

MapleSim User's Guide

**Copyright © Maplesoft, a division of Waterloo Maple Inc.
2022**

MapleSim User's Guide

Copyright

Maplesoft, MapleSim, and Maple are all trademarks of Waterloo Maple Inc.

© Maplesoft, a division of Waterloo Maple Inc. 2008-2022. All rights reserved.

No part of this book may be reproduced, stored in a retrieval system, or transcribed, in any form or by any means — electronic, mechanical, photocopying, recording, or otherwise. Information in this document is subject to change without notice and does not represent a commitment on the part of the vendor. The software described in this document is furnished under a license agreement and may be used or copied only in accordance with the agreement. It is against the law to copy the software on any medium except as specifically allowed in the agreement.

Linux is a registered trademark of Linus Torvalds.

Mac and Macintosh are trademarks of Apple Inc., registered in the U.S. and other countries.

Microsoft, Excel, and Windows are registered trademarks of Microsoft Corporation.

Modelica is a registered trademark of the Modelica Association.

All other trademarks are the property of their respective owners.

This document was produced using Maple and DocBook.

Contents

Introduction	xiii
1 Getting Started with MapleSim	1
1.1 Physical Modeling in MapleSim	1
Topological or “Acausal” System Representation	1
Mathematical Model Formulation and Simplification	1
Advanced Differential Algebraic Equation Solvers	1
Acausal and Causal Modeling	2
1.2 The MapleSim Window	6
1.3 Basic Tutorial: Modeling an RLC Circuit and DC Motor	8
Building an RLC Circuit Model	8
Specifying Component Properties	12
Adding a Probe	12
Simulating the RLC Circuit Model	14
Building a Simple DC Motor Model	15
Simulating the DC Motor Model	16
2 Building a Model	19
2.1 The MapleSim Component Library	19
Viewing Help Topics for Components	20
Updating Models Created in a Previous Release of MapleSim	20
2.2 Browsing a Model	20
Model Tree	21
Model Navigation Controls	23
2.3 Defining How Components Interact in a System	24
2.4 Specifying Component Properties	25
Specifying Parameter Units	25
Specifying Initial Conditions	26
2.5 Creating and Managing Subsystems	28
Example: Creating a Subsystem	29
Viewing the Contents of a Subsystem	30
Adding Multiple Copies of a Subsystem to a Model	31
Editing Subsystem Definitions and Shared Subsystems	34
Working with Standalone Subsystems	38
2.6 Global and Subsystem Parameters	42
Global Parameters	42
Subsystem Parameters	44
Creating Parameter Blocks	45
Creating Parameter Sets	50
Using Advanced Parameter and Variable Settings	51
2.7 Attaching Files to a Model	57
2.8 Creating and Managing Custom Libraries	58
Example: Creating a Custom Library from an Existing Model	58

2.9	Annotating a Model	60
	Example: Adding Text Annotation to a Model	61
2.10	Entering Text in 2-D Math Notation	62
2.11	Creating a Data Set for an Interpolation Table Component	63
	Example: Creating a Data Set in Maple	63
2.12	Best Practices: Building a Model	64
	Best Practices: Laying Out and Creating Subsystems	64
	Best Practices: Building Electrical Models	65
	Best Practices: Building 1-D Translational Models	67
	Best Practices: Building Multibody Models	68
	Best Practices: Building Hydraulic Models	69
	Best Practices: Enforcing Initial Conditions	70
3	Creating Custom Modeling Components	71
3.1	Understanding Custom Components	71
	Creating a Simple Custom Component	72
	Typical Uses	73
	Using The Custom Component Template	74
3.2	Creating Custom Components with Signal-Flow Behavior	74
	Creating a Simple Signal-Flow Custom Component	74
	Using Differential Equations in Custom Components	80
3.3	Creating Custom Components with Physical Connections	80
	Deriving the System Equations for a Resistor	81
3.4	Working with Custom Components in MapleSim	82
	Save a Custom Component as Part of the Current Model	82
	Add a Custom Component to a Custom Library	82
	Edit a Custom Component	83
3.5	Example: Creating a Nonlinear Spring-Damper Custom Component	83
	Opening the Custom Component Template	84
	Defining the Component Name and Equations	84
	Defining Component Ports	85
	Checking Dimensions	87
	Generating the Custom Component	88
4	Simulating and Visualizing a Model	89
4.1	How MapleSim Simulates a Model	89
	Modelica Description	89
	Model Description	89
	System Equations	89
	Simplified Equations	90
	Integration and Event Handling	90
	Simulation Results	90
4.2	Simulating a Model	91
	Simulation and Advanced Simulation Settings	92
	Editing Probe Values	97

Storing Parameter Sets to Compare Simulation Results	97
4.3 Simulation Progress Messages	98
4.4 Managing Simulation Results and Snapshots	99
Storing Results	99
Saving and Using Snapshots	99
4.5 Customizing Plot Window Configurations	100
Example: Plotting Multiple Quantities in Individual Graphs	101
Example: Plotting One Quantity versus Another	103
4.6 Visualizing a Multibody Model	106
3-D Visualization and Multibody Settings	106
The 3-D Workspace	108
Viewing and Browsing 3-D Models	109
Adding Shapes to a 3-D Model	111
Building a Model in the 3-D Workspace	114
Example: Building and Animating a Double Pendulum Model in the 3-D Work- space	118
4.7 Best Practices: Simulating and Visualizing a Model	126
Use an External C Compiler to Run Simulations with Longer Durations	126
Compare Results Generated by Sections of Your Model	126
5 Analyzing and Manipulating a Model	127
5.1 Overview	127
MapleSim Apps and Templates	127
Working with Apps	129
Working with MapleSim Equations and Properties in a Maple Worksheet	129
Using Subsystems	130
5.2 Retrieving Equations and Properties from a Model	130
5.3 Analyzing Linear Systems	131
Linear System Analysis	131
Create Component	132
5.4 Optimizing Parameters	132
5.5 Generating and Exporting C Code from a Model	133
Preparing the Model for Export in MapleSim	134
Opening the Code Generation App	136
Loading the Subsystem	136
Customizing, Defining, and Assigning Parameter Values to Specific Ports	137
Selecting the Code Generation Options	138
Generating and Saving the C code	140
5.6 Generating a Custom Component from External C Code/Library Definition	141
Opening the External C Code/Library Definition App	142
Specifying the C/Library Code Location and Options	142
Defining the C/Library Code Location and Options	143
Component Generation	144
5.7 Working with the MapleSim API and Maple Commands	144

5.8 Working with Maple Embedded Components	144
6 MapleSim Tutorials	147
6.1 Tutorial 1: Modeling a DC Motor with a Gearbox	147
Adding a Gearbox to a DC Motor Model	147
Simulating the DC Motor with the Gearbox Model	149
Grouping the DC Motor Components into a Subsystem	150
Assigning Global Parameters to a Model	151
Changing Input and Output Values	153
6.2 Tutorial 2: Modeling a Cable Tension Controller	154
Building a Cable Tension Controller Model	155
Specifying Component Properties	156
Simulating the Cable Tension Controller	157
6.3 Tutorial 3: Modeling a Nonlinear Damper	158
Generating a Spring Damper Custom Component	158
Providing Damping Coefficient Values	159
Building the Nonlinear Damper Model	160
Assigning a Parameter to a Subsystem	163
Simulating the Nonlinear Damper with Linear Spring Model	164
6.4 Tutorial 4: Modeling a Planar Slider-Crank Mechanism	166
Creating a Planar Link Subsystem	167
Defining and Assigning Parameters	170
Creating the Crank and Connecting Rod Elements	171
Adding the Fixed Frame, Sliding Mass, and Joint Elements	171
Specifying Initial Conditions	173
Simulating the Planar Slider-Crank Mechanism	173
6.5 Tutorial 5: Using the Custom Component Template	174
Example: Modeling a Temperature Dependent Resistor	175
Example: Compliant Contact and Piecewise Functions	180
Example: Custom Ports	185
Advanced Uses for Custom Components	191
6.6 Tutorial 6: Using the External C Code/DLL Custom Component App	196
6.7 Tutorial 7: Using the Equation Extraction App	201
App Description	202
Generating the Equations	204
6.8 Tutorial 8: Modeling Hydraulic Systems	206
Computational Issues	207
Basic Hydraulic Library Components	207
Basic Hydraulic Equations	209
Analysis of Simple Hydraulic Networks	211
Overview of Controlling Hydraulic Flow Path	214
Mechanical and Hydraulic Systems	215
Overview of Compressibility of Hydraulic Liquids	223
Overview of Fluid Inertia Models	224

Overview of Water Hammer Models	226
Overview of Hydraulic Custom Components	234
7 Reference: MapleSim Keyboard Shortcuts	237
Glossary	243
Index	245

List of Figures

Figure 1.1: Causal Model Block Diagram	2
Figure 1.2: Acausal Model Block Diagram	3
Figure 1.3: Simple Through and Across Variable Model	3
Figure 1.4: Simple Through and Across Variable Electrical Model	3
Figure 1.5: RLC Circuit	4
Figure 1.6: RLC MapleSim Circuit	5
Figure 1.7: MapleSim Window	6
Figure 1.8: Voltage Response Plot	14
Figure 1.9: EMF and Inertia connections	16
Figure 1.10: Plots of DC Motor Torque and Speed	17
Figure 2.1: Components view in the Model Tree	21
Figure 2.2: Component selection using the Model Tree	22
Figure 2.3: Exploring a subsystem	23
Figure 2.4: Model Navigational Controls	23
Figure 2.5: Specifying Units using the Conversion Block	26
Figure 2.6: Initial Conditions	27
Figure 2.7: Subsystem Group	28
Figure 2.8: Creating a Subsystem	29
Figure 2.9: Creating Multiple Subsystems	31
Figure 2.10: Subsystem Definition	32
Figure 2.11: Adding Multiple Subsystems to a Model	33
Figure 2.12: DC Motor Subsystem	35
Figure 2.13: Copy Subsystem Dialog	41
Figure 2.14: Attachments	57
Figure 2.15: Verifying Force Arrows	68
Figure 2.16: Center of Mass Placement Best Practice	68
Figure 2.17: Hydraulic Model	69
Figure 3.1: The Add Apps or Templates tab	72
Figure 3.2: Equations Defining a Custom Component	75
Figure 3.3: Port Mappings	76
Figure 3.4: Variable to Port Mapping	76
Figure 3.5: Generated Custom Component	77
Figure 3.6: Completed Custom Component Model	78
Figure 3.7: Double Mass-Spring-Damper Equations	80
Figure 3.8: Port Mapping for Double Mass-Spring-Damper	80
Figure 3.9: Resistor Port Mapping	82
Figure 3.10: Nonlinear Spring-Damper Custom Component	83
Figure 4.1: Simulation Process	91
Figure 4.2: Simulation Results Progress Messages	98
Figure 4.3: Snapshots in the Advanced Simulation Settings	100
Figure 4.4: The Stored Results Palette and Snapshots	100

Figure 4.5: Simulation Graphs	102
Figure 4.6: Custom Plot Window	103
Figure 4.7: Plot One Quantity Versus Another	105
Figure 4.8: 3-D Workspace	108
Figure 4.9: 3-D View Controls	109
Figure 4.10: Perspective View Double Pendulum	110
Figure 4.11: Orthographic View Double Pendulum	110
Figure 4.12: Implicit Geometry Double Pendulum	111
Figure 4.13: Attached Shapes	112
Figure 6.1: Cable Tension Controller	157
Figure 6.2: Nonlinear Damper Model	161
Figure 6.3: Planar Slider-Crank Mechanism	167
Figure 6.4: Temperature Dependent Resistor	179
Figure 6.5: Falling Ball	180
Figure 6.6: Bouncing Ball Dynamics	181
Figure 6.7: Bouncing Ball	183
Figure 6.8: Bouncing Ball Result	185
Figure 6.9: Custom Ports	187
Figure 6.10: Using a Custom Port	188
Figure 6.11: Centrifugal Pump Head Flow Rate Curve	191
Figure 6.12: Centrifugal Pump Custom Component	193
Figure 6.13: External C Code Definition for Windows	197
Figure 6.14: External C Code Definition for Unix	197
Figure 6.15: Flow Through a Pipe	212
Figure 6.16: Controlling Flow Path	215
Figure 6.17: Fixed Flow Rate Source	216
Figure 6.18: Translational Motion with Fixed Pressure Source	218
Figure 6.19: Fixed Pressure Source Results	220
Figure 6.20: Translational Fixed Flange Hydraulic component	220
Figure 6.21: Rotational Fixed Flange Hydraulic component	221
Figure 6.22: Pascal's Principle Example	223
Figure 6.23: Hydraulic Liquids Compressibility	223
Figure 6.24: System without Fluid Inertia	225
Figure 6.25: System with Fluid Inertia	225
Figure 6.26: System with and without Fluid Inertia	226
Figure 6.27: Water Hammer	227
Figure 6.28: Discretized Pipeline Segment	228
Figure 6.29: Water Hammer Pressure Flow Rate	231
Figure 6.30: Pressure Surge with an Accumulator	234
Figure 6.31: Head Flow Rate	235
Figure 6.32: Centrifugal Pump Custom Component Equations	235
Figure 6.33: Gravity Head Custom Component Equations	236

List of Tables

Table 1.1: Through and Across Variable Domain Types	4
Table 1.2: MapleSim Window Components	6
Table 2.1: MapleSim Component Library	19
Table 2.2: Model Navigation Controls	23
Table 2.3: Domain-Specific Connection Line Colors	24
Table 2.4: 2-D Math Notation Key Combinations	62
Table 3.1: Port Map	77
Table 3.2: Signal Flow Components	78
Table 3.3: Characteristics of Through and Across Variables	81
Table 3.4: Through and Across Variable Mathematical Relationship	81
Table 3.5: Resistor Variables and Parameters	81
Table 4.1: Simulation Settings	93
Table 4.2: Advanced Simulation Settings	94
Table 4.3: Multibody Parameter Values	107
Table 4.4: 3-D Visualization Parameter Values	107
Table 4.5: 3-D Workspace Controls	109
Table 5.1: MapleSim Apps	128
Table 5.2: MapleSim Templates	129
Table 6.1: Temperature Dependent Resistor Components	178
Table 6.2: Bouncing Ball Multibody Components	184
Table 6.3: Centrifugal Pump Data	192
Table 6.4: Circular Pipe Parameters	192
Table 6.5: Centrifugal Pump Components	193
Table 6.6: External C Code DLL Custom Components and Required Settings	199
Table 6.7: Basic Hydraulic Library Components	208
Table 6.8: Bernoulli and Darcy Equation Notation	210
Table 6.9: Circular Pipe Parameters	211
Table 6.10: Hydraulic Components	212
Table 6.11: Spool Valve	215
Table 6.12: Translational Motion with Fixed Flow Rate Sources	217
Table 6.13: Translational Motion with a Fixed Pressure Source	219
Table 6.14: Actuating Multibody Components	221
Table 6.15: Hydraulic Liquids Compressibility Components	223
Table 6.16: Confined Hydraulic System Components	224
Table 6.17: Fluid Inertia	224
Table 6.18: Fluid Properties Values	228
Table 6.19: Water Hammer Parameters	229
Table 6.20: Accumulator Parameters Custom Component	233
Table 7.1: Opening, Closing, and Saving a Model	237
Table 7.2: Building a Model in the Block Diagram View	237
Table 7.3: Browsing a Model in the Block Diagram View	237

Table 7.4: Browsing a Model in the 3-D View	239
Table 7.5: Simulating a Model	240
Table 7.6: Navigating the Console Pane	241
Table 7.7: Modifying the Plot Window Layout	241
Table 7.8: Editing a Modelica Custom Component	241
Table 7.9: Miscellaneous	242

Introduction

MapleSim Overview

MapleSim™ 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 hundreds of modeling components, including electrical, hydraulics, 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™ 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 apps and 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 2021.1 requires Maple 2021.1.

Maplesoft™ 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: <ul style="list-style-type: none">• 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. From the Help menu, select Examples to access these examples.
MapleSim User's Guide Examples	Model and Tutorial examples used in the User's Guide. To access these examples, from the Help menu, select Examples > User's Guide Examples . The examples 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 Model Gallery	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/products/maplesim/modelgallery/ .

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.

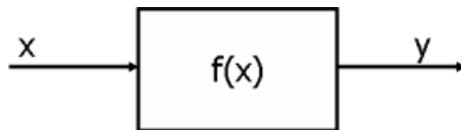


Figure 1.1: Causal Model Block Diagram

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.

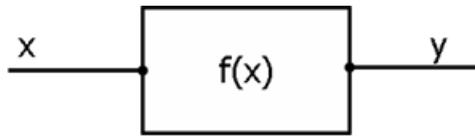


Figure 1.2: Acausal Model Block Diagram

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.

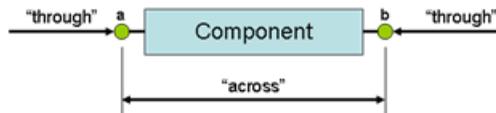


Figure 1.3: Simple Through and Across Variable Model

For an example of sign convention and how arrow direction represents a force acting on the model, from the **Help** menu, select **Examples > User's Guide Examples > Chapter 1**, and then select 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:

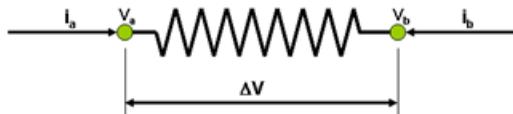


Figure 1.4: Simple Through and Across Variable Electrical Model

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 ($\frac{m}{s}$)
Mechanical (rotational)	Torque ($N.m$)	Angular Velocity ($\frac{rad}{s}$)
Hydraulic	Flow ($\frac{m^3}{s}$)	Pressure ($\frac{N}{m^2}$)
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 Kirchoff'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:

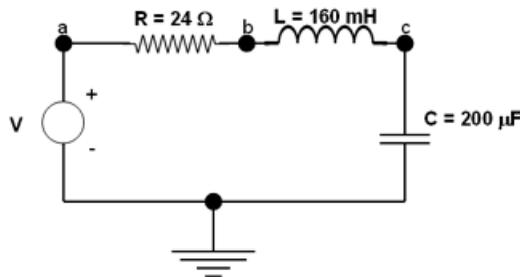


Figure 1.5: RLC Circuit

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

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

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

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

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)

$$V_a = \begin{cases} 0.0 & 0.0 \leq t < 1.0 \\ 1.0 & t \geq 1.0 \end{cases}$$

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.

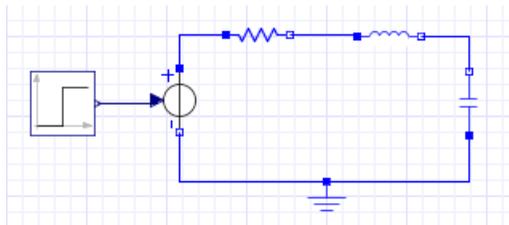


Figure 1.6: RLC MapleSim Circuit

For another example of how a model can be represented using causal and acausal components, from the **Help** menu, select **Examples > User's Guide Examples > Chapter 1**, and then select the **Double Mass Spring Damper** example.

1.2 The MapleSim Window

The MapleSim window contains the following panes and components:

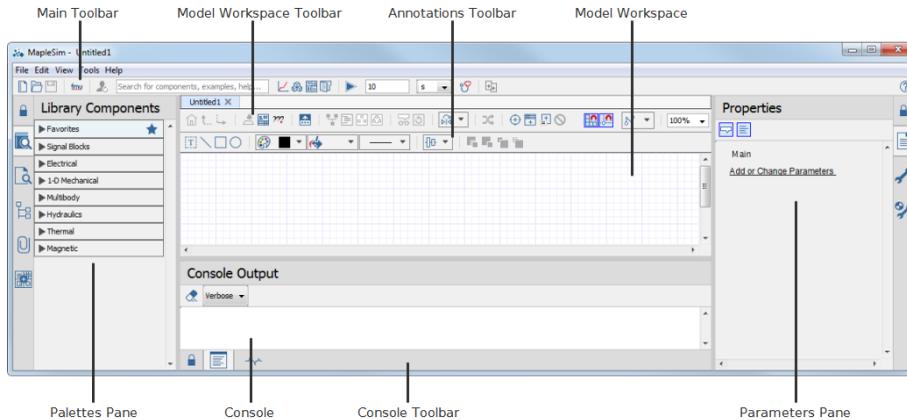


Figure 1.7: MapleSim Window

Table 1.2: MapleSim Window Components

Component	Description
Main Toolbar	Contains tools for running a simulation, viewing simulation results, searching the MapleSim help system, and performing other common tasks.
Model Workspace Toolbar	Contains tools for browsing your model and subsystems hierarchically, changing the model view, viewing the corresponding Modelica code, grouping components, and adding probes.
Annotations Toolbar	Contains tools for adding annotations and laying out objects.
Model Workspace	The area in which you build and edit a model in a block diagram view.

Component	Description	
Palettes Pane	<p>Contains expandable menus with tools that you can use to build a model and manage your MapleSim project. This pane contains five tabs:</p> <ul style="list-style-type: none"> • Library Components (Console	<p>Use buttons on the Console Toolbar to display the following panes:</p> <ul style="list-style-type: none"> • Console Output: displays progress messages indicating the status of the MapleSim engine during a simulation and allows you to clear the console using Clear Console (). • Diagnostics Information: displays diagnostic messages for debugging as you build your model identifying the subsystem in which the errors are located.
Console Toolbar	<p>Contains controls for selecting the type of messages shown in the console ().</p>	

Component	Description
Parameters Pane	<p>Contains the following tabs:</p> <ul style="list-style-type: none"> • Properties ( <h2>1.3 Basic Tutorial: Modeling an RLC Circuit and DC Motor</h2>

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, from the **Help** menu, select **Examples > User's Guide Examples > Chapter 1**, and then select the **RLC Circuit** example. The model you build is identical to the **RLC Circuit** model.

