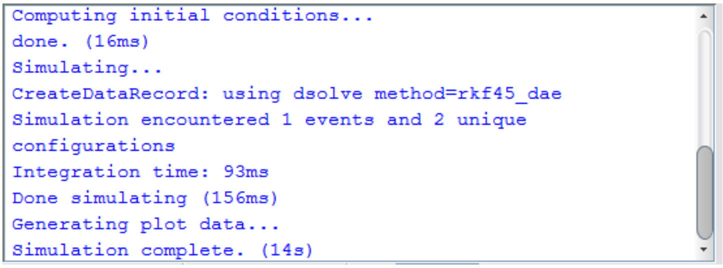
Storing Parameter Sets to Compare Simulation Results

You can store a group of parameter values that are assigned to a model in a parameter set. You can then run a simulation using one parameter set, replace those parameter values with another parameter set, and run another simulation to compare the results.

For more information, see the **Using MapleSim → Building a Model → Storing and Applying Parameter Sets** section in the MapleSim Help system.

4.3 Simulation Progress Messages

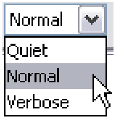
During a simulation, you can view progress messages in the **Console** pane located below the **Model Workspace**. These messages indicate the status of the MapleSim engine as it generates a mathematical model; these messages can help you to debug simulation errors.



```
Computing initial conditions...
done. (16ms)
Simulating...
CreateDataRecord: using dsolve method=rkf45_dae
Simulation encountered 1 events and 2 unique
configurations
Integration time: 93ms
Done simulating (156ms)
Generating plot data...
Simulation complete. (14s)
```

Figure 4.2: Simulation Results Progress Messages

Optionally, before running a simulation, you can specify the amount of detail in progress messages by clicking **Message Console** (☰) at the bottom of the MapleSim window and selecting a level from the drop-down menu.



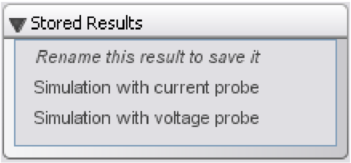
To clear the messages from the console, click **Clear Console** (X).

4.4 Managing Simulation Results and Snapshots

The **Stored Results** palette under the **Project** tab allows you to view, save, and export results generated from multiple simulations. In addition, for each of your stored results, you can save and export snapshots that record all of the state variables for your model at a particular time during the simulation.

Storing and Exporting Results

Whenever you simulate a model, a **Rename this result to save it** entry is added to the **Stored Results** palette. This entry contains all the graphs, progress messages, and (if applicable) the 3-D animation generated from your simulation. However, each new simulation overwrites the results stored in the **Rename this result to save it** entry. To store a simulation result, right-click (**Control-**click for Macintosh) on the **Rename this result to save it** entry, select **Rename** from the context menu, and then provide a name for the stored result. This creates a stored result entry with the name you provided. You can save simulation results to compare and refer to multiple graphs generated during the current MapleSim session.



Stored Results

Rename this result to save it

Simulation with current probe

Simulation with voltage probe

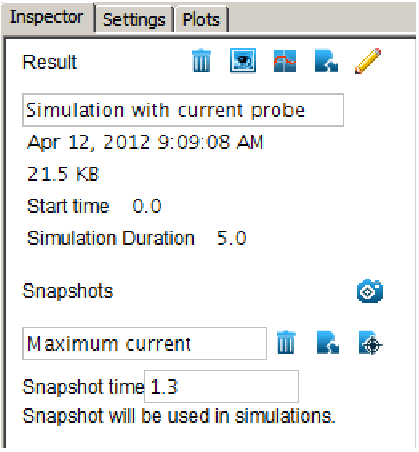
If you save your model with this stored result, you can refer to it in a future MapleSim session as it will be a part of a model. Hence, when you open the model in a future MapleSim session, the graphs, progress messages, and 3-D animation that you saved will be available in the **Stored Results** palette.

If you want to work with your simulation data in another application, you can also export your results to a Microsoft Excel (.xls or .xlsx) or comma-separated value (.csv) file.

Saving and Using Snapshots

You can save the state information by taking a snapshot of your simulation, at a particular time and saving the snapshot as part of a stored result. Saving a snapshot allows you to use the state information in subsequent simulations by selecting it in the **Advanced Simulation** settings (see **Table 4.2**). When you use a snapshot in this way, the information recorded in the snapshot extracts the initial conditions for subsequent simulations.

To manage the snapshots for a stored result, access the **Snapshots** section under the **Inspector** tab for the stored result.



Result

Simulation with current probe

Apr 12, 2012 9:09:08 AM

21.5 KB

Start time 0.0

Simulation Duration 5.0

Snapshots

Maximum current

Snapshot time 1.3

Snapshot will be used in simulations.

Figure 4.3: The Stored Result and Snapshots Properties Section

For more information, see the **Using MapleSim → Simulating a Model → Managing Simulation Results** section in the MapleSim Help system.

4.5 Customizing Plot Windows

By default, quantities are plotted in separate simulation graphs and are alphabetically listed according to probe and quantity names. In each graph, the quantity values are plotted along the y-axis versus the simulation time values along the x-axis.

You can optionally create a custom plot window layout to specify the graph and customized plot titles, and specify the number of columns to appear in the plot window. You may want to create a custom plot window layout if, for example, you want to compare multiple quantities in the same graph, plot one quantity versus another, or view a simulation graph for a specific quantity without editing other probe values.

To create a custom plot window layout, you specify attributes in the **Plots** tab on the right side of the MapleSim window.

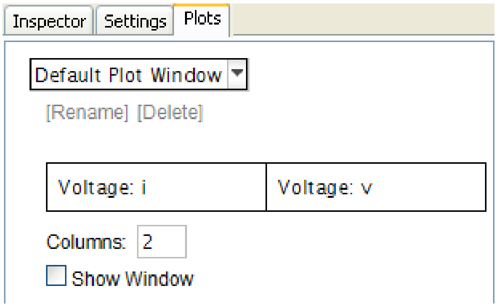


Figure 4.4: Custom Plot Window

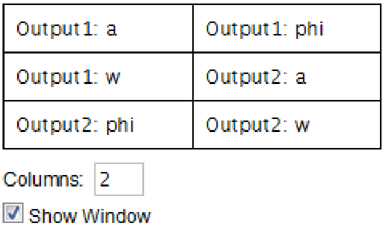
You can then use the new layout in a simulation. When you select the custom plot window layout from the drop-down menu in the **Plots** tab, select the **Show Window** check box, and then simulate your model, a custom plot window with the specified attributes appears in addition to the default plot window. You can store multiple layouts and select the one you want to use when you run a simulation.

Example: Plotting Multiple Quantities in Individual Graphs

In this example, you will view the default plot window layout and then create a custom plot window layout to plot and compare multiple quantities in the generated simulation graphs.

To view the default plot window layout

1. Under the **Libraries** tab, browse to the **Examples** → **Physical Domains** → **Multibody** menu, and then click the **Double Pendulum** example.
2. Click the **Plots** tab located on the right side of the MapleSim window. The following table, which displays all of the selected probe quantities, appears in the pane.



Output1: a	Output1: phi
Output1: w	Output2: a
Output2: phi	Output2: w

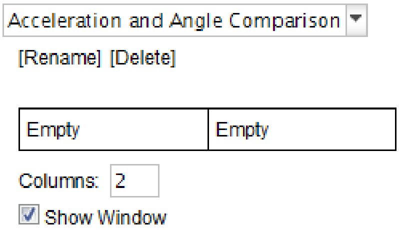
Columns: 2

☑ Show Window

This table displays the default layout of the generated graphs in the plot window. For example, the table shown in the image above indicates that the graph for the **Output1:a** quantity will appear in the top-left corner of the plot window, the graph for the **Output1:phi** quantity will appear in the top-right corner of the plot window, and so on after you run a simulation.

To create a custom plot window layout

1. From the drop-down list at the top of the pane, select **Add Window**.
2. In the **Create Plot Window** dialog box, specify a plot window layout name **Acceleration and Angle Comparison**.
3. In the **Columns** field, type **2** and press **Enter**. The table in the pane now contains two cells; each cell represents a plot window area for which you can specify layout attributes.



Acceleration and Angle Comparison ▾

[Rename] [Delete]

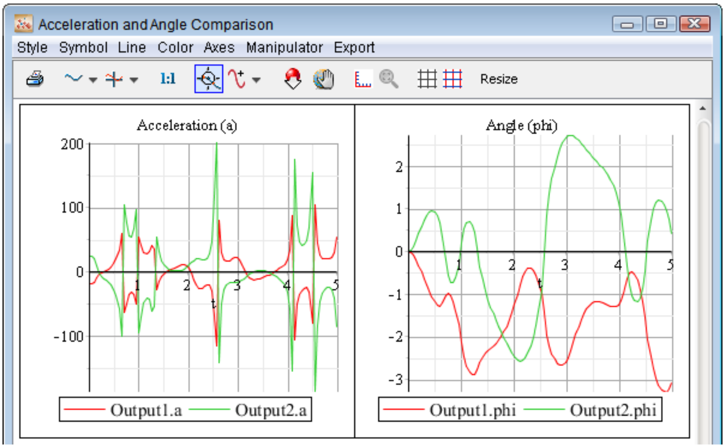
Empty	Empty

Columns: 2

☑ Show Window

4. Click **Empty** in the left cell.
5. In the **Title** field, enter **Acceleration (a)**.
6. From the **Primary Y-axis** drop-down menu, select **Output1: a**.
7. Click **[Add Variable]** below the drop-down menu.
8. From the second **Primary Y-axis** drop-down menu, select **Output2: a**.
9. In the table at the top of the pane, click **Empty** in the top-right cell.
10. In the **Title** field, enter **Angle (phi)**.
11. From the **Primary Y-axis** drop-down menu, select **Output1: phi**.
12. Click **[Add Variable]** below the drop-down menu.

13. From the second **Primary Y-axis** drop-down menu, select **Output2: phi**. You can now simulate the model using the new plot window layout.
14. Make sure that the **Show Window** check box in the **Plots** tab is selected.
15. Click **Run Simulation** (▶) in the main toolbar. The following custom plot window, which compares the acceleration values in one graph and the angle values in another, appears in addition to the default plot window.



If you want to display the default plot window only, clear the **Show Window** check box in the **Plots** tab and simulate your model again.

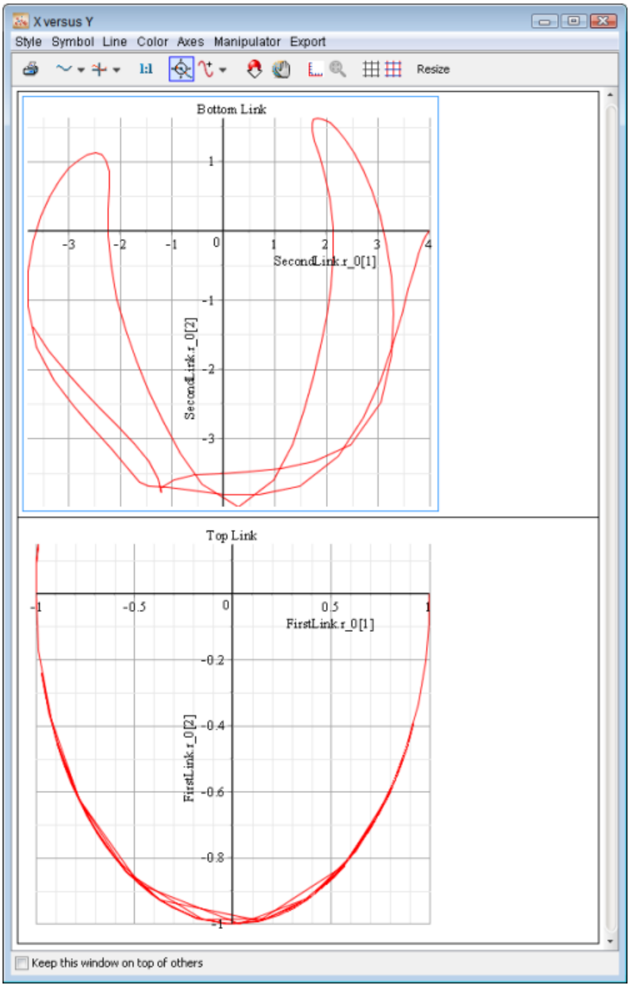
Example: Plotting One Quantity versus Another

In this example, you will create a custom plot window layout to plot the X and Y position of each of the links of a double pendulum.

To plot one quantity versus another in a custom plot window

1. Under the **Libraries** tab, browse to the **Examples** → **Physical Domains** → **Multibody** menu, and then click the **Double Pendulum** example.
2. In the **Model Workspace** toolbar, click **Attach Probe** (✎).
3. Click the right port of the **L$_1$** shared subsystem.
4. Click on an empty area of the **Model Workspace** to position the probe.
5. In the **Inspector** tab, label this probe **FirstLink**, and then select **Length[1]** and **Length[2]**.
6. Add another probe that measures the **Length[1]** and **Length[2]** quantities to the right port of the **L$_2$** shared subsystem, and label this probe **SecondLink**.
7. Click the **Plots** tab on the right side of the MapleSim window.

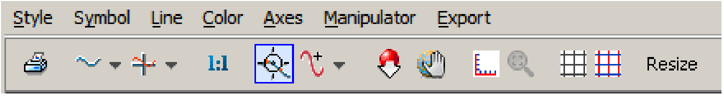
8. From the drop-down list at the top of the pane, select **Add Window**.
9. In the **Create Plot Window** dialog box, assign the plot window layout the name **X versus Y**. Click **OK**.
10. Click **Empty** in the table at the top of the pane.
11. In the **Title** field, enter **Bottom Link** and press **Enter**.
12. From the **X-axis** drop-down menu, select **SecondLink: r_0[1]**
13. From the **Primary Y-axis** drop-down menu, select **SecondLink: r_0[2]**.
14. In the table at the top of the pane, click **Empty** below the **Bottom Link** cell.
15. In the **Title** field, enter **Top Link** and press **Enter**.
16. From the **X-axis** drop-down menu, select **FirstLink: r_0[1]**.
17. From the **Primary Y-axis** drop-down menu, select **FirstLink: r_0[2]**.
18. Make sure that the **Show Window** check box is selected.
19. Click **Run Simulation** (▶) in the **Main Toolbar**. The following custom plot window appears, in addition to the default plot window.



The plots above show the motion of the end point of each link in the pendulum. The bottom link follows a more disorderly path because of the interaction with the top link.

4.6 Plot Window Toolbar and Menus

After simulating a model, you can use the tools and menus in the plot window to customize curves, axes, and gridlines; browse simulation graphs; and export simulation graphs to several image formats. The following menus and toolbar appear at the top of each generated plot window.



You can hover your mouse pointer over any of the toolbar buttons to view their descriptions.

For more information about these tools, see the **Using MapleSim → Simulating a Model → Working with Simulation Graphs** section of the MapleSim Help system.

4.7 Visualizing a Multibody Model

In MapleSim, the 3-D visualization environment allows you to build and analyze 3-D graphical representations of multibody systems. As you build a model and change its parameters, you can validate the 3-D configuration of the model and visually analyze your simulation results. You can build 3-D models by dragging and connecting objects in the **3-D Workspace**, and you can visualize your simulation results by playing animations that depict the movement of the objects.

As you build a block diagram in the **Model Workspace**, the corresponding changes are automatically reflected in the 3-D representation in the **3-D Workspace**. Similarly, when you build a model in the **3-D Workspace**, the corresponding changes are automatically reflected in the block diagram in the **Model Workspace**. Changes that you make in either of the workspaces are shown in both the **Model Workspace** and **3-D Workspace** as you edit your model.

In the **3-D Workspace**, you can view your model from any direction and control playback options to focus on specific components and their motions. Also, you can attach 3-D shapes to parts of your model to create a realistic-looking system representation. These shapes can either be imported from an external CAD file or selected from the **Multibody → Visualization** palette in the **Libraries** tab. You can also attach trace lines to show where components move during an animation.

CAD geometry and visualization shapes are drawn transparent in Construct mode and non-transparent in Playback mode.

For more information about adding 3-D shapes and using the **3-D Workspace**, see the **Using MapleSim → Visualizing a Model** section of the MapleSim Help system.

3-D Visualization and Multibody Settings

The parameters used for 3-D visualization of multibody mechanical components are found under the **Settings** tab in the **Multibody** and **3-D Visualization** sections.

3-D Visualization Settings

You can specify the following 3-D Visualization values for models containing multibody mechanical components:

Table 4.3: 3-D Visualization Parameter Values

Parameter	Default	Description
3-D Animation	☑	Specifies whether a 3-D animation is generated after running a simulation. When this option is cleared, no 3-D animation is generated. If your model is complex and if you do not plan on animating your model, you may want to clear this option to reduce the time required to run a simulation.
3-D Playback Time	-	Specifies the playback duration of the 3-D animation, at a 1x speed rate, in seconds. This value differs from the $\mathbf{t_d}$ value, which in comparison specifies the simulation duration represented in your simulation graphs. You can specify a floating-point value for this option when the **3-D Animation** field is selected. You can specify a value in this field to increase or decrease the speed at which an animation is played. For example, if the $\mathbf{t_d}$ value is set to 0.5 seconds, you can set the **3-D Playback Time** value to 10 seconds to slow down the animation; an animation of the 0.5 second simulation would then be played back over a span of 10 seconds in real time. If no value is specified in this field, the **3-D Playback Time** value is the same as the value entered in the $\mathbf{t_d}$ field: an animation of a 10 second simulation, for example, would be played back over a span of 10 seconds in real time. The number of frames represented in an animation is determined by the value specified in the **Plot Points** field, and the **3-D Playback Time** value multiplied by the **3-D Sampling Rate** value.

Parameter	Default	Description
3-D Sampling Rate	40	Number of frames per second to include in the 3-D animation playback. You can increase this value to create an animation with smoother playback. You can specify a positive integer for this option when the **3-D Animation** field is set to **true**. The number of frames represented in an animation is determined by the value specified in the **Plot Points** field, and the **3-D Playback Time** value multiplied by the **3-D Sampling Rate** value. The '3D Sample Rate' uses a default value of 40 fps for new models.
Enable Translational Snapping	□	When selected, components are positioned at the closest location in 3-D space based on the translation snap delta value.
Translation Snap Delta	1.0	Specifies the translation snap delta spacing.
Enable Rotational Snapping	□	When selected, components are positioned at the closest location in 3-D space based on the rotation snap delta value.
Rotation Snap Delta	0.785398163	Specifies the rotational snap delta spacing.
Perspective Grid Extent	10.0	Specifies the size of the grid drawn in the perspective view. The grid extends this distance in both directions on the horizontal plane.
Grid Spacing	1.0	Specifies the space between grid lines.
Base Radius	0.1	Implicit geometry consists of spheres and cylinders representing multibody components in the **3-D Visualization Area - Construct Mode**. Cylinders are drawn using Base Radius, while spheres (for Rigid Bodies and Joints) are drawn using Base Radius * 2.
Enable view change animations	☑	When enabled, a smooth transition occurs when switching between **Construct** and **Playback** modes, between 3-D orthographic and perspective views, and when using **Fit Scene**, **Fit Selected** or **Fit Animation** in the **3-D Visualization Area**.

Multibody Settings

You can specify the following parameter values for models containing multibody mechanical components:

Table 4.4: Multibody Parameter Values

Parameter	Default	Description
$\widehat{e}_g$	$[0,-1,0]$	Direction of gravity.
g	9.81	The acceleration due to gravity at the surface of the Earth. The default units are in $\frac{m}{s^2}$.

The 3-D Workspace

The **3-D Workspace** is the area in which you build and animate 3-D models in the MapleSim window.

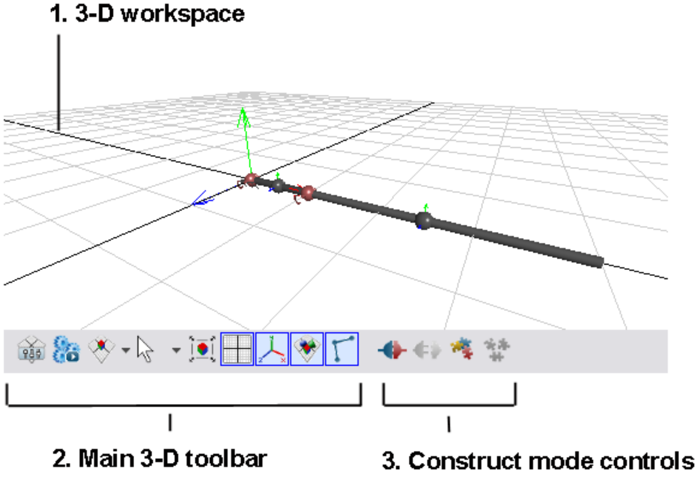
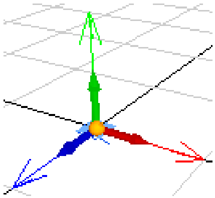


Figure 4.5: 3-D Workspace

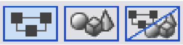
Table 4.5: 3-D Workspace Controls

Component	Description
1. 3-D Workspace	The area in which you build, view, and animate a 3-D model. The arrows at the origin indicate the directions of the world axes and are designated by color:  **X** - red **Y** - green **Z** - blue You can use the grid as a reference to determine the relative sizes and positions of elements in your 3-D model.
2. Main 3-D Toolbar	Contains tools for hiding and displaying components in the **3-D Workspace**, toggling between different modes, selecting camera navigation tools, and changing the 3-D model view.
3. Construct Mode Controls	Controls for building and assembling a 3-D model, and connecting 3-D objects. **Note:** When you switch to playback mode, the construct mode controls are hidden and the playback controls for animating a 3-D model and specifying camera tracking options appear in this toolbar.

You can hover your mouse pointer over any of the buttons to view their descriptions.

Displaying the 3-D Workspace

To display and hide the **3-D Workspace** and **Model Workspace**, use the buttons located at the bottom of the MapleSim window. By default, the **3-D Workspace** does not appear.



Click **3-D View** ( ) if you want to work with your model in the **3-D Workspace** only.

Click **2-D Schematic** ( ) if you want to work in the **Model Workspace** only.

Click **2-D Schematic** and **3-D View** ( ) if you want to work in both the **Model Workspace** and **3-D Workspace**.

Viewing and Browsing 3-D Models

In the **3-D Workspace**, you can view and browse a 3-D model from the *perspective* view or one of the *orthographic* views using the **3-D View Controls** tool.

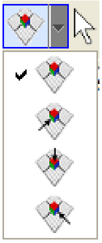


Figure 4.6: 3-D View Controls

The perspective view allows you to examine and browse a model from all angles in 3-D space. It allows you to see 3-D spatial relationships between elements in your model. In the perspective view, objects that are closer to the camera appear larger than those that are further away from the camera.

In the following image, a double pendulum model is shown from the perspective view.

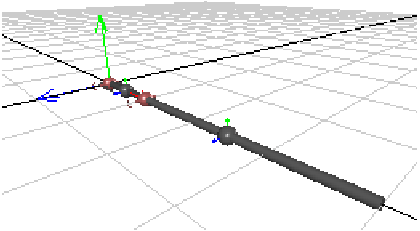


Figure 4.7: Perspective View Double Pendulum

You can also view your 3-D model from front, top, and side orthographic views. Orthographic views use parallel projection as opposed to perspective projection, so your 3-D model appears as a flattened object because no depth information is shown. Orthographic views are sometimes referred to as "true length" views because they display undistorted lines and distances in the view plane that is orthogonal to the camera direction; these views are useful for analyzing spatial relationships or clearances between objects.

In the following image, the double pendulum model is shown from the top orthographic view.

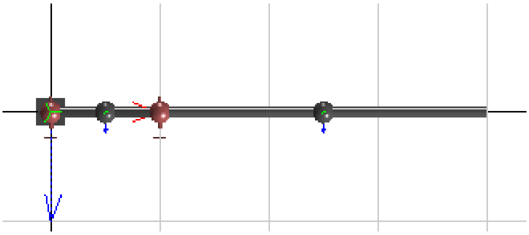


Figure 4.8: Orthographic View Double Pendulum

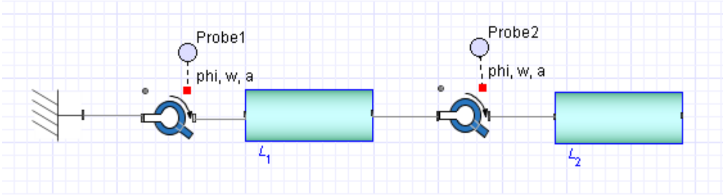
You can browse a model and change the model view while an animation is static or playing. In all of the views, you can pan and zoom into or out from your model. In the perspective view, you can also move the camera to view your model from above or below, and from any direction around your model.

Tip: Before panning, zooming, or moving the camera around a large 3-D model, hover your mouse pointer over the object that you want to focus on. MapleSim adjusts the navigation controls according to the object on which you place the mouse pointer.

Adding Shapes to a 3-D Model

Adding Implicit Geometry

By default, basic spheres and cylinders called *implicit geometry* appear in the **3-D Workspace** to represent physical components in your model. For example, consider the following double pendulum model, which contains two revolute joints and two subsystems that represent planar links.



In the **3-D Workspace**, the implicit geometry of the fully assembled pendulum model appears as follows.

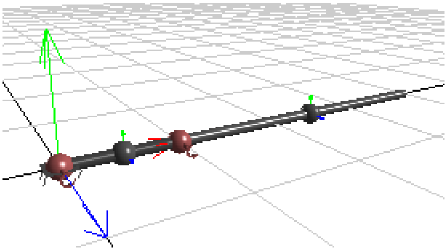


Figure 4.9: Implicit Geometry Double Pendulum

In this example, the spheres represent the revolute joints and rigid bodies, and the cylinders represent the planar links.

Implicit geometry that is not connected to other implicit geometry is drawn in a light gray color; implicit geometry that is assembled is drawn in a dark gray color, with the exception of joint objects, which are drawn in red.

Note: Components that you exclude from a simulation in the **Model Workspace** do not appear in the **3-D Workspace**.

Adding Attached Shapes

If you want to create a more realistic representation of your model, you can add shapes and lines called *attached shapes* to your model. To do so, you first add and connect attached shape components from the **Multibody** →**Visualization** palette to your block diagram in the **Model Workspace**.

When you simulate your model, the attached shapes appear in the **3-D Workspace**, in addition to the implicit geometry. In the following image, attached shapes have been added to represent the pendulum stem and bob pictorially. Also, a trace line - the curved line in the image - can be set to depict the locus of points that will be traced by a particular part of the model during a simulation.

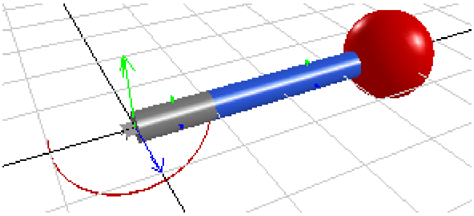


Figure 4.10: Attached Shapes

You can customize the color, size, scale, and other visual aspects of the attached shapes by setting parameter values for individual components in the **Settings** tab before simulating the model.

If you want to view only the implicit geometry in the **3-D Workspace**, you can hide the attached shapes by clicking **Show/hide attached shapes** ( ) in the **Main 3-D Toolbar**. If you want to view the attached shapes only, you can hide the implicit geometry by clicking **Show/hide implicit geometry** ( ).

For more information about attached shape components, see the **MapleSim Component Library** → **Multibody** → **Visualization** → **Overview** topic in the MapleSim Help system.

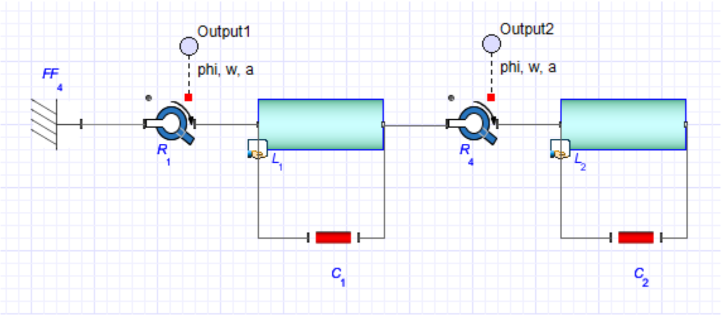
Note: If your model contains **Flexible Beam** components, deflection of the beam will not be depicted in the implicit geometry of your 3-D model.

Example: Adding Attached Shapes to a Double Pendulum Model

In the following example, you will add cylinder shapes to represent the pendulum stem and a sphere component to represent the pendulum bob. You will also add a **Path Trace** component to display the path on which the revolute joint will move during an animation.

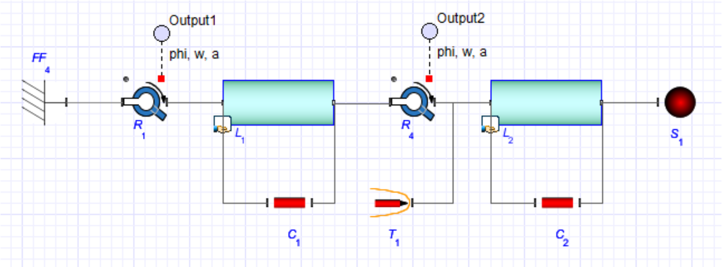
To add attached shapes

1. Under the **Libraries** tab, browse to the **Examples** → **Physical Domains** → **Multibody** menu, and then click the **Double Pendulum** example.
2. Expand the **Multibody** palette and then expand the **Visualization** menu.
3. Add two **Cylindrical Geometry** components below the planar link subsystems in the **Model Workspace**.
4. Connect the components as shown below.



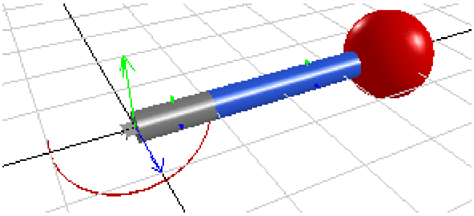
5. From the same menu, add a **Spherical Geometry** component and place it to the right of the **L$_2$** shared subsystem.

6. Right-click (**Control**-click for Macintosh) the **Spherical Geometry** component and select **Flip Horizontal**.
7. Add a **Path Trace** component and place it between the two **Cylindrical Geometry** components.
8. Connect the components as shown below.



9. Select the first **Cylindrical Geometry** component (**C**$_1$ in the previous figure) in the **Model Workspace**.
10. In the **Inspector** tab on the right side of the MapleSim window, change the radius of the cylinder to **0.3**.
11. To select a color for the cylinder, click the box beside the **color** field and click one of the color swatches.
12. Select the second **Cylindrical Geometry** component (**C**$_2$ in the previous figure) in the **Model Workspace**.
13. Change the radius of this cylinder to **0.3** and change the color.
14. To simulate the model, click **Run Simulation** (▶) in the **Main Toolbar**.

When the simulation is complete, the **3-D Workspace** appears. The **3-D Workspace** is set to playback mode automatically and displays your model with the attached shapes.



15. To animate the model, click **Play** (▶) below the **3-D Workspace**.

For an another example of how to use the **Path Trace** component, under the **Libraries** tab, browse to the **Examples** → **User's Guide Examples** menu, and then open the **Lorenz Attractor** example.

Building a Model in the 3-D Workspace

You can build MapleSim models by adding and connecting objects in the **3-D Workspace**. To add components to a 3-D model, you can drag multibody components from the **Multibody** palette, the **Favorites** palette, a custom library that you created, or from the search pane in the **Libraries** tab.

You can toggle between **Construct Mode***, by* clicking ( ) and **Playback Mode**, by clicking ( ) to perform specific tasks in the **3-D Workspace**. In construct mode, you can add, connect, and lay out 3-D objects, and set initial conditions for joints and other multibody components by using graphical controls in the **3-D Workspace**. In playback mode, you can animate your 3-D model and specify camera tracking options to center an object in the **3-D Workspace** during an animation.

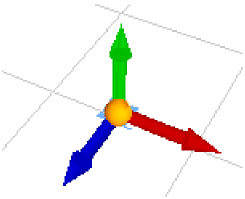
Any changes that you make to your 3-D model are automatically shown in the block diagram representation in the **Model Workspace** and vice versa. For example, if you add and connect a **Flexible Beam** component in the**3-D Workspace**, the block diagram representation of the **Flexible Beam** with the added connection lines appear in the **Model Workspace** at the same time.