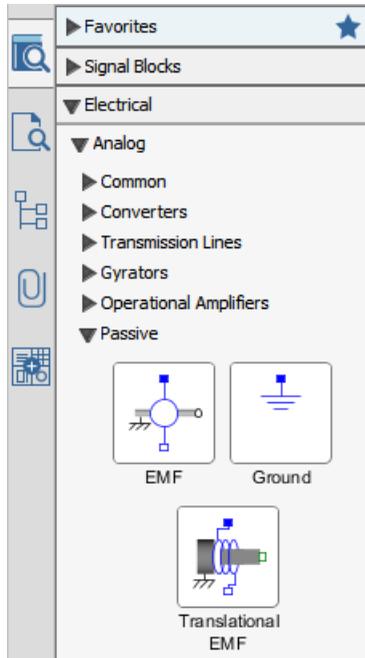
Building an RLC Circuit Model

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

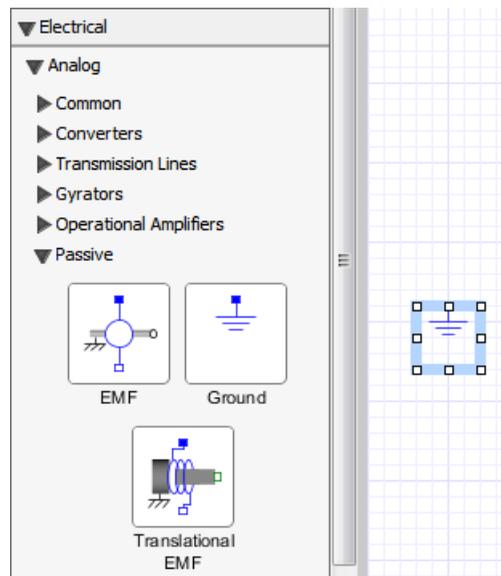
component library. It also contains a step input source, which is a signal generator that drives the input voltage level in the circuit.

To build an RLC circuit:

1. In the **Library Components** 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.



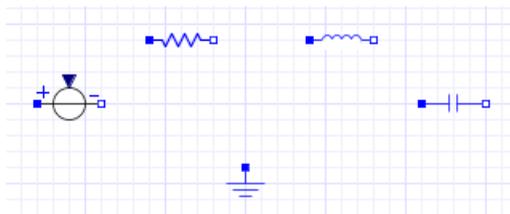
2. From the **Electrical > Analog > Passive** menu, drag the **Ground** component to the **Model Workspace**.



3. Add the following 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 Mac®) the **Signal Voltage** component in the **Model Workspace** and select **Rotate Clockwise**.

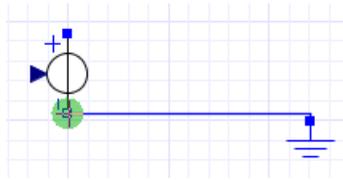
6. To flip the **Signal Voltage** component horizontally, right-click (**Control**-click for Mac) the component again and select **Flip Horizontal**. Make sure that the positive (blue) port is at the top.
7. To rotate the **Capacitor** component clockwise, right-click (**Control**-click for Mac) the **Capacitor** icon in the **Model Workspace** and select **Rotate Clockwise**.

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

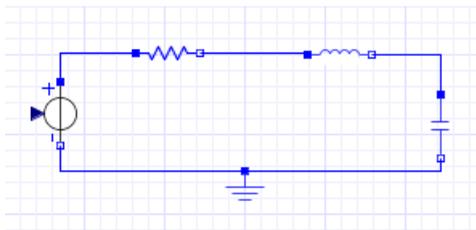
8. Hover your mouse pointer over the **Ground** component port. The port is highlighted in green.



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.



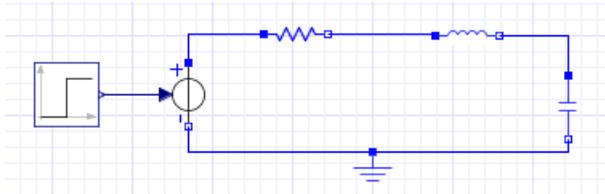
11. Click the port once. The **Ground** component is connected to the **Signal Voltage** component.
12. Connect the remaining components in the arrangement shown below.



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, repres-

ented 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 **Properties** 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 $24\ \Omega$.



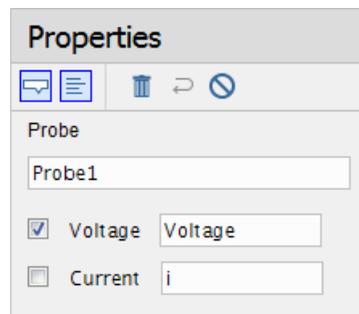
3. Specify the following parameter values for the other components. You can specify units for a parameter by selecting a value from the drop-down menu found beside the parameter value field.
 - For the **Inductor**, specify an inductance of $160\ mH$.
 - For the **Capacitor**, specify a capacitance of $200\ \mu F$.
 - For the **Step** source, specify a **T₀** value of $0.1\ s$.

Adding a Probe

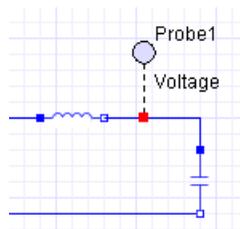
To specify data values for a simulation, you can attach probes to lines or ports to the model. In this example, you will measure the voltage of the RLC circuit.

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. Move 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 **Properties** tab () to the right of the **Model Workspace**.
6. Under the **Properties** 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**.



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, from the **Help** menu, select **Examples > User's Guide Examples > Chapter 1**, and then select 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 (🔧) on the right of the Parameters Pane and in the **Simulation** section, set the simulation duration time (t_d) to 0.5 s.
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.

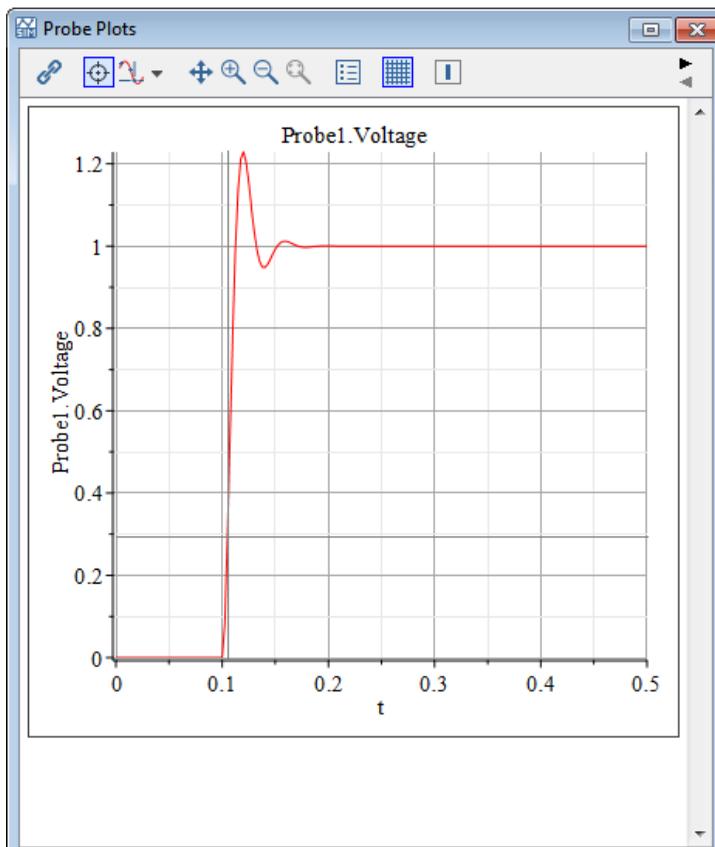


Figure 1.8: Voltage Response Plot

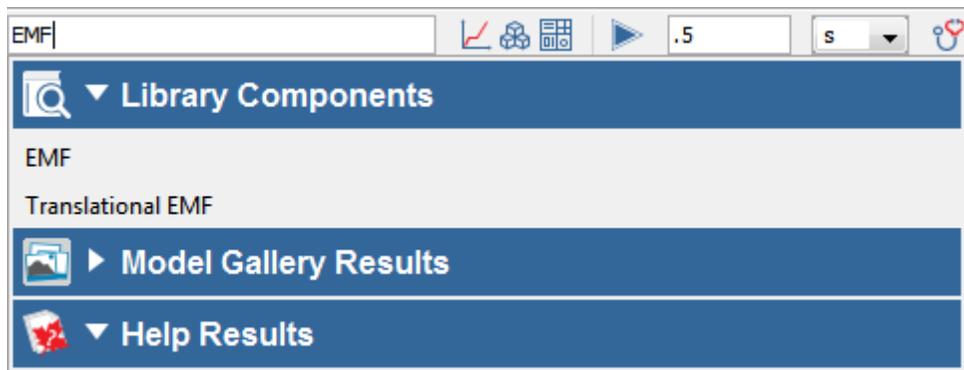
4. Save the model as **RLC_Circuit1.msim**. The probes and modified parameter values are saved as part of the model.

Building a Simple DC Motor Model

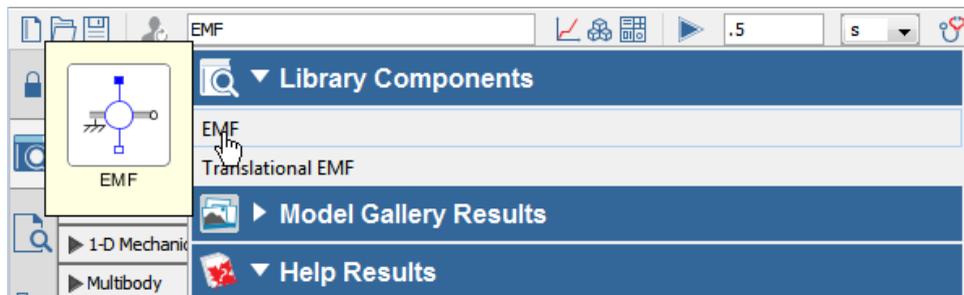
You will now add an electromotive force (EMF) component and a mechanical inertia component to the RLC circuit model to create a DC motor model. In this example, you will add components to the RLC circuit model using the search feature.

To build a simple DC motor:

1. In the **Main** toolbar, type **EMF** in the search bar. A drop-down list displays matches for your search results.



2. Hover over the **EMF** in the **Library Components** section of the drop-down list. The **EMF** component appears in a tool square beside the search field.



3. Drag the **EMF** component to the modeling workspace and place it to the right of the **Capacitor** component.
4. Enter **Inertia** in the search bar.
5. Drag the **Inertia** component to the **Model Workspace** and place it to the right of the **EMF** component.

6. Connect the components as shown below.

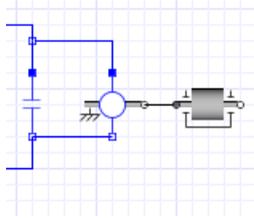


Figure 1.9: EMF and Inertia connections

Note: To connect the positive blue port of the **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 **EMF** component.

8. In the **Properties** tab () , 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, **T₀**, to 1 s.

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 **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 **Properties** tab () , select the **Speed** and **Torque** check boxes and then 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 **Properties** tab.
6. Rename the probe **Output**.
7. Click a blank area in the **Model Workspace**.
8. In the **Settings** tab () , set the simulation duration time (**t_d**) to 5 s.
9. Click **Run Simulation** () in the **Main Toolbar**.
10. Click **Show Simulation Results** () . The following graphs appear.

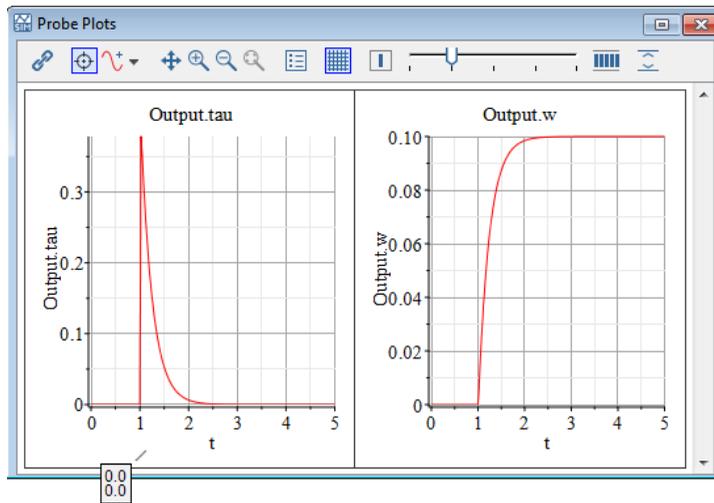


Figure 1.10: Plots of DC Motor Torque and Speed

11. Save the model as **DC_Motor1.msims**.

2 Building a Model

In this chapter:

- *The MapleSim Component Library (page 19)*
- *Browsing a Model (page 20)*
- *Defining How Components Interact in a System (page 24)*
- *Specifying Component Properties (page 25)*
- *Creating and Managing Subsystems (page 28)*
- *Global and Subsystem Parameters (page 42)*
- *Attaching Files to a Model (page 57)*
- *Creating and Managing Custom Libraries (page 58)*
- *Annotating a Model (page 60)*
- *Entering Text in 2-D Math Notation (page 62)*
- *Creating a Data Set for an Interpolation Table Component (page 63)*
- *Best Practices: Building a Model (page 64)*

2.1 The MapleSim Component Library

The MapleSim component library contains hundreds of 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 3.2.3.

Table 2.1: MapleSim Component Library

Library	Description
Signal Blocks	Components to manipulate or generate input and output signals.
Electrical	Components to model electrical analog circuits, single-phase and multiphase systems, and machines.
1-D Mechanical	Components to model 1-D translational and rotational systems.
Multibody	Components, including force, motion, and joint components, to model multibody mechanical systems.
Hydraulics	Components to model hydraulic systems, fluid power systems, cylinders and actuators.
Pneumatics	Components to model Ideal pneumatic systems with cylinders, directional control valves, orifices, and actuators.
Thermal	Components to model heat flow and heat transfer.

Library	Description
Magnetic	Components to model magnetic circuits.

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 71)*.

Viewing Help Topics for Components

To view Help topics in the MapleSim Help system, perform any of the following tasks:

- Right-click (**Control**-click for Mac) a modeling component in any of the palettes and select **Help** from the context menu.
- Search for the topic in the help search box in the main toolbar. Help topics related to your search term are listed in the **Help Results** section.
- Search for the help pages for components in the MapleSim Help system.

Updating Models Created in a Previous Release of MapleSim

In MapleSim 7 or earlier, components from the Modelica Standard Library 3.1 were included in the MapleSim Component Library. MapleSim 2015 was updated to use the Modelica Standard Library 3.2.1, MapleSim 2017 was updated to use the Modelica Standard Library 3.2.2, and MapleSim 2019 was updated to use the Modelica Standard Library 3.2.3.

If you created a model in an earlier version of MapleSim, you can open it in MapleSim 2017 or later. The model will be updated automatically to use equivalent components from the Modelica Standard Library 3.2.3. For more information including how to back up files created in MapleSim 7 or earlier, see **Using MapleSim > Updating Models Created in a Previous Release of MapleSim** in the MapleSim Help system.

2.2 Browsing a Model

Using the **Model Tree** 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

The **Model Tree** tab () is located in the Palettes Pane. Use the **Model Tree** to browse through and optionally search for elements in your model. Nodes in the model tree can represent attachments, search for component types, components, parameters, or probes, depending on the model tree view. To change the model tree view, select a view from the list below the **Find** text box. The following figure shows the drop-down menu available for the *5 DoF Robot* multibody example model.

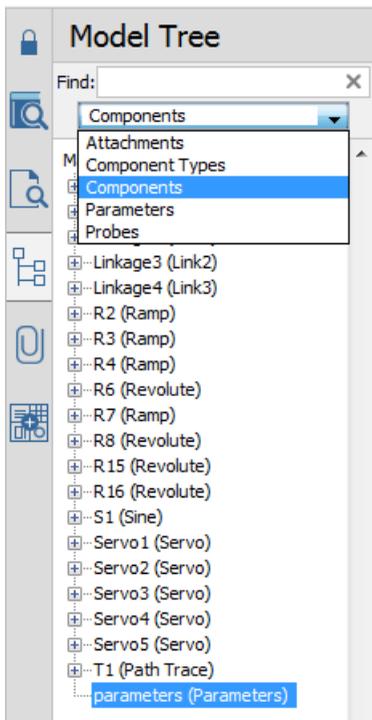


Figure 2.1: Components view in the Model Tree

The following are of the model tree views you can select from.

- **Attachments:** This view shows the files attached to your model. Examples of attachments include worksheets, spreadsheets, and CAD drawings. Double-click an attachment to open the attachment in the appropriate program. Enter a term in **Find** to search for documents that match your term.
- **Component Types:** This view organizes the model tree view according to the type of component or subsystem. Component and subsystem nodes are identified by their type

followed by their name. Enter a term in **Find** to search for component types that match your term (the component's name is ignored in the search).

- **Components:** This view organizes the model tree view according to the *Name* of each component or subsystem. Component and subsystem nodes are identified by their name followed by their type (see **Figure 2.1**). Enter a term in **Find** to search for component and subsystem names that match your term (the component type is ignored in the search). This is the default model tree view.
- **Parameters:** This view shows the parameter definitions in your model. Parameter definitions can come from parameter tables, parameter blocks, Modelica Records, or To Variable components. Enter a term in **Find** to search for parameters names that match your term. For more information about the parameters view (including how to find all references to a parameter), see **Using MapleSim > Building a Model > Navigating and Searching with the Model Tree > Parameters View** in the MapleSim Help system.
- **Probes:** This view shows all the probes in your model. The full path to the probe is given in brackets after the name of the probe. Enter a term in **Find** to search for probes with names that match your term.

To view the parameters associated with a component or subsystem, navigate the model tree to the component node and then select the node. This highlights the element in the **Model Workspace**, changes the **Model Workspace** view to display the element, and populates the **Properties** tab (📄) with the configurable parameters for that element. See **Figure 2.2** for an illustration of component selection.

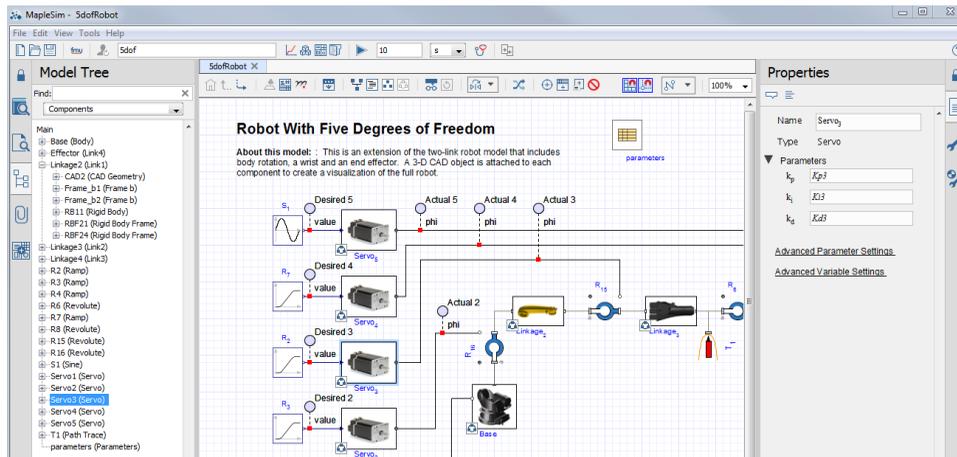


Figure 2.2: Component selection using the Model Tree

If you select more than one component from the model tree, the **Properties** tab displays all the common configurable parameters for the components. If you change a parameter in the **Properties** tab, this updates the parameter for all of the selected components.

To explore a component or subsystem, you can either double-click the node in the model tree or expand the node and then select one of its children. The **Model Workspace** view changes to the appropriate level to explore the component or subsystem.

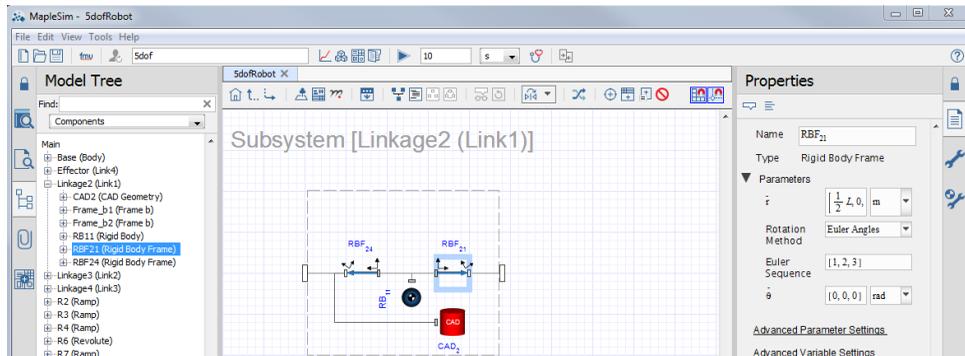


Figure 2.3: Exploring a subsystem

For more information on the **Model Tree** and how to manage complex models, see the **Using MapleSim > Building a Model > Navigating and Searching with the Model Tree** section of the MapleSim Help system.

Model Navigation Controls

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



Figure 2.4: Model Navigational Controls

The following table summarizes what these controls do and gives the keyboard shortcuts associated with them.

Table 2.2: Model Navigation Controls

Control	Keyboard Shortcut	Description
	Home	Return to the Main level of your model.
	Ctrl+Up Arrow	Navigate to the parent component.

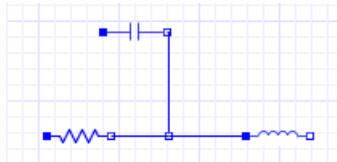
Control	Keyboard Shortcut	Description
	Command+Up Arrow (Mac)	
	Ctrl+Down Arrow Command+Down Arrow (Mac)	Display the contents (or children) of the selected component.

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 2.3: 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
Pneumatic	Light blue

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** in the MapleSim Help system.

Components can have either scalar or vector connection ports. A scalar port has only one quantity associated with it while a vector port can have more than one quantity (or dimension) associated with it. Connections between ports with different dimensions are difficult to manage because what quantities are physically connected is not obvious in the **Model Workspace**. The **Connections Manager** simplifies these types of connections by letting you examine which ports are connected and change them if necessary. To access the **Connections Manager**, select a connection line in the **Model Workspace** and then select the **Properties** tab (). For more information, see **Using MapleSim > Building a Model > Using the Connection Manager** 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 **Properties** tab () located on the right side of the MapleSim window.

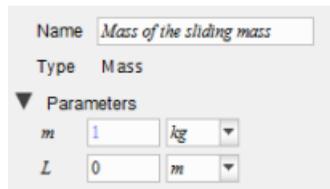
Note: Not all components provide editable parameter values.

You can 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 62)*.

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 > Specifying Parameters** 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*.



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 **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 **Conversion Block** component is connected between a translational **Position Sensor** and a **Feedback** component to convert the units of an output signal.

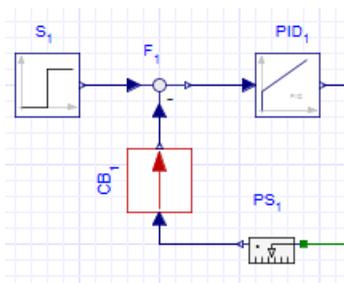


Figure 2.5: Specifying Units using the Conversion Block

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 **Properties** tab, along with the other configurable parameter values for that component.

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

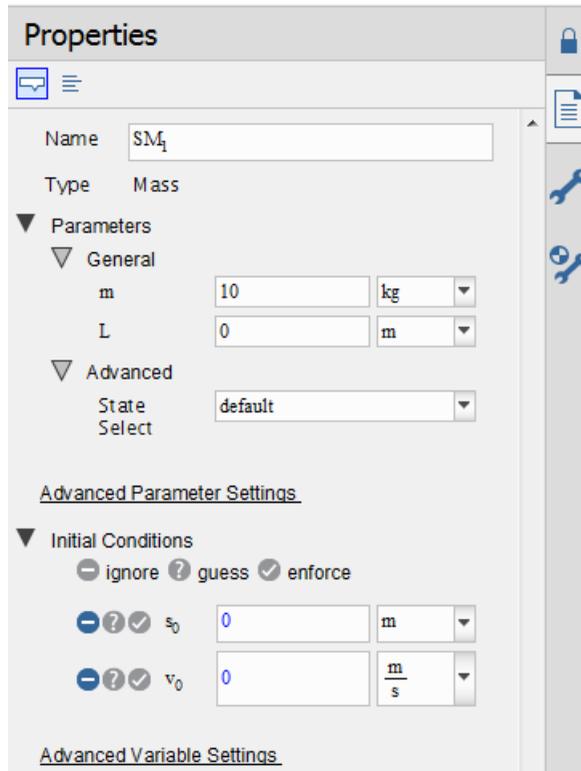


Figure 2.6: Initial Conditions

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 70).

For an example of how initial conditions are enforced, from the **Help** menu, select **Examples > User's Guide Examples > Chapter 2**, and then select 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.

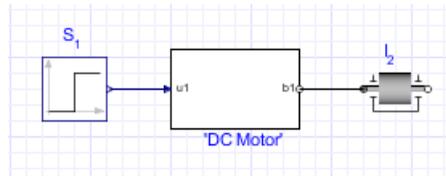
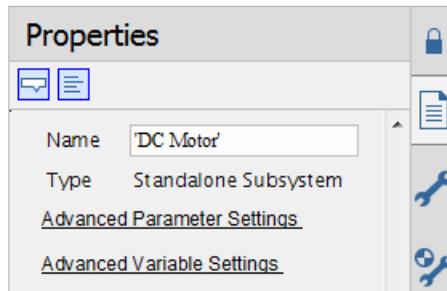


Figure 2.7: Subsystem Group

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.

After 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 tools in the **Properties** tab ()



For best practices on creating subsystems in MapleSim, see *Best Practices: Laying Out and Creating Subsystems* (page 64).

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. From the **Help** menu, select **Examples > User's Guide Examples > Chapter 2**, and then select the **Simple DC Motor** example.
2. Draw a box around the electrical components by dragging your mouse over them.

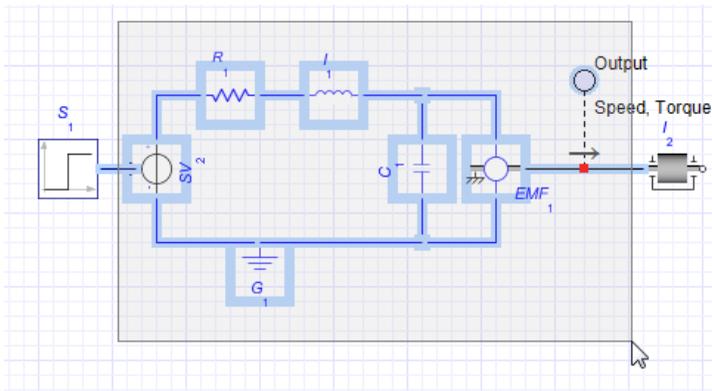
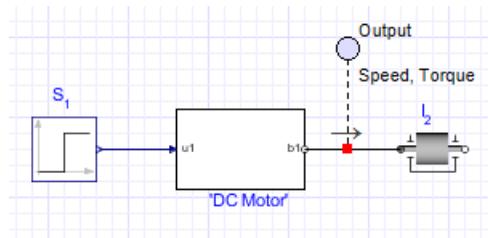


Figure 2.8: Creating a Subsystem

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**.

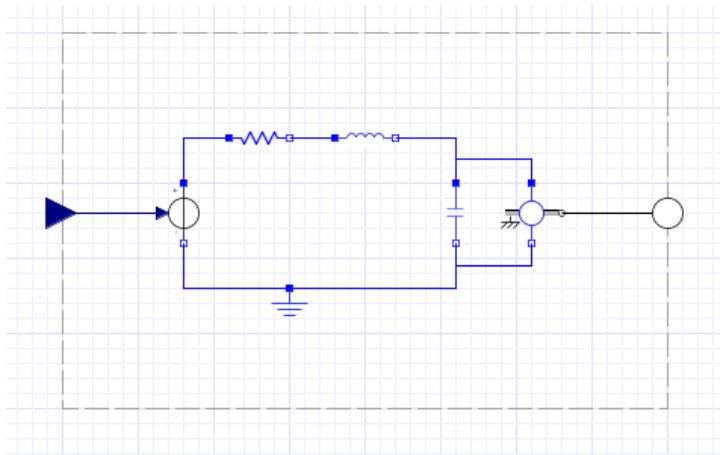
5. Click **OK**. A white block, which represents the DC motor, appears in the **Model Workspace**.



In this example, you created a *standalone 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 31).

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 Model Navigation controls in the **Model Workspace Toolbar**. For details on Model Navigation controls, see *Best Practices: Laying Out and Creating Subsystems* (page 64).



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 **Components** palette under the **Local Components** tab.
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 **Components** palette under the **Local Components** tab (🔍) 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 **Local Components** tab; the copies in the **Model Workspace** are called *shared subsystems* because they share and refer to the configuration specified in their corresponding subsystem definition.

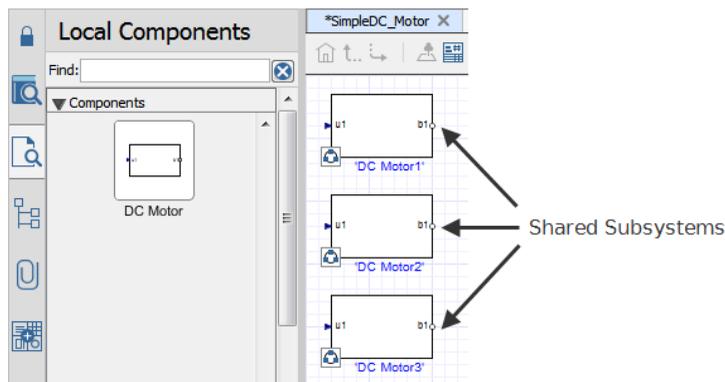


Figure 2.9: Creating Multiple Subsystems

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 **Local Components** tab. A shared subsystem is indicated on the model workspace by the icon .

Using the example shown above, if you change the resistance parameter of the **Resistor** component in the **DC Motor2** shared subsystem from 24Ω to 10Ω , the resistance value of the **Resistor** component in the **DC Motor1** and **DC Motor3** shared subsystems and the **DC Motor** subsystem definition in the **Local Components** tab will also be changed to 10Ω .

For more information, see *Editing Subsystem Definitions and Shared Subsystems (page 34)*.

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 Local Components Tab

To add a subsystem definition

1. In the **Model Workspace**, right-click (**Control**-click for Mac) the standalone DC motor subsystem that you created in *Example: Creating a Subsystem (page 29)*.
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 **Local Components** tab () on the left side of the **Model Workspace**, expand the **Components** palette.

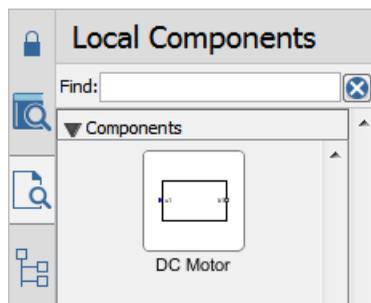


Figure 2.10: Subsystem Definition

The subsystem definition is added to the **Components** palette and the subsystem in the **Model Workspace** is converted into a shared subsystem called **DC Motor₁**. This shared subsystem is linked to the **DC Motor** subsystem definition.

5. Save this model as **DCMotorSubsystem.msimsim**. You will be building on this model in *Example: Editing Shared Subsystems that are Linked to the Same Subsystem Definition (page 34)*.

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 58)*.

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 **Local Components** tab and place it in the **Model Workspace**.

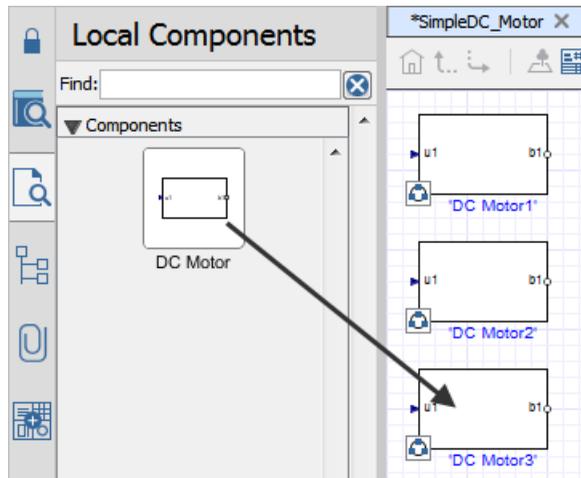


Figure 2.11: Adding Multiple Subsystems to a Model

When you create a new standalone 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.

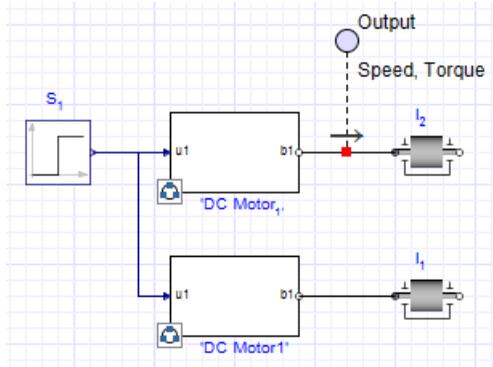
Example: Editing Shared Subsystems that are Linked to the Same Subsystem Definition

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 32)*. 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 32)*.

To use shared subsystems:

1. In MapleSim, open the **DCMotorSubsystem.msimsim** file that you created in *Example: Adding Subsystem Definitions and Shared Subsystems to a Model (page 32)*.
2. Under the **Local Components** tab () , expand the **Components** palette, 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 **Library Components** 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 Motor₁** shared subsystem. The detailed view of the shared subsystem appears.

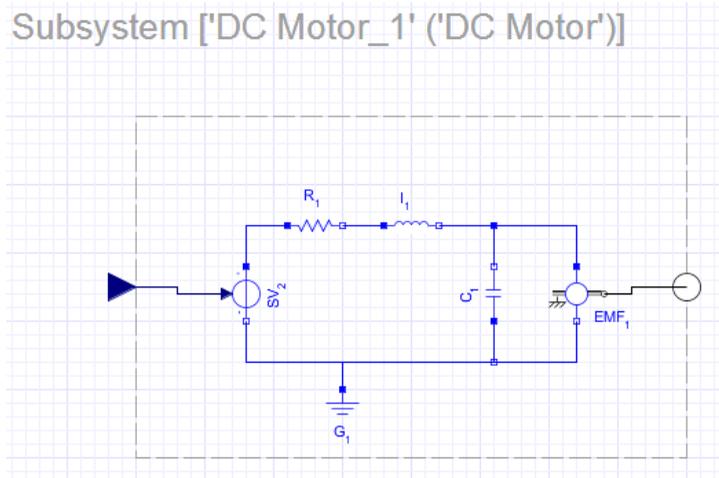
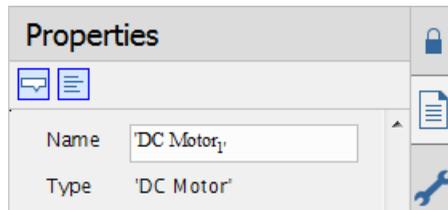
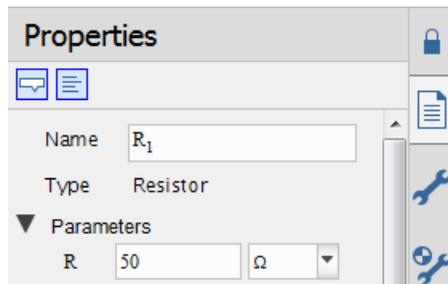


Figure 2.12: DC Motor Subsystem

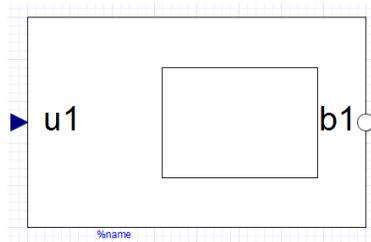
Note that a heading with the shared subsystem name (**DC Motor₁**) followed by the subsystem definition name (**DC Motor**) appears at the top of the **Model Workspace**. In the detailed view of all shared subsystems, this heading also appears to help you identify multiple subsystem copies in your model. Also, when you select a shared subsystem, its subsystem definition name appears in the **Type** field in the **Properties** tab (📄).



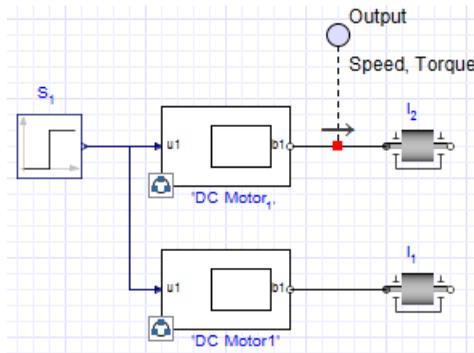
6. Select the **Resistor** component (R_1) and, in the **Properties** tab, click **Parameters**. Change the resistance value to **50 Ω** .



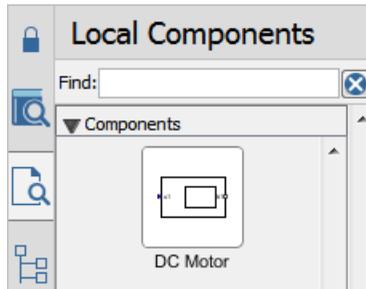
7. In the **Model Workspace Toolbar**, click **Icon** () .
8. Using the **Rectangle Tool** () in the **Model Workspace Toolbar**, click and drag your mouse pointer to draw a shape in the box.



9. In the **Model Workspace Toolbar**, click **Diagram** () .
10. Click **Main** () in the **Model Workspace Toolbar** to browse to the top level of the model. Both of the **DC Motor** shared subsystems now display the square that you drew.



- Under the **Local Components** tab on the left side of the MapleSim window, expand the **Components** palette. 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 .

- From the **Local Components** tab, 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 50Ω , and then delete it from the workspace.
- Save this model as **DCMotorSharedSubsystem.msims**. You will be building on this model in *Example: Removing the Link Between a Shared Subsystem and its Subsystem Definition* (page 37).

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 34) 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 34).
2. In the **Model Workspace**, right-click (**Control**-click for Mac) the **DC Motor1** shared subsystem.
3. Select **Convert to Standalone Subsystem**. The **DC Motor1** subsystem is no longer linked to the **DC Motor** subsystem definition in the **Local Components** tab; it is now called **copy of DC Motor**.
4. Double-click the **DC Motor₁** 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 Motor₁** shared subsystem in the **Model Workspace** and the **DC Motor** subsystem definition in the **Local Components** tab. Note that your change is not shown in the **copy of DC Motor** subsystem that is no longer linked to the **DC Motor** subsystem definition.

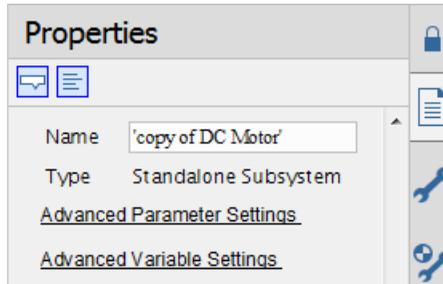
Tip: When you convert a shared subsystem to a standalone subsystem, it is a good practice to assign the standalone subsystem a meaningful name that clearly distinguishes it from existing shared subsystems and subsystem definitions.

Working with Standalone Subsystems

Standalone subsystems are subsystems that are not linked to a subsystem definition. You can create a standalone subsystem in two ways: by creating a new subsystem as shown in *Example: Creating a Subsystem* (page 29) or by converting a shared subsystem to a standalone subsystem as shown in *Example: Removing the Link Between a Shared Subsystem and its Subsystem Definition* (page 37). Standalone subsystems can be edited independently without affecting other subsystems in the **Model Workspace**.

To identify a subsystem as a standalone subsystem, select a subsystem in the **Model**

Workspace and examine the **Properties** tab () . If that subsystem is a standalone subsystem, the **Type** field reads **Standalone Subsystem**.



Standalone subsystems will not show the shared subsystem icon () on the **Model Workspace**. Also, if you double-click a standalone subsystem to browse to its detailed view, no heading is shown for the subsystem in the **Model Workspace**.

When you copy and paste a standalone 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 Standalone Subsystem* (page 40).

Example: Resolving Warning Messages in the Debugging Console

When you convert a shared subsystem into a standalone 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 37).

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
 The stand-alone subsystem 'copy of DC Motor' is identical to the shared subsystem 'DC Motor'.	Main

2. To work with the **copy of DC Motor** subsystem as a standalone subsystem, right-click (**Control**-click for Mac) the warning message and select **Ignore duplication warnings for 'copy for DC Motor'** 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** () above 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** standalone subsystem to the **DC Motor** subsystem definition again, you can right-click (**Control**-click for Mac) the warning message and select **Update 'copy of DC Motor' to use the shared subsystem 'DC Motor'**.

Example: Copying and Pasting a Standalone Subsystem

Note: This example is an extension of *Example: Removing the Link Between a Shared Subsystem and its Subsystem Definition* (page 37).

To copy and paste a standalone subsystem:

1. In the **Model Workspace**, copy and paste the **copy of DC Motor** standalone subsystem. A dialog box appears. (See **Figure 2.13**.)
2. Select **Convert the above stand-alone subsystem to a shared subsystem (Recommended)**. A new subsystem definition called **SharedSubsystem_1** is added to the **Components** palette in the **Local Components** tab ()

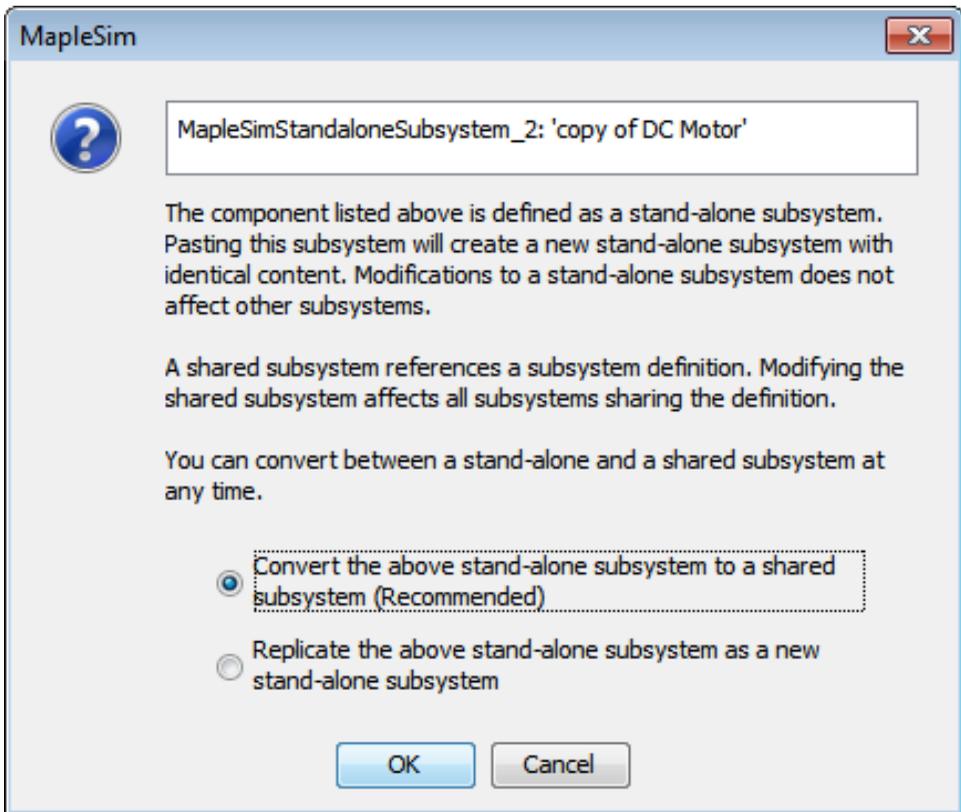


Figure 2.13: Copy Subsystem Dialog

In the **Model Workspace**, the **copy of DC Motor** standalone subsystem has been converted to a shared subsystem called **copy of DC Motor** and another copy of that shared subsystem called **copy of DC Motor₁** has been added to the **Model Workspace**. Both the **copy of DC Motor** and **copy of DC Motor₁** shared subsystems are linked to the new **SharedSubsystem_1** subsystem definition. Therefore, if you edit either **copy of DC Motor** or **copy of DC Motor₁** 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 the above stand-alone subsystem as a new stand-alone subsystem** to add another standalone 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 **Properties** 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 147) 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 **Library Components** 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 **Model Workspace Toolbar**, click **Parameters** () , or click the workspace and from the **Properties** 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
<input type="text"/>				

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. Under **Type**, 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. Click **Diagram** () , then select the **R₁** component. Enter **GlobalResistance** as the resistance value.