Notes:

- Subsystems cannot be created in the **3-D Workspace**. They must be created in the **Model Workspace**.
- Components from the multibody **Forces and Moments**, **Sensors**, and **Visualization** component libraries cannot be dragged into the **3-D Workspace**. You must add these components in the **Model Workspace**.

Moving Objects in the 3-D Workspace

In construct mode, you can position individual objects or groups of objects by clicking and dragging the 3-D manipulators in the **3-D Workspace**.



To display the 3-D manipulator for a single unconnected object, click the object once in the **3-D Workspace**. You can then click and drag the blue arrow of the 3-D manipulator to move the object along the Z axis, the green arrow to move the object along the Y axis, and the red arrow to move the object along the X axis. You can also click and drag the sphere at the center of the 3-D manipulator to move the object in all directions.

For a group of connected objects, the configuration of your model determines where the 3-D manipulators are located.

- If your 3-D model contains a **Fixed Frame** component, click the square that represents the **Fixed Frame** component to display the 3-D manipulator.
- If your model does not contain a **Fixed Frame** component, click the object that defines the initial conditions for your system to display the 3-D manipulator. For example, if your model contains a **Rigid Body** component with its initial condition parameters set to **Strictly Enforce**, that **Rigid Body** component displays the 3-D manipulator. When a model is moved in the **3-D Workspace**, the initial conditions are updated for all of the other **Rigid Body** components that depend on the **Rigid Body** component that has its initial conditions set to **Strictly Enforce**.
- If your model does not contain a **Fixed Frame** component or a **Rigid Body** component with its initial conditions set to **Strictly Enforce**, click any of the objects in your 3-D model to display the 3-D manipulator. When you move the group of objects, the initial conditions of all of the multibody components in your model are set to **Treat as Guess**.

Note: To display 3-D manipulators, the multibody components in your model must contain numeric parameter values. If custom parameter values defined in a parameter block, global parameter, or subsystem parameter are assigned to a multibody component, no 3-D manipulator will appear when you click that component in the **3-D Workspace**.

Assembling a 3-D Model

A 3-D model must be *assembled* before you can animate it in the **3-D Workspace**. Assembling a 3-D model refers to synchronizing the model appears in the **3-D Workspace** with the initial configuration of your model defined by the assigned parameter values and initial condition guess values. The synchronization process occurs automatically when you simulate your model or click **Update 3-D View** ( ) in the **3-D Toolbar**. In the **3-D Workspace**, assembled implicit geometry is drawn in a dark gray color, with the exception of joint objects, which are drawn in red.

Note: You can only assemble 3-D models with a valid configuration and valid connection lines. For example, if you attempt to assemble a 3-D model with missing connection lines, an error message appears in the console pane and no animation is generated.

For more information, see Assembling a 3-D Model in the MapleSim Help system.

Using the Unenforce Constraints Button to Manipulate Joints in the 3-D Workspace

In construct mode, you can select a joint object in the **3-D Workspace** and click **Unenforce Kinematic Constraints** ( ) to specify that the kinematic constraints of the joint are not enforced in the **3-D Workspace** as you build your model. Joints with unenforced kinematic constraints appear in pink in the **3-D Workspace** and their initial conditions are not shown in the **3-D Workspace** as you build your model.

You may want to use **Unenforce Kinematic Constraints** () if, for example, you are creating a closed-loop model in the **3-D Workspace** and you need a joint to remain in a specific position as you are building and laying out your 3-D model.

Notes:

- The unenforce constraints button does not affect the actual initial conditions specified for your joint components in the **Inspector** tab; it affects the initial conditions depicted in the **3-D Workspace** for display purposes only.
- Initial conditions for other joints with enforced kinematic constraints will be shown in the **3-D Workspace**, but will not affect related joints with unenforced kinematic constraints.

For example, consider a double pendulum 3-D model that contains a revolute joint with unenforced kinematic constraints and a second revolute joint with enforced kinematic constraints. If you change the initial angle of the joint with enforced kinematic constraints, the joint with the unenforced kinematic constraints will remain in its original position while the joint with enforced kinematic constraints will be shown at the new initial angle. To display all of the new initial conditions in the **3-D Workspace**, you must assemble your model by running a simulation or clicking **Update 3-D View** ().

Displaying Attached Shapes as You Build a 3-D Model

When you connect **Cylindrical Geometry**, **Tapered Cylinder Geometry**, **Box Geometry**, or **Spherical Geometry** components to your block diagram in the **Model Workspace**, the corresponding attached shapes appear in the **3-D Workspace** in both construct mode and playback mode. The attached shape appears in the **3-D Workspace** after you connect all of its ports to compatible ports of multibody components in the **Model Workspace**.

Working with CAD Geometry

CAD geometry can also be shown in the **3-D Workspace** in both construct mode and playback mode. When you add a **CAD Geometry** component anywhere in the **Model Workspace**, the corresponding CAD image appears in the **3-D Workspace** regardless of whether the **CAD Geometry** component is connected to other components in your model. If a **CAD Geometry** component is not connected to other components, it will be drawn at the origin of the 3-D grid; if a **CAD Geometry** component is connected to another component, it will be drawn at the origin of the coordinate frame of the modeling component to which it is attached.

You can define the translational and rotational offset for CAD images either before or after connecting the corresponding **CAD Geometry** component to your model. To define these offsets, select the **CAD Geometry** component in the **Model Workspace** and specify parameter values in the **Inspector** tab.

Example: Building and Animating a Double Pendulum Model in the 3-D Workspace

In this example, you will build and animate a double pendulum model in the **3-D Workspace**. You will perform the following tasks:

To build and animate a double pendulum

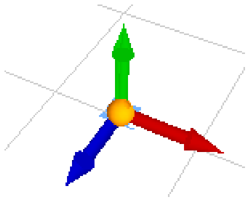
1. Add and move objects in the **3-D Workspace**.
2. Connect the 3-D objects.
3. Set initial conditions for the joints in your model.
4. Animate the 3-D model.

Note: In a new MapleSim document, the **3-D Workspace** is set to construct mode by default.

Adding and Moving Objects in the 3-D Workspace

To add or move an object

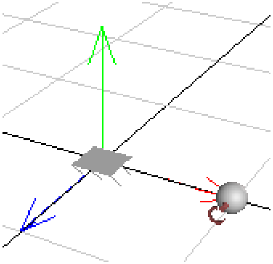
1. Open a new MapleSim document.
2. At the bottom of the MapleSim window, click **3-D view** () to display the **3-D Workspace**.
3. Under the **Libraries** tab, expand the **Multibody** palette, and then expand the **Bodies and Frames** menu.
4. From the palette, drag a **Fixed Frame** component into the **3-D Workspace**. A gray square, which represents the **Fixed Frame** component, is added to the **3-D Workspace** and its 3-D manipulator appears.



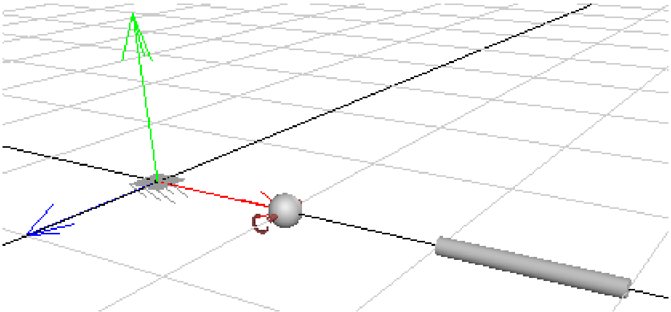
You can use this manipulator to position objects in the **3-D Workspace**.

5. Position the **Fixed Frame** object at the origin of the grid by clicking and dragging the 3-D manipulator arrow controls.

6. From the **Multibody** → **Joints and Motions** menu, drag a **Revolute** component into the **3-D Workspace** and place it to the right of the **Fixed Frame**.



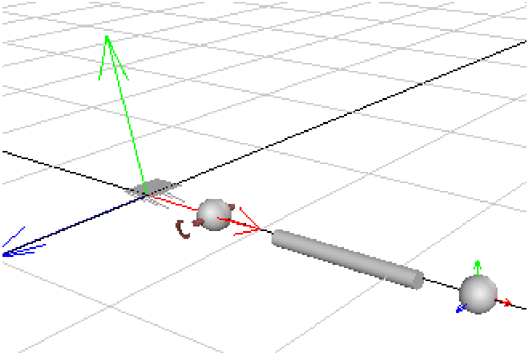
7. From the **Multibody** → **Bodies and Frames** menu, drag a **Rigid Body Frame** component into the **3-D Workspace** and place it to the right of the **Revolute** component.



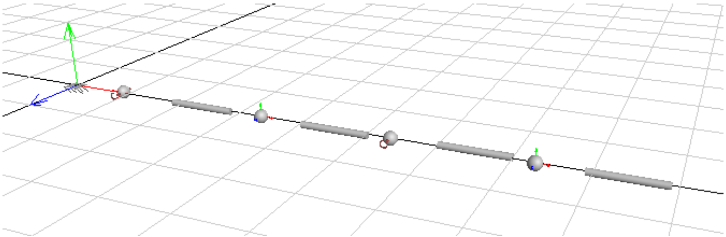
8. From the same menu, drag a **Rigid Body** component into the **3-D Workspace** and place it to the right of the **Rigid Body Frame**.

Tip: To zoom into and out from the **3-D Workspace**, hover your mouse pointer over the object that you want to focus on and rotate your mouse wheel. When zooming with the mouse wheel, the location under the pointer remains in place, allowing you to zoom in on that location. To pan your model, hold the **Shift** key and drag your mouse pointer in the **3-D Workspace**.

9. From the same menu, drag another **Rigid Body Frame** component into the **3-D Workspace** and place it to the right of the **Rigid Body**. The components for the first pendulum link have been added.



10. Repeat steps 6 to 9 to add components for a second pendulum link to the right of the last **Rigid Body Frame** component that you added.

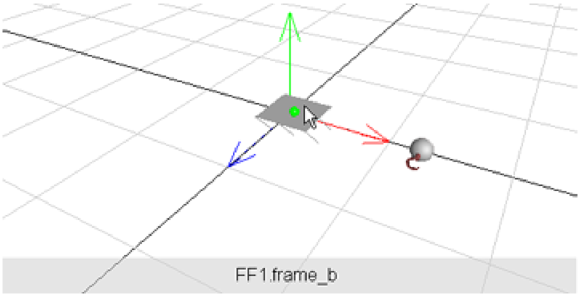


Connecting 3-D Objects

You will now connect the objects that you added in the previous task.

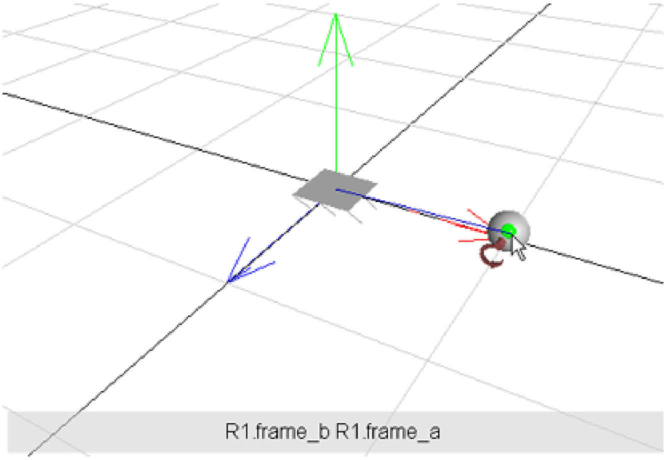
To connect objects

1. Click **Connect ports** (◆).
2. Hover your mouse pointer over the **Fixed Frame** object. A green dot appears.



FF1.frame_b

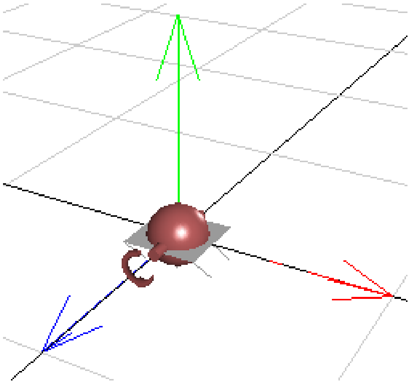
3. Click the green dot once to start the connection line.
4. Hover your mouse pointer over the first **Revolute** joint object. The gray panel at the bottom of the **3-D Workspace** displays the names of the **Revolute** joint frames.



R1.frame_b R1.frame_a

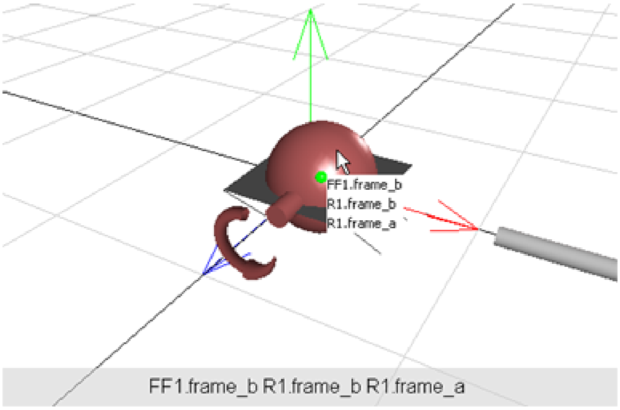
5. Click the **Revolute** joint object once. A context menu displays the names of the frames to which you can connect the line.

6. Select **R1.frame_a**. The objects are connected in the **3-D Workspace**.



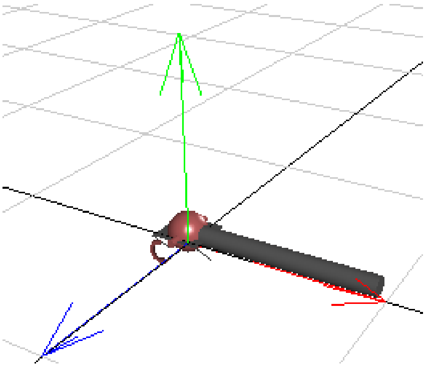
Note that the joint component is drawn in red when it is connected.

7. Click **Connect ports** (◆) to start the next connection line.
8. Click the sphere that represents the **Revolute** joint. A context menu displays the frames of the **Revolute** joint, as well as the **Fixed Frame** to which it is connected.



9. From the context menu, select **R1.frame_b**.

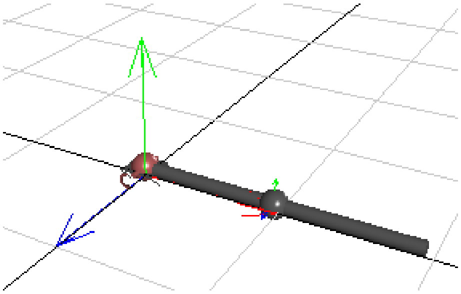
10. Drag your mouse pointer to the end of the cylinder that represents the **Rigid Body Frame** and click the green dot.



frame_b of the first revolute joint, $\mathbf{R_1}$, is now connected to **frame_a** of the first rigid body frame, $\mathbf{RBF_1}$.

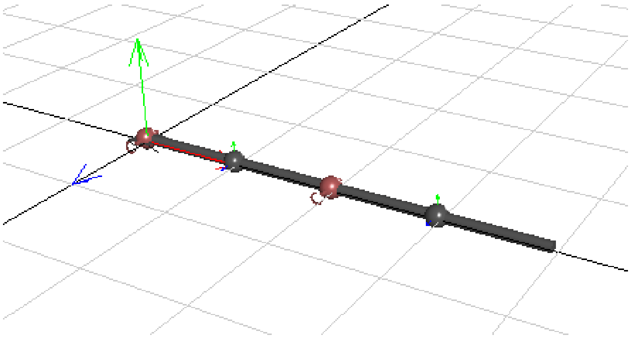
11. Click **Connect ports** to start a new connection line.
12. Hover your mouse pointer over the other end of the cylinder that represents the $\mathbf{RBF_1}$ component and click the cylinder once.
13. Drag your mouse pointer to the sphere that represents the first **Rigid Body** component, $\mathbf{RB_1}$, and click it once. $\mathbf{RB_1}$ is now connected to $\mathbf{RBF_1}$.
14. In the same way, connect **frame_a** of $\mathbf{RB_1}$ to **frame_a** of the second **Rigid Body Frame**, $\mathbf{RBF_2}$.

Note: Click the connect button to start each connection line.



15. Connect **frame_b** of the second **Rigid Body Frame** to **frame_a** of the second **Revolute** joint.
16. Connect **frame_b** of the second **Revolute** joint to **frame_a** of the third **Rigid Body Frame**.

17. Connect third **Rigid Body Frame** to the second **Rigid Body**, and then connect the second **Rigid Body** to the fourth **Rigid Body Frame**. The complete 3-D model appears below.



If you click **2-D Schematic View** (), you will see that all of the components are added and connected accordingly.

Tip: As you are building a 3-D model, it is a good practice to switch to the block diagram view periodically to check whether the block diagram is laid out the way you want.

Setting Initial Conditions for the Joint Components

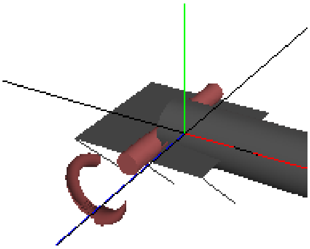
In construct mode, you can set initial conditions for joint components by using graphical controls in the **3-D Workspace**.

Note: Joint components that have been assigned custom parameter values defined in a parameter block, global parameter, or subsystem parameter will not allow the use of graphical controls for setting initial conditions. In these cases, use the fields in the **Inspector** tab to set the initial conditions.

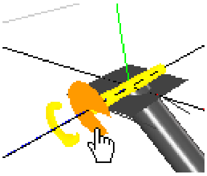
To set initial conditions

1. If you are in the block diagram view, click **3-D View** () at the bottom of the MapleSim window to display the **3-D Workspace**.
2. In the **3-D Workspace**, to set the initial angle of the first revolute joint, click the sphere that represents the first revolute joint in the **3-D Workspace**. The red sphere, which

represents the joint component, is removed temporarily from the **3-D Workspace** and the manipulator for the joint appears.



3. Hover your mouse pointer over the manipulator. The manipulator appears in yellow.
4. Click and drag your mouse pointer around the manipulator to display the meter that represents the initial angle value that you want to set for the revolute joint. A pie graph-shaped meter appears in orange.



When you drag your mouse pointer, you can adjust the initial angle value for the degree of freedom represented by the graphic. The angle value increases if you drag the mouse pointer up or to the right, and decreases if you drag the mouse pointer down or to the left.

5. Release your mouse button when the meter is at the approximate initial condition value that you want. In the **Inspector** tab, the $\boldsymbol{\theta_0}$ parameter displays the value that you selected and the implicit geometry is set to that value in the **3-D Workspace**.

Tips:

- Alternatively, you can set initial conditions for your model by entering a value for the $\boldsymbol{\theta_0}$ parameter in the **Inspector** tab. Initial conditions that you specify in the **Inspector** tab will be shown in the 3-D model.
- To specify precise initial angle conditions, turn on snapping by clicking **Change 3-D Settings** () in the **Main 3-D Toolbar** and selecting **Enable rotational snapping** in the **Settings** tab.

Animating the 3-D Model

You will now simulate your 3-D model to generate the animation that can be viewed in playback mode.

To animate the 3-D model

1. Simulate your model by clicking **Run Simulation** (▶) in the **Main Toolbar**. When the simulation is complete, the **3-D Workspace** is set to playback mode automatically.
2. To play the animation, click **Play** (▶) below the **3-D Workspace**.

Exporting a Movie of the 3-D Model

The Export Movie feature allows you to export a recorded simulation as an .mpeg file to users who may not have MapleSim. For more information, see the **Using MapleSim → Visualizing a 3-D Model → Exporting a Simulation as a Movie** topic in the MapleSim Help system.

4.8 Best Practices: Simulating and Visualizing a Model

This section describes best practices to consider when simulating and visualizing a model.

Use an External C Compiler to Run Simulations With Longer Durations

When you set the **compiler** parameter to **true** in the **Settings** tab/Solver, Maple procedures generated by the simulation engine are translated to C code and then compiled by an external C compiler. As a result, the time required to run a simulation can be reduced. In general, when you use a C compiler to simulate a model, the compilation process will be faster in simulations with longer durations.

Compare Results Generated By Sections of Your Model

For debugging purposes, you may want to view simulation results for a specific section or subsystem in your model. By selecting a section in your model and clicking **Disable** (🚫) above the **Model Workspace**, you can exclude part of your model from the next simulation that you run. When you simulate your model, results will appear only for the model sections that you did not exclude. This feature allows you to view simulation results generated by specific sections in your model and compare results without having to delete components from the **Model Workspace** or build multiple models.

For more information, see the **Using MapleSim → Simulating a Model → Excluding Objects From a Simulation** topic in the MapleSim Help system.

5 Analyzing and Manipulating a Model

In this chapter:

5.1 Overview

MapleSim is fully integrated with the Maple environment. You can use Maple commands, templates, custom components, embedded components, plotting tools, and many other technical features to analyze and manipulate the dynamic behavior of a MapleSim model or subsystem. For example, you can use Maple to retrieve and work with model subsystems and equations, test input and output values, translate your model into C code, and perform many other advanced analysis tasks. You can then attach a file of any format in the **Attachments** palette and save it as part of a MapleSim model.

To start working with your MapleSim model in Maple, you can use the templates available in the Create Attachment window by clicking **Template** (✎) in the **Main Toolbar**. These templates are Maple worksheets with pre-built tools for model building and analysis tasks: you first create a MapleSim model and open it in one of the available templates to perform an analysis task in Maple.

MapleSim Templates

The following MapleSim templates are available from the Main Toolbar:

Table 5.1: MapleSim Templates

Template Name	UI Display Name	Task
C Code Generation	Code Generation	Translate your model into C code. For more information, see *Generating and Exporting C Code from a Model (page 133)*
Custom Component Template	Custom Component	Create a custom modeling component based on a mathematical model. For more information,

Template Name	UI Display Name	Task
		see *Creating Custom Modeling Components (page 65)*.
Discrete State Space Custom Component	Custom Discrete State Space	Define and generate custom components for a MapleSim model from a discrete state space description.
Discrete Transfer Functions Custom Component	Custom Discrete Transfer Function	Define and generate custom components for a MapleSim model from a discrete transfer function.
Data Generation Template	Data Generation	Define and generate a data set to be used in MapleSim, for example, a data set for an interpolation table component. For more information, see *Creating a Data Set for an Interpolation Table Component (page 56)*.
Equation Extraction Template	Equations	Retrieve equations from linear or nonlinear models. For more information, see *Tutorial 6: Using the External C Code/DLL Custom Component Template (page 184)*
Excel Connectivity Template	Excel Connectivity	Import MapleSim parameter sets from, or export MapleSim parameter sets to, an Excel spreadsheet.
External C Code/DLL Definition	External C/Library Block	Define and generate a MapleSim custom component from external C Code/DLL.
Linear System Analysis Template	Linear System Analysis	View and analyze the equations of a linear system. For more information, see *Analyzing Linear Systems (page 129)*
Linearization	Linearization	Create a linear system object from a MapleSim continuous subsystem.
Custom Component Template: Modelica Code Definition	Modelica Custom Component	Define and generate a MapleSim custom component from Modelica code.
Monte-Carlo Simulation Template	Monte Carlo Simulation	Define a random distribution for a parameter and run a simulation using the distribution.
Multibody Analysis Template	Multibody Analysis	Retrieve multibody equations in a form that is suitable for manipulation and analysis.
Parameter Optimization Template	Optimization	Analyze and edit the parameters of a model and view possible simulation results in a graph. For more information, see *Optimizing Parameters (page 131)*
Random Data Template	Random Data	Define and generate a set of random data points to be used in MapleSim, for example, a data set for an interpolation table component.
Sensitivity Analysis	Sensitivity Analysis	Perform parameter sensitivity analysis.
Worksheet	Worksheet	Create a worksheet by opening a MapleSim Model in an embedded component. For more

Template Name	UI Display Name	Task
		information, see *Working with a Maple Standard Worksheet (page 144)*

Alternatively, to edit and analyze a model, you can insert a MapleSim Model embedded component into a Maple worksheet and open an existing MapleSim model in that component. For more information about the MapleSim Model component, see *Working with Maple Embedded Components (page 144)*.

Both methods of performing analysis tasks allow you to use commands from any Maple packages, including **MapleSim** and **DynamicSystems**, to work with your model programmatically.

Note: After using a MapleSim template, save the .mw file and then save the .msim file to which the .mw file is attached.

Tip: The pre-built analysis tools available in each template are Maple embedded components, which allow you to interact with Maple code through graphical interactive components. The code associated with each embedded component uses commands from Maple packages, including **MapleSim** and **DynamicSystems**.

To view the code associated with an embedded component, right-click (**Control**-click for Macintosh) any of the tools in the Maple worksheet, select **Component Properties**, and click **Edit**. For more information about embedded components, see the **EmbeddedComponents** topic in the Maple help system.

Working with MapleSim Equations and Properties in a Maple Worksheet

When viewing and working with MapleSim equations or properties in a Maple template, corresponding parameters, variables, connectors, subscripts and superscripts are mapped and represented differently.

Mapping MapleSim Programmatic Names to Maple

The programmatic names of certain parameters, variables, and connectors displayed in the Maple worksheet differ from the names displayed for the corresponding elements in the MapleSim interface. For example, if an **Inertia** component is included in a model, the parameter for the initial value of the angular velocity appears as ω_0 in the MapleSim interface and *w_start* in a Maple worksheet. For more information about the mappings of parameter, variable, and connector names, see the **MapleSim Component Library** in the MapleSim help system.

Representing MapleSim Subscripts and Superscripts in Maple

Subscripts and superscripts in the MapleSim interface are represented differently in a Maple worksheet. Subscripts in the MapleSim interface appear with an underscore character in a

Maple worksheet. For example, a connector called *$flange_a$* in the MapleSim interface appears as *flange_a* in a Maple worksheet. Also, superscripts are formatted as regular characters in a Maple worksheet. For example, a variable called a^2 in the MapleSim interface would be displayed as *a2* in a Maple worksheet.

Using Subsystems

The basic structure for exporting models is the subsystem. A template allows you to select a complete subsystem for which you want to analyze and manipulate. By converting your model or part of your model into a subsystem you can more easily identify the set of modeling components that you want to explore, define the set of inputs and outputs for the subsystem, or identify the components that you want to export as a block component. For best practices on creating subsystems in MapleSim, see *Best Practices: Laying Out and Creating Subsystems (page 58)*.

For an example of a basic structure for exporting models, under the **Libraries** tab, browse to the **Examples → User's Guide Examples** menu, and then open the **Preparing a Model for Export** example.

Note: When generating code for a subsystem, any included ports must be real input or real output ports. When generating code for the top-level system, the system is considered to have no inputs, but all probed values are treated as outputs.

Tip: If you want to use your complete model, group all of the components at the top level of your model into a single subsystem.

5.2 Retrieving Equations and Properties from a Model

You can use the **Equation Analysis Template** to retrieve, define and analyze equations and properties such as parameters and variables in your model. You can also set initial equations, set the level of equation optimization and generate custom equations. By using Maple commands you can perform detailed equation analysis, assign model equations to a variable or parameter, and define additional system variables and parameters. The features within this template are useful in generating reusable equations when there is more than one subsystem.

For a complete tutorial on how to use the **Equation Analysis Template**, see Chapter 6 *Tutorial 6: Using the External C Code/DLL Custom Component Template (page 184)*.

To retrieve equations and properties

1. In MapleSim, open the model for which you want to retrieve equations or properties.
2. Click **Templates** (✎) in the **Main Toolbar**. The **Create Attachment** window appears.
3. From the list, select **Equations**.

4. In the **Attachment** field, enter a name for the template and click **Create Attachment**. Your model opens in the **Equation Extraction Analysis Template** in Maple.
5. Using the navigation tools above the model diagram, select the subsystem for which you want to view equations. If you want to retrieve equations from the complete system, click **Main**.
6. Click **Load Selected Subsystem**. The model equations are extracted and the system parameters and variables are loaded. The system equations appear in the **View Core Equations** area and are automatically stored in the variable DAEs.

You can now manipulate the equations using any Maple packages, for example, **DynamicSystems** and **MapleSim**. For more information about these packages, see the **DynamicSystems** and **MapleSim** topics in the Maple help system.

5.3 Analyzing Linear Systems

You can use the **Linear System Analysis Template** to retrieve, view and analyze the equations of a linear system, test system input and output values, and view possible simulation results in a Bode or root locus plot.

Note: Linear analysis cannot be performed on the entire system. To perform linear analysis using the tools in the **Analysis and Simulation** section of the template, you must select a subsystem.

To analyze a linear system model from MapleSim

1. In MapleSim, open the linear system model that you want to analyze.
2. Click **Templates** (✎) in the **Main Toolbar**. The **Create Attachment** window appears.
3. From the list, select **Linear System Analysis**.
4. In the **Attachment** field, enter a name for the template and click **Create Attachment**. Your model is opened in the **Linear System Analysis Template** in Maple.
5. Using the navigation tools above the model diagram, select the subsystem for which you want to view equations.
6. In the **Model Input from MapleSim** section, click **Retrieve System**.
7. In the **System IO and Probes**, define the system input and output values by selecting the inputs and outputs for your linear system. To add a value as an input or output, select an entry from the **System IO and Probes** menu, and click (>). To add all of the entries in the **System IO Probes** list as inputs or outputs, click (>>). You can use the < and << buttons to remove inputs and outputs if needed.
8. From the **Equation Type** drop-down list select the equation type that you want to analyze and click **Build Linear System Object** to create the linear system.

Model Input from a .msys File

As an alternative to creating the linear system in the **Model Input from MapleSim** section, you can select a .msys file that you generated previously from a **Linear System Analysis Template** by clicking **Save** in the **Save the Linear System Object** section, or from the **MapleSim Linearization Template**. The **File Name** list is populated with the .msys files associated with the MapleSim model currently open in this template.

Model Statistics

The **Model Statistics** section contains a table summarizing the details of your MapleSim model. When you click **Retrieve System**, this table is populated with information about your system. You can now select a parameter from the list, change the value of that parameter, and use the plotting tools in the **Linear System Analysis Tools** section to view possible simulation results.

Analysis and Simulation

You can use the **Analysis and Simulation** tools in the **Linear System Analysis Tools** section to view the effects of different inputs on the outputs of your system by selecting them from a list. Optionally, you can use the **Parameter Management** table to select parameters that you want to analyze, in addition to the inputs and outputs that you select in the **Linear System Analysis Tools** section. When you click **Retrieve System** in the **Model Input from MapleSim** section or build the linear system object, the **Parameter Management** list is populated with parameters from the generated linear system. You can change the parameter values to view the effects of the parameter values on your system and perform further analysis using the linear system analysis tools.

You must click **Update** each time you change a parameter value.

Saving the Linear System Object

You can save a linear system object in the **Attachments** palette in MapleSim for future reference. To save the linear system object, enter a name and description for the linear system object and click **Save**. A .msys file is saved in the **Attachments** palette associated with the model currently open in this template.

Note: If a linear system representation is already listed in the **Attachments** palette with the same name, the existing file will be overwritten.

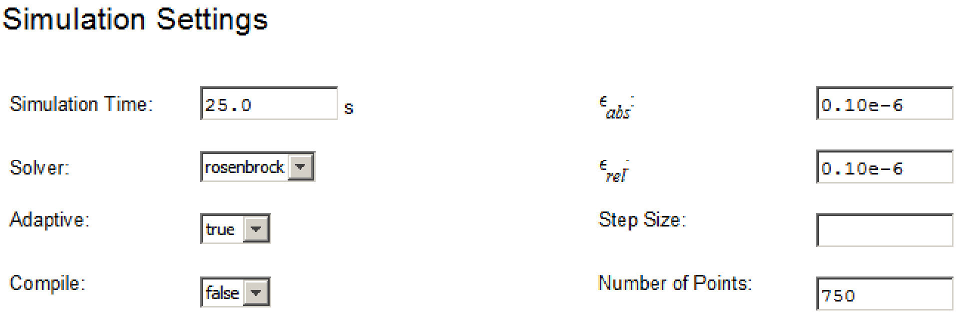
5.4 Optimizing Parameters

You can use the **Parameter Optimization Template** to test the model parameters, view simulation plots, and assign parameters to a Maple procedure to perform parameter sweeps and other advanced optimization tasks.