Properties

Name

Type Resistor

▼ Parameters

R Ω

T_{ref} °C

α $\frac{1}{K}$

Use Heat Port

T °C

9. Repeat this step for the **R₂** component.

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 **R₁** and **R₂** components.

The **R₁** and **R₂** 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 **R₁** and **R₂** components will also be changed

to **5**. Any change to the **GlobalResistance** parameter value will not apply to the **R₃** 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 **Properties** tab (). Parameters can only be assigned to components in the subsystem in which they are defined. If you select 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 158)* 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

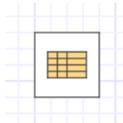
1. From the **Help** menu, select **Examples > Physical Domains > Multibody**, and then select the **Double Pendulum** model. This model contains two shared subsystems, **L₁** and **L₂**, which are linked to a subsystem definition called **L**.
2. Double-click the **L₁** shared subsystem.
3. Click **Parameters** ()
4. In the **L subsystem default settings** table, click the empty field at the bottom of the table.
5. Enter **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 **Properties** tab () for the **L₁** shared subsystem.
7. Click **Main** () , select the **L₂** subsystem, and then examine the **Properties** tab. The new subsystem parameter is also displayed for the **L₂** shared subsystem.
8. In the **Properties** tab, change the value of **c** to **50**.
9. Click the **L₁** shared subsystem in the **Model Workspace** and examine the **Properties** 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 default settings

Name	Type	Default Value	Default Units	Description
<input type="text"/>				

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 58).

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. From the **Help** menu, select **Examples > Physical Domains > 1-D Mechanical**, and then select the **PreLoad** example.
2. Under the **Settings** tab () , enter **0.012** seconds for t_d , the simulation duration time.
3. Click the **SM₁** Mass component on the workspace, and then click the **Properties** 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 **Properties** 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 default settings

Name	Type	Default Value	Default Units	Description
<input type="text"/>				

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.

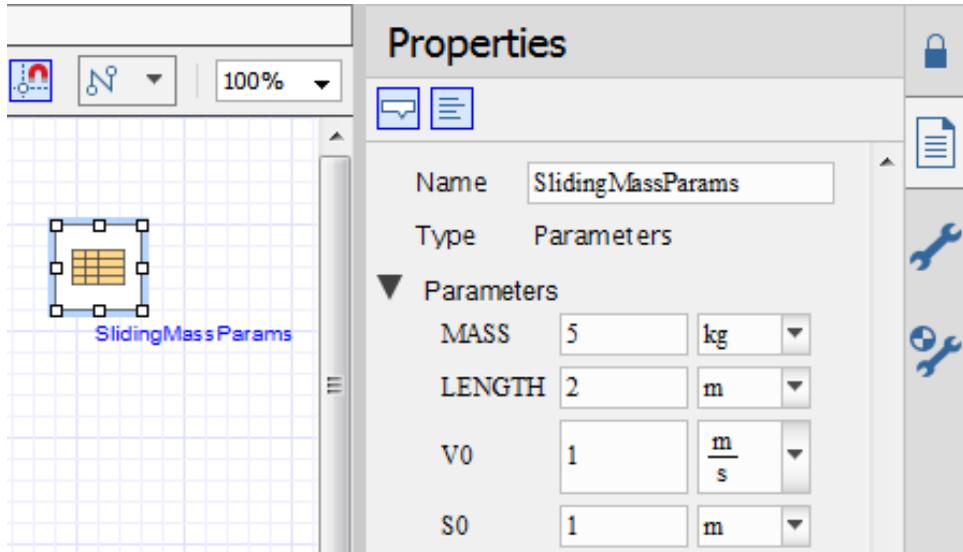
Name	Type	Default Value	Default Units	Description
LENGTH	Length [[m]]	2	m	Length of the sliding mass
V0	Velocity [[$\frac{m}{s}$]]	1	$\frac{m}{s}$	Initial velocity of the sliding mass
S0	Position [[m]]	1	m	Initial position of the sliding mass

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

Parameters 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 [[$\frac{m}{s}$]]	1	$\frac{m}{s}$	Initial velocity of the sliding mass
S0	Position [[m]]	1	m	Initial position of the sliding mass

14. Click **Diagram** () . When you select the parameter block in the **Model Workspace**, the defined parameters appear in the **Properties** tab on the right side of the MapleSim window.

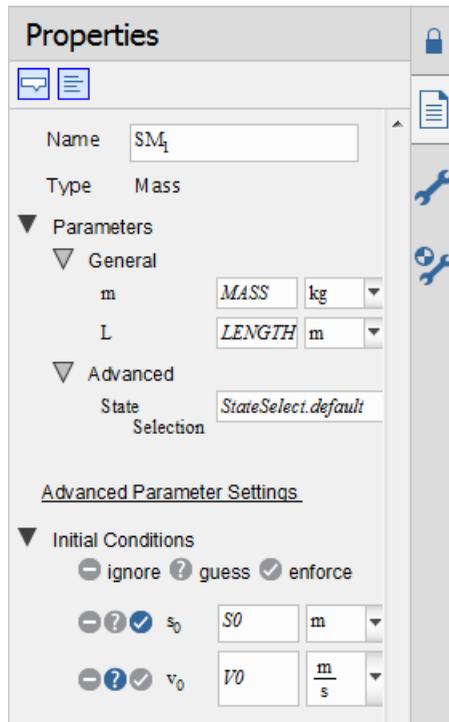


The screenshot shows the MapleSim interface. On the left, a diagram in the Model Workspace shows a parameter block labeled "SlidingMassParams" selected. On the right, the Properties tab is open, displaying the following information:

- Name: SlidingMassParams
- Type: Parameters
- Parameters:
 - MASS: 5 kg
 - LENGTH: 2 m
 - V0: 1 $\frac{m}{s}$
 - S0: 1 m

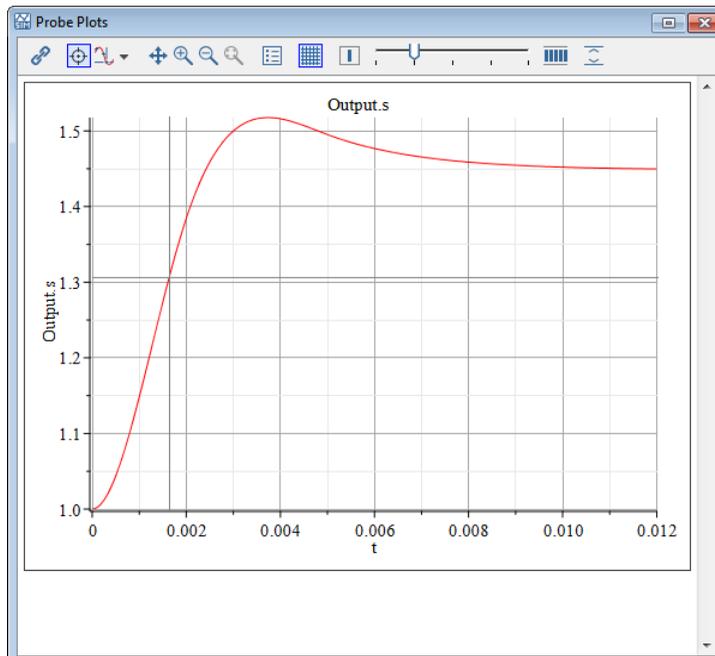
15. In the **Model Workspace**, select the **SM₁** mass component in the diagram.

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

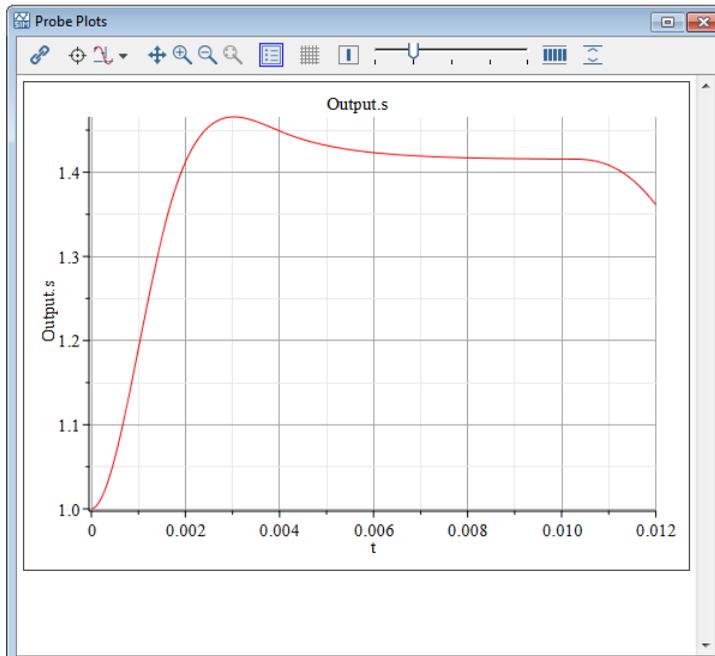


The parameters of this **Mass** component now inherit the numeric values that you defined in the parameter block.

17. In the same way, assign the same values to the parameters of the **SM₂** and **SM₃** mass components in the model.
18. In the **Model Workspace**, delete the probe labeled **Input**.
19. Select the probe labeled **Output**.
20. In the **Properties** tab, clear the check box beside **Velocity**.
21. To simulate the model, click **Run Simulation** (▶) in the **Main Toolbar**.
22. Click **Show Simulation Results** (📈). The following graph appears in the Analysis window.



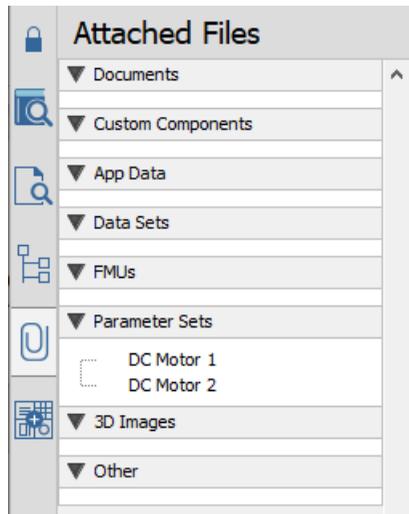
23. In the **Model Workspace**, click the parameter block.
24. In the **Properties** tab, change the mass to **3.5** and the initial velocity to **5**. Press **Enter**.
These changes apply to all of the **Mass** components to which you assigned the symbolic parameter values.
25. Simulate the model again, then bring the Analysis window to the front. 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 can 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 **Attached Files** 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 Mac) on a Parameter Set. For more information, see the **Using MapleSim > Building a Model > Using Parameter Sets > Saving and Applying Parameter Sets** section in the MapleSim Help system.

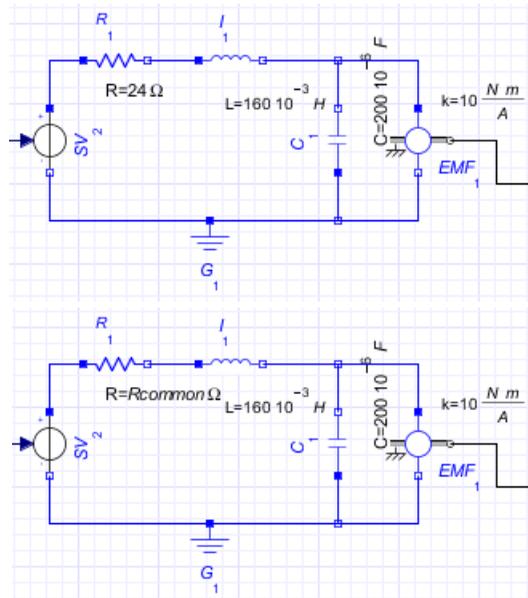
Using Advanced Parameter and Variable Settings

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** tools in the **Properties** tab (). Advanced Settings lets you override one or more default values.

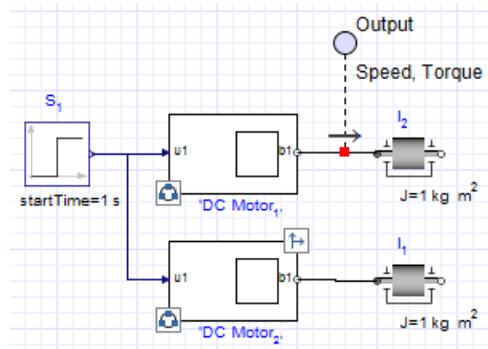
Advanced Parameter Settings

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**.



In the **Model Workspace**, components with a parameter override are identified with an override icon (⏏). In the following model, the **DC Motor₂** subsystem has a parameter override.



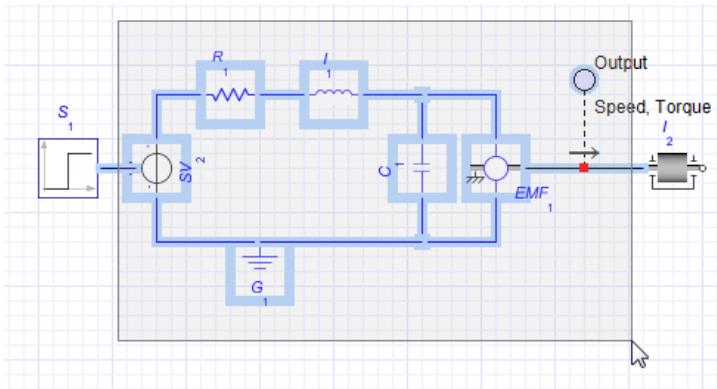
Advanced Variable Settings

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.

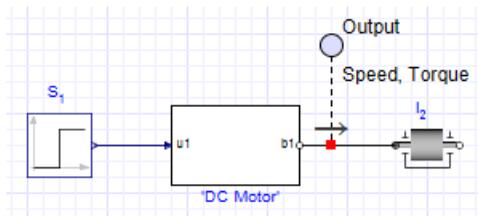
Example: Creating a Parameter Override

To create a parameter override:

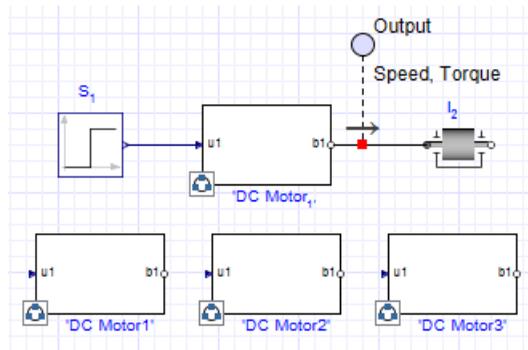
1. From the **Help** menu, select **Examples > User's Guide Examples > Chapter 2**, and then select the **Simple DC Motor** example.
2. Draw a box around the electrical components by dragging your mouse over them.



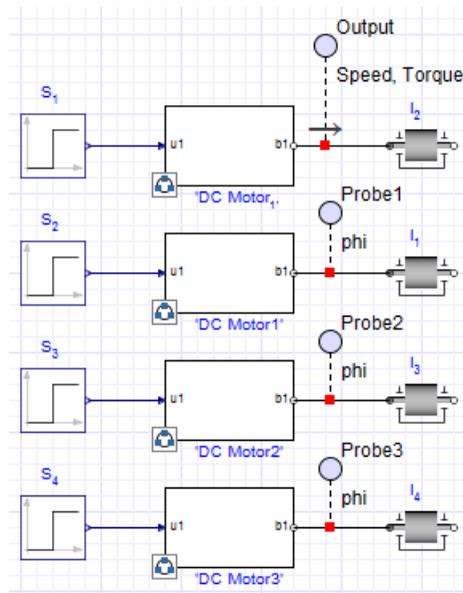
3. From the **Edit** menu, select **Create Subsystem** or right-click (**Control-click** for Mac) the boxed area and select **Create Subsystem**.
4. In the dialog box, enter **DC Motor**, and then click **OK**. The DC Motor subsystem appears.



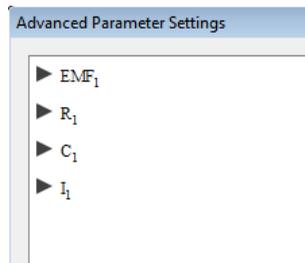
5. Right-click (**Control-click** for Mac) the **DC Motor** subsystem, select **Convert to Shared Subsystem**, and then click **OK**. This creates the shared subsystem definition and adds it **Components** palette under the **Local Components** tab.
6. Under the **Local Components** tab () , expand the **Components** palette, and then drag three copies of the DC Motor shared subsystem to your **Model Workspace**.



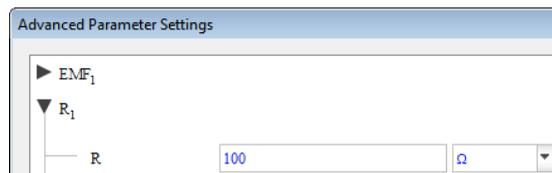
7. Create three additional DC Motor subsystems as shown below.



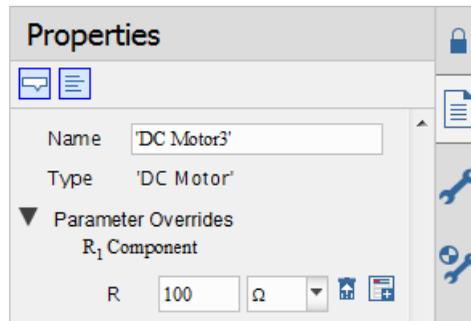
8. Click S_2 , and, under the **Properties** tab (☰), set T_0 to **1**. Do the same for S_3 and S_4 .
9. Click the **DC Motor3** subsystem, and then click **Advanced Parameter Settings** under the **Properties** tab (☰). The **Advanced Parameter Settings** window appears, showing all of the subsystem components.



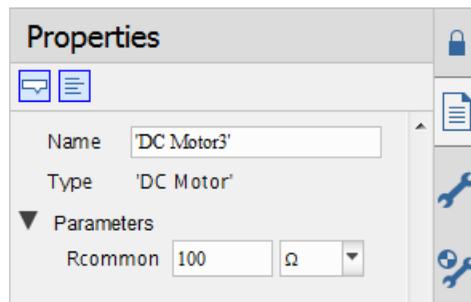
10. Expand R_1 and enter a value of **100** for the Resistance parameter (**R**).



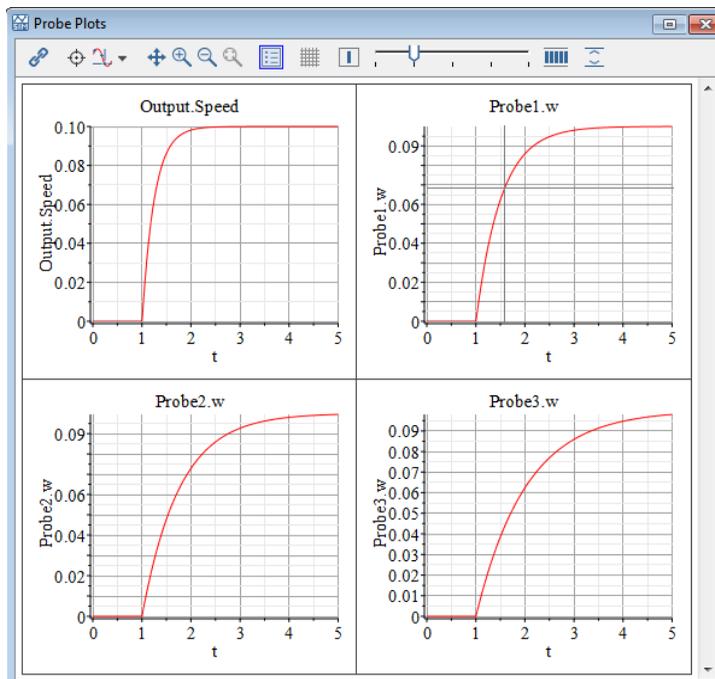
11. Click **OK**. The new parameter appears in the **Properties** tab as an override.



12. 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 **Properties** tab as a parameter that can now be reused in the other subsystems. Note that it is no longer an override.



13. For each of the other subsystems, click the subsystem and in the **Properties** tab enter the following values for **Rcommon**: of
- For **DC Motor₁**, set **Rcommon** to 25Ω
 - For **DC Motor₁**, set **Rcommon** to 50Ω
 - For **DC Motor₂**, set **Rcommon** to 75Ω
14. For each of the subsystems, select the probe, and in the **Properties** tab select the **Speed** check box and clear all of the other check boxes.
15. 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 **Properties** 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 **Attached Files** 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 **Attached Files** tab 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 Mac) the category in which you want to save the attachment and select **Attach File**.

You can also attach a file to a model from the menu bar by selecting **Edit > 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 **Attached Files** tab that contains files called *CustomComponent.mw*, *NonLinearMSD.mw*, and *DamperCurve.xlsx*.

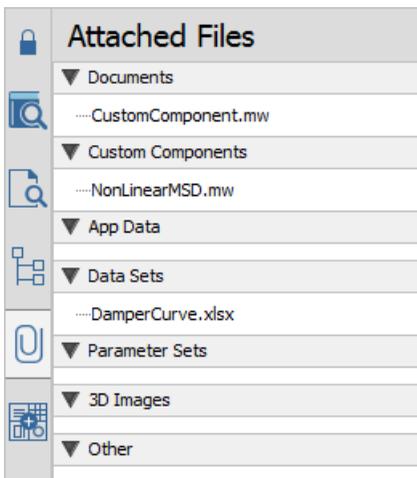


Figure 2.14: Attachments

You can also use the **Attached Files** tab to open MapleSim templates to create custom modeling components and ports for a model. For more information, see *Analyzing and Manipulating a Model* (page 127) in this guide.

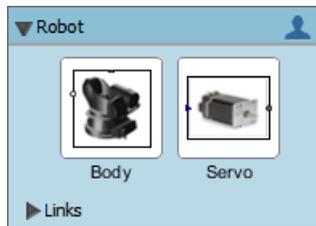
2.8 Creating and Managing Custom Libraries

You can create a custom library to save a collection of subsystems and custom modeling components that you plan to reuse in multiple files or MapleSim sessions. Custom libraries that you create appear in custom palettes in the **Library Components** 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 use the subsystems and components from the custom library palette when building a model, in the same way you add components from other palettes.

You can also share a custom library with other users. For example, if you store a custom library on a network drive, other users with access to that location can load your custom library in their MapleSim session.

Custom library palettes appear in the **Library Components** tab (🔍) and are indicated with an icon: 👤. A sample custom palette is shown below.



For information on creating custom libraries, see **Using MapleSim > Building a Model > Custom Libraries > Creating a Custom Library** in the MapleSim Help system.

If you used a third-party tool to create models or model libraries based on the Modelica 3.2.2 programming language, you can import the .mo files for the models or model libraries into MapleSim as .msimlib files. 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 > Importing and Opening Modelica Models and Libraries > Importing Modelica Libraries**.

Example: Creating a Custom Library from an Existing Model

In this example, you will create a custom library from the shared subsystem definitions of an existing MapleSim model. The components added to the custom library will then be available in future MapleSim sessions.

To create a custom library from a model:

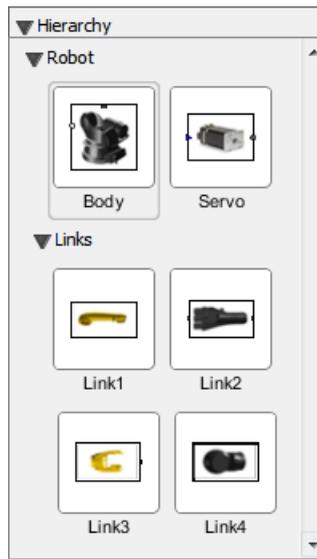
1. From the **Help** menu, select **Examples > Physical Domains > Multibody**, and then select the **5 DoF Robot** example.

This model has six shared subsystems, which are listed in the **Components** palette of the **Local Components** tab ()

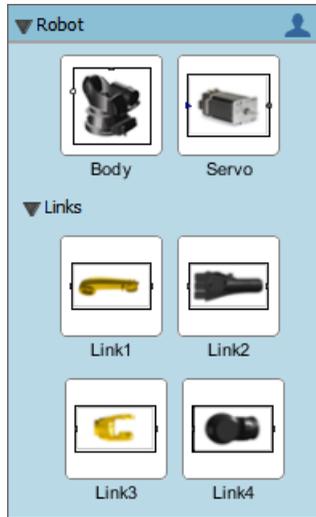
2. Save the model.
3. From the **Tools** menu, select **Export to MapleSim Library**.
4. Enter the name **Robot** for the library under **Package**.
Note: The package name that you specify will appear as the custom palette name in the MapleSim interface.
5. Click **OK**.
6. After the library has been exported, click **Close**.

You are now in *library edit mode* (indicated by the watermark on the workspace and the library properties in the **Properties** tab in the Parameters pane). A new custom library palette appears in the **Library Components** tab () on the left side of the MapleSim window. The palette is empty because we have not defined a hierarchy for its elements.

7. Switch to the **Local Components** tab () , and drag the components to the **Hierarchy** tab under the root name (Robot). If desired, organize the elements into subgroups.

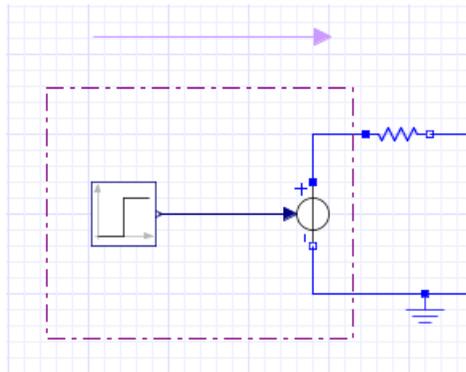


8. Click **Reload** (🔄) in the **Main Toolbar** to save your changes and reload the palette for the custom library in the **Library Components** tab with the updates. The custom library palette now contains all these components.



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 62).

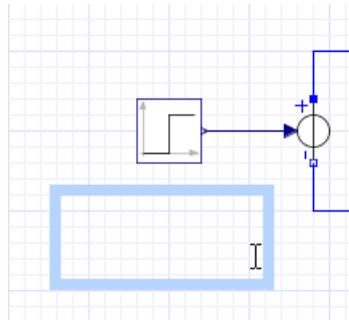
Example: Adding Text Annotation to a Model

To add text annotation to a model:

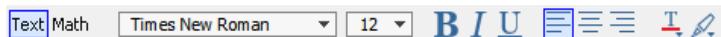
1. From the **Help** menu, select **Examples > User's Guide Examples > Chapter 2**, and then select the **Simple DC Motor** example.
2. From the **Annotations Toolbar**, click **Text Tool** ().

Note: If the **Annotation Toolbar** is not visible, click **Show/Hide Drawing Tools** () in the **Model Workspace Toolbar**.

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.

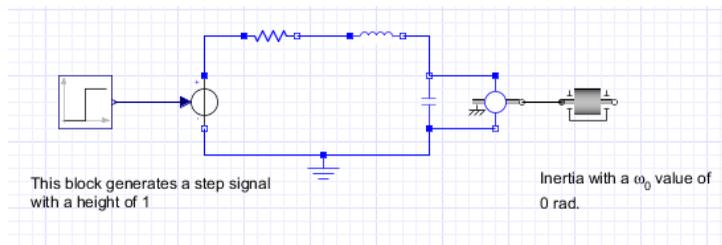


4. Enter the following text: **This block generates a step signal with a height of 1.**
5. Select the text that you entered and change the font to **Arial**.
6. Click anywhere outside of the text box.
7. Draw another text box below the **Inertia** component.
8. Enter the following text: **Inertia with a ω_0 value of 0 rad.**

Tip: To enter the omega character (ω), click **Math** on the context bar  to switch to the 2-D math mode, type **omega**, and then press **Esc**. To enter the subscript, press **Ctrl + Shift + the underscore key** (Windows and Linux) or **Command + Shift + the underscore key** (Mac) followed by **0**. Press the right arrow key to move the cursor from the subscript position. Toggle back to text entry mode by clicking **Text** on the context bar, and enter the remaining text.

9. Select the text that you entered and change the font to **Arial**.

10. Click anywhere outside of the text box to complete the annotation.



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.

To enter 2-D math notation, select **Math** (Math) in the text formatting toolbar.

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

Table 2.4: 2-D Math Notation Key Combinations

Task	Key Combination	Example
Command and symbol completion (parameter values and annotations only)	<ol style="list-style-type: none"> Enter the first few characters of a symbol name, Greek character, or Maple command. Enter the key combination for your platform: <ul style="list-style-type: none"> • Esc, Mac, Windows, and Linux • Ctrl + Shift + Space, Linux 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 fraction	forward slash (/)	$\frac{1}{8}$

For more information, see **Using MapleSim > Building a Model > Annotating a Model > Key Combinations for 2-D Math Notation** 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 App or Random Data App. These apps are found in the **Apps** palette under the **Add Apps or**

Templates tab (.

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

Example: Creating a Data Set in Maple

In this example, you will use the Data Generation App to create a data set for a MapleSim **1D Lookup Table** component. In this app, 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 **Library Components** tab () , expand the **Signal Blocks** palette, and then expand the **Interpolation Tables** menu.
3. Add a **Lookup Table 1 D** component to the **Model Workspace**.
4. Click **Add Apps or Templates** (.
5. In the **Apps** palette, double-click **Data Generation**. The Data Generation App opens in the Apps tab of the Analysis Window.
6. At the bottom of the app, in the **Data set name** field, enter **TestDataSet**.
7. To make the data set available in MapleSim, click **Attach Data in MapleSim**.
8. In MapleSim, under the **Attached Files** tab () , expand the **Data Sets** palette. The data set file appears in the list. You can now assign this data set to the interpolation table component in the **Model Workspace**.
9. In the **Model Workspace**, select the **Lookup Table 1 D** component.
10. In the **Properties** tab () , from the **datasource mode** list, select **attachment**.

11. From the **data** drop-down menu, select the **TestDataSet.csv** file. The data set is now assigned to the **Lookup Table 1 D** component.
12. 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 127) 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.

After you run the simulation, the debugging pane displays diagnostic messages that can help you troubleshoot potential errors as you build a model. When you click **Run diagnostic tests** (✓) above the debugging pane (or from the **Edit** 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 Mac) the message in the debugging pane to display options that can help you to resolve the issue.

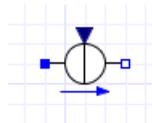
Best Practices: Building Electrical Models

Include a Ground Component in Electrical Circuits

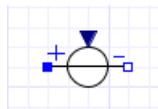
In each electrical circuit model, you must add and connect a **Ground** component to provide a reference for the voltage signals.

Verify the Connections of Current and Voltage Sources

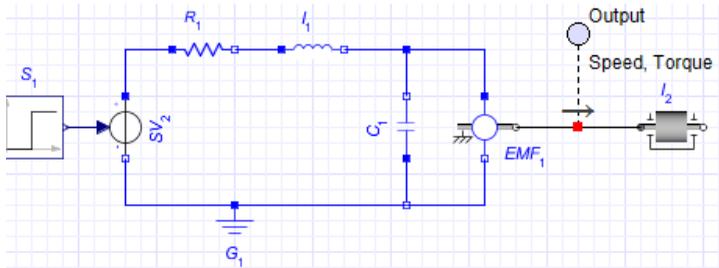
Simulation results can be affected by the way in which a current or voltage source is connected in your model. If you receive unexpected simulation results, verify the connections between electrical sources and other components in your model. All of the current sources in the MapleSim Component Library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



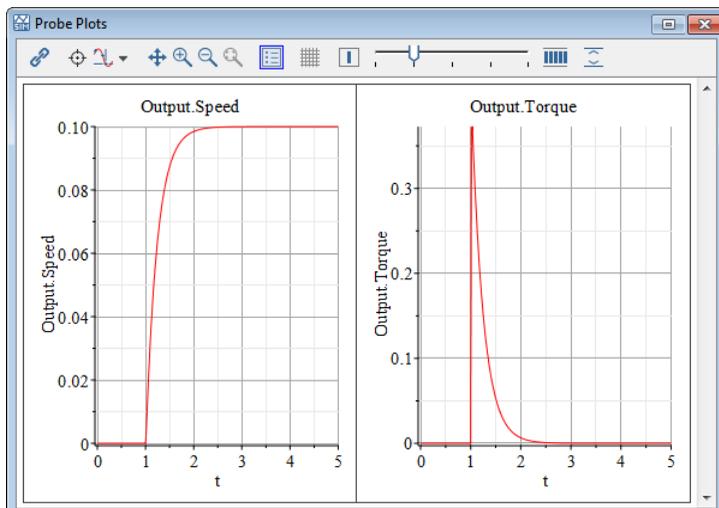
Also, all of the voltage sources display a plus sign indicating the location of the positive voltage and a minus sign indicating the location of the negative voltage.



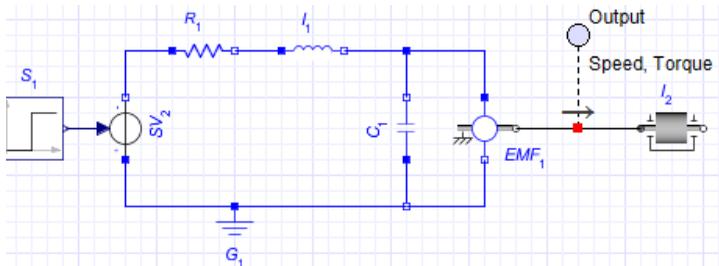
Consider the following **Simple DC Motor** model. Note that the positive port of the **Signal Voltage** source at the left of the diagram is connected to the positive port of the **Resistor** component.



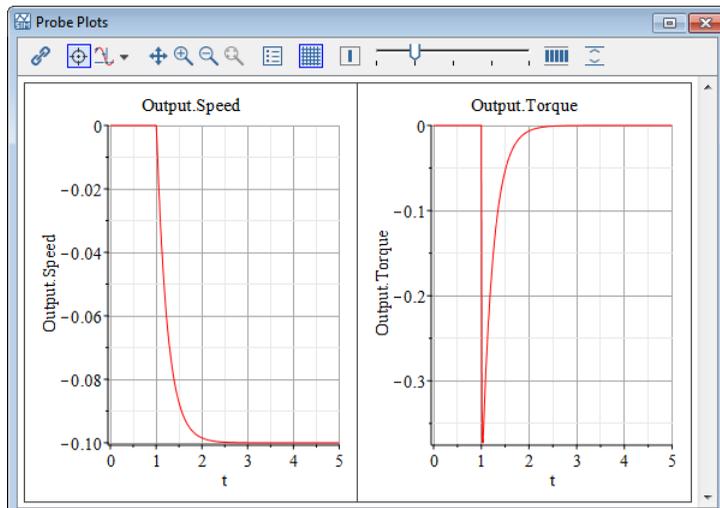
When this model is simulated, MapleSim returns the following results for the torque and speed quantities.



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 coordinate 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.

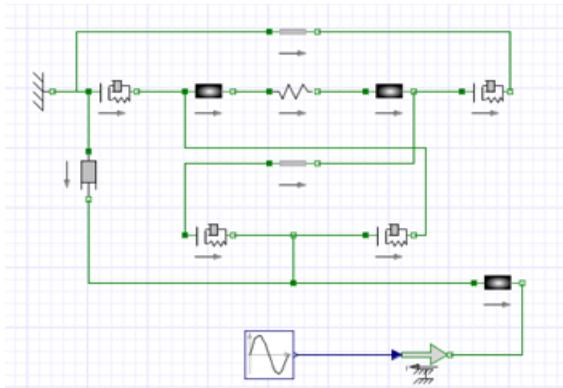


Figure 2.15: Verifying Force Arrows

For an example of sign convention and how arrow direction represents a force acting on the model, from the **Help** menu, select **Examples > User's Guide Examples > Chapter 2**, and then select one of the **Constant Acceleration**, **Sign Convention**, or **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  icon) are both connected to a **Rigid Body** component.

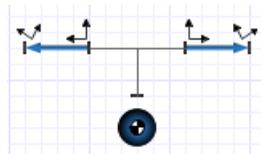


Figure 2.16: Center of Mass Placement Best Practice

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.

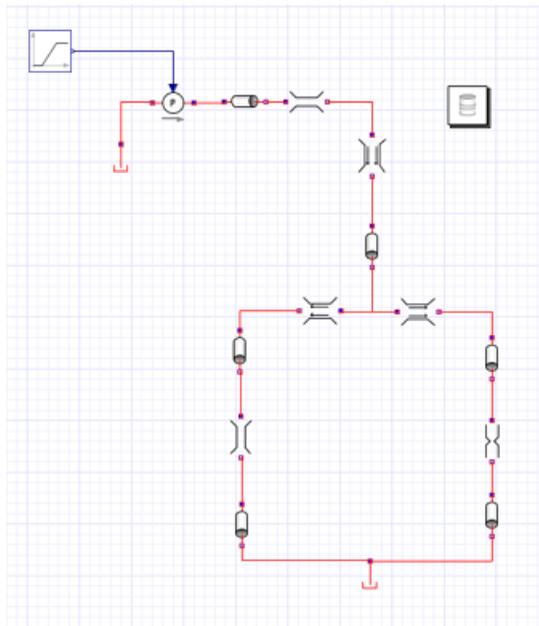


Figure 2.17: Hydraulic Model

For a complete tutorial on how to model hydraulic systems, see *Tutorial 8: Modeling Hydraulic Systems* (page 206).

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:

- *Understanding Custom Components (page 71)*
- *Creating Custom Components with Signal-Flow Behavior (page 74)*
- *Creating Custom Components with Physical Connections (page 80)*
- *Working with Custom Components in MapleSim (page 82)*
- *Example: Creating a Nonlinear Spring-Damper Custom Component (page 83)*

3.1 Understanding Custom Components

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.

For a complete tutorial on how to create domain specific custom components, see *Tutorial 5: Using the Custom Component Template (page 174)*.

There are several different Custom Component templates. These are found in the **Templates** palette, under the **Add Apps or Templates** tab ()

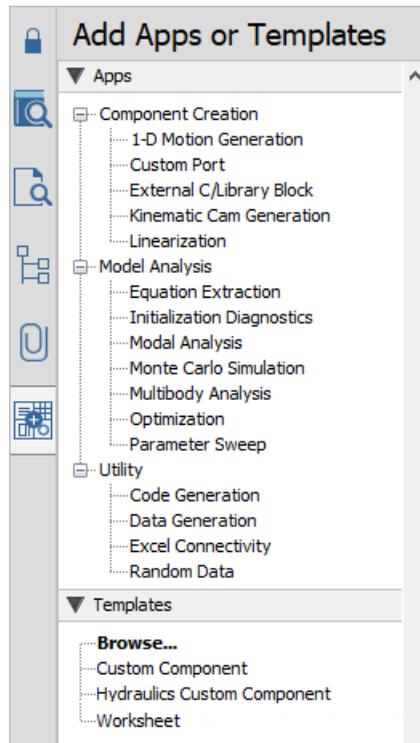


Figure 3.1: The Add Apps or Templates tab

Creating a Simple Custom Component

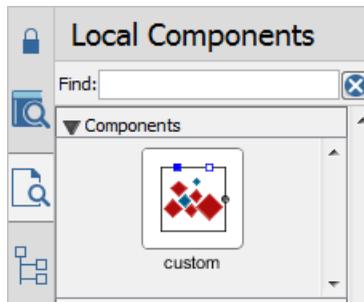
The general process of creating a custom component for a MapleSim model consists of specifying the component equations for the custom component, component parameters and system model, specifying the port types and their values, and generating the component.

To create a custom component:

1. Start a new MapleSim model and select the **Add Apps or Templates** tab (.
2. From the **Templates** palette, double-click on **Custom Component**.
3. Enter a name for the template in the **Create Attachment** window, and then click **Create Attachment** (). Maple opens with the **DAE Custom Component** template.
4. In the **Define Equations** area, enter the equations for your custom component. Equations, parameters, and initial conditions are all entered here. Press **Enter** at the end of the line.

$$\begin{aligned}
 eq &:= \left[s(t) = sa(t) - sb(t), 0 = Fa(t) + Fb(t), Fa(t) \right. \\
 &= \text{piecewise}\left(s(t) < 0, K \cdot s(t) + B \cdot \frac{d}{dt}s(t), 0\right), K = 1000, B \\
 &= 10 \left. \right]
 \end{aligned}$$

5. In the **Configuration** section, select **Parameters**, click **Refresh All**, and then assign default values and types to model parameters.
6. Select **Variables**, click **Refresh All**, and then assign initial values and types for model variables.
7. Select **Ports**, and then add ports to the custom components by clicking **Add Port**. You can also control the layout of ports and the icon to use for the custom component. It is possible to define custom ports. To do so, you must first define the custom port using the **Custom Port** app. Then you can use the custom port in this app. For a complete tutorial on how to use custom ports, see *Example: Custom Ports (page 185)* in Chapter 6.
8. Provide the details for the port type, style, name, and port signals.
9. In the **Component Generation** section, enter a name for the component. This will be the name given to the custom component in the **Components** palette, under the **Local Components** tab () in MapleSim.
10. Click **Generate MapleSim Component** to create your component and to return to the MapleSim environment. The custom component now appears in the **Local Components** tab in the **Components** palette.



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 **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 the 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.

Using The Custom Component Template

For details on using this template, refer to the help page [Using the MapleSim DAE Custom Component Template](#).

3.2 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 then click **Add Apps or Templates** ()
2. In the **Templates** palette, double-click on **Custom Component**.
3. Enter **custom** for the name of the attachment, and then click **Create Attachment** (✓).
4. In the **Define Equations** section, enter the following equation:

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

Note: The equation here is enclosed in square brackets because *eq* must be assigned a *list* of equations.

5. Press **Enter** to register the equation.

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

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

Figure 3.2: Equations Defining a Custom Component

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:

$$eq := [x(t) + \log(x(t)^2) = y(t) + z(t)];$$

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

6. In the **Configuration** section, select **Ports**, and then click **Refresh All** to update the tables.
7. Click **Clear All Ports**.
8. Add three new ports by clicking **Add Port** three times and then drag them into following positions.

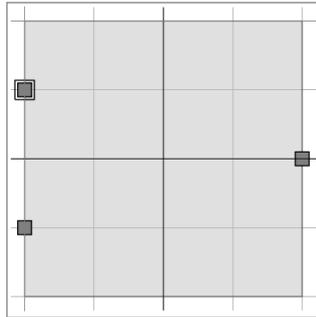


Figure 3.3: Port Mappings

9. Click on the top port on the left-hand side to select it.
10. From the **Type** drop-down list, select **Real Signal**. For the **Style**, select the **in** radio button. The port is given the default name **real_i**.
11. Next, associate a variable with the port signal. Select **y(t)** from the drop-down list as shown in **Figure 3.4**.