You can also use commands from the Global Optimization Toolbox to perform parameter optimization tasks. This product is not included with MapleSim. For more information, visit the Maplesoft Global Optimization Toolbox web site at **http://www.maplesoft.com/products/toolboxes/globaloptimization/**.

To optimize parameters

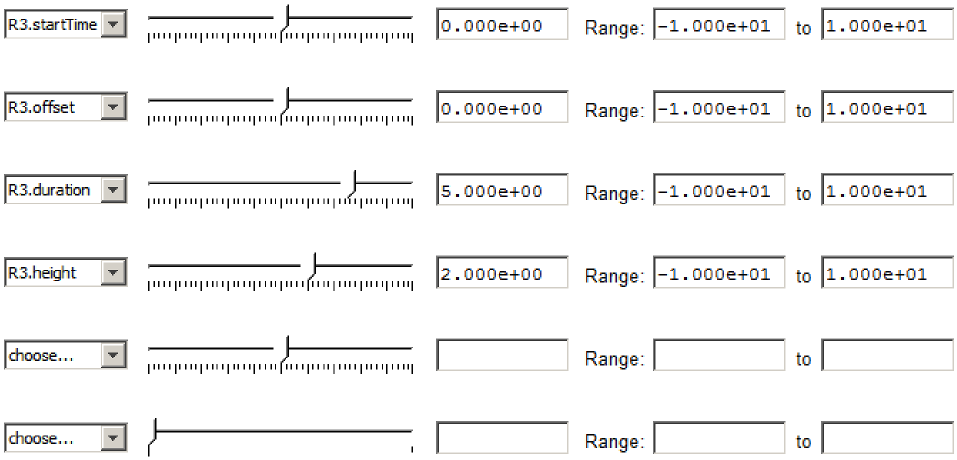
1. In MapleSim, open the linear system model that you want to analyze.
2. Click **Templates** ( ) in the **Main Toolbar**. The **Create Attachment** window appears.
3. From the list, select **Optimization**.
4. In the **Attachment** field, enter a name for the template and click **Create Attachment**. Your model is opened in the Parameter Optimization template in Maple.
5. Using the navigation tools above the model diagram, select the subsystem that you want to analyze. Your subsystem appears in the **Model Workspace**.
6. In the **Parameter Investigation** section of the template, click **Retrieve System Parameters**. The model simulation settings are imported.
7. In the **Simulation Settings** section, specify the simulation options to be used by the plotting analysis tools.



Simulation Settings

Simulation Time:	25.0 s	ϵ_{abs}:	0.10e-6
Solver:	rosenbrock	ϵ_{rel}:	0.10e-6
Adaptive:	true	Step Size:	
Compile:	false	Number of Points:	750

8. In the **Parameter Values** section, from the first drop-down menu, select the parameter that you want to test.



R3.startTime 0.000e+00 Range: -1.000e+01 to 1.000e+01

R3.offset 0.000e+00 Range: -1.000e+01 to 1.000e+01

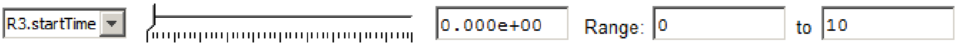
R3.duration 5.000e+00 Range: -1.000e+01 to 1.000e+01

R3.height 2.000e+00 Range: -1.000e+01 to 1.000e+01

choose... Range: to

choose... Range: to

Note: When a parameter is selected, its assigned value is shown in the field next to the slider.



R3.startTime 0.000e+00 Range: 0 to 10

9. In the **Range** fields beside the slider and parameter value field, specify the range of the slider. By default, the range is 0 to 10 unless the selected parameter value is outside of this range.
10. Using the process described above, select other parameters that you want to test.

When you have defined all of the parameters, you can move the sliders to test different values and view possible simulation results in the model by clicking the **Update Parameters in MapleSim Model** and running your simulation in MapleSim. You can also generate procedures by assigning the parameters you defined to a Maple procedure to perform further analysis tasks in the **Parameter Optimization** section of the template.

When a procedure is generated you will be able to use that procedure, when called, to run a simulation with the specified parameter values. The procedure returns the simulation result as an array of *y* values, one column per probed quantity. This procedure can also be used for parameter sweeps and optimizations.

For more information about Maple procedures, see the Procedures help topic in the Maple help system.

5.5 Generating and Exporting C Code from a Model

If you want to use or test your model in an application that supports the C programming language, you can use the **Code Generation Template** to translate your model or a subsystem in your model into C code. Access to the basic C code, and the ability to compile and run it, is available in Maple. Extensions of this code are available for a variety of software tools as additional Connector toolboxes.

With this template, you can define inputs and outputs for the system, set the level of code optimization, generate the source code, and choose the format of the resulting component and library code. You can use any Maple commands to perform task analysis, assign model equations to a variable, group inputs and outputs, and define additional input and output ports for variables.

Note: C code generation now handles all systems modeled in MapleSim, including hybrid systems with defined signal input (RealInput) and signal output (RealOutput) ports.

Whenever you export code or generate equations you often only see a subset of the parameters for that model. The following parameters cannot be exported:

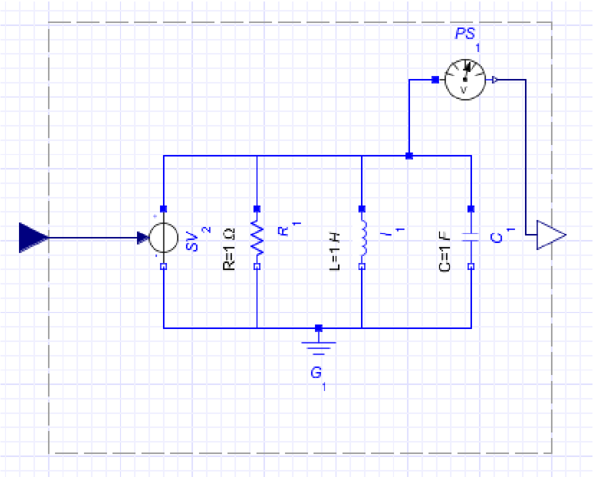
- Multibody parameters cannot be directly exported. Only user-generated parameters that the multibody parameters are assigned using Inspector tab are able to be exported.
- Dependent parameters cannot be exported. If the parameter A is a function of b (A=b, A=sin(b), A=1+3/b, etc.), then A will not be able to be exported. It will be directly substituted for in the equations as a function of b. You will be able to export b.
- Parameters that change the number of equations cannot be exported.
- Parameters for discrete values cannot be exported.

The process of generating C code from a MapleSim model consists of the following steps:

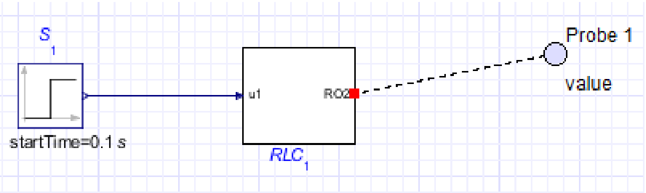
- Preparing the MapleSim model
- Attaching the code generation template
- Loading the subsystem
- Customizing, defining and assigning parameter values to specific ports
- Selecting the code generation options
- Generating and saving the C code

Preparing the Model for Export in MapleSim

The basic structure for exporting models is the subsystem where you define the input and output signals from the generated code. By creating a subsystem you also improve the visual layout of a system in the **Model Workspace**. The following figure shows a subsystem with a defined input (blue arrow) and a defined output (white arrow). When generating code for a subsystem, all ports must be defined as real input or real output ports.

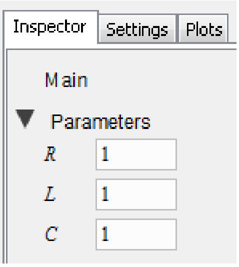


When generating code for the top-level system, there are no inputs, but all probed values are considered as outputs, as shown in the following figure.



Tip: If you want to generate code for your complete model, group all of the components at the top level of your model into a single subsystem.

In addition to inputs and outputs, generated code may have user-modifiable parameters defined for it. By default, not all parameters are selected to be modifiable in the exported code. In general, the fewer the parameters left modifiable, the less time it will take to generate and run the exported code. By default, only parameters defined in the exported subsystem are selected to be modifiable in the generated code. In the above example, generating code for the RLC subsystem, only the parameters R, L and C will default to being modifiable in the exported code.



R	1
L	1
C	1

Note: Since not all parameters are modifiable in the generated code, parameters that change the structure of the equations, by adding or removing variables from the system, are automatically removed from the list of parameters that can be exported. This is true even if the parameters are defined in the exported subsystem.

Initialization

All discrete events initialize to the same values as the corresponding MapleSim model. For example, if a clutch is initialized as 'locked' in the MapleSim model, then the generated code assumes that the clutch starts in the 'locked' configuration. The same is true for continuous variables and their derivatives.

Since exported code obtains its initial conditions from an initialized MapleSim model, that code may only be exported for subsystems that are part of a model that can be simulated.

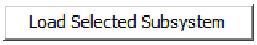
Note: If you are unable to run or initialize your model in MapleSim, you will not be able to export code for that model or any of its subsystems.

Attaching the Code Generation Template

To attach your model into the template, click **Templates** ( ) in the **Main Toolbar** and select **Code Generation**. In the Attachment field, enter a name for the template, and then click **Create Attachment**. Your model opens in the C Code Generation template in Maple.

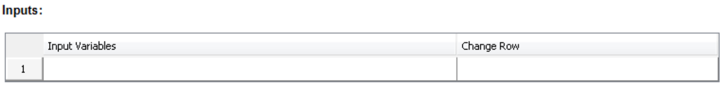
Loading the Subsystem

This part of the template identifies the subsystems that you want to generate and export code for. After selecting a subsystem, click **Load Selected Subsystem**. All defined input and output ports are loaded.



Customizing, Defining and Assigning Parameter Values to Specific Ports

The Port and Parameter Management interface lets you customize, define and assign parameter values to specific ports. Subsystem *Working with Maple Embedded Components (page 144)*components to which you assign the parameter inherit a parameter value defined at the subsystem level.

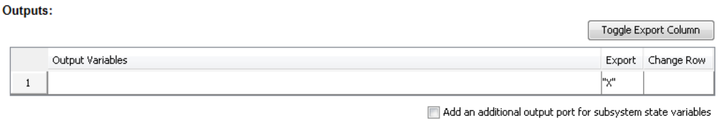


Inputs:

	Input Variables	Change Row
1		

Input Variables. Contains the model input variables.

Change Row. Select the equations with the specified row.



Outputs:

	Output Variables	Export	Change Row
1		"X"	

□ Add an additional output port for subsystem state variables

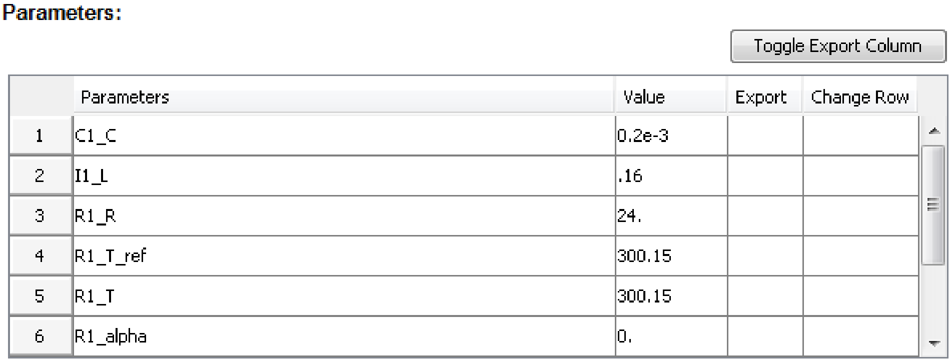
Toggle Export Column. Allows you to either select or remove all of the parameters for export.

Output Variables. Contains the model output variables.

Export. Enter "X" to select which variables you want to leave in the symbolic form.

Change Row. Select the equations with the specified row.

Add an additional out port. Select this option to add an additional port for the selected subsystem state variable.



Parameters:

	Parameters	Value	Export	Change Row
1	C1_C	0.2e-3		
2	I1_L	.16		
3	R1_R	24.		
4	R1_T_ref	300.15		
5	R1_T	300.15		
6	R1_alpha	0.		

Parameters. Contains the model parameters.

Value. Displays the value for the system parameter.

Export. Enter "X" to select which parameters you want to export in the symbolic form.

Change Row. Select the equations with the specified row.

After the subsystem is loaded you can group individual input and output variable elements into a vector array and add additional input and output ports for customized parameter values. Input ports can include variable derivatives and output ports can include subsystem state variables.

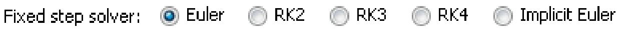
Note: If the parameters are not marked for export they will be numerically substituted.

Selecting the Code Generation Options

The Generation Options settings specify the advanced options for the code generation process.

Solver Options

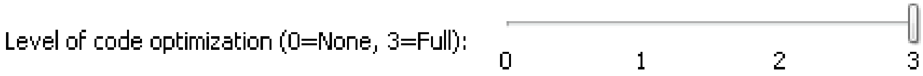
In this section you can specify the type of solver.



Fixed step solver: ● Euler ○ RK2 ○ RK3 ○ RK4 ○ Implicit Euler

Optimization Options

Set the level of code optimization to specify whether equations are left in their implicit form or converted to an ordinary differential equation (ODE) system during the code generation process. This option specifies the degree of simplification applied to the model equations during the code generation process and eliminates redundant variables and equations in the system.



Level of code optimization (0=None, 3=Full):

Select one of the following options:

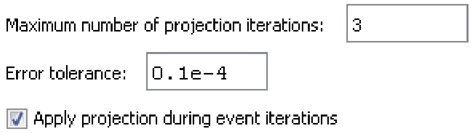
None (0): no optimization is performed; the default equations will be used in the generated code.

Partial (1, 2): removes redundant equations from the system.

Full (3): performs index reduction to reduce the system to an ODE system or a differential algebraic equation (DAE) system of index 1, and removes redundant equations.

Constraint Handling Options

The **Constraint Handling Options** specify whether the constraints are satisfied in a DAE system by using constraint projection in the generated file. Use this option to improve the accuracy of a DAE system that has constraints. If the constraint is not satisfied, the system result may deviate from the actual solution and could lead to an increase in error at an exponential rate.



Maximum number of projection iterations: 3

Error tolerance: 0.1e-4

☑ Apply projection during event iterations

Set the **Maximum number of projection iterations** to specify the maximum number of times that a projection is permitted to iterate to obtain a more accurate solution.

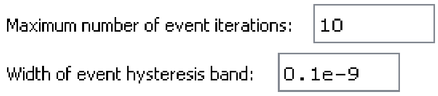
Set the **Error tolerance** to specify the desirable error tolerance to achieve after the projection.

Select **Apply projection during event iterations** to interpolate iterations to obtain a more accurate solution.

Constraint projection is performed using the **constraint projection** routine in the External Model Interface as described on The MathWorks™ web site to control the drift in the result of the DAE system.

Event Handling Options

The **Event Handling Options** section specifies whether the events are satisfied in a DAE system by using event projection in the generated file. Use this option to improve the accuracy of a DAE system with events. If the constraint is not satisfied, the system result may deviate from the actual solution and could lead to an increase in error at an exponential rate.



Maximum number of event iterations: 10

Width of event hysteresis band: 0.1e-9

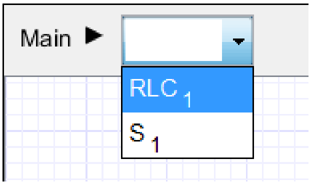
Set the **Maximum number of event iterations** to specify the maximum number of times that a projection is permitted to iterate to obtain a more accurate solution.

Set the **Width of event hysteresis band** to specify the desirable error tolerance to achieve after the projection.

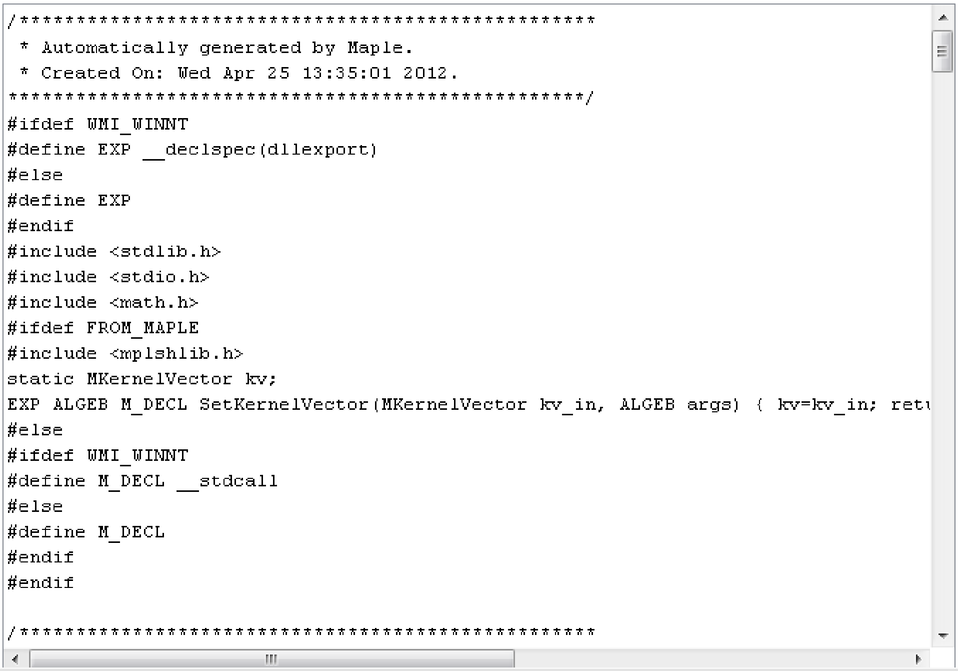
Generating and Saving the C code

To generate C code

1. In MapleSim, open the model for which you want to generate code.
2. In the **Model Workspace**, make sure that the components for which you want to generate code are grouped in a subsystem.
3. Click **Templates** (📎) in the **Main Toolbar**. The **Create Attachment** window appears.
4. From the list, select **Code Generation**.
5. In the **Attachment** field, enter a name for the template and click **Create Attachment**. Your model opens in the **Code Generation** template in Maple.
6. From the drop down list, select the subsystem for which you want to generate code. The subsystem and its contents appears in the **Subsystem Selection** window.



7. Click the **Load Selected Subsystem** located directly below the model diagram. The subsystem, along with all input and output variables are now loaded into the Code Generation template.
8. In the **C Code Generation Options** section of the template, choose the solver**.** By default, the Euler solver is selected.
9. Choose where you want to save the code and the name of the file. The file is automatically given a “c” prefix and a “c” extension.
10. Click **Generate C Code**. Once the C code is generated, the code can be viewed in the View C Code area at the bottom of the template. The C code is also automatically saved to your specified location.



```
/****************************************************
 * Automatically generated by Maple.
 * Created On: Wed Apr 25 13:35:01 2012.
***************************************************/
#ifdef WMI_WINNT
#define EXP __declspec(dllexport)
#else
#define EXP
#endif
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#ifdef FROM_MAPLE
#include <mplshlib.h>
static MKernelVector kv;
EXP ALGEB M_DECL SetKernelVector(MKernelVector kv_in, ALGEB args) { kv=kv_in; ret
#else
#ifdef WMI_WINNT
#define M_DECL __stdcall
#else
#define M_DECL
#endif
#endif

/****************************************************
```

5.6 Generating a Custom Component from External C Code/Library

MapleSim can call external code directly within your model. By using the External C Code/DLL template you can create a custom component to call external C Code and DLL functions directly into your model or subsystem. You can access the basic C code, and then compile and run the code in Maple. Extensions of this code are available for a variety of software tools as additional Connector toolboxes.

With this template, you can define the external inputs and outputs, specify the function name and arguments, generate the source code, and choose the format of the resulting component and library code. You can use any Maple commands to perform task analysis, assign model equations to a variable, group inputs and outputs, and define additional input and output ports for variables.

Changes to the parameters, inputs and outputs are remembered when you re-load your system using the External C Code/DLL template.

The process of creating an external code custom component for a MapleSim model consists of the following steps:

- Specify the custom component name

- Specify the location of the external C/library
- Define the external C/Library code options
- Specify the directory of the generated Modelica code
- Generate and save the external code custom component

Opening the External C Code/DLL Template

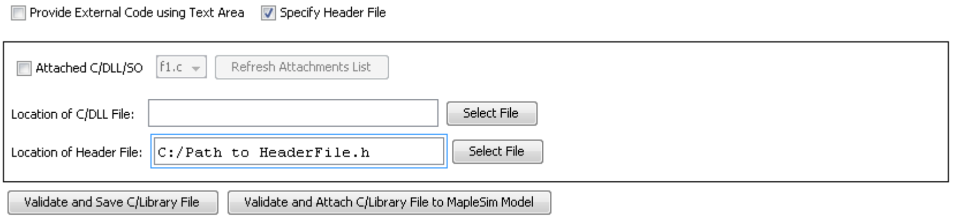
Click **Templates** ( ) in the **Main Toolbar** and select **External C/Library Block**. In the **Attachment** field, enter a name for the template, and then click **Create Attachment**. The template opens in Maple.

Specifying the C/Library Code Location and Options

Use the External C Code/DLL template to define the library code location, and/or validate and assign the code to a model. You can specify a header file, use an existing C or shared library file, or create a new C file using the text area.

Specifying a Header file

If required you can select **Specify Header File** and provide the location of the existing header file in the **Location of Header File** text box.



□ Provide External Code using Text Area ☑ Specify Header File

□ Attached C/DLL/SO f1.c

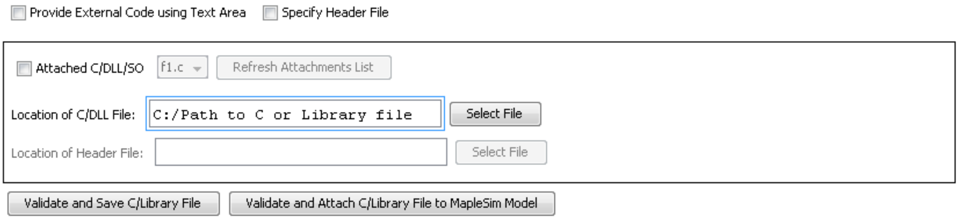
Location of C/DLL File:

Location of Header File: C:/Path to HeaderFile.h

Refresh Attachments List
Select File
Select File
Validate and Save C/Library File
Validate and Attach C/Library File to MapleSim Model

Using an Existing C or Library File

Provide the location of the existing C or Library file in the **Location of C/DLL File** text box.



□ Provide External Code using Text Area □ Specify Header File

□ Attached C/DLL/SO f1.c

Location of C/DLL File: C:/Path to C or Library file

Location of Header File:

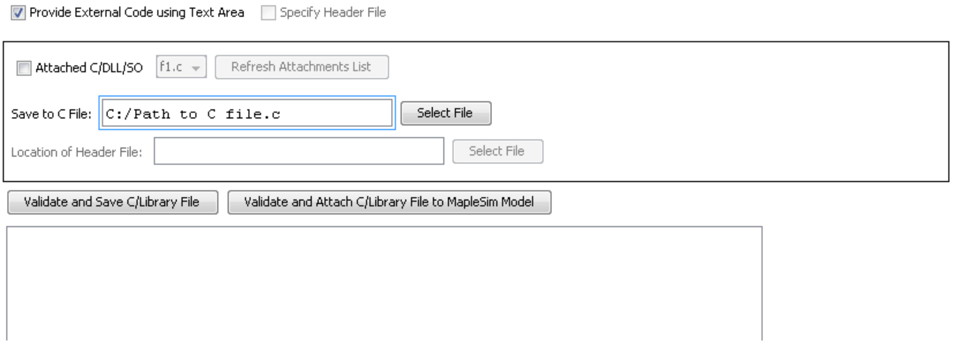
Refresh Attachments List
Select File
Select File
Validate and Save C/Library File
Validate and Attach C/Library File to MapleSim Model

Click **Validate and Save C/Library File** to verify the validity of the provided C or Library file.

Click **Validate and Attach C/Library File to MapleSim Mode**l to verify the validity of the provided C or Library file, and attach it to the MapleSim model. The attachment is saved in the **Project** > **Attachments** > **Other** folder.

Providing External Code in the Text Area

Select **Provide External Code using Text Area** to expand the text area. Enter the C code in the text area and specify the file location in the **Save to C File** text box.



☑ Provide External Code using Text Area □ Specify Header File

□ Attached C/DLL/SO f1.c

Save to C File: C:/Path to C file.c

Location of Header File:

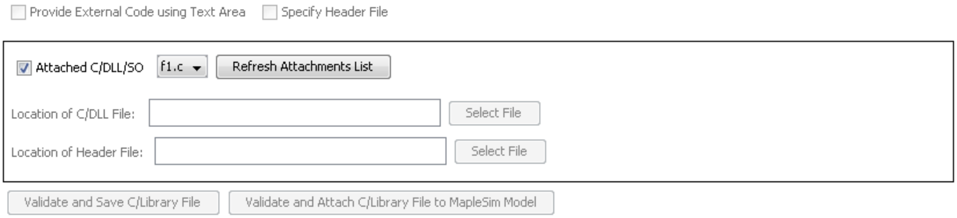
Refresh Attachments List
Select File
Select File
Validate and Save C/Library File
Validate and Attach C/Library File to MapleSim Model

Click **Validate and Save C/Library File** to verify the validity of the C file. The file is saved to the location specified in the **Save to C File** text box.

Click **Validate and Attach C/Library File to MapleSim Mode**l to verify the validity of the provided C file and attach it to the MapleSim model. The attachment is saved to the **Attachments** → **Other** subpalette located in the **Project** tab.

Using an Attached C or Library File

If there is an attached C or Library file you can select **Attached C/DLL/SO** and then choose the specific C or Library file from the drop down list.



□ Provide External Code using Text Area □ Specify Header File

☑ Attached C/DLL/SO f1.c

Location of C/DLL File:

Location of Header File:

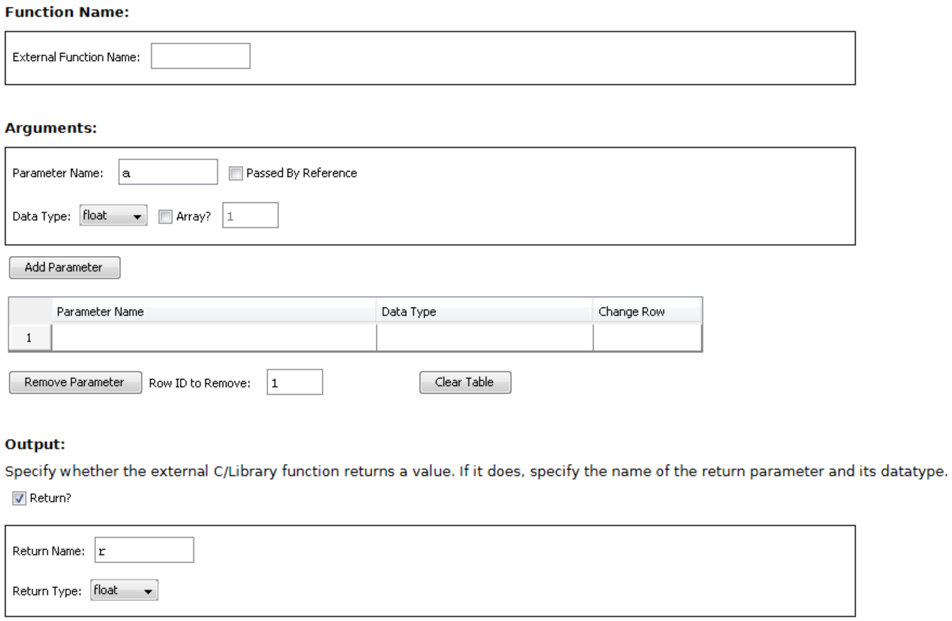
Refresh Attachments List
Select File
Select File
Validate and Save C/Library File
Validate and Attach C/Library File to MapleSim Model

Defining the C/Library Code Location and Options

You can define the external C/Library function name, specify the external C/Library prototype, choose the parameter name, data type, whether it is an array, and whether it is an input or an output of the external function. The specified inputs and outputs form the inputs and outputs of the custom component.

If necessary click **Remove Parameter** to remove a parameter by its row ID.

If necessary click **Clear Table** to reset the parameter table.



Function Name:

External Function Name:

Arguments:

Parameter Name: a □ Passed By Reference

Data Type: float □ Array? 1

	Parameter Name	Data Type	Change Row
1			

Row ID to Remove: 1

Output:

Specify whether the external C/Library function returns a value. If it does, specify the name of the return parameter and its datatype.

☑ Return?

Return Name: r

Return Type: float

For an example on how to create a external code custom component and its use, under the **Libraries** tab, browse to the **Examples** → **User's Guide Examples** menu, and then open the **Simple External Function** example.

Generating and Saving the C code

To generate the custom component

1. Specify where you want to save the generated Modelica custom component code.

Target directory: C:\Users\

2. Click **Generate External Code Component**. Once the custom component is generated, the code can be viewed in the Source Details area at the bottom of the template. The template is also automatically saved to the **Definitions → Custom Components** sub-palette located in the **Project** tab.

5.7 Working with Maple Embedded Components

Embedded Components are simple graphical interface elements that you embed into a worksheet or document to view, edit, create actions, display information, and analyze the properties of MapleSim models. You can also associate model properties with other Maple embedded components, including sliders and plots to create custom analysis tools.

For example, you can view and change parameter values using commands in the **DocumentTools** package. Model or subsystem equations can be retrieved using commands from the **MapleSim** package and you can manipulate your model as a **DynamicSystems** object to analyze the model or subsystem behavior using any input functions. Embedded Components are inserted using the Maple Components palette.

For more information about advanced analysis tasks, first open the **Sliding Table** example from the **Examples → User's Guide Examples** palette under the **Libraries** tab, and then open the **AdvancedAnalysis.mw** worksheet attachment (from MapleSim, under the **Project** tab, browse to the **Attachments → Documents** menu).

For more information about the MapleSim Model component, see the **MapleSimModel** topic in the Maple help system.

5.8 Working with a Maple Standard Worksheet

Once you create an attachment for your MapleSim model, you can further interact with and analyze your model in a Maple worksheet by using the application programming interface (API) commands. The MapleSim API is a collection of specialized procedures for manipulating, simulating, and analyzing a MapleSim model in a Maple standard worksheet.

There are two ways to link to a MapleSim model: linking to a MapleSim model using the LinkModel command or through the embedded component facility. The LinkModel command returns a 'connection module' that allows access to a MapleSim model. For more information on how to use the LinkModel command or the embedded component facility, see MapleSim Application Programming Interface Overview and the examples section in the LinkModel description.

6 MapleSim Tutorials

MapleSim Tutorials help you get started with MapleSim and learn about the key features, tools, templates and systems available in MapleSim, by leading you through a series of descriptive tasks, problems and examples using best practices. Many of these examples can be found in the **Examples → User's Guide Examples** palette in the **Libraries** tab on the left side of the MapleSim window, in the order that they are presented in the User's Guide.

In this chapter:

6.1 Tutorial 1: Modeling a DC Motor with a Gearbox

In this tutorial, you will extend a DC motor model and perform the following tasks:

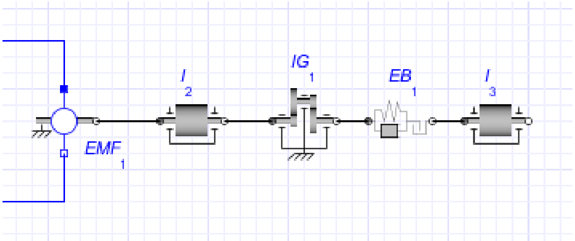
- Add a gearbox to the DC motor model
- Simulate the DC motor with gearbox model
- Group the DC motor components into a subsystem
- Assign global parameters to the model
- Add signal block components and a PI controller to the model
- Simulate the modified DC motor model using different conditions

Adding a Gearbox to a DC Motor Model

In this example, you will build the gearbox by adding and connecting an ideal gearbox component, a backlash component with a linear spring and damper, and an inertia component from the 1-D Mechanical library. You can use the selection tool to drag and position components in the **Model Workspace**.

To add a gearbox

1. In the **Libraries** tab, expand the **Examples** palette, expand the **User's Guide Examples** menu, and then open the **Simple DC Motor** example.
2. Delete the existing probe from the workspace.
3. Perform the following tasks:
   - From the **1-D Mechanical** → **Rotational** → **Bearings and Gears** menu, add an **Ideal Gear** component to the **Model Workspace** and place it to the right of the **Inertia** component.
   - From the **1-D Mechanical** →**Rotational** → **Springs and Dampers** menu, add an **Elasto-Backlash** component to the **Model Workspace** and place it to the right of the **Ideal Gear** component.
   - From the **1-D Mechanical** → **Rotational** → **Common** menu, add another **Inertia** component to the **Model Workspace** and place it to the right of the **Elasto-Backlash** component.
4. Connect the components as shown below.

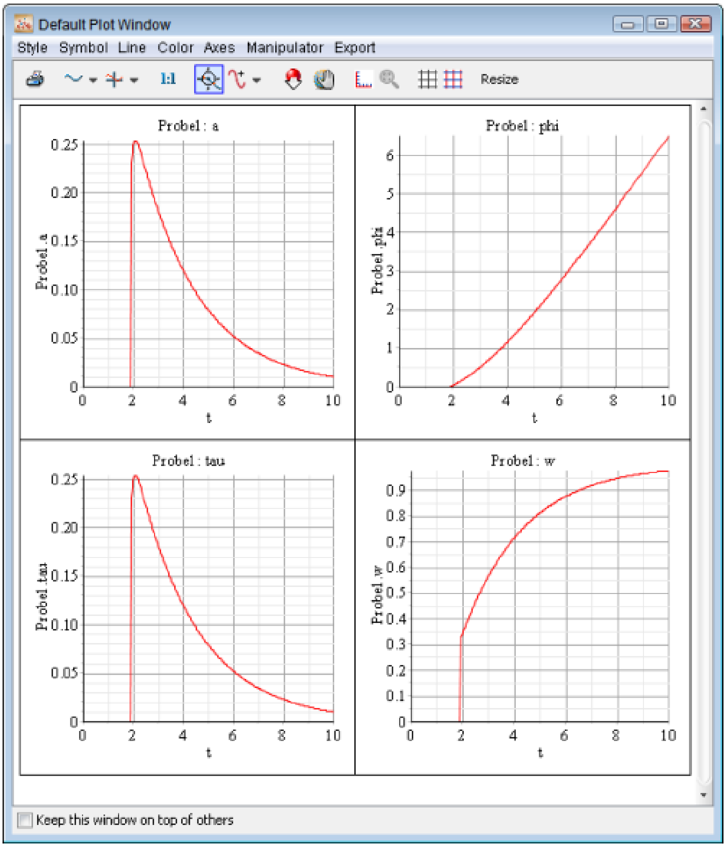


5. In the **Model Workspace**, click the **Ideal Gear** component.
6. In the **Inspector** tab, change the transmission ratio, **r**, to **10** and then press **Enter** to accept the value.
7. Specify the following parameter values for the other components:
   - For the **Elasto-Backlash** component, in the **b** field, change the total backlash value to **0.3**rad. In the **d** field, change the damping constant to $\mathbf{10^4}\,\frac{N\cdot m\cdot s}{rad}$.
   - For the first **Inertia** component ($\mathbf{I_2}$), in the **J** field, change the moment of inertia value to **10**$kg\cdot m^2$.
   - For the second **Inertia** component ($\mathbf{I_3}$), in the **J** field, change the moment of inertia value to **1**$kg\cdot m^2$.
   - For the **Step** source, in the **height** field, change the value to **100**.

Simulating the DC Motor with the Gearbox Model

To simulate the DC motor

1. From the **Model Workspace Toolbar**, click **Attach Probe** ( ).
2. Hover your mouse pointer over the line that connects the **Elasto-Backlash** component and the second **Inertia** component (**I_3**). The line is highlighted.
3. Click the line once, and then click a spot in the workspace to anchor the probe.
4. Select the probe in the **Model Workspace**.
5. To include the angle (phi), speed (w), acceleration (a), and torque (tau) values in the simulation graphs, in the **Inspector** tab, select **Angle**, **Speed**, **Acceleration**, and **Torque**.
6. Click a blank area in the **Model Workspace**.
7. Under the **Settings** tab, set the **t_d** parameter to **10** seconds and press **Enter**.
8. Click **Run Simulation** (▶) in the **Main Toolbar**. When the simulation is complete, the following graphs appear.

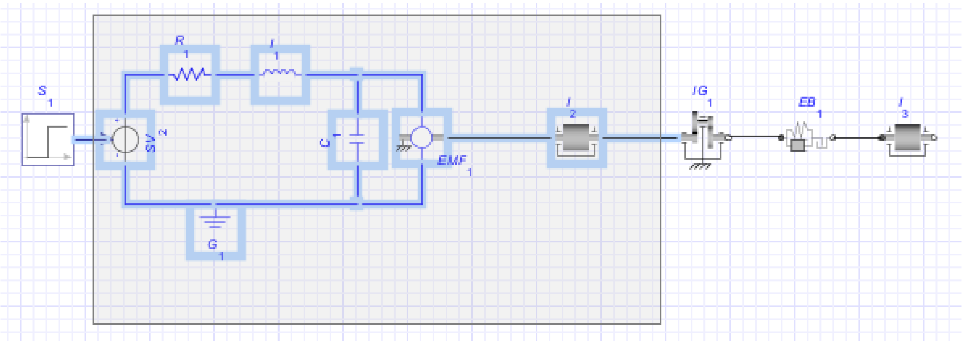


9. To verify the results, in the **Libraries** tab, expand the **Examples** palette, expand the **User's Guide Examples** menu, and then open the **DC Motor with Gearbox** example.

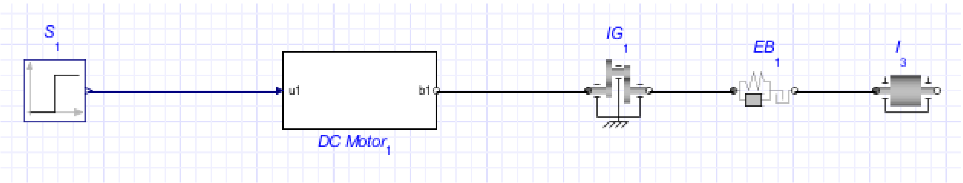
Grouping the DC Motor Components into a Subsystem

To group the DC motor components

1. Using the **Selection Tool** ( ), draw a box around the electrical components and the first inertia component.



2. From the **Edit** menu, select **Create Subsystem**.
3. In the **Create Subsystem** dialog box, enter **DC Motor**.
4. Click **OK**. A white block, which represents the DC motor, appears in the **Model Workspace**.



Tip: To view the components in the subsystem, double-click the **DC Motor** subsystem in the **Model Workspace**. To browse to the top level of the model, click **Main** (Main ▶) in the **Navigation Toolbar**.

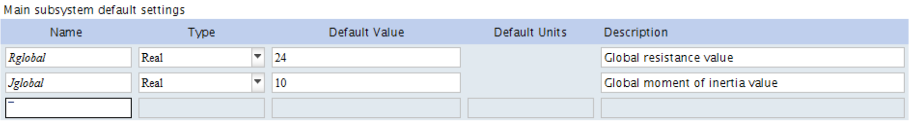
Assigning Global Parameters to a Model

You can define a global parameter and assign its value to multiple components in your model.

To assign global parameters

1. Click **Main** in the **Navigation Toolbar** to browse to the top level of the model.
2. From the **Navigation Toolbar**, click **Parameters** (▦) to switch to the parameter editor view.
3. In the first row of the **Main subsystem default settings** table, define a parameter called **Rglobal** and press **Enter**.
4. Specify a default value of **24** and enter **Global resistance value** as the description.
5. In the second row of the table, define a parameter called **Jglobal** and press **Enter**.

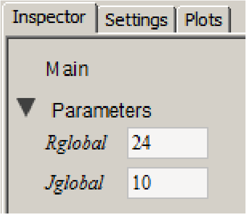
6. Specify a default value of **10** and enter **Global moment of inertia value** as the description.



Main subsystem default settings

Name	Type	Default Value	Default Units	Description
Rglobal	Real	24		Global resistance value
Jglobal	Real	10		Global moment of inertia value

7. To switch to the diagram view, click **Diagram** () in the **Navigation Toolbar**. The new **Rglobal** and **Jglobal** parameters appear in the **Inspector** tab. You can now assign these parameter values to other components in your model.



Main

Parameters

Rglobal 24

Jglobal 10

8. In the **Navigation Toolbar**, click **Parameters** ().
9. In the $\mathbf{I_3}$ **component** table, in the **Value** field for the moment of inertia parameter, enter **Jglobal** and press **Enter**. The moment of inertia parameter now inherits the numeric value of the global parameter **Jglobal** (in this example, **10**).
10. Switch to the diagram view and double-click the **DC Motor** subsystem.
11. In the **Navigation Toolbar**, click **Parameters** ().
12. In the $\mathbf{EMF_1}$ **component** table, in the **Value** field for the transformation coefficient, enter **Rglobal·Jglobal** and press **Enter**.

Note: This value is an approximation of the transformation coefficient.

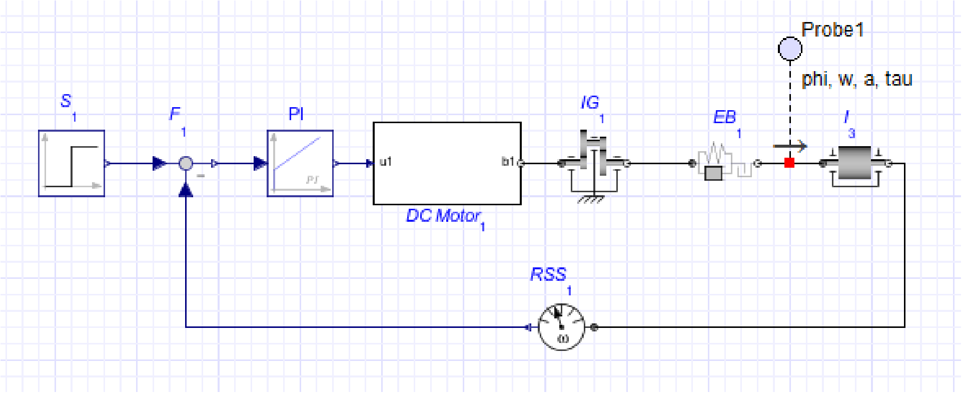
13. In the $\mathbf{R_1}$ **component** table, in the **Value** field for the resistance parameter, enter **Rglobal** and press **Enter**.
14. Switch to the diagram view and browse to the top level of your model.
15. Save the model as **DC_Motor2.msim**.

Changing Input and Output Values

In this example, you will change the input and output values of the model to simulate different conditions.

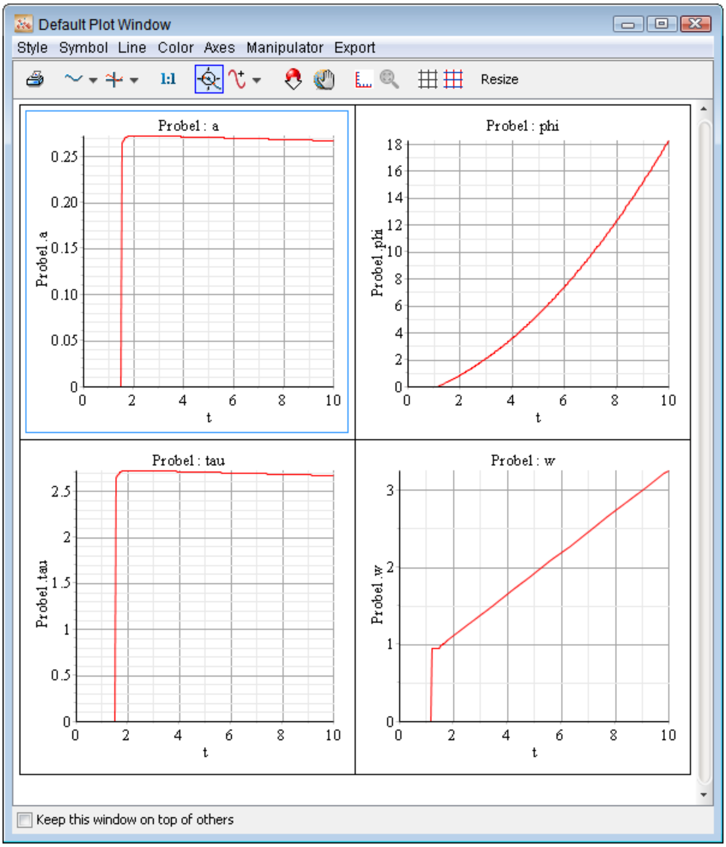
To change input and output values

1. Under the **Libraries** tab, browse to the **1-D Mechanical** → **Rotational** → **Sensors** menu and then add the **Angular Velocity Sensor** component to the **Model Workspace** and place it below the gearbox components.
2. Right-click (**Control**-click for Macintosh) the **Angular Velocity Sensor** component and select **Flip Horizontal**.
3. Delete the connection line between the **Step** source and the **DC Motor** subsystem.
4. From the **Signal Blocks** → **Controllers** menu, add the **PI** component to the **Model Workspace** and place it to the left of the **DC Motor** subsystem.
5. From the **Signal Blocks** → **Mathematical** → **Operators** menu, add the **Feedback** component to the **Model Workspace** and place it to the left of the **PI** component.
6. Connect the components as shown below.



To draw a perpendicular line, click a point in the **Model Workspace** to anchor the line and then move your mouse cursor in a different direction to draw the second line segment.

7. Click the **PI** component in the **Model Workspace**.
8. In the **Inspector** tab, specify a gain of **20** in the **k** field, and a time constant of **3** seconds in the **T** field.
9. Simulate the model again. When the simulation is complete, the following graphs appear.



10. Save the model as **DC_Motor3.msim**.

11. To verify the results, in the **Libraries** tab, expand the **Examples** palette, expand the **User's Guide Examples** menu, and then open the **DC Motor with Gearbox and PI Controller** example.

6.2 Tutorial 2: Modeling a Cable Tension Controller

In this tutorial, you will extend the DC motor example to model a cable that is stretched with a pre-defined tension. The tension is defined by a **Constant** source and the **PI** controller provides the voltage to drive the motor. You will perform the following tasks:

- Build a cable tension controller model
- Specify component properties
- Simulate the cable tension controller model

Building a Cable Tension Controller Model

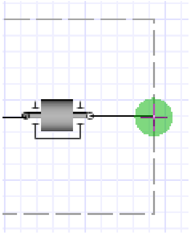
In this example, you will build the cable tension controller model using a combination of 1-D mechanical rotational and translational components. You will also group components into a **Gear** subsystem and add subsystem ports.

To build the cable tension controller

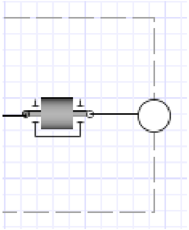
1. Open the **DC_Motor3.msim** file that you created in the previous tutorial and save the file as **Cable_Tension.msim**.
2. Delete the probe attached to the line that connects the **Elasto-Backlash** and **Inertia** components.
3. Delete **Angular Velocity Sensor** component and its connection lines.
4. Select the **Elasto-Backlash**, **Ideal Gear**, and **Inertia** components and group them into a subsystem called **Gear Components**.
5. Add the following components to the **Model Workspace**:
    - From the **1-D Mechanical → Rotational → Bearings and Gears** menu, add the **Ideal Rotation to Translation Gear** component and place it to the right of the **Gear Components** subsystem.
    - From the **1-D Mechanical → Translational → Sensors** menu, add the **Force Sensor** component and place it to the right of the **Ideal Rotation to Translation Gear** component.
    - From the **1-D Mechanical → Translational → Springs and Dampers** menu, add the **Translational Spring** component and place it to the right of the **Force Sensor** component.
    - From the **1-D Mechanical → Translational → Common** menu, add the **Translational Fixed** component and place it to the right of the **Translational Spring** component.
6. Right-click (**Control**-click for Macintosh) the **Translational Fixed** component in the **Model Workspace** and select **Rotate Counterclockwise**.
7. Delete the **Step** source and replace it with a **Constant** source from the **Signal Blocks → Sources → Real** menu.

Tip: You can connect the **Constant** source by dragging it onto the unconnected line end.

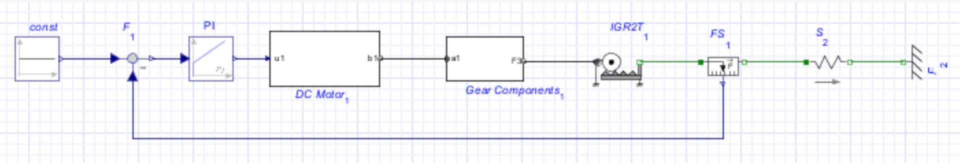
8. Double-click the **Gear Components** subsystem. You will now add a port to connect this subsystem with other components.
9. Click the negative (white) flange of the **Inertia** component and drag your mouse cursor to the boundary that surrounds the subsystem components.



10. Click the line once. The subsystem port is added to the line.



11. Click **Main** in the **Navigation Toolbar** to browse to the top level of your model.
12. Connect the components as shown below.



Specifying Component Properties

To specify component properties

1. In the **Model Workspace**, double-click the **Gear Components** subsystem.
2. Specify the following parameter values for the subsystem components:
   - For the **Ideal Gear** component, change the transmission ratio value to **0.01**.
   - For the **Inertia** component, change the moment of inertia value to **0.1**$kg{\cdot}m^2$.
3. Click **Main** in the **Navigation Toolbar** to browse to the top level of the model.
4. Specify the following parameter values for the other components:
   - For the **Translational Spring** component, in the **c** field, change the spring constant value to **210**$\cdot$**10**$^9\,\frac{N}{m}$.
   - For the **PI** controller, change the **T** value to **0.1**s.

- For the **Constant** source, change the constant output value **k** to **77.448**.

Simulating the Cable Tension Controller

To simulate the cable tension controller

1. Click **Attach Probe** ( ).
2. Click the line that connects the **Feedback** and **PI** components and then click the workspace to position the probe.
3. Select the probe in the **Model Workspace**.
4. In the **Inspector** tab, select the **Real** quantity and change its name to **Error**.
5. Add another probe that measures the **Real** quantity to the line connecting the **PI** component and **DC Motor** subsystem. Change the quantity name to **Controller**.

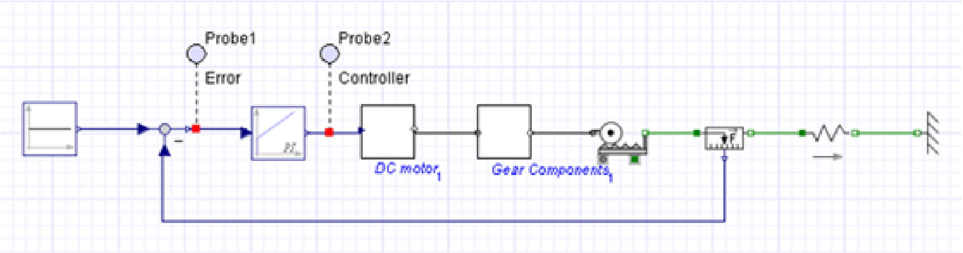
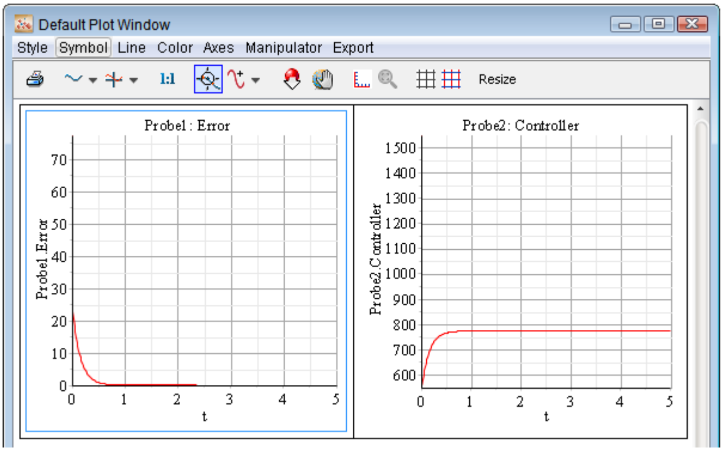


Figure 6.1: Cable Tension Controller

6. Click a blank area in the **Model Workspace**.
7. In the **Settings** tab, specify the following parameters:
   - Set the simulation duration time, $\mathbf{t_d}$, to **5**s.
   - Select **Fixed** from the **Solver Type** drop-down menu.
   - Select **Implicit Euler** from the **Solver** drop-down menu.
8. Click the **Run Simulation** (▶) in the **Main Toolbar**. When the simulation is complete, the following graphs appear.



9. Save the file as **CableTension.msim**.

6.3 Tutorial 3: Modeling a Nonlinear Damper

In this tutorial, you will model a nonlinear damper with a linear spring. This tutorial builds upon the concepts demonstrated in the previous tutorials. You will perform the following tasks:

- Generate a custom spring damper defined by differential equations
- Provide custom damping coefficient values as input signals
- Build the nonlinear damper with linear spring model
- Assign a variable to a subsystem
- Simulate the nonlinear damper with linear spring model

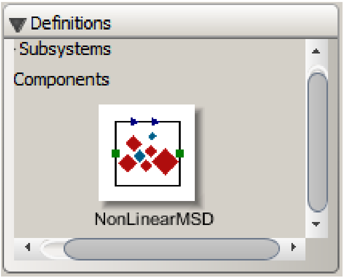
Generating a Spring Damper Custom Component

In MapleSim, you can create a custom component that is based on a mathematical model. Typically, you would define the component equations and ports before making the component available in MapleSim. For the the purpose of this tutorial, you will generate a sample custom component with pre-defined differential equations.

To generate a spring damper custom component

1. Open a new MapleSim document.
2. In the **Main Toolbar**, click **Templates** ( ) .
3. Click **Browse...** and in the **Browse Templates** dialog box, open the **Component Templates** folder.

4. Select the **NonLinearMSD.mw** file and click **Use Template**. The Create Attachment window appears.
5. In the **Attachment** field, enter **NonLinearMSD**.
6. Click **Create Attachment**. The Custom Component Template opens in Maple.
7. To generate the custom component, click **Generate MapleSim Component** at the bottom of the worksheet. In MapleSim, the **NonLinearMSD** custom component appears under the **Project** tab, in the **Definitions** → **Components** palette, on the left side of the MapleSim window. You will use this component later in this tutorial.

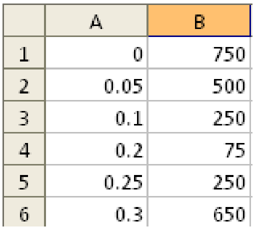


Providing Damping Coefficient Values

You can provide custom values for interpolation table components that you add to your model. In this example, you will provide damping coefficient values in an external file.

To create damping coefficient values

1. Create either a Microsoft Excel spreadsheet (.xls) or comma-separated values (.csv) file that contains the following values:



	A	B
1	0	750
2	0.05	500
3	0.1	250
4	0.2	75
5	0.25	250
6	0.3	650

The first column contains values for the relative displacement of the damper and the second column contains values for the damping coefficients.

2. Save the file as **DamperCurve.xlsx** or **DamperCurve.csv**.
3. In MapleSim, expand the **Attachments** palette located under the **Project** tab.
4. Right-click (**Control**-click for Macintosh) **Data Sets** and select **Attach File**.

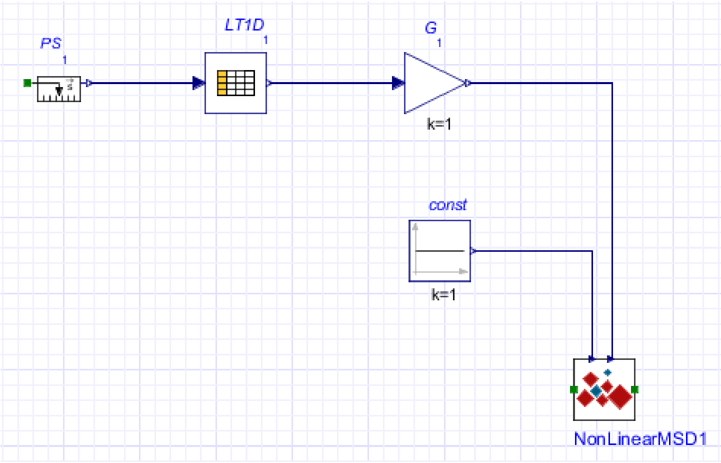
5. Browse to and select the Excel spreadsheet or .csv file that you created, and click **Attach...**. The file containing the data set is attached to your model. You will use this file in the next task.

Building the Nonlinear Damper Model

In this example, you will build the nonlinear damper using components from the component library.

To build the nonlinear damper

1. From the **Definitions** → **Components** palette, drag the **NonLinearMSD** custom component into the **Model Workspace**.
2. Add the following components to the **Model Workspace**:
   - From the **Signal Blocks** → **Mathematical** → **Operators** menu, add a **Gain** component and place it above the **NonLinearMSD** component.
   - From the **Signal Blocks** → **Sources** → **Real** menu, add a **Constant** component and place it between the **NonLinearMSD** and **Gain** components.
   - From the **Signal Blocks** → **Interpolation Tables** menu, add a **1D Lookup Table** component and place it to the left of the **Gain** component.
   - From the **1-D Mechanical** → **Translational** → **Sensors** menu, add a **Position Sensor** component and place it to the left of the **1D Lookup Table** component.
3. Connect the components as shown below.



4. Add the following components to the **Model Workspace**:
   - From the **1-D Mechanical** → **Translational** → **Common** menu, add a **Translational Fixed** component, place it to the right of the **NonLinearMSD** component, and then rotate it counterclockwise.
   - From the same menu, add **Mass** and **Force** components and place them to the left of the **Position Sensor** component.
   - From the **Signal Blocks** → **Sources** → **Real** menu, add a **Step** source.
5. Connect the components as shown below.

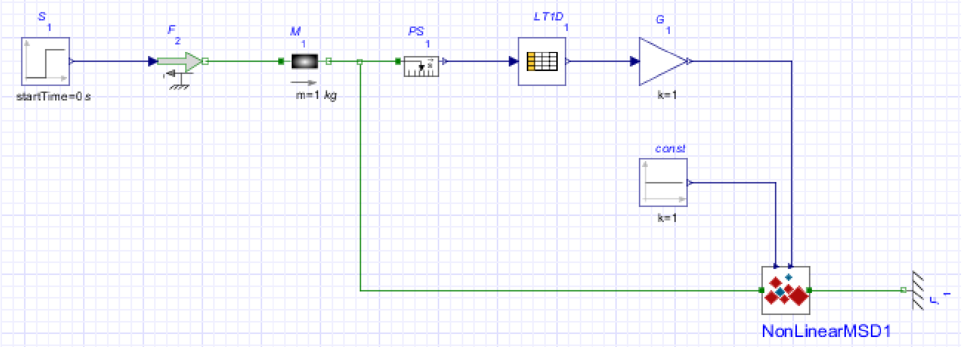
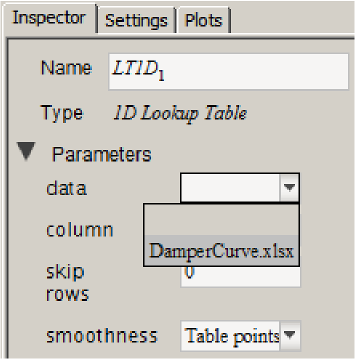


Figure 6.2: Nonlinear Damper Model

6. In the **Model Workspace**, select the **1D Lookup Table** component. In the **Inspector** tab, the **data** drop-down menu lists all of the documents that you have attached to the model.



Name $LT1D_1$

Type *1D Lookup Table*

Parameters

data

column

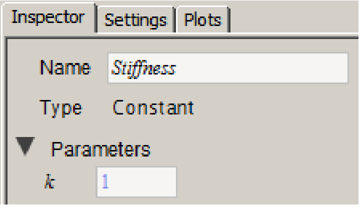
skip rows 0

smoothness Table points

DamperCurve.xlsx

7. Select the **DamperCurve.xlsx** or **DamperCurve.csv** file that you created in the previous task. You will now define the stiffness of the spring.
8. In the **Model Workspace**, select the **Constant** component.

9. In the **Inspector** tab, in the **Name** field, change the component name to **Stiffness**.



Name *Stiffness*

Type Constant

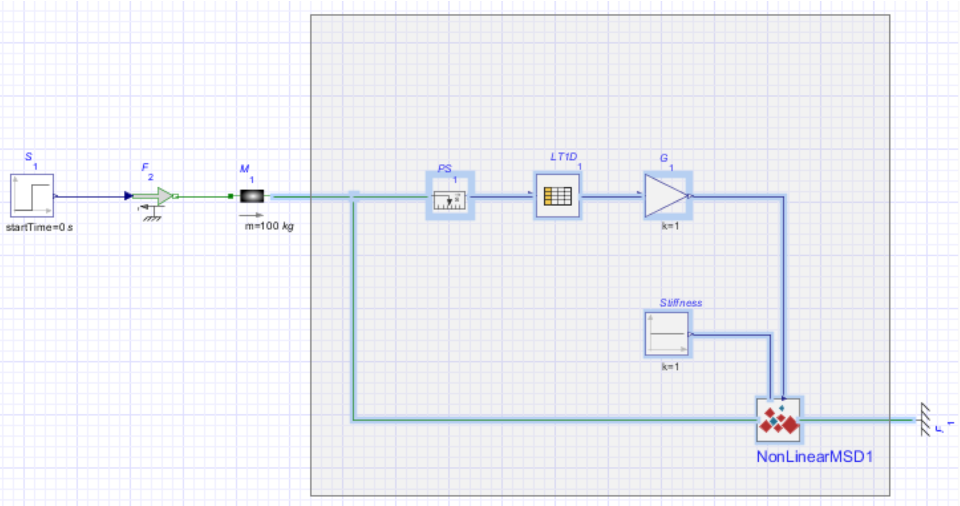
▼ Parameters

k 1

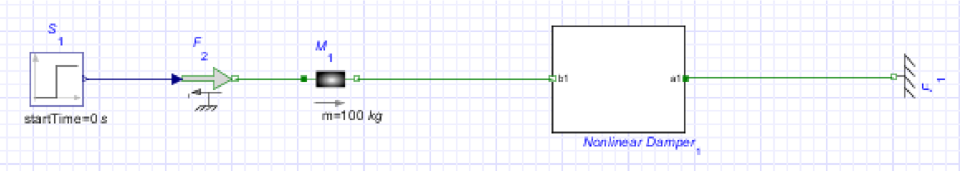
10. Select the **Step** component and change the step height to **100**.

11. Select the **Mass** component and change the mass, **m**, to **100kg**.

12. Using the **Selection Tool** ( ), draw a box around all of the components in the nonlinear damper model.



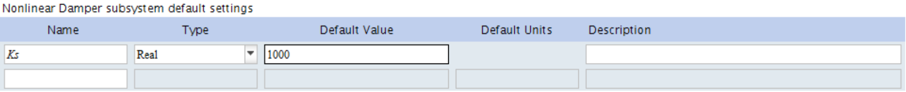
13. Group the selected components into a subsystem called **Nonlinear Damper**. The complete model is shown below.



Assigning a Parameter to a Subsystem

To assign a parameter to a subsystem

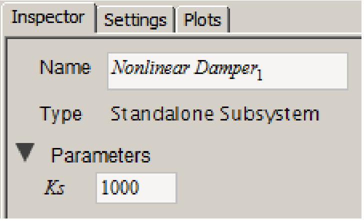
1. In the **Model Workspace**, double-click the **Nonlinear Damper** subsystem.
2. In the **Navigation Toolbar**, click **Parameters** ().
3. In the first row of the **Nonlinear Damper subsystem default settings** table, define a spring constant parameter called **Ks** and press **Enter**.
4. In the same row, specify a default value of **1000** and enter **Spring constant** as the description. You can now assign the parameter value **Ks** to other components in the **Nonlinear Damper** subsystem.