Type:

Dimension: Style: in out

Name:

Signal:

- Choose...
- x(t)
- y(t)
- z(t)

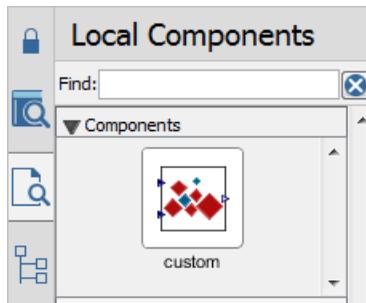
Figure 3.4: Variable to Port Mapping

12. Assign the remaining port mappings using the settings in **Table 3.1**.

Table 3.1: Port Map

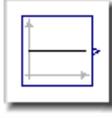
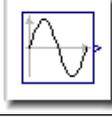
Port Name	Port Location	Port Type	Port Style	Port Component
real_i	Top left	Real Signal	in	$y(t)$
real1_i	Bottom left	Real Signal	in	$z(t)$
real_o	Right	Real Signal	out	$x(t)$

13. From the **Icon** list, select **Use default**.
14. Under the **Component Generation** section, enter **custom** in the **Name** field.
15. Click **Generate MapleSim Component**. The custom component equations are generated and assigned to the model. The custom component icon appears in MapleSim under the **Local Components** tab, in the **Components** palette.

**Figure 3.5: Generated Custom Component**

16. Drag the custom component into the workspace from the **Components** palette as shown in **Table 3.2**.

Table 3.2: Signal Flow Components

Component	Number of Components	Symbol	Library Location	Required Settings
Custom Component	1		Local Components () > Components	Custom settings
Constant	1		Library Components () > Signal Blocks > Sources > Real	Use default settings
Sine	1		Library Components () > Signal Blocks > Sources > Real	Use default settings

17. 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 **Attach Probe**), as shown below.

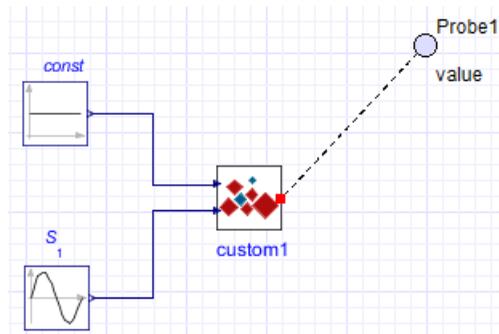
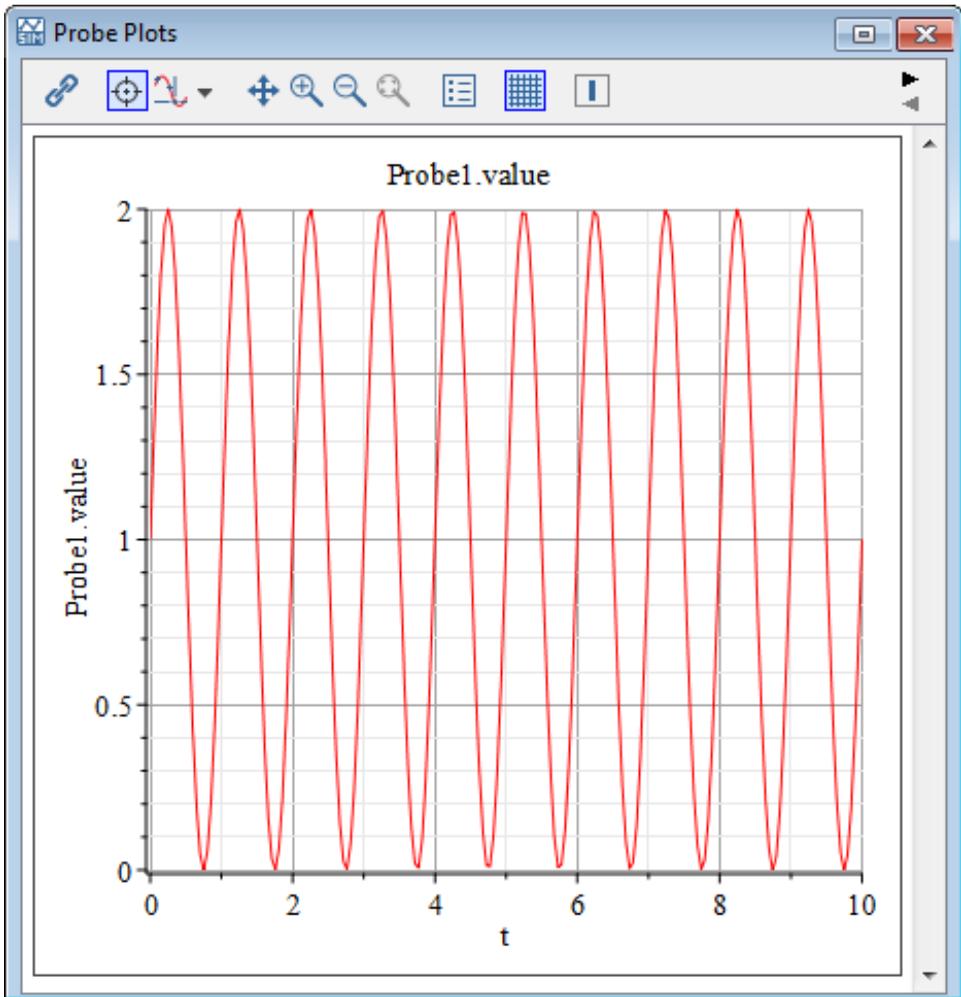


Figure 3.6: Completed Custom Component Model

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



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 your model, you can also use differential equations to define your custom component. For example, **Figure 3.7** shows the equations that describe the motion of two coupled mass-spring-dampers subjected to a driving force.

$$\begin{aligned}
 \text{eq} := & \left[m1 \cdot \frac{d^2}{dt^2} x1(t) = -K1 x1(t) - B1 \frac{d}{dt} x1(t) - K2 (x1(t) - x2(t)) - B2 \left(\frac{d}{dt} x1(t) \right. \right. \\
 & \left. \left. - \frac{d}{dt} x2(t) \right), m2 \frac{d^2}{dt^2} x2(t) = -K2 (x2(t) - x1(t)) - B2 \left(\frac{d}{dt} x2(t) - \frac{d}{dt} x1(t) \right) + F(t) \right]; \\
 \text{params} := & [K1 = 1, B1 = 1, K2 = 1, B2 = 1, m1 = 1, m2 = 1]; \\
 \text{initialconditions} := & [x1(0) = 0, x2(0) = 0, D(x1)(0) = 0, D(x2)(0) = 0];
 \end{aligned}$$

Figure 3.7: Double Mass-Spring-Damper Equations

Figure 3.8 shows how the parameters are mapped to component ports:

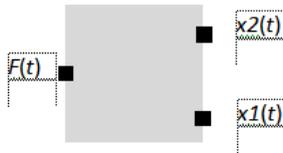


Figure 3.8: Port Mapping for Double Mass-Spring-Damper

3.3 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 across variable represents the driving force in a system (temperature difference, pressure difference, voltage drop, velocity or relative angular velocity), while the through variable represents a flow of a conserved quantity (such as heat, mass, current, force or torque).

Table 3.3: Characteristics of Through and Across Variables

Characteristics of Through Variables	Characteristics of Across Variables
Conserved quantity (like heat or mass)	Drives the flow of the conserved quantity
Has a direction of flow	Is a scalar
Satisfies the relationship input = output + accumulation	Defined as the difference between two points within a physical domain
Uniform across a 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 3.9 shows the equations mapped to the custom component ports.



Figure 3.9: 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 **Properties** tab (📄).

3.4 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:

- Save a Custom Component as Part of the Current Model
- Add a Custom Component to a Custom Library
- Edit a Custom Component
- Opening Custom Component Examples

Save a Custom Component as Part of the Current Model

When you save a model, the custom component is saved as part of the model. If you click the **Edit** menu > **Prune Model...** then any unused custom component definition will be removed from the model. To protect the custom component from potentially being cleaned up by the **Prune Model** feature, do one of the following:

- Move the custom component from the **Components** palette of the **Local Components** tab to the **Hierarchy** palette.
- Use the custom component in your model: drag the custom component from the **Components** palette to the Model Workspace.

For more information, see **Using MapleSim > Building a Model > Pruning a Model** in the MapleSim Help System.

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 58).

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 where you added the custom component.

3.5 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.

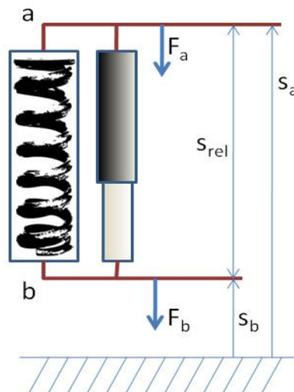


Figure 3.10: Nonlinear Spring-Damper Custom Component

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 is:

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

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 from the **Main Toolbar**.

To open the custom component template:

1. In MapleSim, open the model to which you want to add the custom component.
2. Select the **Add Apps or Templates** tab ()
3. In the **Templates** palette, double-click on **Custom Component**.
4. 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 **Equations** section, define the nonlinear system by entering 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];
```

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.

2. With your cursor on the equation, press **Enter**.
3. In the **Configuration** section, select **Variables**, and then click **Refresh All** to see an updated list of variables.

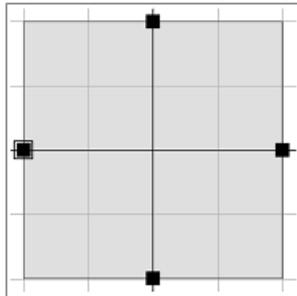
You can now assign these input and output variables to ports that you will include in your generated custom component.

Defining Component Ports

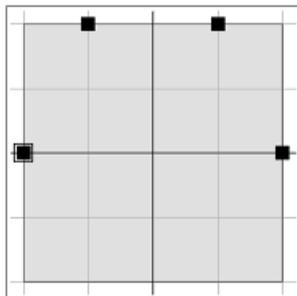
In the **Ports** section of the template, you assign input and output variables to ports that will appear in the generated component, and specify the layout of these ports.

To define the ports:

1. In the **Configuration** section, select **Ports**.
2. Click **Clear All Ports**.
3. Click **Add Port** four times. Four squares appear on the diagram. These represent the ports that you will lay out and define.



4. Drag the ports to position them with one on each side and two on the top of the diagram.



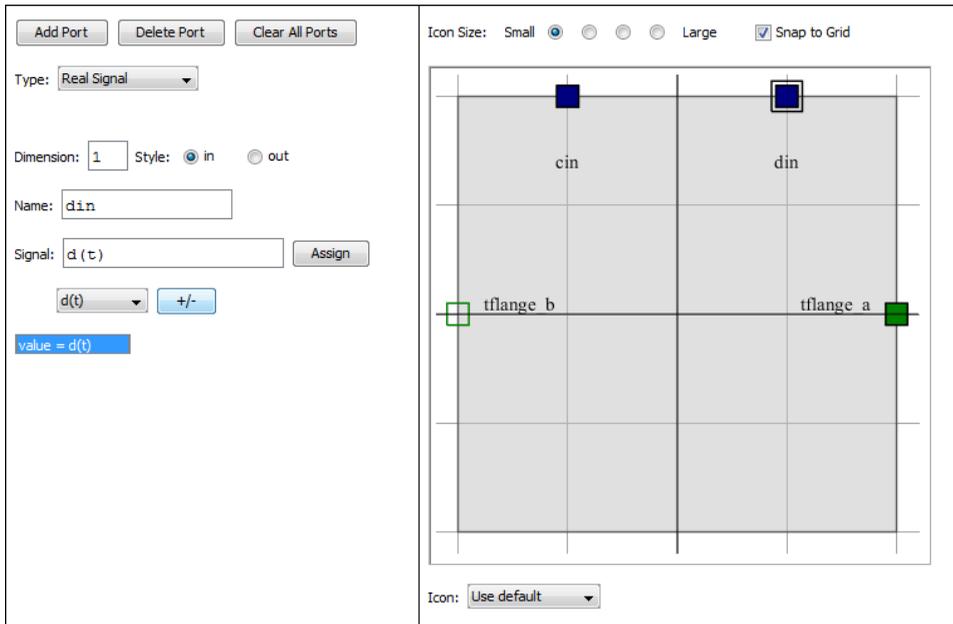
5. Select the port on the left side of the diagram.
6. From the **Type** drop-down menu, select **Translational**.
7. For the style of the port, select **b**. The port's default name is **tflange_b**.
8. First, define the signal *position*. Ensure **Position = unassigned** is selected in the list under **Signal**. From the drop-down list under **Signal**, select **s_b(t)**.
9. Next, define the signal *force*. Select **Force = unassigned** in the list under **Signal**. From the drop-down list under **Signal**, select **F_b(t)**. 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)**.
10. Select the port on the right side of the diagram.
11. From the **Type** drop-down menu, select **Translational**.
12. For the style of the port, select **a**. The port's default name is **tflange_a**.
13. First, define the signal *position*. Ensure **Position = unassigned** is selected in the list under **Signal**. From the drop-down list under **Signal**, select **s_a(t)**.
14. Next, define the signal *force*. Select **Force = unassigned** in the list under **Signal**. From the drop-down list under **Signal**, select **F_a(t)**. 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)**.
15. Select the port at the top left of the diagram and then do the following.
 - From the **Type** drop-down menu, select **Real Signal**.
 - For **Style**, select **in**.
 - Change **Name** to **cin**.
 - Under **Signal**, select **c(t)** from the drop-down list.

This port is now defined as a signal input and associated with the stiffness variable **c(t)**.

16. Select the port at the top right of the diagram and then do the following:
 - From the **Type** drop-down menu, select **Real Signal**.
 - For **Style**, select **in**.
 - Change **Name** to **din**.
 - Under **Signal**, 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)**.

17. From the **Icon** list, select **Use default**.



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

Checking Dimensions

This step is optional.

1. In the **Configuration** section, select **Dimensional Analysis**, and then click **Check Dimensions**. Algebraic expressions in the system equations that are dimensionally inconsistent are displayed in the math-container.
2. To correct the dimensions, select **Variables**, and then assign the following dimensions for the system variables in the **Type** column:
 - For $c(t)$, enter **Force/Distance**.
 - For $d(t)$, enter **Force/Velocity**.
 - For $F(t)$, enter **Force**.
 - For $s_{rel}(t)$, enter **Length**.
 - For $v_{rel}(t)$, enter **Velocity**.
3. Click **Refresh All**. The algebraic expressions **Force/Distance** and **Force/Velocity** are converted to the corresponding Modelica types, **TranslationalSpringConstant** and **TranslationalDampingConstant**, respectively (you could have entered these directly).

4. Select **Dimensional Analysis**, and then click **Check Dimensions**. The result should be the two equations shown below.

$$cin(t) = c(t) \frac{\text{kg}}{\text{s}^2}$$

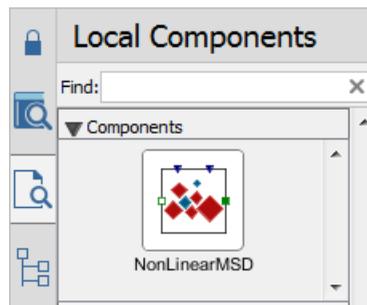
$$din(t) = d(t) \frac{\text{kg}}{\text{s}}$$

This is a benign inconsistency indicating that the real input signals $cin(t)$ and $din(t)$ actually have implied units.

Generating the Custom Component

Generate the custom component to add it to the component to the **Components** palette in the **Local Components** tab.

1. In the **Component Generation** section, enter **NonLinearMSD** in the **Name** text box.
2. Click **Generate MapleSim Component**. The generated custom component appears in MapleSim in the **Components** palette of the **Local Components** tab (.



You can add the custom component to a model by dragging it into the **Model Workspace** as you would any other component.

3. Save your MapleSim model as **NonlinearSpringDamper.msim**. *Tutorial 3: Modeling a Nonlinear Damper (page 158)* in Chapter 6 uses this custom component in a model.

4 Simulating and Visualizing a Model

In this chapter:

- *How MapleSim Simulates a Model (page 89)*
- *Simulating a Model (page 91)*
- *Simulation Progress Messages (page 98)*
- *Managing Simulation Results and Snapshots (page 99)*
- *Customizing Plot Window Configurations (page 100)*
- *Visualizing a Multibody Model (page 106)*
- *Best Practices: Simulating and Visualizing a Model (page 126)*

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 **Properties** tab on the right side of the MapleSim window. If the specified initial conditions are inconsistent, an error will be detected during the simulation. To help solve this issue, use the Initialization Diagnostics App.

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:

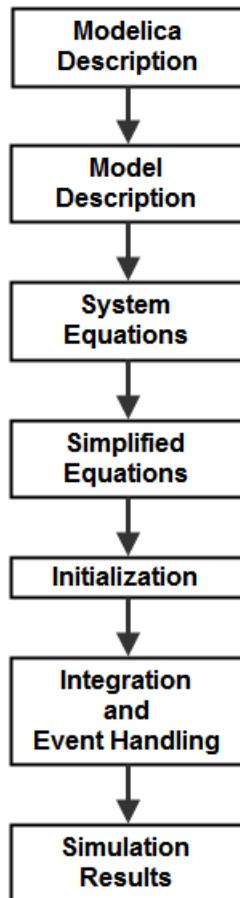


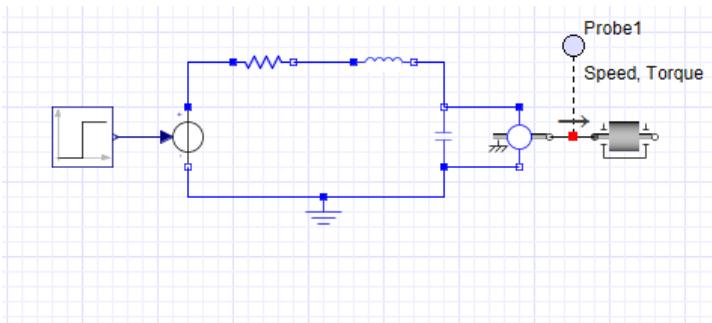
Figure 4.1: Simulation Process

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, select the **Help** menu > **Examples** > **User's Guide Examples** > **Chapter 4**, and then select one of the **Constant Acceleration**, **Sign Convention**, or **Arrow Convention** examples.

Simulation and Advanced Simulation Settings

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

To access the simulation settings, click **Simulation Settings** tab () on the right of the Parameters pane.

For a description of the Multibody and 3-D Visualization settings see *3-D Visualization and Multibody Settings* (page 106).

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	<p>The duration time of the simulation. You can specify any positive value, including floating-point values.</p> <p>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).</p>
Solver Type	Variable	<p>The type of solver to use for the simulation.</p> <ul style="list-style-type: none"> • Variable: use a variable time step to maintain error tolerances. • Fixed: use a fixed time step and disregard integration error. <p>Note: The fixed step solvers are identical to those used by MapleSim's exported code.</p>
Solver	Variable: CK45 (semi-stiff) Fixed: Euler	<p>DAE solver used during the simulation. The following choices are available when Solver Type is set to Variable.</p> <ul style="list-style-type: none"> • 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). <p>If your model is complex, you may want to use a stiff DAE solver to reduce the time required to simulate a model.</p> <p>The following choices are available when Solver Type is set to Fixed.</p> <ul style="list-style-type: none"> • 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}	0.00001	<p>The limit on the absolute error tolerance for a successful integration step if you are using a variable-step solver to run the simulation. You can specify a floating-point value for this option.</p>

Parameter	Default	Description
ϵ_{rel}	0.00001	The limit on the relative error tolerance for a successful integration step if you are using a variable-step solver to run the simulation. You can specify a floating-point value for this option.
Step size	0.001	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	2000	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). 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 <i>Managing Simulation Results and Snapshots</i> (page 99) for more information on snapshots.

Parameter	Default	Solver Type	Description
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 be used with stiff solvers (Rosenbrock or Implicit Euler).
Baumgarte	<input type="checkbox"/>	All	Apply Baumgarte constraint stabilization to your model. Select to enter values for the derivative gain (α) and the proportional gain (β) that are appropriate for your model.
Projection	<input checked="" type="checkbox"/>	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	0.00001	Fixed	The tolerance value at which the projection iterations are terminated. You can specify any positive floating-point value.
Event Projection	<input checked="" type="checkbox"/>	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	$1. \times 10^{-7}$	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.
Initial Hysteresis	$1. \times 10^{-10}$	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. Note: Used by variable step solvers.
Index 1 Error Control	<input type="checkbox"/>	Variable	When selected, error control is applied to all algebraic variables. By default, error control is only applied to algebraic variables that trigger events, are plotted, or are outputs of functions.

Parameter	Default	Solver Type	Description
Index 1 Tolerance	1	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.
Minimum Step Size	0	Variable	Set the minimum step size. This option should be used with caution, as if the solver is unable to achieve the requested error tolerance with this minimum then it will take the step anyway.
Maximum Step Size	0 / no limit	Variable	Set the maximum step size. (0 indicates no limit)
Scaling	Minimum	Variable	Specifies the method of variable scaling to apply to the system. The available choices are: <ul style="list-style-type: none"> • 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	<input type="checkbox"/>	Variable	This option specifies whether 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	<input checked="" type="checkbox"/>	All	Specifies whether to include extra plot points at event points during the simulation.
Solver Diagnostics	<input type="checkbox"/>	All	When selected, the simulation generates diagnostics describing constraint iterations, constraint residual, event iterations, and step size, and plots them after the simulation is complete in a Solver Diagnostics plot configuration in the Simulation Results tab. For systems with the projection option cleared, this incurs additional computational cost. For models with an inconsistent system of equations or run-time issues, select this option to display details about the variables, equations, and components that are causing the errors.

Parameter	Default	Solver Type	Description
Compiler	<input checked="" type="checkbox"/>	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.
Compile Optimized	<input checked="" type="checkbox"/>	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

Click the **Model Tree** tab () and then select **Probes** from the list. This lists all of the probes that you have added to the current MapleSim model.



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, two probes have been attached to a model: **Probe1** is at the top level of the model and **Probe2** is in a subsystem called **Main.Nonlinear Damper_1** (that is, the **Nonlinear Damper_1** subsystem in **Main**).

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 **Properties** tab (). You can also right-click (**Control-click** for Mac) 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 > Using Parameter Sets > Saving 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. Messages for each step of the simulation are placed in their own section (for example, *Generating Equations* or *Computing Initial Values*). When a new step starts, the preceding section is automatically collapsed. To see the messages for a section, expand the section by clicking on the gray arrow for that section. Alternatively, use the arrow keys to navigate the Console pane:

- Press the Right Arrow key to expand a section.
- Press the Left Arrow key to collapse a section.
- Press the Down Arrow key to move to the following section.
- Press the Up Arrow key to move to the preceding section.

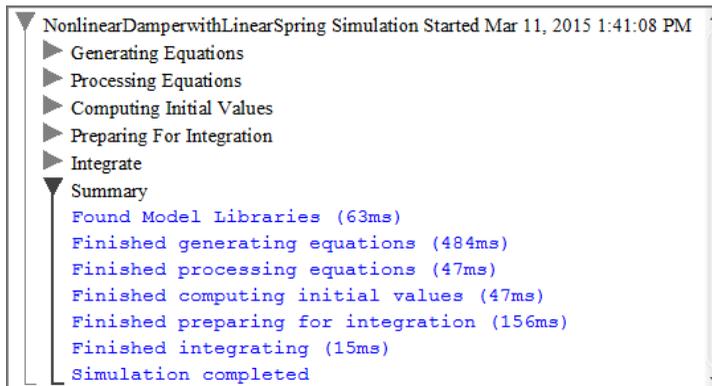
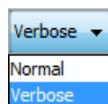


Figure 4.2: Simulation Results Progress Messages

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



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

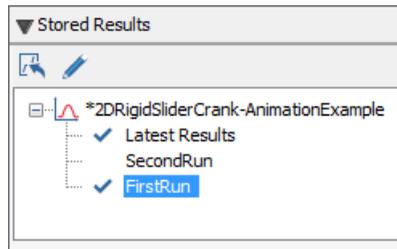
4.4 Managing Simulation Results and Snapshots

The **Stored Results** palette in the **Simulation Results** 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 Results

Whenever you simulate a model, the results are saved as **Latest Results** in 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 **Latest Results** entry.

To store a simulation result under a new name, right-click (**Control**-click for Mac) on the **Latest Results** entry, select **Save** from the context menu, and then provide a name for the stored result. You can save simulation results to compare and refer to multiple graphs generated during the current MapleSim session.



If you save the stored result and then save your model, when you open the model in a future MapleSim session, the stored result will be available in the **Stored Results** palette. However, **Latest Results** are not saved when you close and reopen the model.

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. To manage snapshots, including creating new snapshots and selecting a snapshot to use in the next simulation, see the **Advanced Simulation** settings (see **Table 4.2**).

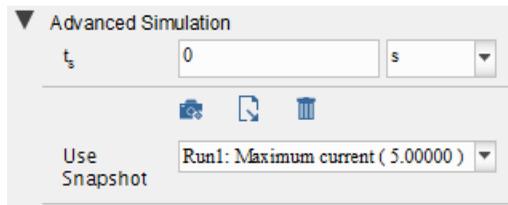


Figure 4.3: Snapshots in the Advanced Simulation Settings

When you use a snapshot in this way, the information recorded in the snapshot provides the initial conditions for subsequent simulations.

When no snapshot is selected for use in simulation, the Run Simulation icon in the toolbar is . When a snapshot is selected for use in simulation, the Run Simulation icon changes to . When a snapshot is used in a simulation, that information is available in the tooltip of the resulting entry in the **Stored Results** palette.

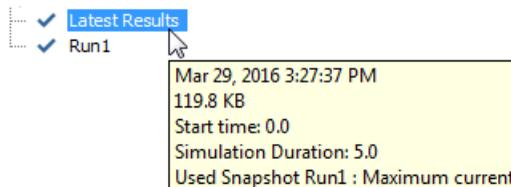


Figure 4.4: The Stored Results Palette and Snapshots

For more information, refer to the **Using MapleSim > Simulating a Model > Managing Simulation Results > Managing Simulation Snapshots** section of the MapleSim Help system.

4.5 Customizing Plot Window Configurations

By default, probed quantities are plotted in separate simulation graphs in a plot window called **Probe Plots**. 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 custom plot window configurations. You might want to create a custom plot window 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. You can even compare quantities from two different models. You can further customize a plot window by for instance customizing plot titles and specifying the number of columns to appear in the plot window.

For details on creating a plot window, refer to **Generating a New Plot Window Configuration**. For more information on plot windows, refer to the **Using MapleSim > Simulating a Model > Working with Plot Window Configurations** section of the MapleSim Help system.

When you create a new plot window, it will be populated with data without having to run the simulation again.

Tip: When you save and reopen a model with stored results, the plots may initially show the message *No data available*. You can load the plots by right-clicking on an entry in the **Stored Results** palette and selecting **Show Probe Plots**.

If you want to work with your simulation data in another application, you can export your results to a Microsoft Excel (.xls) or comma-separated value (.csv) file. For details, refer to **Exporting Simulation Graph Data**.

In the following examples, you will create custom plot window configurations.

Example: Plotting Multiple Quantities in Individual Graphs

In this example, you will view the **Probe Plots** and then create a plot window configuration in which you will add a second variable to each custom plot.

To view the Probe Plots:

1. From the **Help** menu, select **Examples > Physical Domains > Multibody**, and then select the **Double Pendulum** example.
2. Click **Run Simulation** () in the **Main Toolbar**.
3. Click **Show Simulation Results** () . The Analysis window opens with the Simulation Results tab selected. The 3-D Playback Window and the Probe Plots are in the Simulation Results tab. See **Figure 4.5**.

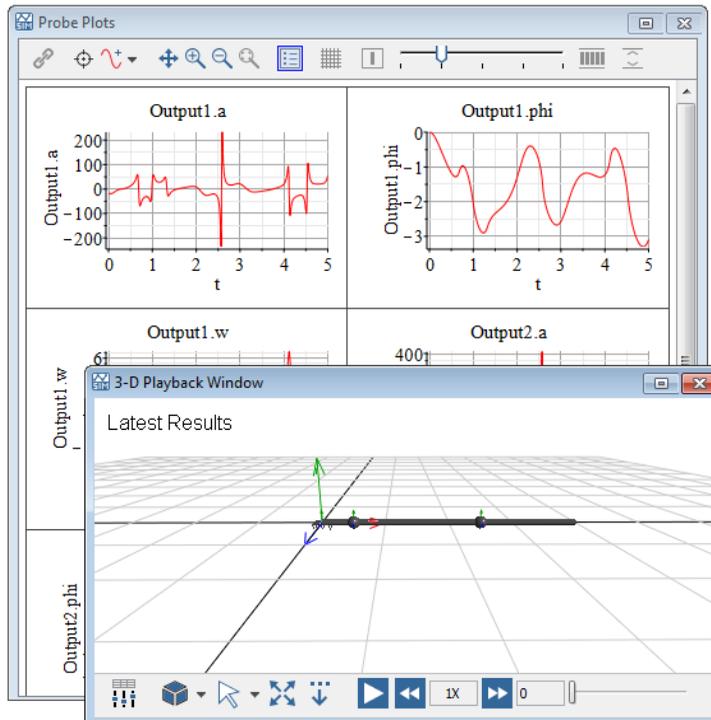


Figure 4.5: Simulation Graphs

To create a custom plot window configuration:

1. In the **Plot Windows** palette, double-click on the following plots to hide them from view: Output1.w, Output2.a, Output2.phi, Output2.w. The plot window should now show just two plots, **Output1.a** and **Output1.phi**.
2. Click the **Duplicate the Plot Window** button (📄).
3. Enter the name **Acceleration and Angle Comparison** in the Duplicate Plot Window dialog box. Click **OK**. A new plot window with the name Acceleration and Angle Comparison is created containing the currently visible plots.
Tip: To return the Probe Plots to its original view, right-click on **Probe Plots** in the **Plot Windows** palette and select **Show All Plots**.
4. In the plots window, select the plot **Output1.a** from the **Acceleration and Angle Comparison** plot window.
5. In the **Variables** palette, under **Output2**, select **a**.
6. Click **Add Selected Variable to Selected Plot** (📌). The plot now shows two curves, Output1.a and Output2.a.

7. In the plots window, select the plot **Output1.phi** from the **Acceleration and Angle Comparison** plot window.
8. In the **Variables** palette, under **Output2**, select **phi** and click **Add Selected Variable to Selected Plot** () . (Alternatively, drag **phi** onto the plot.) The plot now shows both curves.
9. In the **Plot Windows** palette, under **Acceleration and Angle Comparison**, right-click on **Output1.a** (the name of the first plot), and select **Rename**.
10. Enter the name **Acceleration (a)**. This changes the title of the plot.
11. In the **Plot Windows** palette, under **Acceleration and Angle Comparison**, right-click on **Output1.phi** (the name of the second plot), and select **Rename**.
12. Enter the name **Angle (phi)**.

The resulting plots are shown in **Figure 4.6**.

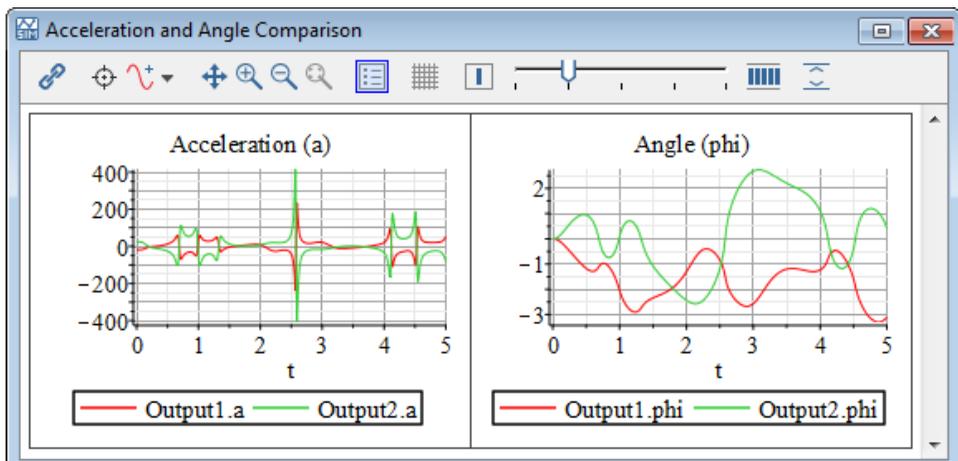


Figure 4.6: Custom Plot Window

Example: Plotting One Quantity versus Another

In this example, you will create a custom plot window 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. From the **Help** menu, select **Examples > Physical Domains > Multibody**, and then select 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 **Properties** 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 **Run Simulation** () in the **Main Toolbar**.
8. Click **Show Simulation Results** () . The Analysis window opens, displaying the Probe Plots in the Simulation Results tab.
9. We want the plots window to only show two plots, **FirstLink.r_0[2]** and **SecondLink.r_0[2]**, which will be the basis for creating a new plot window configuration. In the **Plot Windows** palette, double-click on the names of the other plots to hide them from view.
10. Click **Duplicate the Plot Window** () .
11. Enter the name **X versus Y** in the Duplicate Plot Window dialog box. Click **OK**. A new plot window with the name X versus Y is created containing the currently visible plots.
Tip: To return the Probe Plots to its original view, right-click on **Probe Plots** in the **Plot Windows** palette and select **Show All Plots**.
12. In the **Plot Windows** palette, under **X versus Y**, right-click on **FirstLink.r_0[2]** (the name of the first plot), and select **Rename**.
13. Enter the name **Top Link**. This changes the title of the plot.
14. Similarly, change the name of the plot **SecondLink.r_0[2]** to **Bottom Link**.
15. In the plots window, select the plot **Top Link** from the **X versus Y** plot window.
16. In the **Variables** palette, select **FirstLink: r_0[1]**.
17. Click **Place Selected Variable on x-Axis** () .
18. In the plot window, select the plot **Bottom Link** from the **X versus Y** plot window.
19. In the **Variables** palette, select **SecondLink: r_0[1]**.
20. Click **Place Selected Variable on x-Axis** () .
21. In the **Plotting Toolbar**, use the slider to change the number of plot columns to 1.
The resulting plots are shown in **Figure 4.7**.

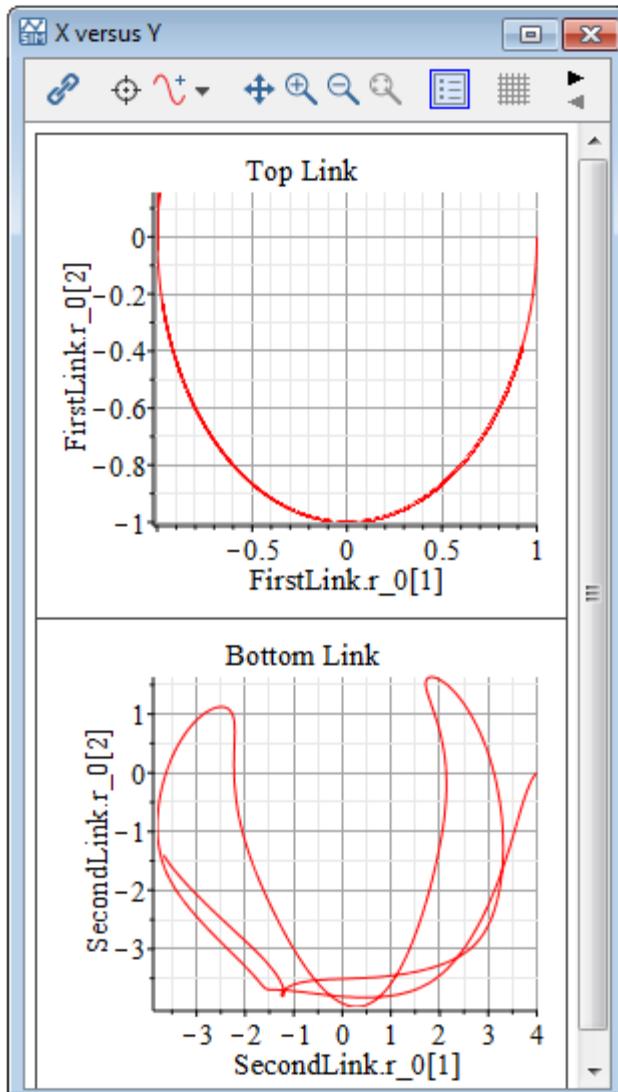


Figure 4.7: Plot One Quantity Versus Another

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 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. 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 **Library Components** tab.

After simulating your model, you can animate your 3-D model in the **3-D Playback Window**. You can control playback options to focus on specific components and their motions, for instance by specifying camera tracking options to center an object in the **3-D Playback Window** during an animation. You can also attach trace lines to show where components move during an animation.