Nonlinear Damper subsystem default settings

Name	Type	Default Value	Default Units	Description
Ks	Real	1000		

5. In the **Navigation Toolbar**, click **Diagram** (). The **Ks** parameter appears as a field in the **Inspector** tab with the defined default value.



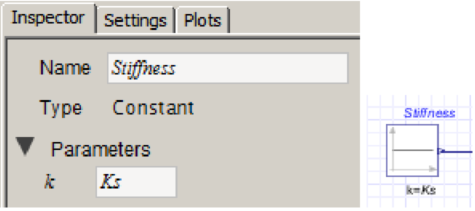
Name *Nonlinear Damper$_1$*

Type Standalone Subsystem

▼ Parameters

Ks 1000

6. In the **Model Workspace**, select the **Stiffness** component and change the constant output parameter, **k**, to **Ks**. This component now inherits the numeric value of **Ks** (in this example, **1000**). Therefore, if you edit the numeric value of **Ks** at the subsystem level, the **k** parameter that has been assigned the variable, **Ks**, also inherits that change.



Name *Stiffness*

Type Constant

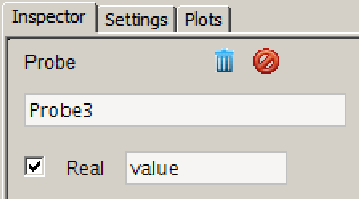
▼ Parameters

k *Ks*

Simulating the Nonlinear Damper with Linear Spring Model

To simulate the Nonlinear Damper

1. In the **Model Workspace**, double-click the **Nonlinear Damper** subsystem.
2. From the **Model Workspace Toolbar**, click **Attach probe** ( ). The cursor changes to the probe icon when you move into the workspace.
3. To attach the probe click the line that connects the **Gain** and **NonLinearMSD** custom component and then click a spot in the workspace to anchor the probe.
4. In the **Model Workspace**, select the probe. The Inspector tab opens.

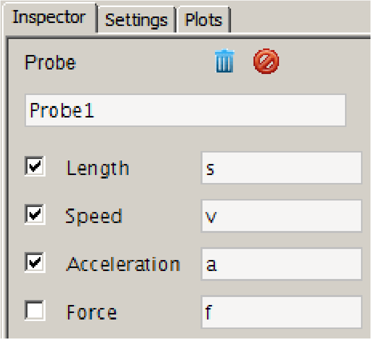


Probe

Probe3

☑ Real value

5. In the **Inspector** tab, select the **Real** quantity and change its name to **Damping**.
6. Click **Main** to browse to the top level of the model.
7. Add a probe to the line that connects the **Mass** component and the **Nonlinear damper** subsystem.
8. Select the probe added in the previous step, and have it measure the length, speed, and acceleration quantities.



Probe

Probe1

☑ Length s

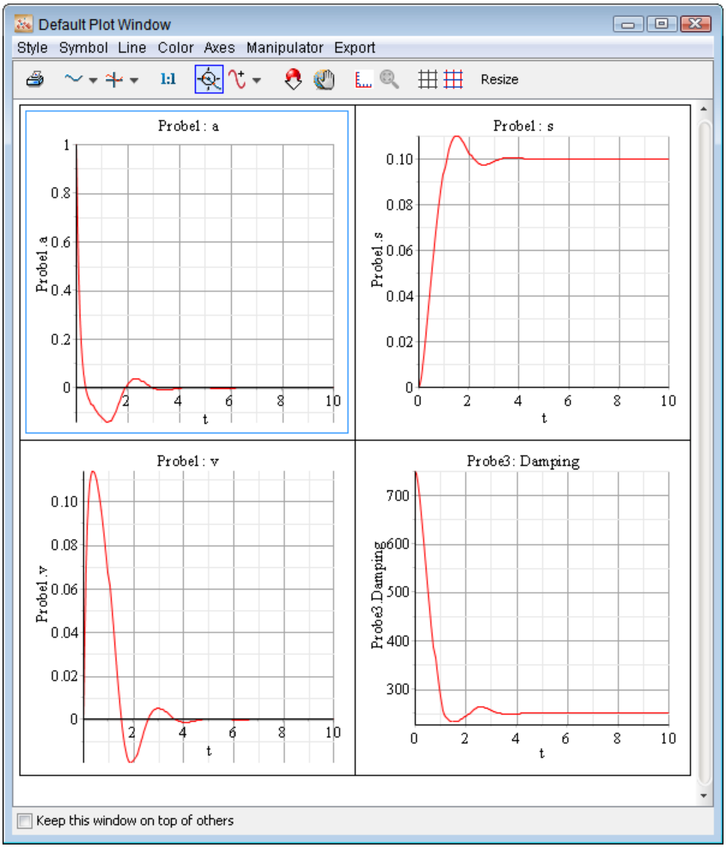
☑ Speed v

☑ Acceleration a

□ Force f

9. Click a blank area in the **Model Workspace**. The length, speed, and acceleration quantities (s, v, a) appear beside the probe.
10. Under the **Settings** tab, set the $\mathbf{t_d}$ parameter to **10** seconds.
11. Under the **Plots** tab, set the **Columns** parameter to **2**.

12. Click **Run Simulation** ( ) in the **Main Toolbar**. When the simulation is complete, the following graphs appear.



13. Save the file as **NonLinearMSD.msim**.

6.4 Tutorial 4: Modeling a Planar Slider-Crank Mechanism

Using components from the Multibody mechanical library, you will model the planar slider-crank mechanism shown in the following figure.

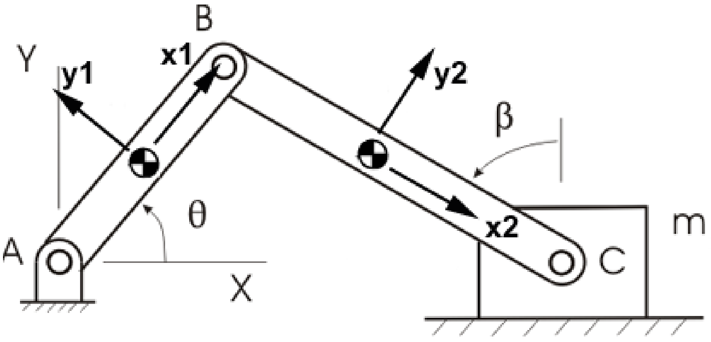


Figure 6.3: Planar Slider-Crank Mechanism

This model consists of a revolute joint, *A*, which is attached to a planar link. This planar link is attached to a connecting rod by a second revolute joint, *B*. The connecting rod connects to a sliding mass by a third revolute joint, *C*, and the sliding mass connects to ground by a prismatic joint. In practice, this mechanism converts rotational motion at the crank to translational motion at the sliding mass or vice versa. For the system shown in the diagram, gravity is assumed to be the only external force, acting along the negative Y axis (the y axis for the inertial frame).

In this tutorial, you will perform the following tasks:

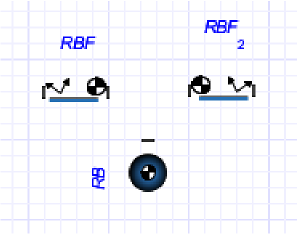
- Create a planar link subsystem
- Define and assign subsystem parameters
- Create the crank and connecting rod elements
- Add the fixed frame, sliding mass, and joint elements to the model
- Specify initial conditions
- Simulate the planar slider-crank mechanism

Creating a Planar Link Subsystem

The diagram above shows that the slider-crank has two associated planar links: the crank (the link from point A to B) and the connecting rod (the link from B to C). In both cases, these links have their longitudinal axis along their local x axis (**x1** and **x2**, respectively). Therefore, you will first create a generic planar link with two ports. The inboard port (base) will be located $-\frac{L}{2}$ units along the x axis of the link, and the outboard port (tip) will be located $\frac{L}{2}$ units along the x axis of the link. In this example, *L* refers to the length of the link and the center-of-mass is assumed to be in the middle of the link.

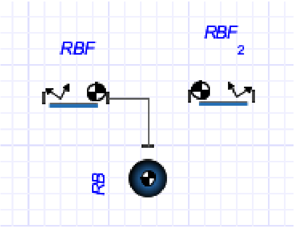
To create a planar link subsystem

1. Open a new MapleSim document.
2. Under the **Libraries** tab, browse to the **Multibody** → **Bodies and Frames** menu, and then add two **Rigid Body Frame** components and a **Rigid Body** component.
3. In the **Model Workspace**, right-click (**Control**-click for Macintosh) one of the **Rigid Body Frame** components, and then select **Flip Horizontal**.
4. Right-click (**Control**-click for Macintosh) the **Rigid Body** component, and then select **Rotate Counterclockwise**.
5. Drag the components in the arrangement shown below. You can now connect the components.

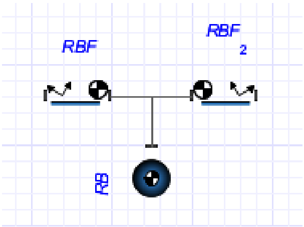


Note: The labels for your components may differ from the labels in the preceding figure (that is, **RB**, **RBF**, and **RBF$_2$**). You can change the labels in your model by selecting the component, and then entering the new label in the **Name** field under the **Inspector** tab. For this tutorial, the labels shown in the preceding figure will be used when referring to specific components.

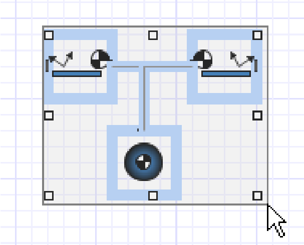
6. Draw a connection line between the **RB** component and the right frame of the **RBF** component.



7. Draw another connection line between the **RB** component and the left frame of the **RBF$_2$** component.



8. Using the **Selection Tool** ( ), draw a box around the components.



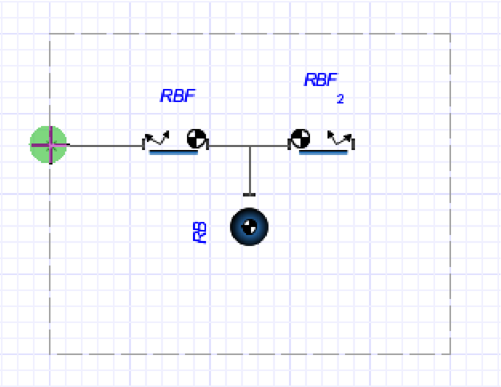
9. From the **Edit** menu (or right-click the highlighted components) select **Create Subsystem**.

10. In the **Create Subsystem** dialog box, enter **Link**, and then click **OK**.

You will now add ports to connect this subsystem to other components.

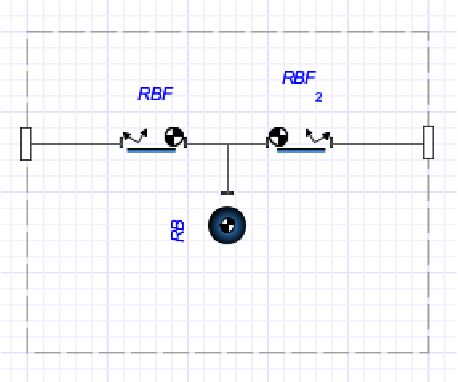
11. Double-click the **Link$_1$** subsystem.

12. Click the left frame of the **RBF** component and drag your mouse pointer to the left of the subsystem boundary.



13. Click the line once. A subsystem port is added.

14. In the same way, using the right frame of the **RBF$_2$** component, create another port on the right side of the subsystem boundary.



Defining and Assigning Parameters

In this task, you will define a subsystem parameter, *L*, to represent the length of the link and assign the parameter value as a variable to the parameters of the **Rigid Body Frame** components. The **Rigid Body Frame** components will then inherit the numeric value of *L*.

To define and assign parameters

1. Double-click the **Link$_1$** subsystem and in the **Navigation Toolbar**, click **Parameters** (▦), or from the **Inspector** tab click **Add or Change Parameters**. The **Link subsystem default settings** window appears.
2. In the first row of the **Link subsystem default settings** table, define a parameter called **L**, and press **Enter**.
3. Specify a default value of **1** and enter **Length** as the description.
4. Specify the following parameter values (to enter a fraction, use the forward slash key(/)):
   - For the **RBF** component, in the **Value** field for $\overline{r}_{XYZ}$, specify a position offset of

     $\left[-\frac{L}{2}, 0, 0\right]$, and then select **m** from the **Units** drop-down menu.
   - For the **RBF$_2$** component, in the **Value** field for $\overline{r}_{XYZ}$, specify a position offset of

     $\left[\frac{L}{2}, 0, 0\right]$, and then select **m** from the **Units** drop-down menu.

Creating the Crank and Connecting Rod Elements

In this task, to create the crank and connecting rod elements, you will add a **Link** subsystem definition to your model and create **Crank** and **ConnectingRod** shared subsystems. You will also assign a different length value to the connecting rod element.

To create the crank and connecting rod elements

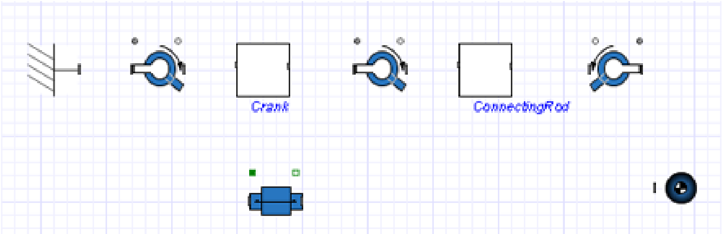
1. Click **Main** in the **Navigation Toolbar** to browse to the top level of your model. The **Link$_1$** subsystem appears in the **Model Workspace**.
2. Right-click (**Control**-click for Macintosh) the **Link$_1$** subsystem and select **Convert to Shared Subsystem**. The **Create Shared Subsystem** window appears. Click **OK**. A **Link** subsystem definition is added to the **Definitions** → **Subsystems** palette under the **Project** tab and the **Link** subsystem in the **Model Workspace** is converted to a shared subsystem.
3. Select the **Link** shared subsystem in the **Model Workspace** and in the **Inspector** tab, in the **Name** field, change the name of the shared subsystem to **Crank**.
4. From the **Definitions** → **Subsystems** palette, drag the **Link** icon to the **Model Workspace** and place it to the right of the **Crank** shared subsystem.
5. In the **Model Workspace**, select the second copy of the **Link** shared subsystem.
6. In the **Inspector** tab, change the shared subsystem name to **ConnectingRod** and change the length value to **2**.

Adding the Fixed Frame, Sliding Mass, and Joint Elements

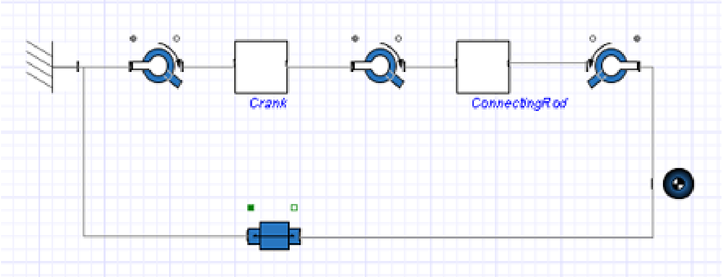
In this task, you will add a **Fixed Frame** component, a **Rigid Body** component to represent the sliding mass, and the **Revolute** joint components.

To add the fixed frame, sliding mass, and joint elements

1. From the **Multibody** → **Bodies and Frames** menu, select the **Fixed Frame** component and place it to the left of the **Crank** shared subsystem.
2. From the same menu, select the **Rigid Body** component and place it slightly below and to the right of the **ConnectingRod** shared subsystem.
3. Add the following joints:
   - From the **Multibody** → **Joints and Motions** menu, add a **Revolute** joint between the **Fixed Frame** component and the crank, a second **Revolute** joint between the crank and the connecting rod, and a third **Revolute** joint between the connecting rod and the rigid body.
   - From the same menu, add a **Prismatic** joint and place it below the **Crank** subsystem.
4. Select the **Rigid Body** component in the **Model Workspace** and rename it **SlidingMass**.
5. Right-click (**Control**-click for Macintosh) the **SlidingMass** component and select **Flip Horizontal.** In the same way, flip horizontally the revolute joint that is located between the connecting rod and the rigid body.



6. Connect the components as shown below.



Tip: In this example, the default axes of motion for the revolute and prismatic joints line up with the desired axes of motion. For example, the revolute joints initially assume that they rotate about the z axis of the inboard frame, which always coincides with the inertial Z axis for XY-planar systems. If you create nonplanar models, you may need to change these axes to make sure that they allow motion along or about the correct directions.

Specifying Initial Conditions

You can specify initial condition values for certain components in your model.

To specify initial conditions

1. For the first revolute joint, in the $\boldsymbol{\theta_0}$ field, change the initial angle to $\frac{\pi}{4}$ *rad*.

Tip: To enter π, type **pi**, press **Ctrl** + **Space** (or **Command** + **Shift** + **Space** for Macintosh), and then select the π symbol from the menu.

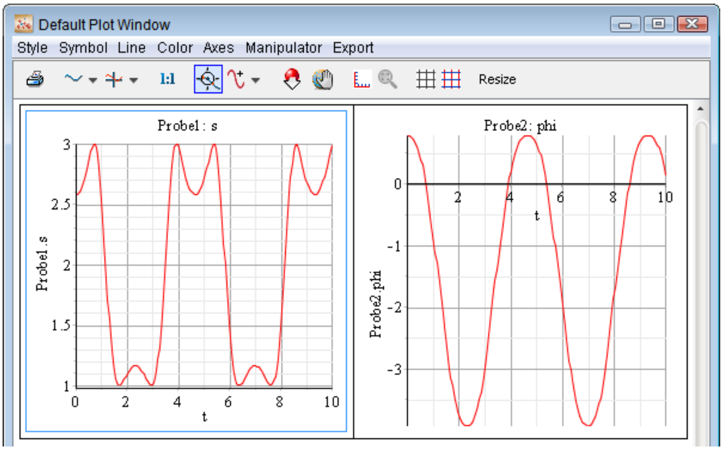
2. From the $\mathbf{IC}_{\boldsymbol{\theta,\omega}}$ drop-down menu, select **Strictly Enforce**.

When MapleSim solves for the initial conditions, the first angle will be set to $\frac{\pi}{4}$ *rad* before the angles are set for the other joints.

Simulating the Planar Slider-Crank Mechanism

To simulate the planar slider-crank mechanism

1. From the **Model Workspace Toolbar**, click **Attach Probe** ().
2. In the **Model Workspace**, click the white 1-D translational flange (flange_b) at the top right of the **Prismatic** component icon and position the probe.
3. Click the probe in the **Model Workspace**.
4. In the **Inspector** tab, select the **Length** quantity to measure the displacement.
5. In the same way, add a probe that measures the **Angle** quantity to the white 1-D rotational flange (flange_b) at the top right of the **R** component icon (that is, the revolute joint between the **Fixed Frame** and **Crank** components).
6. Click a blank area in the **Model Workspace**.
7. In the **Settings** tab, expand **Simulation** and set the **t_d** parameter to **10** seconds.
8. Click **Run Simulation** (▶) in the **Main Toolbar**. When the simulation is complete, the following graphs appear.



9. Save the file as **SliderCrank.msim**.

6.5 Tutorial 5: Using the Custom Component Template

This tutorial describes the use of the Custom Component template in various domains in MapleSim. With this template, you can define system parameters and variables, set the level of equation optimization, generate the equations, and then further analyze the resulting equations. You can use any Maple commands to perform detailed equation analysis, assign model equations to a variable or parameter, and define additional system variables and parameters. These features are especially useful in generating reusable equations when there is more than one subsystem.

The Custom Component Templates contain pre-built embedded components that let you extract, manipulate and analyze the symbolic system equations generated by any MapleSim model. Using various components from the library, you will create models, set initial conditions and component properties, and assign new values to parameters and variables.

In this tutorial, you will use the Custom Component template to extract the equations for various models by performing the following tasks:

- Create the model
- Attach a Custom Component template for the model
- Enter your governing equations
- Set initial conditions by specifying the component properties
- Assign new values to parameters and variables
- Generate the Equations Template
- View, manipulate, and reassign equations
- Simulate and translate an equation to a transfer function
- Map variables from your equations to the ports
- Specify ports for your block
- Create the Custom Component template

For a description of the Custom Component Template see *Using The Custom Component Template (page 68)*.

Example: Modeling a Temperature Dependent Resistor

In this tutorial example, you will create a model of an RL circuit using a custom component for the temperature dependent resistor whose resistance varies as $r(t) = 0.1\, T(t)^2$.

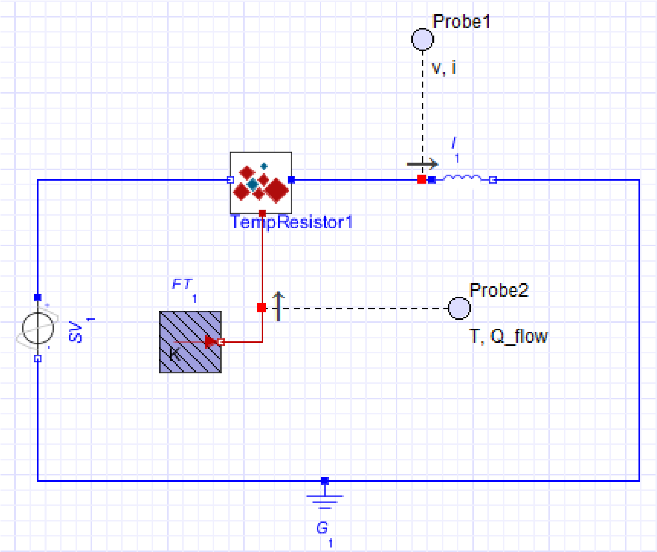


Figure 6.4: Temperature Dependant Resistor

To create the custom component

1. Start a new MapleSim model and click **View** → **Create Attachment...**.
2. Select the **Custom Component** template and then press **Create Attachment**. The Maple **Custom Component** template is loaded.
3. Under **Component Description**, change the component name to **TempResistor**.
4. Under **Component Equations**, enter in the following system equation, parameters and initial conditions to define your component. Press **Enter** at the end of each line.

$$eq := \left[v(t) = vp(t) - vn(t), r(t) = 0.1 \cdot T(t)^2, v(t) = i(t) \cdot r(t), qdot(t) = i(t) \cdot v(t)\right]$$

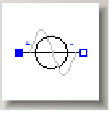
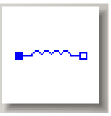
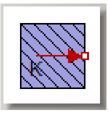
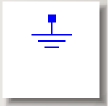
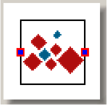
$$params := [\ ]$$

$$initialconditions := [\ ]$$

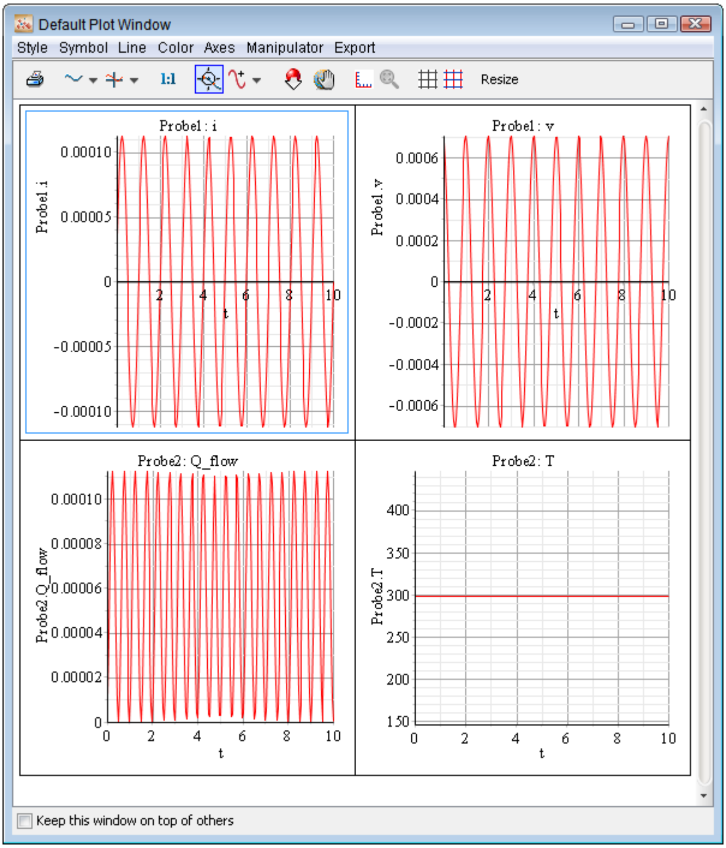
5. Scroll down to **Component Ports** and press **Clear All Ports**. The ports are removed from the component.
6. Press **Add Port** three times.
7. With the bottom port selected, from the **Port Type** drop down box, select **Heat Port**.
8. From the **Temperature** drop down box, select **T(t)**.

9. From the **Heat Flow Rate** drop down box, select **qdot(t)**.
10. With the right port selected, from the **Port Type** drop down box, select **Negative Pin**.
11. From the **Voltage** drop down box, select **vn(t)**.
12. From the **Current** drop down box, select **i(t)** and change this to **–i(t)**.
13. With the left port selected, from the **Port Type** drop down box, select **Positive Pin**.
14. From the **Voltage** drop down box, select **vp(t)**.
15. From the **Current** drop down box, select **i(t)**.
16. Press **Generate MapleSim Component** to create your component and to bring you back into the MapleSim environment. The custom component now appears under the **Project** tab, in the **Definitions** → **Components** menu.
17. Drag the custom component into your model area.
18. Create the model shown in **Figure 6.4** and specified model components and their settings from **Table 6.1**.

Table 6.1: Temperature Dependant Resistor Components

Component	Symbol	Component Location	Required Settings
Sine Voltage		Electrical > Analog > Common	Use default settings
Inductor		Electrical > Analog > Common	Use default settings
Fixed Temperature		Thermal > Sources	Set T = 298K
Ground		Electrical > Analog > Common	Use default settings
TempResistor Custom Component		Project > Definitions > Components	Use default settings

19. Click **Run Simulation** (▶) in the **Main Toolbar**. When the simulation is complete, the following graphs appear.



Example: Compliant Contact and Piecewise Functions

In this tutorial example, you will create a model of a bouncing ball using a custom component to model the compliant ground contact.

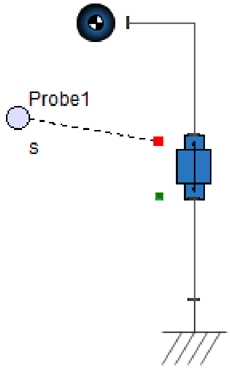


Figure 6.5: Falling Ball

The prismatic joint in **Figure 6.5** models a falling ball by allowing translation of a rigid body along the vertical y axis. To change the falling ball into a bouncing ball, a custom component models the compliant ground contact using a spring-damper arrangement. The custom component attaches to the 1D translational ports on the prismatic joint with the following conditions:

- The ball will hit ground at s=0 and cause the spring damper to compress (and hence ball position will be at s<0)
- The spring-damper will impart a restoring force of F(t) to the ball until it is above at s=0

Figure 6.6 shows a diagram of this process.

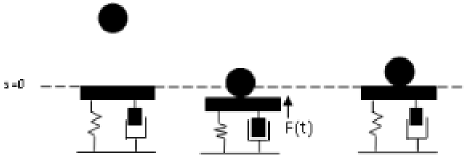


Figure 6.6: Bouncing Ball Dynamics

where

$$f(x) = K s(t) + B\frac{ds(t)}{dt}, \textbf{if}\, s(t) < 0$$

$$f(x) = 0 \;\textbf{if}\, s(t) \geq 0$$

To create the custom component

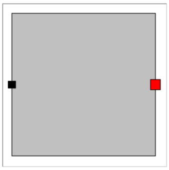
1. Start a new MapleSim model and click **View** → **Create Attachment...**.
2. Select the **Custom Component** template and then press **Create Attachment**. The Maple **Custom Component** template is loaded.
3. Under **Component Description**, change the component name to **custom**.
4. In the **Component Equations** area, enter in the following equation, parameters and initial conditions to define your custom component. Press **Enter** at each line.

$$eq := \left[s(t) = sa(t) - sb(t), 0 = Fa(t) + Fb(t), Fa(t) = piecewise\left(s(t) < 0, K \cdot s(t) + B \cdot \frac{\mathrm{d}}{\mathrm{d}t} s(t), 0 \right) \right]$$

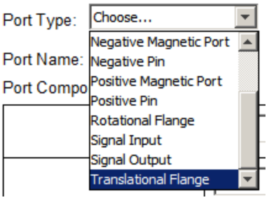
$$params := [\mathrm{K} = 1000, B = 10]$$

$$initialconditions := [\ ]$$

5. Scroll down to **Component Ports** and press **Clear All Ports**. The ports are removed from the component.
6. Press **Add Port** twice and position the ports as shown below.

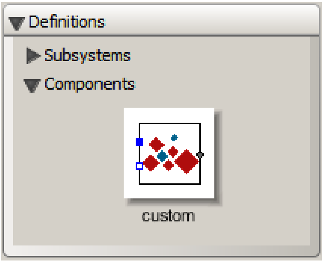


7. Add a new translational flange for each port.



8. For the right port (**tflange0**), from the **Position** drop down box, select **sb(t)** and from the **Force** drop down box, select **Fb(t)**.
9. For the left port (**tflange**), from the **Position** drop down box, select **sa(t)** and from the **Force** drop down box, select **Fa(t)**.

10. Press **Generate MapleSim Component** to create your component and to bring you back into the MapleSim environment. The custom component now appears in the **Project** → **Definitions** → **Components** menu.



11. Drag the custom component into your workspace and assemble the components shown in **Figure 6.7** using the specified model components and their settings from **Table 6.2**. Ensure that the prismatic joint translates along the y direction.

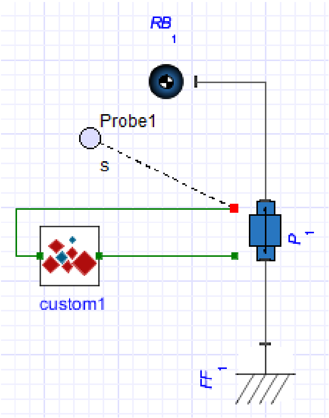
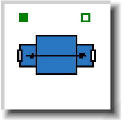
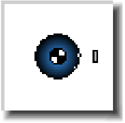
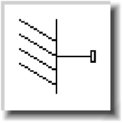
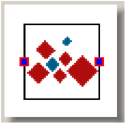


Figure 6.7: Bouncing Ball

Table 6.2: Bouncing Ball Multibody Components

Component	Symbol	Component Location	Required Settings
Prismatic		Multibody > Joints and Motions	For the prismatic joint to translate along the y direction, set $\hat{e}_1$ to $[\mathbf{0, 1, 0}]$ For an initial displacement, the prismatic joint requires a value > 0, set s_0 to $\mathbf{10}\boldsymbol{m}$
Rigid Body		Multibody > Bodies and Frames	Use default settings
Fixed Frame		Multibody > Bodies and Frames	Use default settings
Custom Component		Project > Definitions > Components >	Use default settings

12. Click **Run Simulation** (▶) in the **Main Toolbar**. When the simulation is complete, the following graph appears.

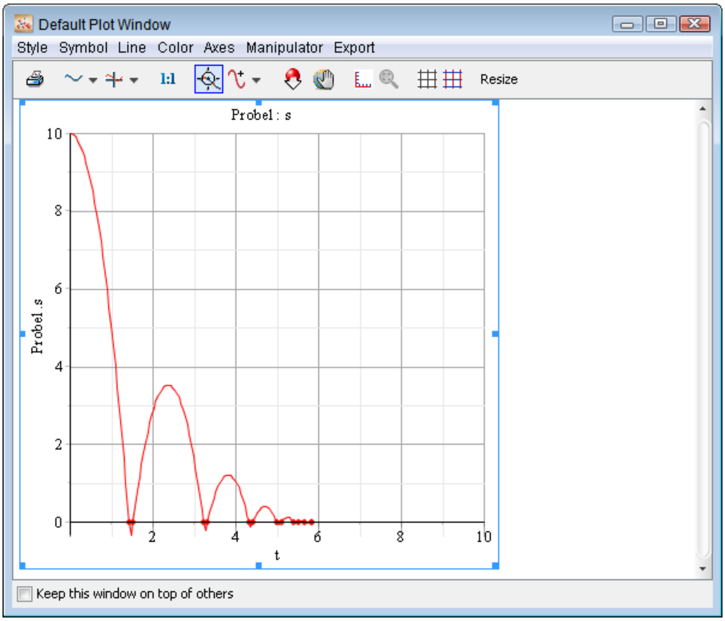

Default Plot Window
Style Symbol Line Color Axes Manipulator Export
1:1
Resize
Probel : s
Probel.s
t
0
2
4
6
8
10
Keep this window on top of others

Figure 6.8: Bouncing Ball Result

13. To play the animation, click **Play** (▶) in the **Main 3-D Toolbar**. To create a smoother animation, select **Interpolate Intermediate Frames** from the **View** menu.

Advanced Uses for Custom Components

You can use the entire range of Maple functionality to derive your system equations in the Custom Component template. This section summarizes a few advanced applications.

Example: Modeling a Centrifugal Pump from a Head Flow Rate Curve

The following hydraulics example demonstrates how to apply extrapolated data from a centrifugal pump into a custom component. Creating a centrifugal pump custom component involves the following tasks.

- Obtain data from a graph
- Generate an equation by fitting the best curve for your data set
- Obtain multi-argument operators
- Apply operators and generate custom component

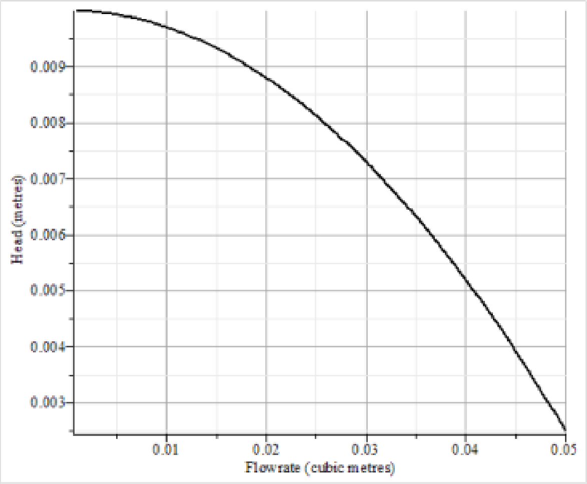


Figure 6.9: Centrifugal Pump Head Flow Rate Curve

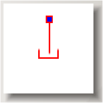
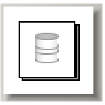
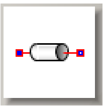
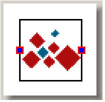
Table 6.3: Centrifugal Pump Data

Flow Rate (cubic meters)	Pressure Head (meters)
0.01	0.0098
0.02	0.00874
0.03	0.00725
0.04	0.005
0.05	0.0025

Table 6.4: Circular Pipe Parameters

Symbol	Description	Values
D	Pipe hydraulic diameter	0.01 *m*
L	Pipe length	5 *m*
ϵ	Height of internal pipe roughness	$1.5 \cdot 10^{-5}$ *m*
ReL *ReT*	Maximum Reynolds number in laminar regime Minimum Reynolds number in turbulent regime	2000 4000
ρ	Fluid density	*rhoFluid*
ν	Fluid kinematic viscosity	*nuFluid*

Table 6.5: Centrifugal Pump Components

Component	**Symbol**	**Component Location**	**It's Use**	**Required Settings**
Atmospheric Pressure		Libraries > Hydraulic > Reference Components	This component defines a base pressure (similar to ground in the electrical domain) and represents a connection to atmosphere	Use default settings
Hydraulic Fluid Properties		Libraries > Hydraulic > Reference Components	All hydraulic models need a Hydraulic Fluid Properties component. Similar to a Parameter block, it is placed in the **Model Workspace** to define the following hydraulic fluid properties: • **rhoFluid**: liquid density • **EIFluid**: Bulk Modulus defines the fluid compressibility • **nuFluid**: Kinematic Viscosity defined as dynamic viscosity divided by liquid density	**rhoFluid**: $850\frac{kg}{m^3}$ **EIFluid**: $8.0\ x\ 10^8 Pa$ **nuFluid**: $0.000018\frac{m^2}{s}$
Circular Pipe		Libraries > Hydraulic > Pipes and Valves	The circular pipe defines a pressure drop in the hydraulic line. The pressure drop is given by the Darcy equation.	See **Table 6.4**
Custom Component		Project > Definitions > Components >	This custom component defines hydraulic pressure and flow rate properties for the hydraulic line.	User defined

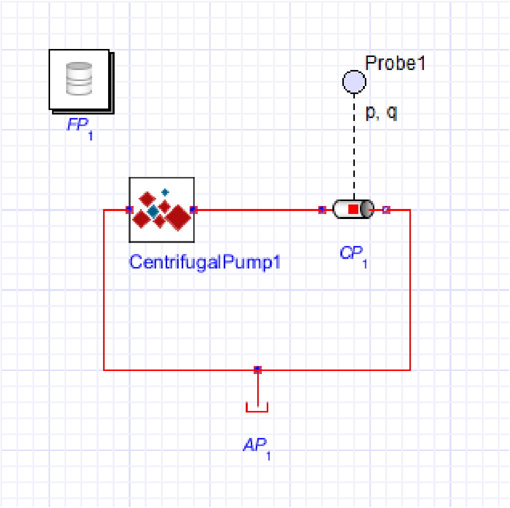


Figure 6.10: Centrifugal Pump Custom Component

To create the custom component

1. Start a new MapleSim model and click **View** > **Create Attachment...**.
2. Select the **Custom Component** template and then press **Create Attachment**. The Maple **Custom Component** template loads.
3. Under **Component Description**, change the component name to **CentrifugalPump**.
4. Place your cursor immediately below the **Component Equations** table, and then insert two execution groups after the cursor (from the main menu, select **Insert** → **Execution Group** → **After Cursor** twice).
5. In the first execution group you added, place the values from **Table 6.3** into the following list, and then press **Enter** at the end of the line to register the list.

$$L := [[0.01, 0.0098], [0.02, 0.00874], [0.03, 0.00725], [0.04, 0.005], [0.05, 0.0025]]$$

6. In the second execution group you added, fit a quadratic curve to the data points using the following Maple command, and then press **Enter** at the end of the line.

$$f := unapply\Big(CurveFitting[LeastSquares]\big(L, x, curve = a{\cdot}x^2 + b{\cdot}x + c\big), x\Big)$$

7. Implement the polynomial in a custom component by defining your equations in terms of the regression curve and parameters for the block. That is, enter in the following system equations, parameters, and initial conditions to define your component. Press **Enter** at the end of each line.

$$eq := \left[P(t) = f(Q(t))\cdot\rho\cdot g, P(t) = P_r(t) - P_l(t)\right]$$

$$params := \left[\rho = 1000, g = 9.81\right]$$

$$initialconditions := [\ ]$$

8. Scroll down to **Component Ports** and click **Clear All Ports**.
9. Click **Add Port** twice to add two new ports, one on the left and one on the right.
10. For each port set the **Port Type** to **Hydraulic Port** and define each port using the settings from **Table 6.6**.

Table 6.6: Hydraulic Port Type

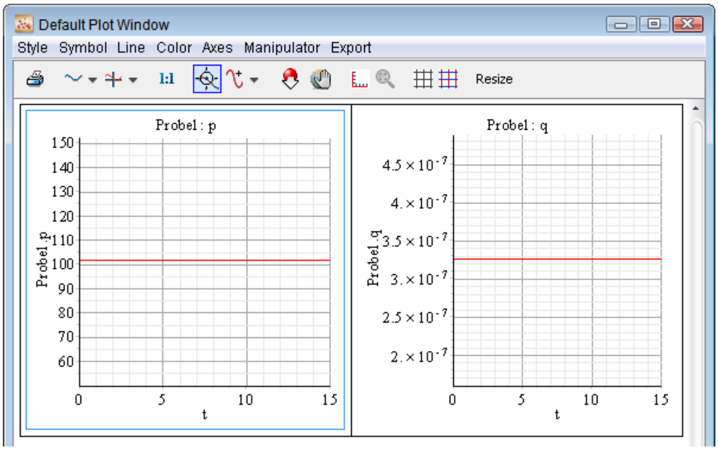
	Port Type	**Left Port**	**Right Port**
Pressure	**Hydraulic Port**	$\mathbf{P}_l(\boldsymbol{t})$	$\mathbf{P}_r(\boldsymbol{t})$
Volume Flow Rate	**Hydraulic Port**	$\boldsymbol{Q}(\boldsymbol{t})$	$-\boldsymbol{Q}(\boldsymbol{t})$

11. Click **Generate MapleSim Component** to create your component and to bring you back into the MapleSim environment. The custom component now appears in the **Project > Definitions > Components**.
12. Drag the custom component into the **Model Workspace** and create the model shown in **Figure 6.10** using the specified model components and their settings from **Table 6.5**.

Tip: To attach the probe on the Circular Pipe component, right-click (**Control**-click for Macintosh) on component, select **Attach Probe**, and then position the probe by clicking on the workspace.

Note: To display the pressure and volume flow rate quantities for your output, select the probe, and then select the **Pressure** and **Volume Flow Rate** quantities from under the **Inspector** tab.

13. Click **Run Simulation** (▶) in the **Main Toolbar**. When the simulation is complete, the following graph appears.



6.6 Tutorial 6: Using the External C Code/DLL Custom Component Template

In this tutorial, you will use the **External C Code/DLL** template to import external C Code parameters and build your model by performing the following tasks:

- Specify the custom component name
- Specify the location of the external C/library
- Define the external C/Library code options
- Specify the directory of the generated Modelica code
- Generate and save the external code custom component
- Build the Simple External Function model

This model consists of three components, a **Step** function, a **Constant Vector**, and an **External C Code/DLL** custom component.

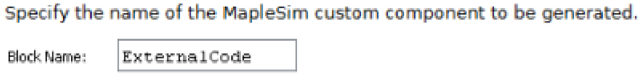
The external C Code parameters are defined by a function that takes in

- a double scalar input
- an *input* double array of size 2
- an *output* double array of size 3

and then returns a double scalar

To create the external code custom component

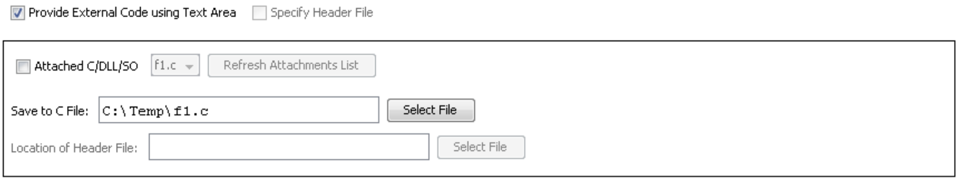
1. Start a new MapleSim model that will call the external code.
2. Click **Templates** ( ) in the **Main Toolbar**. The **Create Attachment** window appears.
3. From the list, select **External C/Library Block**.
4. In the **Attachment** field, enter a name for the template and click **Create Attachment**. The **External C/Library Block** template opens in Maple.
5. Provide 'ExternalCode' as a name for the custom component.



Specify the name of the MapleSim custom component to be generated.

Block Name: ExternalCode

6. Select **Provide External Code using Text Area** and specify the location for the external C/library in **Save to C File**.



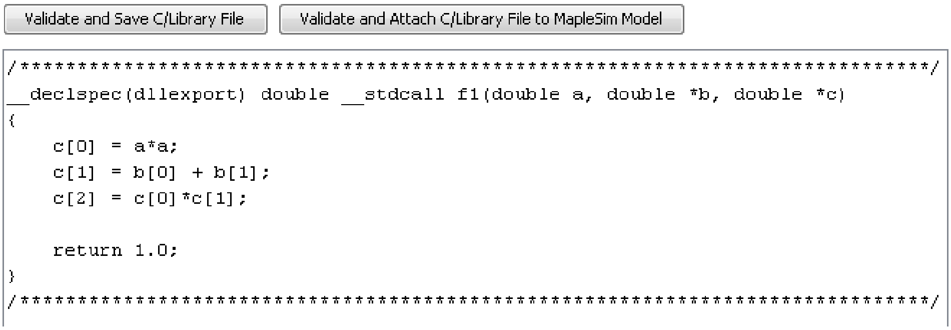
☑ Provide External Code using Text Area ☐ Specify Header File

☐ Attached C/DLL/SO f1.c [Refresh Attachments List]

Save to C File: C:\Temp\f1.c [Select File]

Location of Header File: [Select File]

7. For a Windows platform, in the text area, enter the C code using the function declaration as shown in **Figure 6.11**. For a Unix platform enter the C code using the function declaration shown in **Figure 6.12**.

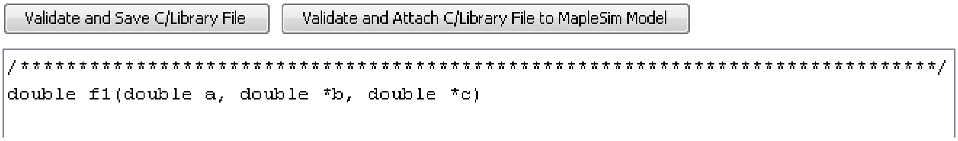


[Validate and Save C/Library File] [Validate and Attach C/Library File to MapleSim Model]

```
/*******************************************************************************/
__declspec(dllexport) double __stdcall f1(double a, double *b, double *c)
{
    c[0] = a*a;
    c[1] = b[0] + b[1];
    c[2] = c[0]*c[1];

    return 1.0;
}
/*******************************************************************************/
```

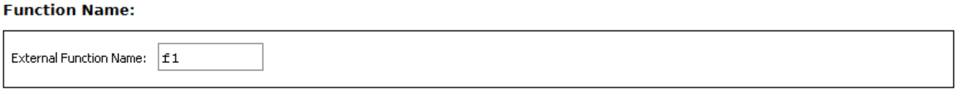
Figure 6.11: External C Code Definition for Windows


Validate and Save C/Library File
Validate and Attach C/Library File to MapleSim Model

```
/*******************************************************************************/
double f1(double a, double *b, double *c)
```

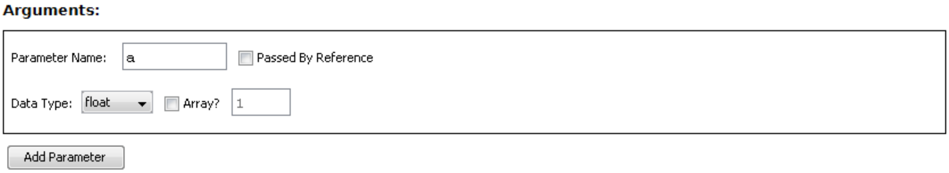
Figure 6.12: External C Code Definition for Unix

8. Click **Validate and Save C/Library File**.
9. In the **ExternalC/Library Function Name and Definition** section, enter 'f1' for the **External Function Name**.



External Function Name: f1

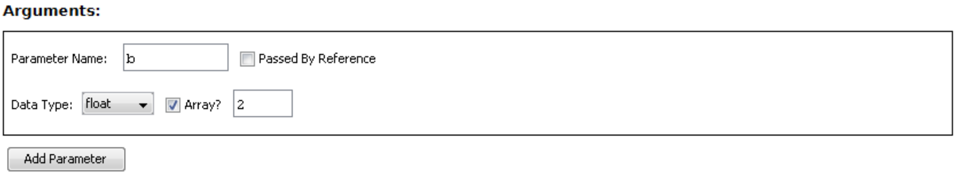
10. Use the following values for the *double* scalar input 'a' as follows and then click **Add Parameter**. The parameter appears in the **Arguments Table** as shown in **Figure 6.13**.



Parameter Name: a ☐ Passed By Reference

Data Type: float ☐ Array? 1

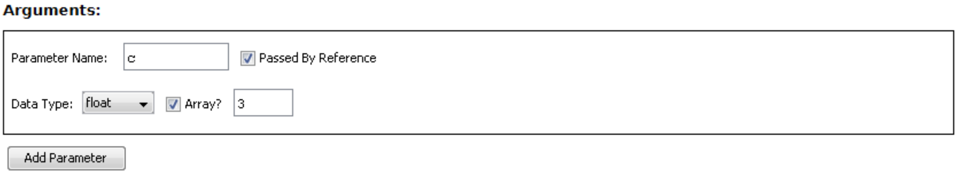
11. Use the following values for the input *double* array 'b' of size 2 as follows and then click **Add Parameter**. The parameter appears in the **Arguments Table** as shown in **Figure 6.13**.



Parameter Name: b ☐ Passed By Reference

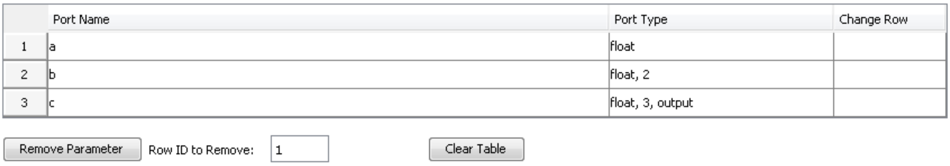
Data Type: float ☑ Array? 2

12. Use the following values for the output *double* array 'c' of size 3 as follows and then click **Add Parameter**. The parameter appears in the **Arguments Table** as shown in **Figure 6.13**.



Parameter Name: c ☑ Passed By Reference

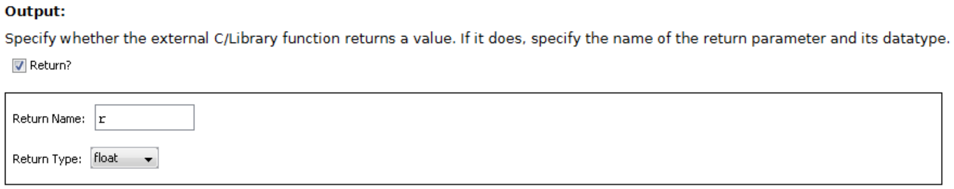
Data Type: float ☑ Array? 3



	Port Name	Port Type	Change Row
1	a	float	
2	b	float, 2	
3	c	float, 3, output	

Figure 6.13: Arguments Table

13. Use the following values for the C function return parameter.



Output:

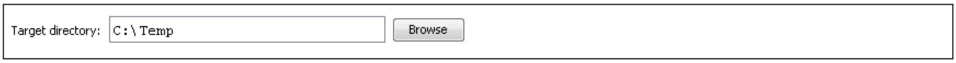
Specify whether the external C/Library function returns a value. If it does, specify the name of the return parameter and its datatype.

☑ Return?

Return Name: r

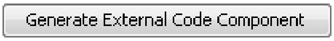
Return Type: float

14. Specify where you want to save the generated Modelica custom component code.



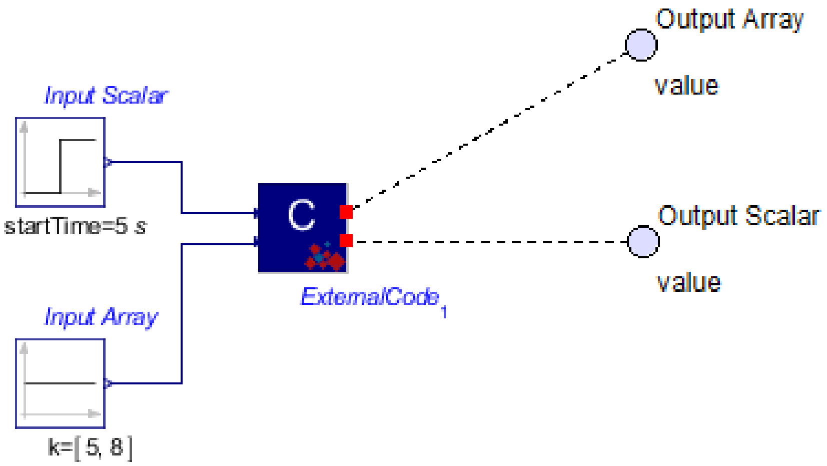
Target directory: C:\Temp

15. Click **Generate External Code Component**. In MapleSim, the custom component appears under the **Project** tab, in the **Definitions** → **Components** palette, on the left side of the MapleSim window.



To use the external code custom component

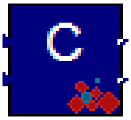
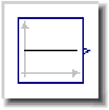
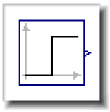
1. Using the specified model components and their settings from **Table 6.7** drag the components into the **Model Workspace** and set their values values.



Note: Ensure that the model component parameter values are set in your model. When you select a component in the **Model Workspace**, the configurable parameter values for that component appear in the **Inspector** tab located on the right side of the MapleSim window.

Table 6.7 shows the required components and their settings.

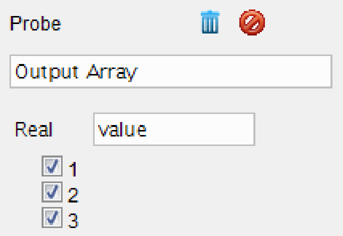
Table 6.7: External C Code DLL Custom Components and Required Settings

Component	Symbol	Component Location	Required Settings
Custom Component		Project palette > Definitions > Components	Use default settings
Constant Vector		Component Library > Signal Blocks > Sources > Real	**Constant output value**, set ***K* to** [**5, 8**]
Step		Component Library > Signal Blocks > Sources > Real	**Height**: 4 **Offset**: 0 $\mathbf{T_0}$: 5

2. Connect the **Step** component to custom component **input port a**.

3. Connect the **Constant Vector** component to custom component **input port b**.

4. Attach a probe to the custom component **output port c** and enter the following values:



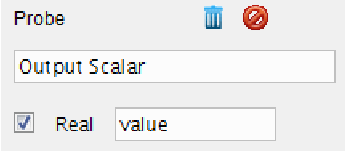
Probe

Output Array

Real value

☑ 1
☑ 2
☑ 3

5. Attach a probe to the custom component **output port r** and enter the following values:

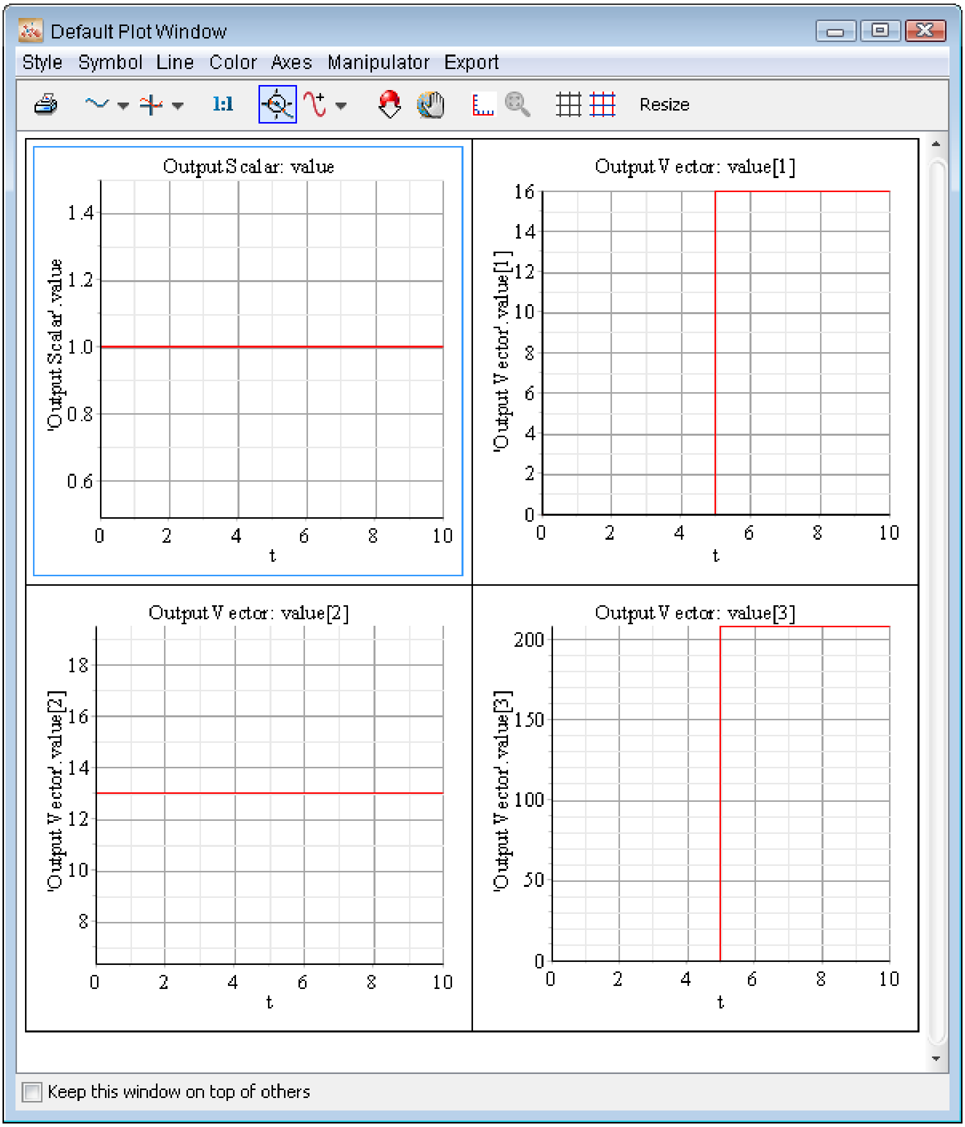


Probe

Output Scalar

☑ Real value

6. Click **Run Simulation** (▶) in the **Main Toolbar**. When the simulation is complete, the following graphs appear.



7. To verify the results, in the **Libraries** tab, expand the **Examples** palette, expand the **User's Guide Examples** menu, and then open the **Simple External Code Function** example.

6.7 Tutorial 7: Using the Equations Template

In this tutorial, you will use the Equations template to extract the equations for a Spring Damper model by performing the following tasks:

- Create the model
- Generate the Equations Template
- View, manipulate and reassign equations

The Equations Template contains pre-built embedded components that lets you extract, manipulate and analyze the symbolic system equations generated by any MapleSim model.

With this template, you can define system parameters and variables, set the level of equation optimization, generate the equations, and further analyze the resulting equations. You can use any Maple commands to perform detailed equation analysis, assign model equations to a variable and define additional system variables and parameters. These features are useful in generating reusable equations when there is more than one subsystem.

Template Description

The Equations Template is a collection of pre-built controls and procedures associated with specific Maple commands to easily generate equations from MapleSim models. When you simulate a model, all parameter units convert to SI units, allowing you to select more than one system of units for parameter values throughout the model.

The Equations Template consists of two main areas, **Equation Display** and **Equation Manipulation**.

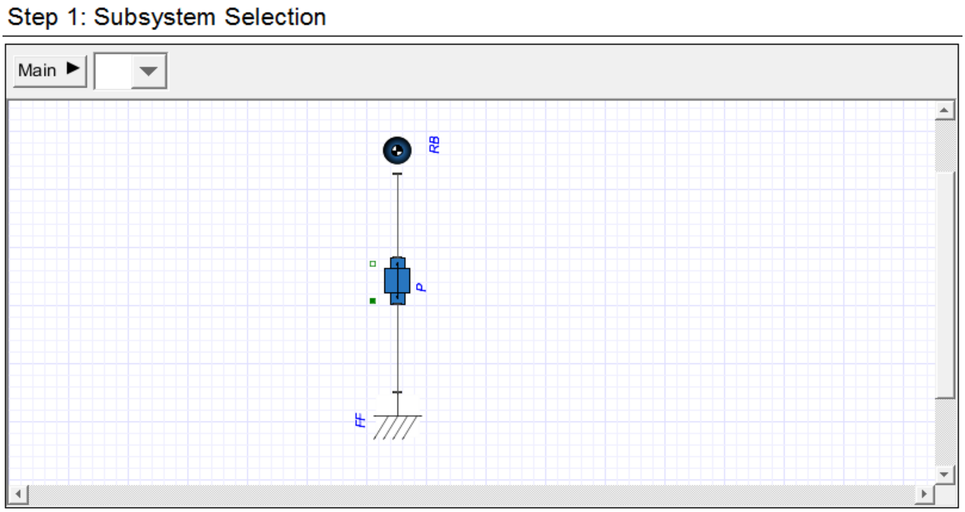
Equation Display Area

The **Equation Display** area consists of three sections, **Subsystem Selection**, **Parameter and Variable Manipulation** and **View Equations**. Use this section to extract the equations for an entire model or one of its subsystems.

Subsystem Selection

This section loads the MapleSim model and shows all subsystems and their components. From the toolbar you can select a subsystem and load its subsystem equations.

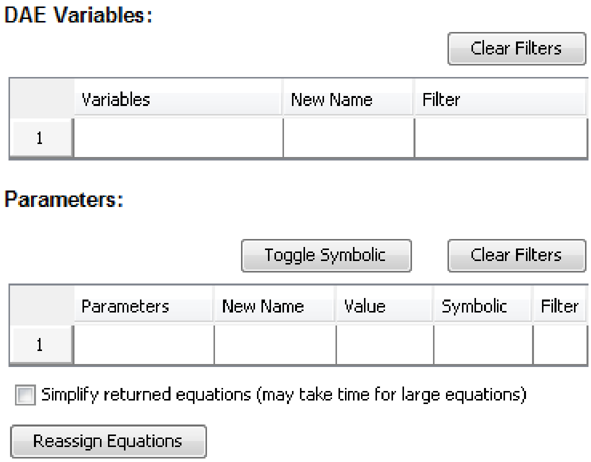
If no subsystem is selected, equations for the whole model will be loaded when you click **Load Selected Subsystems**.



Load Selected Subsystem. Extracts the model equations and loads the system parameters and variables.

Parameter and Variable Manipulation

In this area you can customize and define parameter and DAE variables for the generated equations.


DAE Variables:

Clear Filters

	Variables	New Name	Filter
1			

Parameters:

Toggle Symbolic | Clear Filters

	Parameters	New Name	Value	Symbolic	Filter
1					

☐ Simplify returned equations (may take time for large equations)

Reassign Equations

DAE Variables

You can rename the DAE variables by specifying a new variable name in the **New Name** column. For example, variable s(t) can be changed to x(t).

Variables. Contains the model DAE variables.

New Name. Provide a new name for the DAE variable.

Filter. Select the equations with the specified variable.

Clear Filters. Clears all selections to show all of the equations.

Parameters

You can rename the parameters in the **New Name** column. By default, all equation parameters are substituted by their numeric values. You can manually select which parameters you want to leave in the symbolic form by entering “X” under the **Symbolic** column beside the parameter of interest. In the **Filter** column only equations with the selected variable or parameter are shown. Alternatively, if you want all the parameters in symbolic form, select **Toggle Symbolic**. An “X” will appear beside every parameter. Once a symbolic parameter is selected for filtering, it will appear in symbolic form.

Parameters. Contains the model parameters.

New Name. Provide a new name for the model parameter.

Value. Displays the new parameter value as specified in MapleSim.

Symbolic. Enter "X" to select which parameters you want to leave in the symbolic form.

Filters. Select the equations by the specified parameter.

Toggle Symbolic. Switches the parameter symbolic setting selection to all or none.

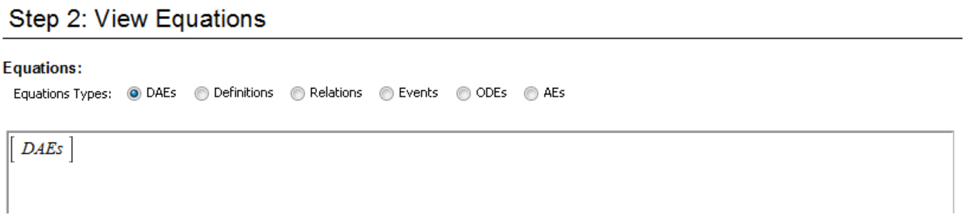
Clear Filters. Clears all selections to show all of the equations.

Simplify returned equations. Invokes the appropriate simplification procedures, rules and symbolic algorithms to reduce the index of the equation, and searches for function calls, square roots, radicals, and powers. For more information, in Maple, enter `?simplify` at a prompt in a worksheet.

Reassign Equations. Select **Reassign Equations**, to update the new equation names and show the changes to the variables and parameters.

View Equations

This area shows the system of equations in symbolic form with the assigned parameters. You can select which equations you want to look at by selecting one of the equation types (DAEs, Definitions, Relations, Events, ODEs, AEs). These equations are automatically assigned to their respective equation variables and can be manipulated in the main body of the Maple worksheet. DAEs are assigned to DAEs, Definitions are assigned to Definitions, and so forth. For more information about equation types, in Maple, enter `?GetEquations` at a prompt in a worksheet.



Step 2: View Equations

Equations:

Equations Types: ● DAEs ○ Definitions ○ Relations ○ Events ○ ODEs ○ AEs

$[DAEs]$

Equation Manipulation Area

You can also use this template to manually generate additional equations using the GetEquations command to get active MapleSim subsystem equations. For additional equation extraction options see the GetEquations command. Equations in the template are automatically assigned to the variable DAEs, allowing them to be manipulated in the main body of the Maple worksheet.

Note: Maple applies simplification rules to equations stored in their respective equation types by using Maple's simplify command set to *true*. For large sets of equations this can take significant amounts of time and memory. For these situations set Maple's simplify command to *false*.

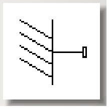
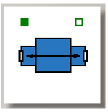
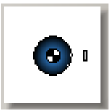
Generating the Equations for the Spring Damper

This model consists of a prismatic joint (also called a slider or translational joint) attached to a rigid body, both from the Multibody library. The prismatic joint connects to the Fixed Frame and allows one translational degree of motion along the Y-axis. For this system, gravity is assumed to be the only external force, acting along the Y-axis. You will group these components into a subsystem, define the system parameters and reference the subsystem parameters. For the purpose of this tutorial, you will generate sample equations with predefined differential equations.

Ensure that the model component parameter values are set in your model. When you select a component in the **Model Workspace**, the configurable parameter values for that component appear in the **Inspector** tab located on the right side of the MapleSim window.

Table 6.8 shows the required components and their settings.

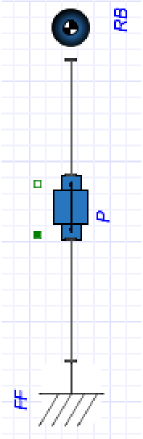
Table 6.8: Spring Damper Components and Required Settings

Component	Symbol	Library Location	Required Settings
Fixed Frame		Multibody > Bodies and Frames	Use default settings
Prismatic		Multibody > Joints and Moments	Translational axis in the y direction, set $\hat{\boldsymbol{e}}_1$ to [0,1,0] Spring constant, set $\boldsymbol{K_S}$ Damping constant, set $\boldsymbol{K_d}$
Rigid Body		Multibody > Bodies and Frames	Mass, set $\boldsymbol{m}$ to1 kg

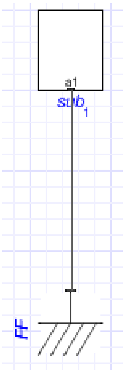
Generating the System Equations

To generate the system equations

1. Create the following model using the specified model components and their settings from **Table 6.8** .

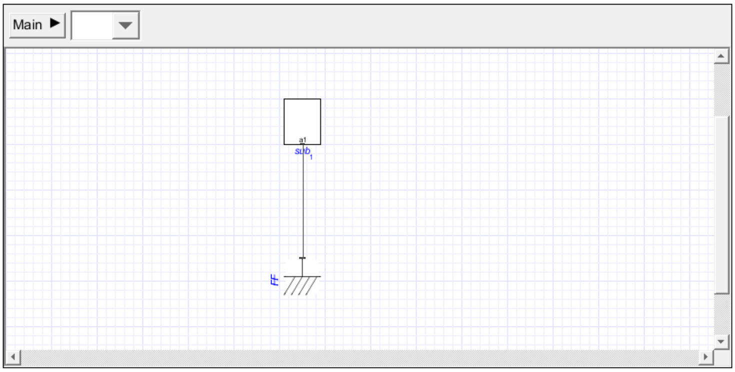


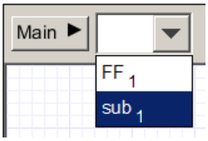
2. Place the **Prismatic** joint and **Rigid Body** in a subsystem called **sub**. This allows the Equation Template to generate equations specifically for the selected subsystem.



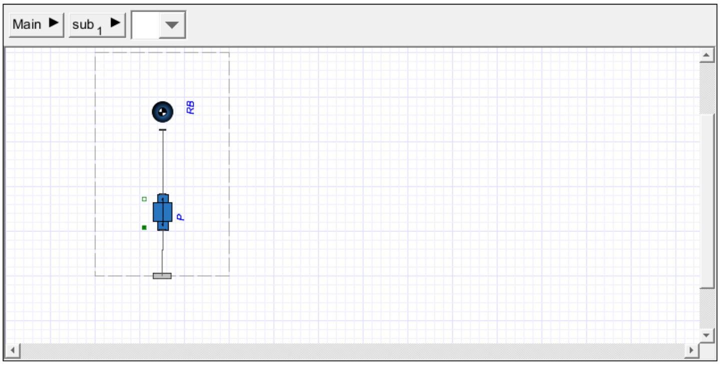
3. Click **Templates** (📎) in the **Main Toolbar** and select the **Equations** template.

4. In the **Attachment** field, provide a worksheet name and click **Create Attachment**. Your MapleSim model opens in a Maple worksheet, with the **Subsystem Selection** window. The toolbar shows all of the subsystems.





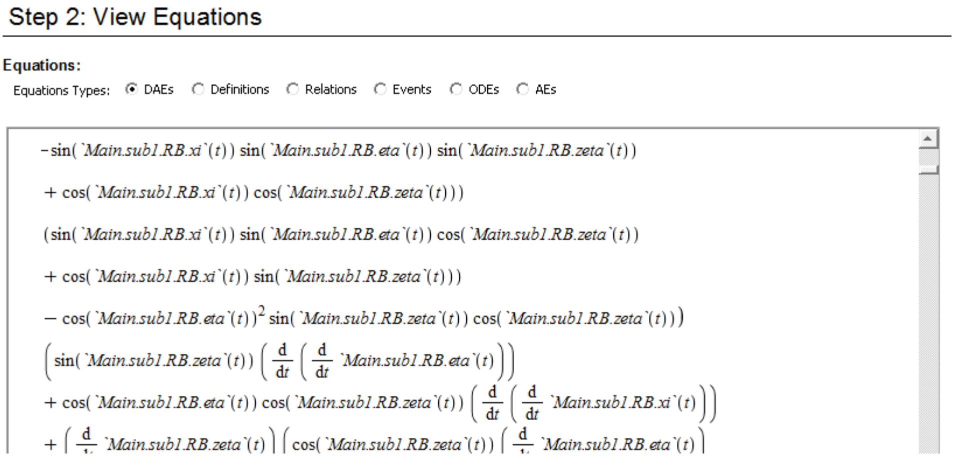
5. From the toolbar select the **sub$_1$** subsystem. The *sub$_1$* subsystem components appear.



6. Click **Load Selected Subsystem**. The sub_1 component parameter and variables load automatically in the **Parameter** and **Variable Manipulation** areas.

View Equations Area

The system equations appear in the **View Equations** area.



Step 2: View Equations

Equations:

Equations Types: ● DAEs ○ Definitions ○ Relations ○ Events ○ ODEs ○ AEs

$$-\sin(\text{`Main.sub1.RB.xi`}(t))\sin(\text{`Main.sub1.RB.eta`}(t))\sin(\text{`Main.sub1.RB.zeta`}(t))$$

$$+\cos(\text{`Main.sub1.RB.xi`}(t))\cos(\text{`Main.sub1.RB.zeta`}(t)))$$

$$(\sin(\text{`Main.sub1.RB.xi`}(t))\sin(\text{`Main.sub1.RB.eta`}(t))\cos(\text{`Main.sub1.RB.zeta`}(t))$$

$$+\cos(\text{`Main.sub1.RB.xi`}(t))\sin(\text{`Main.sub1.RB.zeta`}(t)))$$

$$-\cos(\text{`Main.sub1.RB.eta`}(t))^2\sin(\text{`Main.sub1.RB.zeta`}(t))\cos(\text{`Main.sub1.RB.zeta`}(t)))$$

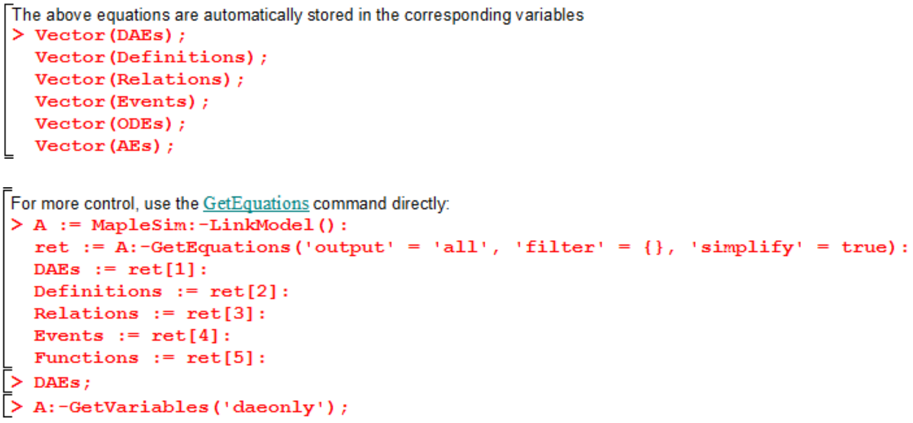
$$\left(\sin(\text{`Main.sub1.RB.zeta`}(t))\left(\frac{d}{dt}\left(\frac{d}{dt}\text{`Main.sub1.RB.eta`}(t)\right)\right)\right.$$

$$\left.+\cos(\text{`Main.sub1.RB.eta`}(t))\cos(\text{`Main.sub1.RB.zeta`}(t))\left(\frac{d}{dt}\left(\frac{d}{dt}\text{`Main.sub1.RB.xi`}(t)\right)\right)\right)$$

$$+\left(\frac{d}{dt}\text{`Main.sub1.RB.zeta`}(t)\right)\left(\cos(\text{`Main.sub1.RB.zeta`}(t))\left(\frac{d}{dt}\text{`Main.sub1.RB.eta`}(t)\right)\right.$$

Manipulating the Equations for the Spring Damper

By default the above equations are automatically stored in the variable DAEs. In the example below, only part of the equation is shown in the **Equation Manipulation** section.



The above equations are automatically stored in the corresponding variables

```
> Vector(DAEs);
  Vector(Definitions);
  Vector(Relations);
  Vector(Events);
  Vector(ODEs);
  Vector(AEs);
```

For more control, use the GetEquations command directly:

```
> A := MapleSim:-LinkModel():
  ret := A:-GetEquations('output' = 'all', 'filter' = {}, 'simplify' = true):
  DAEs := ret[1]:
  Definitions := ret[2]:
  Relations := ret[3]:
  Events := ret[4]:
  Functions := ret[5]:
> DAEs;
> A:-GetVariables('daeonly');
```

6.8 Tutorial 8: Modeling Hydraulic Systems

This tutorial provides you with a basic description of hydraulic systems and helps you understand how to model these systems in MapleSim. Using components from the Hydraulic library, you will create models, set initial conditions and component properties, and assign new values to parameters and variables.

The hydraulic components are designed primarily to convert hydraulic flow into mechanical motion, but can also be used to model pure hydraulic circuits.

In this tutorial, you will perform tasks based on the following basic principles and concepts:

- Basic Hydraulic Library Components
- Basic Hydraulic Equations
- Analysis of Simple Hydraulic Networks
- First Principles Modeling
- Mechanical and Hydraulic Systems

The following sections provide conceptual models that you can build using the hydraulic library components.

- Controlling Hydraulic Flow Path
- Actuating Multibody Systems with Hydraulic Components
- Compressibility of Hydraulic Liquids
- Fluid Inertia Models
- Water Hammer Models
- Hydraulic Custom Components

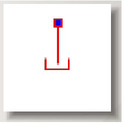
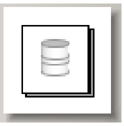
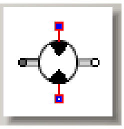
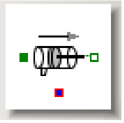
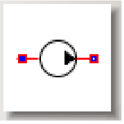
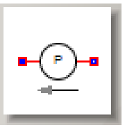
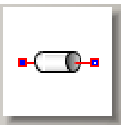
Computational Issues

Hydraulic networks tend to be numerically stiff. Generally, the stiff rosenbrock solver is recommended.

Basic Hydraulic Library Components

This tutorial uses the following basic Hydraulic library components.

Table 6.9: Basic Hydraulic Library Components

Component	Symbol	Library Location	Its Use
Atmospheric Pressure		Hydraulic > Reference Components	This component defines a base pressure (similar to ground in the electrical domain) and represents a connection to the atmosphere
Hydraulic Fluid Properties		Hydraulic > Reference Components	All hydraulic models need a Hydraulic Fluid Properties component. Similar to a Parameter block, it is placed in the **Model Workspace** to define the following hydraulic fluid properties: • **rhoFluid**: fluid density • **EIFluid**: Bulk Modulus is the fluid compressibility • **nuFluid**: Kinematic Viscosity is the dynamic viscosity divided by liquid density
Hydraulic Motor		Hydraulic > Actuators	Actuators convert hydraulic flow into the motion of a mechanical body. MapleSim offers a Hydraulic Cylinder (for translational motion) and a Hydraulic Motor (for rotational motion).
Hydraulic Cylinder			
Fixed Flow Source		Hydraulic > Sources	You can specify either the flow rate or the pressure of the hydraulic source (with MapleSim calculating the other quantity). If a Pressure Source is used, then MapleSim balances the load in the hydraulic system against the pressure source to find the flow rate, and vice-versa.
Fixed Pressure Source			
Circular Pipe		Hydraulic > Pipes and Valves	The circular pipe introduces a pressure drop in a hydraulic line. The pressure drop is given by the Darcy equation, with the friction factor being determined by using predefined equations.

Basic Hydraulic Equations

The Bernoulli and the Darcy equations are the fundamental equations necessary to analyze hydraulic systems and define the fluid pressure and flow rate characteristics for any point along a flow. This tutorial uses the following basic fluid equations.

- Bernoulli Equation
- Darcy Equation
- Friction Factor

Bernoulli Equation

The Bernoulli Equation defines the pressure and flow rate characteristics of incompressible fluid flow in a pipe. For any point along a streamline, the following relationship applies.

$$\frac{p}{\rho \cdot g} + \frac{V^2}{2 \cdot g} + z = constant$$

Darcy Equation

For an incompressible fluid flowing through a pipe with a constant diameter, the pressure drop due to pipe friction is given by the Darcy equation.

$$\triangle P = f \cdot \frac{L \cdot V^2}{\mathrm{D} \cdot 2 \cdot g}$$

Hence

$$\frac{p}{\rho \cdot g} + \frac{V^2}{2 \cdot g} + z + f \cdot \frac{L \cdot V^2}{\mathrm{D} \cdot 2 \cdot g} = constant$$

Table 6.10: Bernoulli and Darcy Equation Notation

Symbol	Description	Units
P	Pressure	Pa
ρ	Density	$\frac{kg}{m^3}$
g	Gravitational constant	$\frac{m}{s^2}$
V	Velocity	$\frac{m}{s}$
z	Elevation	m
L	Pipe length	m
D	Pipe diameter	m
f	Friction factor	*dimensionless*

Hence pressure must be applied to overcome internal frictional effects within the liquid (in laminar flow), and the effect of the surface roughness of the pipe (in turbulent flow). Frictional losses (and any other loads in the system) have to be balanced against the applied pressure to determine the flow rate.

In MapleSim's mechanical-hydraulic systems, the vertical displacement **z** is insignificant compared to the other terms, and is ignored.

Friction Factor

In laminar flow, the internal frictional (***f***) effect is determined by the following equations:

$$f = \frac{64}{\mathrm{Re}}$$

$$\mathrm{Re} = \frac{\mathrm{D} \cdot V}{v}$$

where

f is the internal friction

Re is the Reynolds number

D is the pipe diameter

V is the fluid velocity and

v is the dynamic viscosity

In turbulent flow, the frictional effects of the surface roughness of the pipe are characterized by the **Halaand Equation**.

$$f = \frac{1}{\left(1.8\,\mathrm{log10}\left(\frac{6.9}{\mathrm{Re}} + \left(\frac{\epsilon}{3.7 \cdot \mathrm{D}}\right)^{1.11}\right)\right)^{2}}$$

The Reynolds number (**Re**) indicates whether flow in a pipe is in laminar or turbulent flow, or is in transition between the two. For example, the circular pipe parameters in **Table 6.11** gives the Reynolds number for laminar (**ReL**) and turbulent (**ReT**) flow. Between these two parameters, the friction factor is determined by linear interpolation.

Table 6.11: Circular Pipe Parameters

Symbol	Description	Values
D	Pipe hydraulic diameter	0.01 *m*
L	Pipe length	5 *m*
ϵ	Height of internal pipe roughness	$1.5 \cdot 10^{-5}$ *m*
ReL *ReT*	Maximum Reynolds number in laminar regime Minimum Reynolds number in turbulent regime	2000 4000
ρ	Fluid density	*rhoFluid*
ν	Fluid kinematic viscosity	*nuFluid*

Analysis of Simple Hydraulic Networks

This section simulates a simple hydraulic system and analyzes the results from first principles and explains how to:

- Create a simple laminar pipe flow hydraulic system
- Analyze the governing equations by applying various laws (for example, conservation of mass, Bernoulli Equation, Darcy Equation)

Flow Through a Pipe

Figure 6.14 analyzes pressure and laminar flow rate characteristics through a pipe when pressure is applied to overcome internal frictional effects.

To analyze flow through a pipe

1. Create the following model using the specified model components and their settings from **Table 6.12** .

Tip: To attach the probe on the Circular Pipe component, right-click (**Control**-click for Macintosh) on component, select **Attach Probe**, and then position the probe by clicking on the workspace.

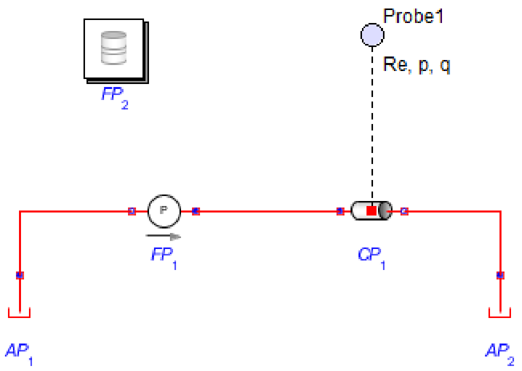
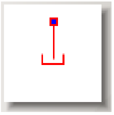
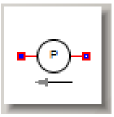
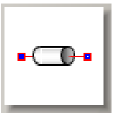
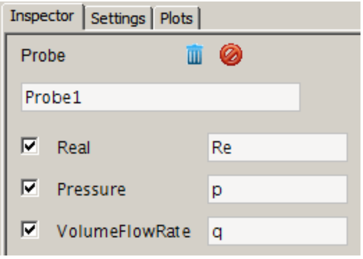


Figure 6.14: Flow Through a Pipe

Table 6.12: Hydraulic Components and Required Settings

Component	Quantity	Symbol	Library Location	Required Settings
Atmospheric Pressure	**2**		Hydraulic > Reference Components	Use default settings
Hydraulic Fluid Properties	**1**		Hydraulic > Reference Components	**rhoFluid**: $850\frac{kg}{m^3}$ **EIFluid**: $8.0\ x\ 10^8\ Pa$ **nuFluid**: $0.000018\frac{m^2}{s}$
Fixed Pressure Source	**1**		Hydraulic > Sources	Use default settings
Circular Pipes	**1**		Hydraulic > Pipes and Valves	See **Table 6.11**

2. Click the probe and select the **Real**, **Pressure** and **Volume Flow Rate** probe parameters.



Probe

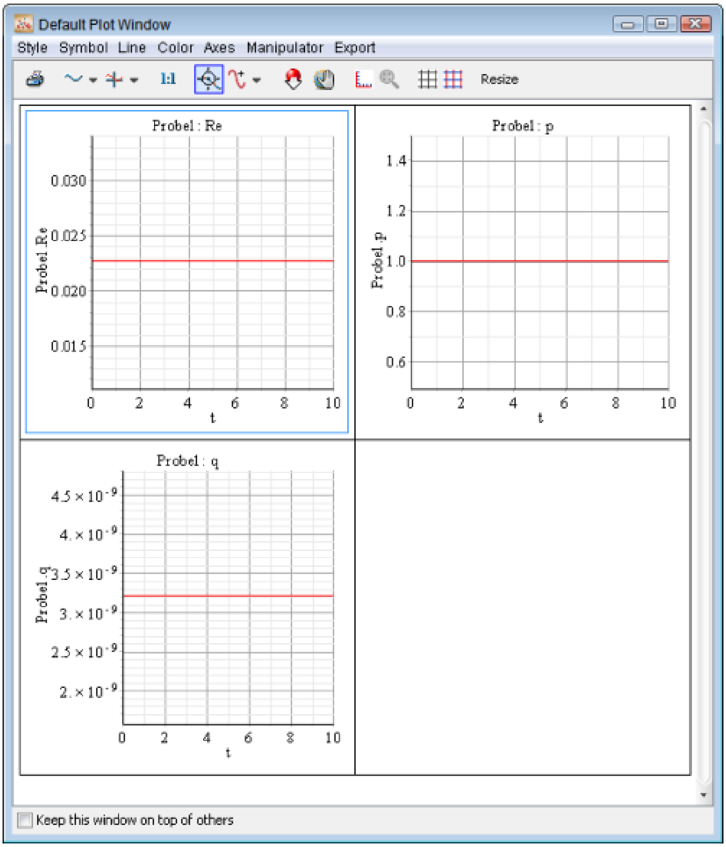
Probe1

☑ Real Re

☑ Pressure p

☑ VolumeFlowRate q

3. Click **Run Simulation** (▶) in the **Main Toolbar**. When the simulation is complete, the following graph appears showing the predicted flow rate of $Q = \dfrac{3.2x10^{-9}\,m^3}{s}$.



Confirming the Modeling Results from First Principles

When analyzing the system shown in **Figure 6.14**, apply the Darcy Equation,

$$\frac{\Delta P}{\rho\cdot g} = f\cdot\frac{L\cdot V^2}{D\,2\,g}$$

Assuming that the system is in laminar flow, then

$$f = \frac{64}{\mathrm{Re}}$$

Hence

$$\frac{1}{850\times 9.81} = \frac{64}{\frac{0.01\times V}{0.000018}}\times\frac{5}{0.01}\times\frac{V^2}{2\times 9.81}$$

$$0.0001199256461 = 2.935779816\,V$$

Where

$$\mathrm{V} = 0.0000408\ \frac{m}{s}$$

Using **V** in the flow rate equation yields the following result

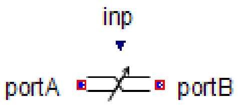
$$Q = \frac{1}{4}\,V\pi\,\mathrm{D}^2 = 0.0000408\times\frac{\pi\times 0.01^{\,2}}{4} = 3.2\,x\,10^{-9}\ \frac{m^3}{s}$$

This is the same value given by MapleSim. Using the calculated value of **V** gives **Re**= 0.02. This is far less than the critical value of 2000, and hence the system is in laminar flow.

Overview of Controlling Hydraulic Flow Path

A spool valve has a sharp-edged variable area orifice that enables or partially restricts flow in a pipe, and can assist in switching flow from one part of a hydraulic network to another. A spool valve has three ports.

Table 6.13: Spool Valve

Spool Valve	Name	Description	ID
	*PortA*	Upstream port	portA
	PortB	Downstream port	portB
	inp	Input signal	inp

The top port (**inp**) accepts a signal input that is equal to the open valve area. By regulating the valve area, flow switches on or off. The left and right ports (**portA** and **portB**) are hydraulic connectors. In the following diagram, the model switches flow from the top leg to the bottom leg when the simulation time reaches five seconds. That is, initially, the top spool valve is open and the bottom is closed. After 5 seconds, the top spool valve closes, and the bottom opens.

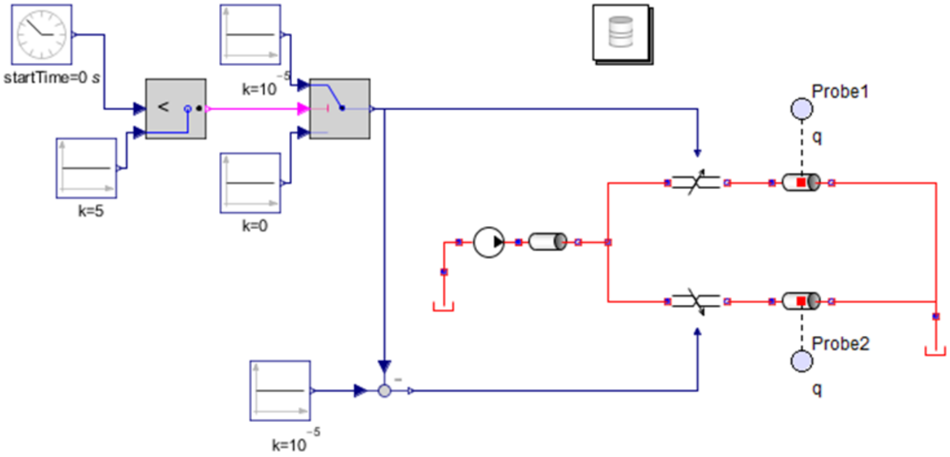


Figure 6.15: Controlling Flow Path

Mechanical and Hydraulic Systems

In the following examples you will use multidomain components to simulate translational motion in mechanical and hydraulic models with the following sources:

- Fixed Flow Rate Source
- Fixed Pressure Source

Simulating Translational Motion with a Fixed Flow Rate Source

The following model converts flow from a fixed flow source to translational motion using the components and their settings from **Table 6.14**.

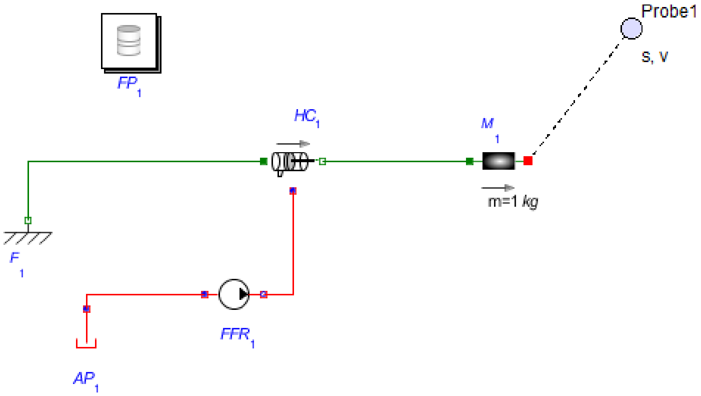
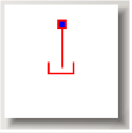
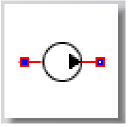
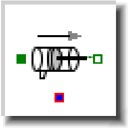
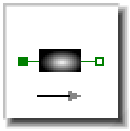
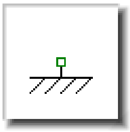


Figure 6.16: Fixed Flow Rate Source

Table 6.14: Translational Motion with Fixed Flow Rate Sources

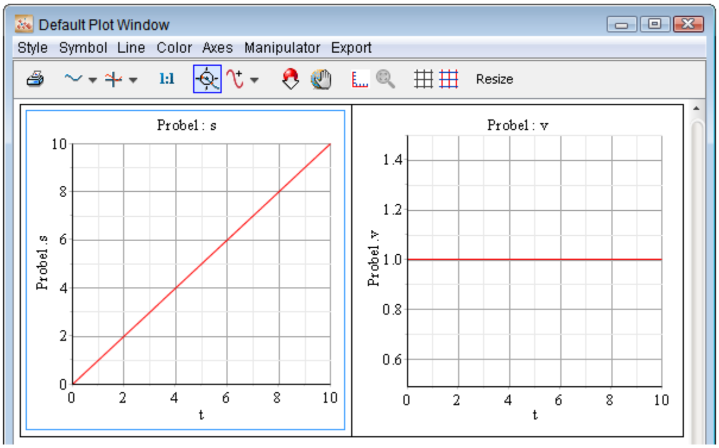
Component	Quantity	Symbol	Library Location	Required Settings
Atmospheric Pressure	**1**		Hydraulic > Reference Components	Use default settings
Hydraulic Fluid Properties	**1**		Hydraulic > Reference Components	**rhoFluid**: $850\ \frac{kg}{m^3}$ **EIFluid**: $8.0\ x\ 10^8\ Pa$ **nuFluid**: $18\ x\ 10^{-6}\ \frac{m^2}{s}$
Fixed Flow Source	**1**		Hydraulic > Sources	Use default settings
Hydraulic Cylinder	**1**		Hydraulic > Actuators	Use default settings
Mass	**1**		1-D Mechanical > Translational > Common	Use default settings
Translational Fixed	**1**		1-D Mechanical > Translational > Common	Use default settings

Note: The Hydraulic cylinder has a cross-sectional area **A** of 1 m^2, while the Fixed Flow Source has a flow **Q** of $1\frac{m^3}{s}$.

The cylinder pushes the sliding mass at a speed of

$$V=\frac{Q}{A}=\frac{1\,m^3 s^{-1}}{1\,m^2}=1\,\frac{m}{s}$$

This is confirmed by running the simulation and probing the speed of the sliding mass.



Simulating Translational Motion with a Fixed Pressure Source

Replace the Fixed Flow Source with the Fixed Pressure Source component shown in **Figure 6.17** and **Table 6.15**. The following model converts flow from a fixed pressure source to translational motion.

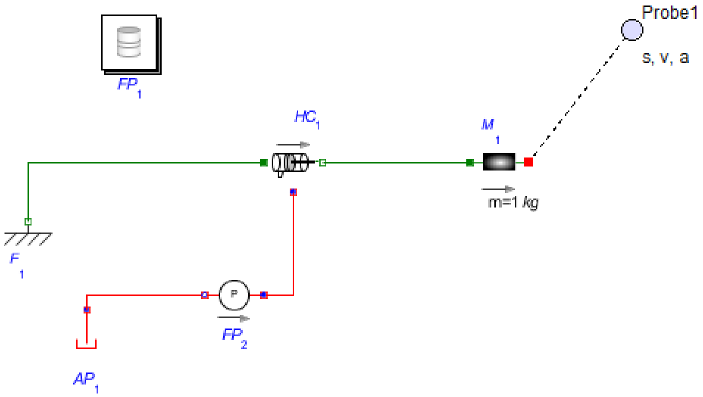
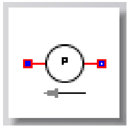


Figure 6.17: Translational Motion with Fixed Pressure Source

Table 6.15: Translational Motion with a Fixed Pressure Sources

Component	Symbol	Library Location	Required Settings
Fixed Pressure Source		Hydraulic > Sources	Use default settings

The force on the Sliding Mass is equal to the cross-sectional of the hydraulic cylinder **A** multiplied by the pressure **P** of the hydraulic fluid.

$$F = A \cdot P = 1\,m^2 \cdot 1\,Pa\ = 1\,N$$

The acceleration of the Sliding Mass is giving by

$$F = m \cdot a = 1\,m^2 \cdot 1\,Pa\ = 1\,N$$

$$1\,N = 1\,kg \cdot a$$

Therefore

$$a = 1\,\frac{m}{s^2}$$

By probing the acceleration, speed and displacement of the Sliding Mass, these values are confirmed with the results in **Figure 6.18**.

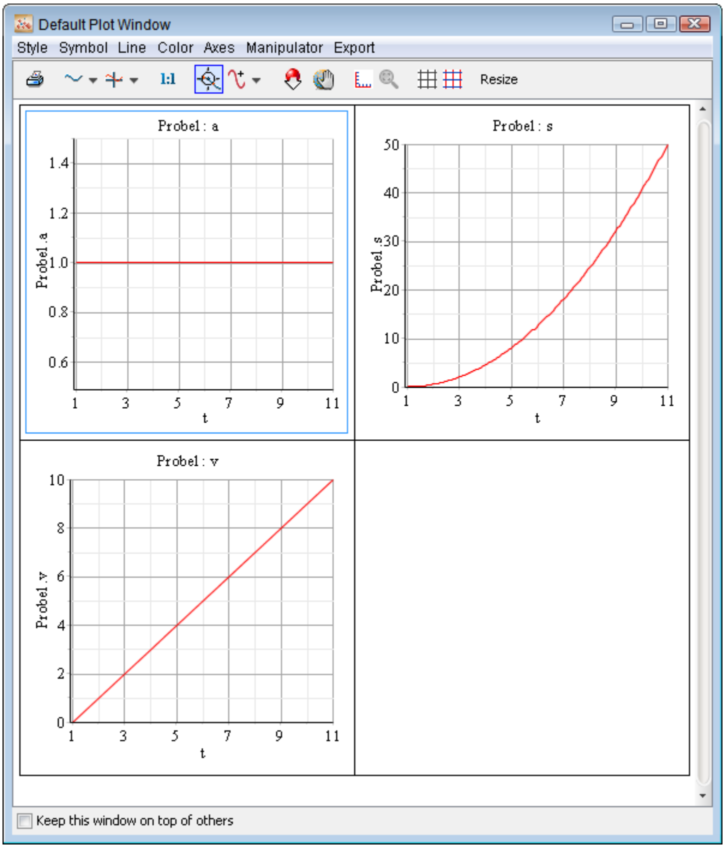


Figure 6.18: Fixed Pressure Source Results

Overview of Actuating Multibody Systems with Hydraulic Components

In the following model connect the 1-D translational port on the hydraulic cylinder to the 1-D translational port on the multibody prismatic joint using a Translation Fixed flange and a Rigid Body mass from **Table 6.16**.

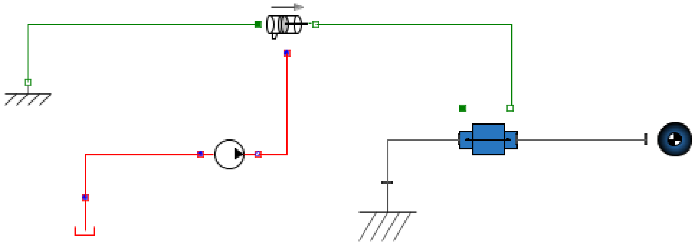


Figure 6.19: Translational Fixed Flange Hydraulic component

Similarly in the following model, connect the 1-D rotational port on the hydraulic motor to the 1-D rotational port on the multibody revolute joint using a Rotational Fixed flange and a Rigid Body mass from **Table 6.16**.

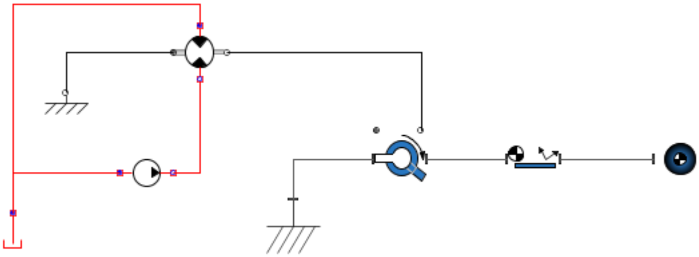
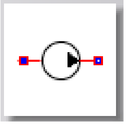
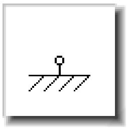
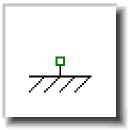
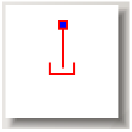
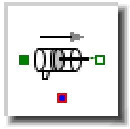
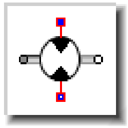
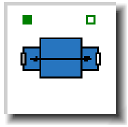
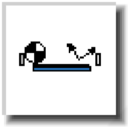
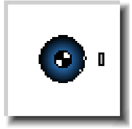
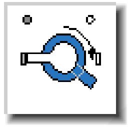
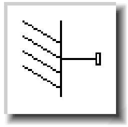


Figure 6.20: Rotational Fixed Flange Hydraulic component

Table 6.16: Actuating Multibody Components

Component	Symbol	Library Location	Required Settings
Fixed Flow Source		Hydraulic > Sources	Use default settings
Rotational Fixed Flange		1-D Mechanics > Rotational > Common	Use default settings
Translational Fixed Flange		1-D Mechanics > Translational > Common	Use default settings

Component	Symbol	Library Location	Required Settings
Atmospheric Pressure		Hydraulic > Reference Components	Use default settings
Hydraulic Cylinder		Hydraulic > Actuators	Use default settings
Hydraulic Motor		Hydraulic > Actuators	Use default settings
Prismatic		Multibody > Joints and Motions	Use default settings
Rigid Body Frame		Multibody > Bodies and Frames	Use default settings
Rigid Body		Multibody > Bodies and Frames	Use default settings
Revolute		Multibody > Joints and Motions	Use default Settings
Fixed Frame		Multibody > Bodies and Frames	Use default settings

Pascal's Principle

Pascal's Principle states that pressure applied to a closed hydraulic system is transmitted everywhere equally. This principle shows that an applied force can be amplified to move loads that would otherwise not be possible.

The model in **Figure 6.21** demonstrates a simple example of Pascal's Principle. A 1 N force (acting on a 0.1 m^2 hydraulic cylinder) transmits hydraulic pressure to a 1 m^2 hydraulic cylinder, which lifts a 1kg load vertically. Normally, a 9.81 N force maintains the height of a 1 kg force, but this simple hydraulic system multiplies the magnitude of a 1 N load by a factor of 10 or $\frac{1\ m^2}{0.1\ m^2}$.

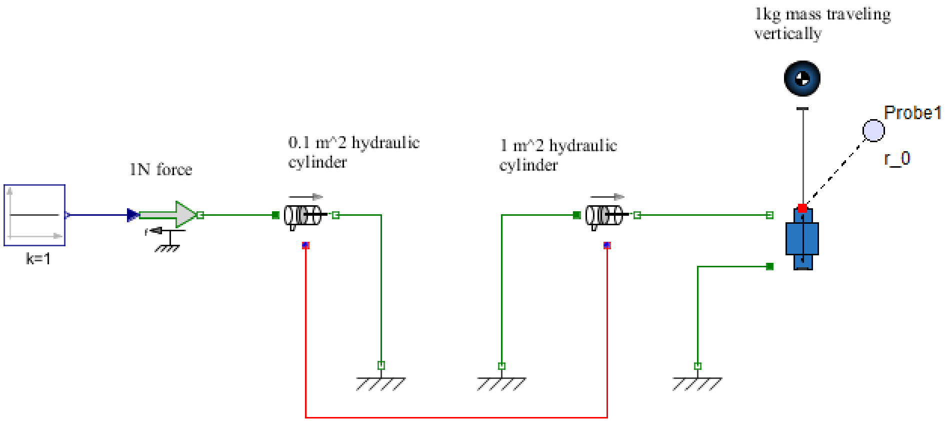


Figure 6.21: Pascal's Principle Example

Overview of Compressibility of Hydraulic Liquids

The compliant cylinder and constant volume chamber components **Table 6.18**) model the compressibility of hydraulic liquids under high pressure. The compliant cylinder component also models pipe wall compliance. Both have to be attached to a node between pipes or inertias as shown in **Figure 6.22**.

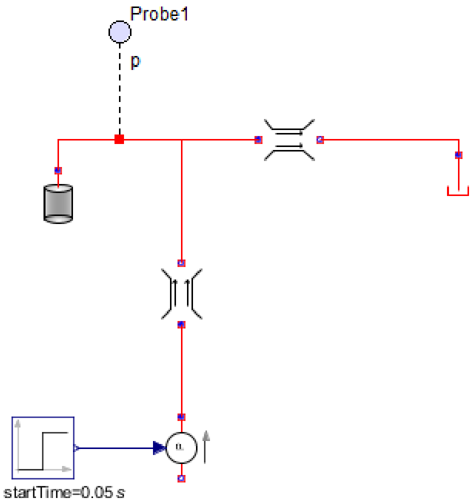


Figure 6.22: Hydraulic Liquids Compressibility

Table 6.17: Hydraulic Liquids Compressibility Components

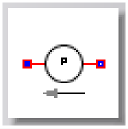
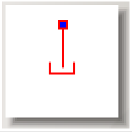
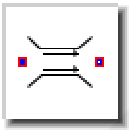
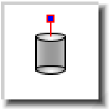
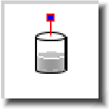
Component	Symbol	Library Location	Required Settings
Fixed Pressure Source		Hydraulic > Sources	Use default settings
Atmospheric Pressure		Hydraulic > Reference Components	Use default settings
Linear Resistance		Hydraulic > Pipes and Valves	Use default settings

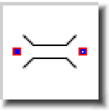
Table 6.18: Confined Hydraulic System Components

Component	Quantity	Symbol	Library Location	Required Settings
Compliant Cylinder	1		Hydraulic > Chambers	Use default settings
Constant Volume	1		Hydraulic > Chambers	Use default settings

Overview of Fluid Inertia Models

The Fluid Inertia component models the inertia of liquid accelerating or decelerating in a pipe and is analogous to mechanical inertia. Fluid Inertia can be significant for large diameter pipes and when the acceleration/deceleration is large. This component is useful when modeling water hammer.

Table 6.19: Fluid Inertia

Component	Quantity	Symbol	Library Location	Required Settings
Fluid Inertia	1		Hydraulic > Pipes and Valves	Use default settings

System Without Fluid Inertial

Figure 6.23 shows a system without fluid inertia.

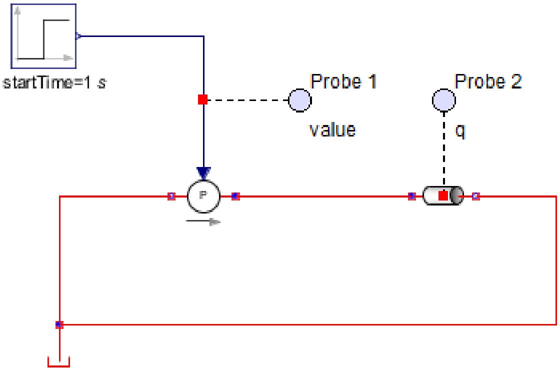


Figure 6.23: System Without Fluid Inertia

System With Fluid Inertial

Figure 6.23 shows a system with fluid inertia.

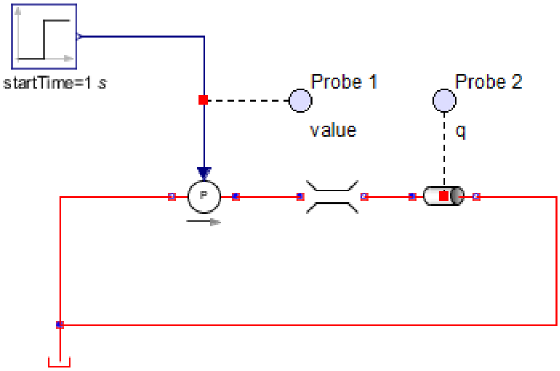


Figure 6.24: System With Fluid Inertia

Figure 6.25 shows typical system flow rates with (green) and without (red) fluid inertia. Introducing fluid inertia adds a lag into the system.

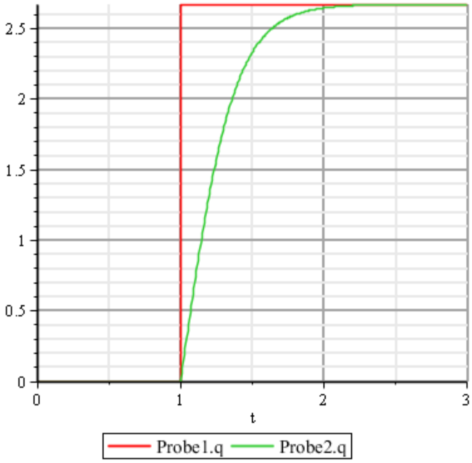


Figure 6.25: System With and Without Fluid Inertia

Overview of Water Hammer Models

Water hammer occurs when a valve suddenly stops (or significantly restricts) flow in a pipe, resulting in a pressure surge due to a momentum change in the fluid inertia. This pressure surge bounces off the closed valve and travels up and down the pipe, potentially causing significant damage to the entire pipe. Water hammer is traditionally modeled by the numerical solution of the following equations.

$$\frac{dV(x,t)}{dt} + \frac{1}{\rho}\,\frac{(dP(x,t))}{dt} + \frac{fV(x,t)|V(x,t)|}{2\,\mathrm{D}}$$

$$\frac{dV(x,t)}{dx} + \frac{1}{Ks}\,\frac{dP(x,t)}{dt} = 0$$

$$Ks = \frac{1}{\frac{1}{K} + \frac{\mathrm{D}}{E\,t}}$$

Where

V(x,t) is the pipe velocity

***P*(x,t)** is the pipe pressure

ρ is the liquid density

D is the pipe diameter

t is the pipe wall thickness

K is the liquid bulk modulus

E is Young’s modulus for the pipe

f is the friction factor

These equations (with the appropriate boundary and initial conditions) are typically solved numerically, requiring custom code to solve the equations using the method of characteristics.

Example: Water Hammer

Another method of simulating water hammer involves building a lumped parameter pipeline model. The pipeline model includes effects such as flow inertia, flow resistance (through pipe friction), pipe compliance, and fluid compressibility.

The following figure shows a discretized pipeline with inertial and resistive properties, initially pressurized at one end to create flow. After two seconds, a valve at the other end is closed, resulting in a pressure surge.

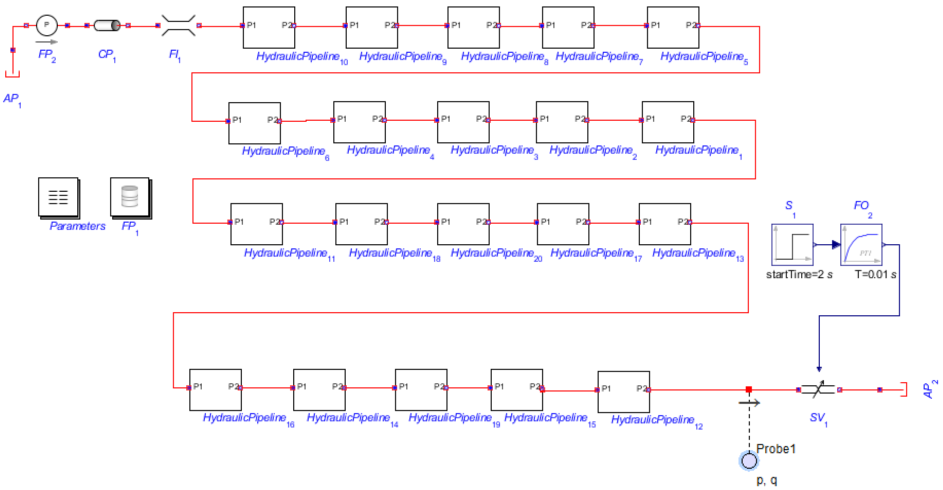


Figure 6.26: Water Hammer

Each subsystem consists of a constant volume chamber, a pipe, and a fluid inertia component as shown in **Figure 6.27**. A pipeline (of total length L and volume V) with N segments has $N + 1$ pipes, each with a length,

$$\frac{L}{N+1}$$

N+1 fluid inertia components, each with a length,

$$\frac{L}{N+1}N$$

N constant volume chambers, each with a volume,

$$\frac{V}{N}$$

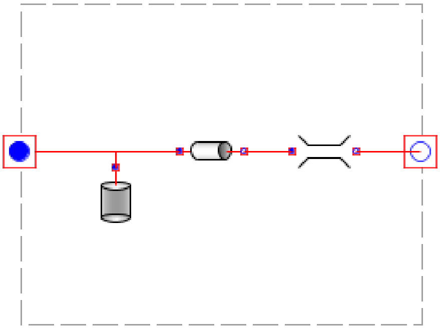


Figure 6.27: Discretized Pipeline Segment

If pipe wall compliance is significant, replace the constant volume chamber with a compliant cylinder. If necessary, the water hammer simulation in MapleSim could be compared to the results given by solving the following partial differential equations to determine pressure transients.

Figure 6.28 plots the pressure and flow rate at the end of a pipe for a valve that rapidly closes after 2 seconds. **Table 6.20** shows the model parameters.

Table 6.20: Water Hammer Parameters

Symbol	Description	Values
D	Pipe hydraulic diameter	0.1 m
L	Pipe length	25 m
N	Number of segments	20
ϵ	Pipe internal roughness	$1.0 \, x \, 10^{-4}$ m
t	Pipe wall thickness	.001 m
E	Pipe Young's Modulus	$70 \, x \, 10^{9} \, Pa$
ρ	Fluid density	$1000 \, \frac{kg}{m^3}$
ν	Fluid kinematic viscosity	$10^{-3} \, \frac{m^2}{s}$
K	Fluid Bulk Modulus	$200 \, x \, 10^{6} \, Pa$
-	Number of pipe nodes	20
	Upstream pressure	$500 \, kPa$

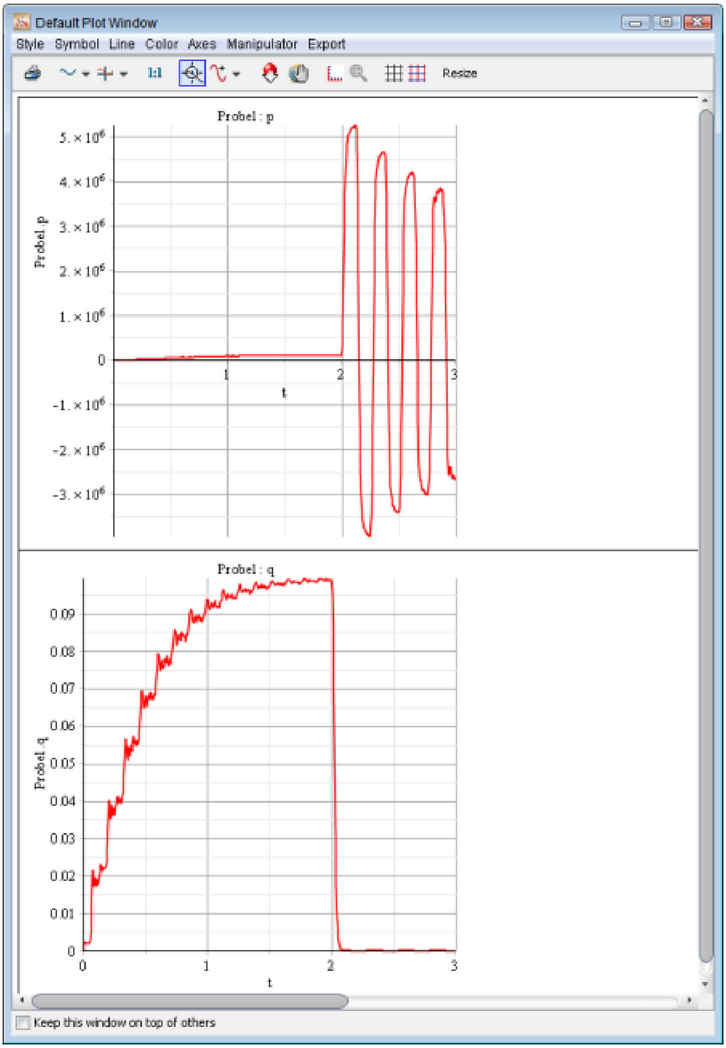


Figure 6.28: Water Hammer Pressure Flow Rate

The maximum pressure is about 5 x $10^6 Pa$, with the liquid reaching a flow rate of $0.099\ \frac{m^3}{s}$. The maximum pressure can also be calculated using the Joukousky equation,

$$\Delta P = \rho c \frac{\Delta Q}{A}$$

$$c = \sqrt{\frac{Ke}{\rho}}$$

$$Ke = \frac{1}{\frac{1}{K} + \frac{D}{E\,t}}$$

Substituting in the parameters from **Table 6.20** and assuming

$$\Delta Q = 0.099\,\frac{m^3}{s}$$

gives

$$\Delta P \approx 5\times 10^6 Pa$$

This result agrees with the MapleSim model.

Example: Attenuating Water Hammer with an Accumulator

A hydraulic accumulator is a reservoir, often located near a valve, that stores non-compressible hydraulic fluid under pressure. An accumulator acts as a safety valve by allowing fluid to enter the reservoir when the pressure increases beyond a certain threshold value. This action attenuates the magnitude and frequency of the pressure waves.

MapleSim does not have a built-in Accumulator block, but this functionality is easily modeled with the Custom Component template using the following equations. For a complete description on how to create custom components, see *Creating Custom Modeling Components (page 65)* **Creating Custom Modeling Components**.

$$eq := \left[q(t) = \dot{V}F(t),\ VF(t) = \begin{cases} Ks\,p(t) & p(t) \le ppr \\ Vpr + (k\cdot(p(t) - ppr)) & ppr < p(t) \quad p(t) < pmax \\ Vmax + (Ks\,(p(t) - pmax)) & pmax \le p(t) \end{cases},\ k = \frac{(Vmax - Vpr)}{(pmax - ppr)} \right]$$

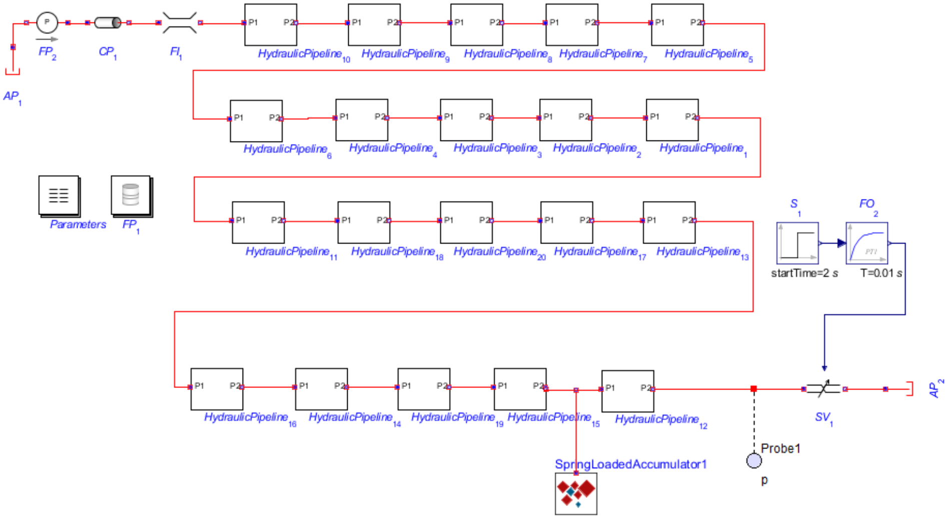
$$params := \left[\mathrm{Vmax} = 0.1, \mathrm{ppr} = 10^5, pmax = 3\cdot 10^6, Ks = 4\cdot 10^{-10}, Vpr = 0\right]$$

$$initialconditions := [VF(0) = 0]$$

Table 6.21: Accumulator Parameters Custom Component

Description	Values
Vmax	0.1 m
ppr	100000
Pmax	3000000
Ks	$10\ x\ 10^{-10}$ m
Vpr	0

The following figure shows the same pipeline with a pressure accumulator. After two seconds, a valve at the other end is closed, resulting in a pressure surge. **Figure 6.29** shows the pressure surge at the end of a pipeline with an accumulator.



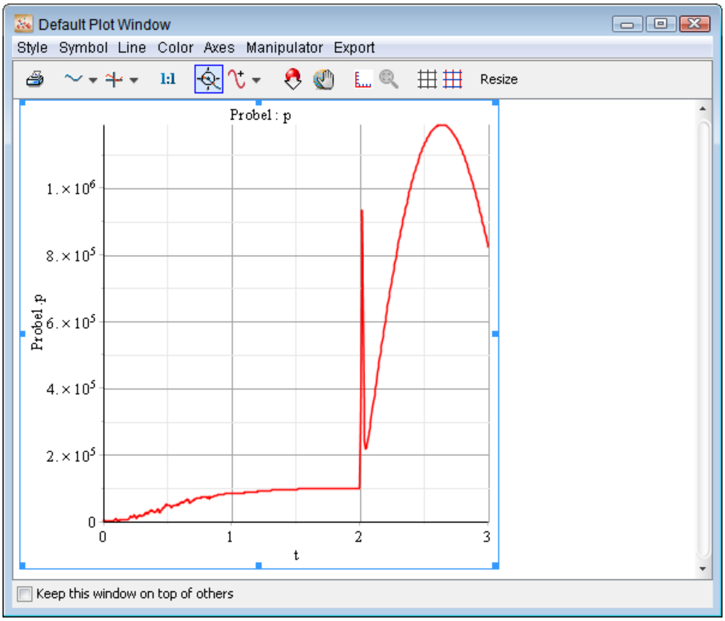


Figure 6.29: Pressure Surge With an Accumulator

Overview of Hydraulic Custom Components

Two examples of hydraulic custom components are centrifugal pumps and vertical pumps. For a complete description on how to create custom components see *Creating Custom Modeling Components (page 65)* **Creating Custom Modeling Components**.

Centrifugal Pumps

Typically, manufacturers provide head flow rate charts for centrifugal pumps, as shown in **Figure 6.30**.

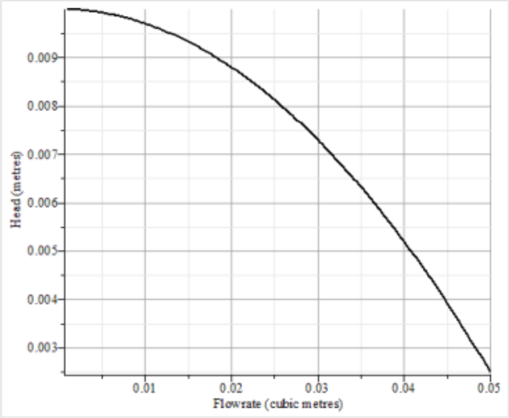


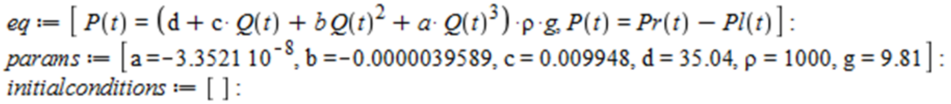
Figure 6.30: Head Flow Rate

Data from these charts is easily implemented into a custom component.

To implement chart data

1. Read several sets of head flow rate points from the plot.
2. Fit these data points to a polynomial using the Maple curve-fitting functionality.
3. Implement the polynomial into a custom component. **Figure 6.31**, for example, shows the custom component equations for a centrifugal pump (including the best-fit parameters).

Note: Since the equation is a polynomial, several solutions may exist.



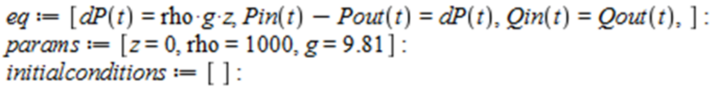
$$eq := \left[\, P(t) = \left(\mathrm{d} + \mathrm{c}\cdot Q(t) + b\,Q(t)^2 + a\cdot Q(t)^3\right)\cdot \rho\cdot g,\ P(t) = Pr(t) - Pl(t)\right]:$$
$$params := \left[\mathrm{a} = -3.3521\ 10^{-8},\ \mathrm{b} = -0.0000039589,\ \mathrm{c} = 0.009948,\ \mathrm{d} = 35.04,\ \rho = 1000,\ \mathrm{g} = 9.81\right]:$$
$$initialconditions := [\ ]:$$

Figure 6.31: Centrifugal Pump Custom Component Equations

Note: Assigning the value rhoFluid to the Density parameter assigns the values the Hydraulic Parameter block to density.

Vertical Pipes

Since gravity head is usually insignificant in mechanical-hydraulic systems, the base pipe component in MapleSim does not model vertical pipe travel. For low-pressure systems, gravity head can be significant. **Figure 6.32** shows the custom component equations to simulate gravity head.



```
eq := [dP(t) = rho·g·z, Pin(t) − Pout(t) = dP(t), Qin(t) = Qout(t), ] :
params := [z = 0, rho = 1000, g = 9.81] :
initialconditions := [ ] :
```

Figure 6.32: Gravity Head Custom Component Equations

7 Reference: MapleSim Keyboard Shortcuts

Table 7.1: Opening, Closing, and Saving a Model

Task	Windows and Linux	Macintosh
Create a new model	**Ctrl** + **N**	**Command** + **N**
Open an existing model	**Ctrl** + **O**	**Command** + **O**
Close the active document	**Ctrl** + **F4**	**Command** + **W**
Save a model as an .msim file	**Ctrl** + **S**	**Command** + **S**

Table 7.2: Building a Model in the Block Diagram View

Task	Windows and Linux	Macintosh
Rotate the selected modeling component 90 degrees clockwise	**Ctrl** + **R**	**Command** + **R**
Rotate the selected modeling component 90 degrees counter-clockwise	**Ctrl** + **L**	**Command** + **L**
Flip the selected modeling component vertically	**Ctrl** + **F**	**Command** + **F**
Flip the selected modeling component horizontally	**Ctrl** + **H**	**Command** + **K**
Group the selected modeling components into a subsystem	**Ctrl** + **G**	**Command** + **G**
Display or hide probes in the model workspace	**Ctrl** + **D**	**Command** + **D**

Table 7.3: Browsing a Model in the Block Diagram View

Task	Windows and Linux	Macintosh
View the selected modeling component or subsystem in detail	**Ctrl** + **M**	**Command** + **M**
Zoom into the model workspace	**Ctrl** + numeric keypad **plus** key	**Command** + numeric keypad **plus** key
Zoom out from the model workspace	**Ctrl** + numeric keypad **minus** key	**Command** + numeric keypad **minus** key

Table 7.4: Browsing a Model in the 3-D View

Task	Windows and Linux	Macintosh
Move the camera around a 3-D model in the perspective view	**Ctrl** + left mouse button click and drag	**Command** + mouse click and drag
Panning a 3-D model	**Shift** + left mouse button click and drag	**Shift** + mouse click and drag
Zoom into or out from the 3-D workspace	**Alt** + left mouse button click and drag, or move the mouse wheel forward (zoom in), or backward (zoom out).	**Alt** + mouse click and drag, or mouse wheel

Glossary

Term	Description
2-D math notation	Formatting option that allows you to enter mathematical text, such as superscripts, subscripts, and Greek characters.
3-D workspace	The area of the MapleSim window in which you can build and edit a 3-D model.
Attached shapes	Shapes that you can display in a 3-D model to create a realistic representation of a system model. Attached shapes include cylinders, trace lines, and CAD geometry that you import from another file.
Camera	The point of view from which a 3-D scene is viewed.
Camera tracking	The process by which a camera follows the movement of a target 3-D component that you select. The target component is centered in the 3-D workspace during an animation.
Custom component	A user-defined component that you can create and add to a MapleSim model using the Custom Component Template.
Custom library	A collection of modeling components and subsystems that can be saved in a user-defined palette and used in a future MapleSim session.
Embedded component	Configurable graphical controls, buttons, meters, and other interactive components that you can add to a Maple standard worksheet to analyze, manipulate, and visualize equations and Maple commands.
Implicit geometry	Default cylinders and spheres that are displayed in a 3-D model to represent modeling components.
Maple package	A collection of routines or commands that can be used in Maple. Most Maple packages provide a set of commands for a particular mathematical or scientific domain, or field of study.
MapleSim component library	The default collection of domain-specific modeling components included in MapleSim. These modeling components can be found in the gray palettes in the **Libraries** tab.
Model workspace	The area of the MapleSim window in which you can build and edit a model in a block diagram view.
Orthographic view	A type of 3-D view that uses parallel projection and displays lines in the view plane at their "true length." In MapleSim, you can view a model from front, top, and side orthographic views.
Perspective view	A 3-D view that allows you to examine and browse a model from any direction in 3-D space.

Term	Description
Probe	The tool used to identify quantities of interest in order to simulate a MapleSim model.
Shared subsystem	A subsystem copy that shares the same configuration as other subsystems. All shared subsystems are linked to a particular *subsystem definition*, which defines the configuration.
Stand-alone subsystem	A subsystem that is not linked to a *subsystem definition* and can be edited and manipulated independent of other subsystems in a model.
Subsystem	A collection of modeling components grouped in a single block.
Subsystem definition	A subsystem block that defines the configuration for a series of *shared subsystems*.

Index