Tip: The quality of the visualization is affected if any open plot windows are behind the **3-D Playback Window**. If you are experiencing playback issues, try moving the **3-D Playback Window** so that it does not overlap a plot window. Alternatively, minimize or close any open plot windows.

CAD geometry and visualization shapes are drawn transparent in the 3-D Workspace and non-transparent in the 3-D Playback Window.

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 **Multibody Settings** tab of the Parameters pane in sections for **Animation**, **Multibody**, and **Visualization** settings.

To access the simulation settings, click **Multibody Settings** tab () on the right of the Parameters pane.

Animation Settings

Under Animation, ensure the **3-D Animation** check box is enabled if you want to see an animation of your simulation runs.

Multibody Settings

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

Table 4.3: Multibody Parameter Values

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

Visualization Settings

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

Table 4.4: 3-D Visualization Parameter Values

Parameter	Default	Description
Enable Translational Snapping	<input type="checkbox"/>	When selected, components are positioned at the closest location in 3-D space based on the translation snap delta value.
Translation Snap Delta	1	Specifies the translation snap delta spacing.
Enable Rotational Snapping	<input type="checkbox"/>	When selected, components are positioned at the closest location in 3-D space based on the rotation snap delta value.
Rotation Snap Delta	Pi/2	Specifies the rotational snap delta spacing.
Perspective Grid Extent	10	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	Specifies the space between grid lines.

Parameter	Default	Description
Base Radius	0.02	Implicit geometry consists of spheres and cylinders representing multibody components in the 3-D Workspace . Cylinders are drawn using Base Radius, while spheres (for Rigid Bodies and Joints) are drawn using Base Radius * 2.
Enable View Change Animations	<input checked="" type="checkbox"/>	When enabled, a smooth transition occurs when switching between 3-D orthographic and perspective views and when using Fit Scene , Fit Selected , or Fit Animation in the 3-D Workspace and 3-D Playback Window .

The 3-D Workspace

The **3-D Workspace** is the area in which you build and view 3-D models in the MapleSim window. It is found in the 3-D Workspace tab of the Analysis window.

To open 3-D Workspace:

- From the **View** menu, select **Show 3-D Workspace...** The Analysis window opens showing the 3-D Workspace. Alternatively, from the MapleSim main toolbar, click **Show 3-D Workspace** (🔍).

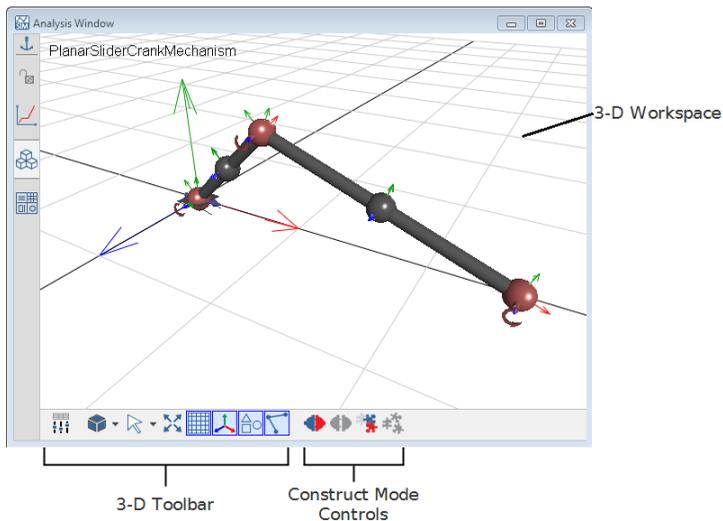
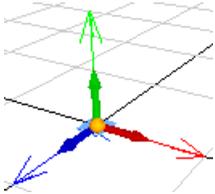


Figure 4.8: 3-D Workspace

Table 4.5: 3-D Workspace Controls

Component	Description
3-D Workspace	<p>The area in which you build and view a 3-D model. The arrows at the origin indicate the directions of the world axes and are designated by color:</p> <div style="display: flex; align-items: center; justify-content: center;">  <div style="margin-left: 20px;"> <p>X - red</p> <p>Y - green</p> <p>Z - blue</p> </div> </div> <p>You can use the grid as a reference to determine the relative sizes and positions of elements in your 3-D model.</p>
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.
Construct Mode Controls	Part of the 3-D toolbar, these are controls for building and assembling a 3-D model, and connecting 3-D objects.

You can hover your mouse pointer over any of the buttons on the toolbar to view its tooltip.

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 4.9: 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.

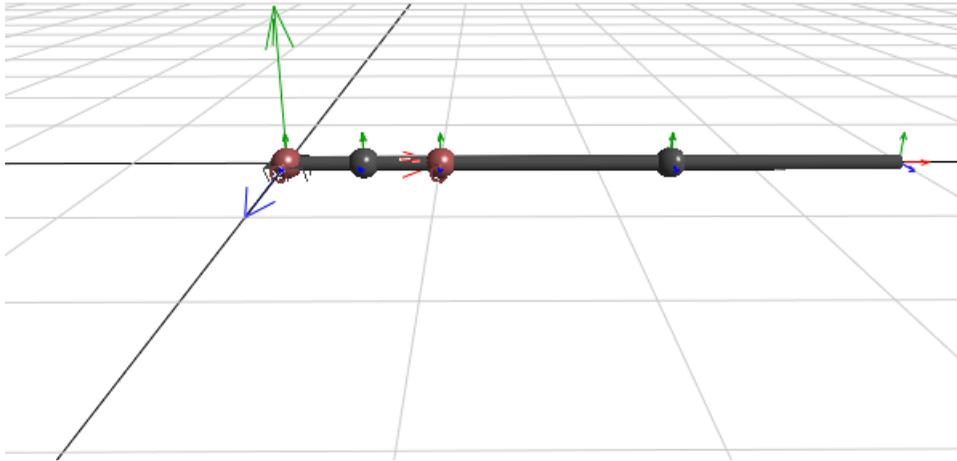


Figure 4.10: Perspective View Double Pendulum

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.

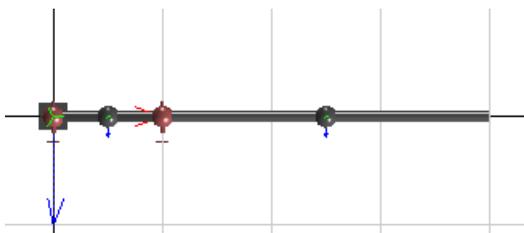


Figure 4.11: Orthographic View Double Pendulum

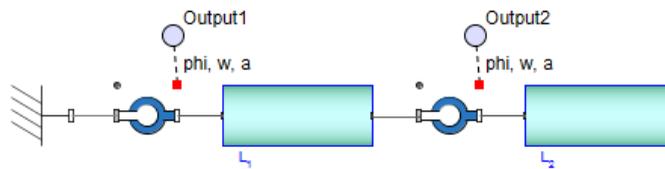
You can browse a model and change the model view while an animation is static or playing. In all of the views, you can pan and zoom into or out from your model. In the perspective view, you can also move the camera to view your model from above or below, and from any direction around your model.

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.

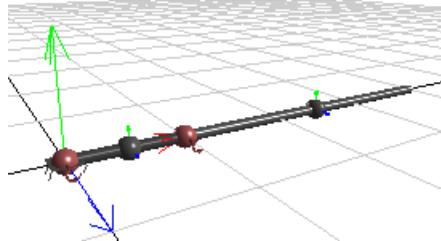


Figure 4.12: Implicit Geometry Double Pendulum

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.

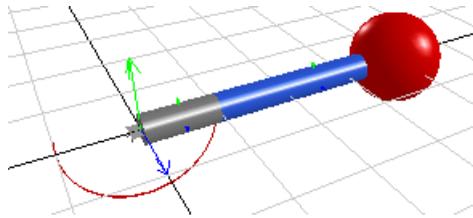


Figure 4.13: Attached Shapes

You can customize the color, size, scale, and other visual aspects of the attached shapes by setting parameter values for individual components in the **Properties** 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 **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** in the MapleSim Help system.

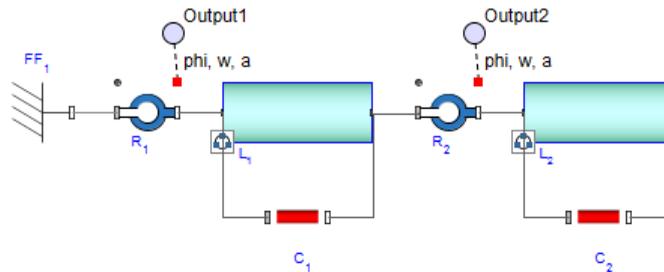
Note: If your model contains **Flexible Beam** components, deflection of the beam will not be depicted in the implicit geometry of your 3-D model.

Example: Adding Attached Shapes to a Double Pendulum Model

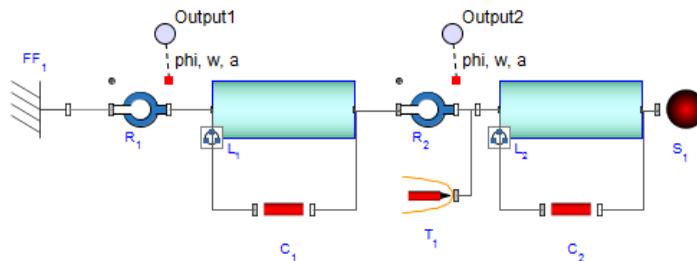
In the following example, you will add cylinder shapes to represent the pendulum stem and a sphere component to represent the pendulum bob. You will also add a **Path Trace** component to display the path on which the revolute joint will move during an animation.

To add attached shapes

1. From the **Help** menu, select **Examples > Physical Domains > Multibody**, and then select 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.



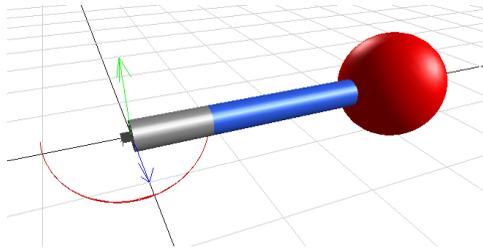
5. From the same menu, add a **Spherical Geometry** component and place it to the right of the L_2 shared subsystem.
6. Right-click (**Control-click** for Mac) the **Spherical Geometry** component and select **Flip Horizontal**.
7. Add a **Path Trace** component and place it between the two **Cylindrical 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 **Properties** tab () on the right side of the MapleSim window, change the radius of the cylinder to **0.2**.
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 **0.2** and change the color.
14. Select the **Spherical Geometry** component (S_1 in the previous figure).
15. Change the radius of the sphere to **0.8** and change the color.
16. To simulate the model, click **Run Simulation** () in the **Main Toolbar**.

When the simulation is complete, the Analysis window opens with the Simulation Results tab selected. The **3-D Playback Window** displays your model with the attached shapes.



17. To animate the model, click **Play** () in the **3-D Playback Window**.

For another example of how to use the **Path Trace** component, from the **Help** menu, select **Examples > User's Guide Examples > Chapter 4**, and then select 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 multibody components to a 3-D model, put the **3-D Workspace** beside the main MapleSim window, then drag the desired components from the **Multibody** palette, the **Favorites** palette, a custom library that you created, or from the search pane in the **Library Components** tab to the **3-D Workspace**.

In the **3-D Workspace**, 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**.

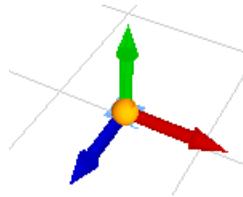
Any changes that you make to your 3-D model are automatically shown in the block diagram representation in the **Model Workspace** and vice versa. For example, if you add and connect a **Flexible Beam** component in the **3-D Workspace**, the block diagram representation of the **Flexible Beam** with 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 the **3-D Workspace**, you can position individual objects or groups of objects by clicking and dragging the 3-D manipulators.



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. Assembling a 3-D model refers to synchronizing the model that 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. Alternatively, click **Update 3-D View** () in the **3-D Toolbar** to assemble the model. 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 Unenforced Constraints Button to Manipulate Joints in the 3-D Workspace

You can select a joint object in the **3-D Workspace** and click **Do Not Enforce 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 **Do Not Enforce 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 do not enforce constraints button does not affect the actual initial conditions specified for your joint components in the **Properties** 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 both the **3-D Workspace** and the **3-D Playback Window**. 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 both the **3-D Workspace** and the **3-D Playback Window**. 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 **Properties** 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. You will perform the following tasks:

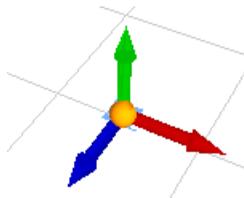
To build and animate a double pendulum:

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.

Adding and Moving Objects in the 3-D Workspace

To add or move an object:

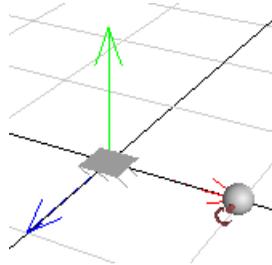
1. Open a new MapleSim document.
2. From the MapleSim main toolbar, click **Show 3-D Workspace** () and arrange that window beside the main MapleSim window.
3. Click **Always-On-Top** (). This keeps the Analysis window in focus as you work in the MapleSim window.
4. Under the **Library Components** tab, expand the **Multibody** palette, and then expand the **Bodies and Frames** menu.
5. 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.



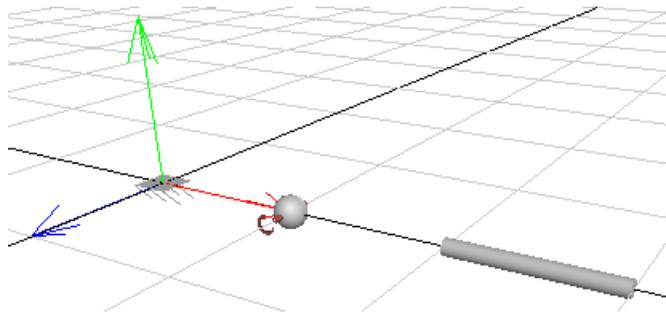
You can use this manipulator to position objects in the **3-D Workspace**.

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

7. 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**.



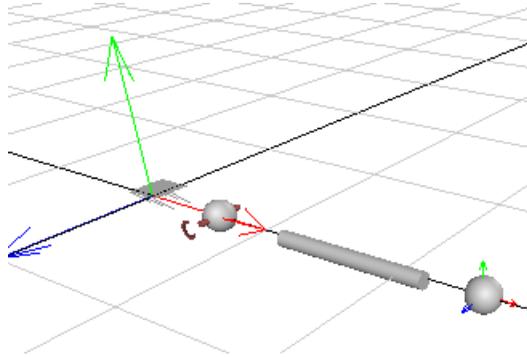
8. From the **Multibody > Bodies and Frames** menu, drag a **Rigid Body Frame** component into the **3-D Workspace** and place it to the right of the **Revolute** component.



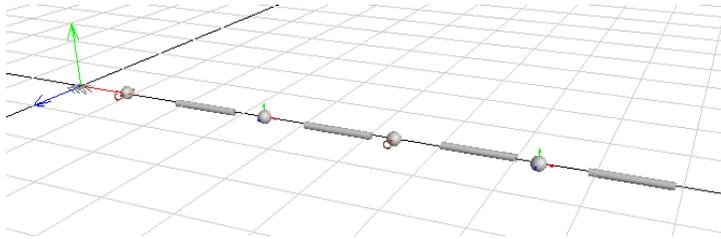
9. From the same menu, drag a **Rigid Body** component into the **3-D Workspace** and place it to the right of the **Rigid Body Frame**.

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**.

10. 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.



11. 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.

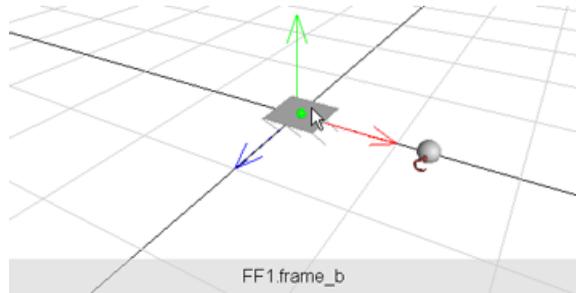


Connecting 3-D Objects

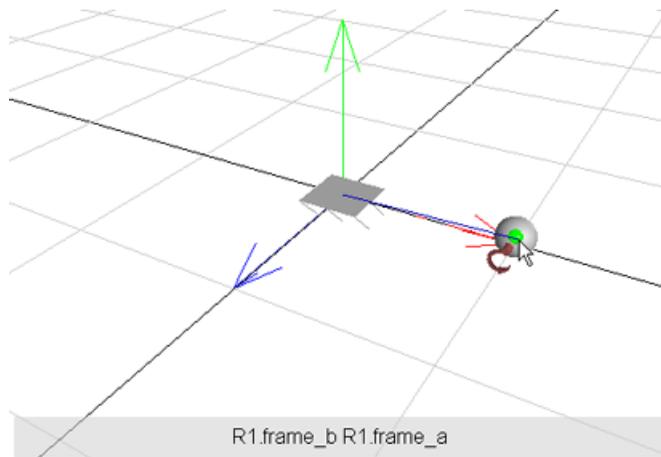
You will now connect the objects that you added in the previous task.

To connect objects:

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

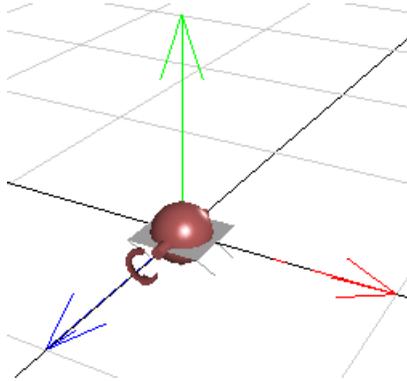


3. Click the green dot once to start the connection line.
4. Hover your mouse pointer over the first **Revolute** joint component. The gray panel at the bottom of the **3-D Workspace** displays the names of the **Revolute** joint frames.



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

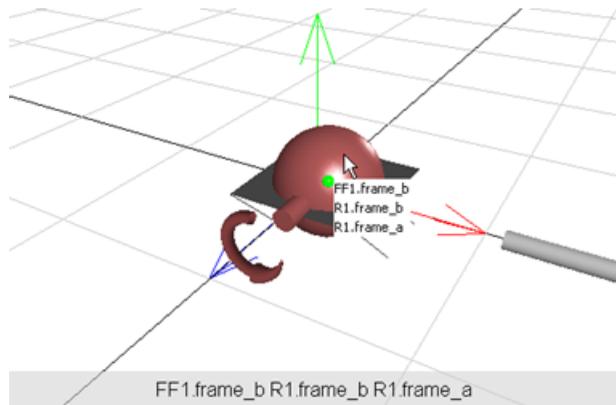
6. Select **R1.frame_a**. The components are connected in the **3-D Workspace**.



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

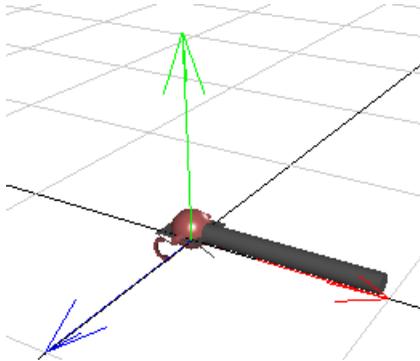
7. Click **Connect ports** () to start the next connection line.

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**.

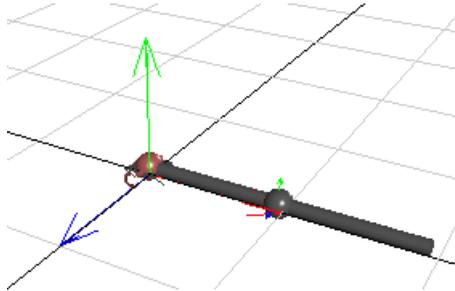
10. Drag your mouse pointer to the end of the cylinder that represents the **Rigid Body Frame** and click the green dot.



frame_b of the first revolute joint, R_1 , is now connected to **frame_a** of the first rigid body frame, RBF_1 .

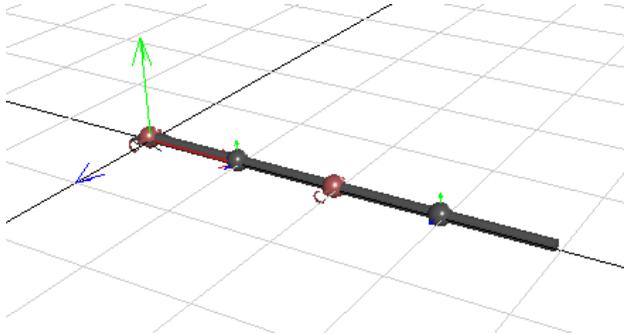
11. Click **Connect ports** to start a new connection line.
12. Hover your mouse pointer over the other end of the cylinder that represents the RBF_1 component and click the cylinder once.
13. Drag your mouse pointer to the sphere that represents the first **Rigid Body** component, RB_1 , and click it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



In the 2-D model workspace, 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.

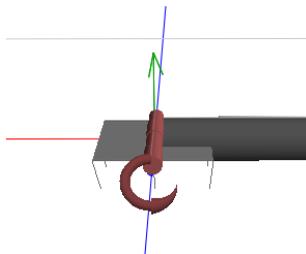
Setting Initial Conditions for the Joint Components

You can set initial conditions for joint components by using graphical controls in the **3-D Workspace**.

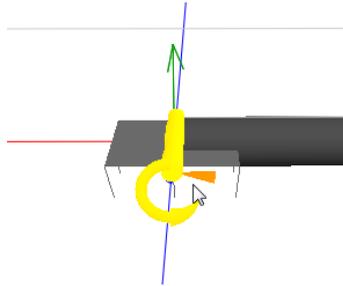
Note: Joint components that have been assigned custom parameter values defined in a parameter block, global parameter, or subsystem parameter will not allow the use of graphical controls for setting initial conditions. In these cases, use the fields in the **Properties** tab to set the initial conditions.

To set initial conditions:

1. 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.



2. Hover your mouse pointer over the manipulator. The manipulator appears in yellow.
3. 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.

4. Release your mouse button when the meter is at the approximate initial condition value that you want. In the **Properties** tab (📄), the θ_0 parameter displays the value that you selected. 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 θ_0 parameter in the **Properties** tab. Initial conditions that you specify in the **Properties** tab will be shown in the 3-D model.
- To specify precise initial angle conditions, turn on snapping by selecting **Enable Rotational Snapping** under **Visualization** in the **Multibody Settings** tab (🔧).

Animating the 3-D Model

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

To animate the 3-D model:

1. Simulate your model by clicking **Run Simulation** () in the **Main Toolbar**. When the simulation is complete, the Analysis window opens, containing both simulation plots and the **3-D Playback Window**.
2. Select the 3-D Playback Window. (If the playback window is not visible, double-click **3-D Playback Window** in the **Plot Windows** palette in the left pane of the Analysis window to view it.)
3. To play the animation, click **Play** () in the **Playback Toolbar**.

Exporting a Movie of the 3-D Model

The Export Movie feature allows you to export a recorded simulation as a movie file (for example AVI) for playback outside of MapleSim or to share with others who may not have MapleSim. For more information, see **Using MapleSim > Visualizing a 3-D Model > Exporting a Simulation as a Movie** in the MapleSim Help system.

4.7 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 **Simulation Settings** tab () , 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** or pressing **Ctrl + E** (**Command + E** for Mac), 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 **Using MapleSim > Simulating a Model > Excluding Objects From a Simulation** in the MapleSim Help system.

5 Analyzing and Manipulating a Model

In this chapter:

- *Overview (page 127)*
- *Retrieving Equations and Properties from a Model (page 130)*
- *Analyzing Linear Systems (page 131)*
- *Optimizing Parameters (page 132)*
- *Generating and Exporting C Code from a Model (page 133)*
- *Generating a Custom Component from External C Code/Library Definition (page 141)*
- *Working with Maple Embedded Components (page 144)*

5.1 Overview

MapleSim is fully integrated with the Maple environment.

- You can use Maple-based **Apps** within the MapleSim interface.
- You can attach **Templates** to a model, which open in the Maple interface.
- You can attach **Maple worksheets** to a MapleSim model, allowing you full access to commands, embedded components, plotting tools, and many other technical features to analyze and manipulate the dynamic behavior of a MapleSim model or subsystem.
- You can use the MapleSim application programming interface (**API**) in a Maple worksheet to manipulate, simulate, and analyze an existing MapleSim model programmatically.

MapleSim Apps and Templates

To start working with your MapleSim model in Maple, you can use the apps and templates available in the **Add Apps or Templates** tab (). Apps are pre-built tools for model building and analysis tasks: you first create a MapleSim model and open it in one of the available apps to perform an analysis task. Apps open in the **Analysis** window in MapleSim. For example, you can use apps to perform parameter sweeps, Monte Carlo simulations, or code generation.

Templates are Maple worksheets that you attach to a model. Templates open in Maple. Templates can be used to retrieve and work with equations for a subsystem or build custom components.

The following tables list the MapleSim Apps and Templates available in the **Add Apps or Templates** tab (.

Table 5.1: MapleSim Apps

App Name	Task
Component Creation	
1-D Motion Generation	Create motion profiles for 1-D motion that adhere to defined velocity, acceleration, and jerk constraints.
Custom Port	Create a custom port for a custom component. For more information, see <i>Creating Custom Modeling Components (page 71)</i> .
DCV Builder	Create a custom hydraulic directional control valve.
External C/Library Block	Define and generate a MapleSim custom component from external C Code/DLL.
Kinematic Cam Generation	Model the kinematic behavior of cams and followers.
Linearization	Linearize a MapleSim continuous subsystem. Perform linear analysis on the linear system object, such as generating Bode plots and Root Locus plots.
Model Analysis	
Equation Extraction	Retrieve equations from linear or nonlinear models. For more information, see <i>Tutorial 7: Using the Equation Extraction App (page 201)</i> .
Initialization Diagnostics	Resolve inconsistent initial conditions and errors detected during the simulation, as well as give insight into the original configuration of the system.
Modal Analysis	Visualize the vibration modes of a multibody model.
Monte Carlo Simulation	Define a random distribution for a parameter and run a simulation using the distribution.
Multibody Analysis	Retrieve multibody equations in a form that is suitable for manipulation and analysis.
Optimization	Analyze and edit the parameters of a model and view possible simulation results in a graph. For more information, see <i>Optimizing Parameters (page 132)</i> .
Parameter Sweep	Execute a parameter sweep.
Utility	
Code Generation	Translate your model into C code. For more information, see <i>Generating and Exporting C Code from a Model (page 133)</i> .
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 <i>Creating a Data Set for an Interpolation Table Component (page 63)</i> .
Excel Connectivity	Import MapleSim parameter sets from an Excel spreadsheet, or export MapleSim parameter sets to an Excel spreadsheet.
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.

App Name	Task
Heat Transfer (The MapleSim Heat Transfer Library is available as a separate add-on.)	
Board and Parts Generation	Generate a component of a board with parts.
Temperature Distribution	Plot a 3-D visualization of the temperature distribution of a Heat Transfer shape component, mapping temperatures to colors.

Table 5.2: MapleSim Templates

Template Name	Task
Custom Component	Create a custom modeling component based on a mathematical model. For more information, see <i>Creating Custom Modeling Components (page 71)</i> .
Worksheet	Create a worksheet by opening a MapleSim model in an embedded component.

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

Working with Apps

If you close and reopen an app, the Apps Manager remembers the previous state of the app.

The Apps Manager displays three options:

- **Refresh** (

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. An app or 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 64).

For an example of a basic structure for exporting models, from the **Help** menu, select **Examples > User's Guide Examples > Chapter 5**, and then select 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 Extraction App** to retrieve, define, and analyze equations and properties such as parameters and variables in your model. Additional features within this app are useful in generating reusable equations when there is more than one subsystem.

For a complete tutorial on how to use the **Equation Extraction App**, see *Tutorial 7: Using the Equation Extraction App* (page 201).

To retrieve equations and properties:

1. In MapleSim, open the model for which you want to retrieve equations or properties.
2. Click the **Add Apps or Templates** tab () .
3. From the **Apps** palette, select **Equation Extraction**.

4. Use the navigation tools under **Subsystem Selection** to select the subsystem for which you want to view equations. If you want to retrieve equations from the complete system, click **Main**.
5. Click **Load Selected Subsystem**. The model equations are extracted and the system parameters and variables are loaded. Under **View Equations**, click **Extract Equations**. The system equations are shown and are automatically stored in the variable DAEs.

5.3 Analyzing Linear Systems

You can use the **Linearization** app 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, Nyquist, 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 the **Add Apps or Templates** tab () .
3. From the **Apps** palette, select **Linearization**. The App opens in the **Analysis** window.
4. Using the navigation tools above the model diagram, select the subsystem for which you want to view equations.
5. Click **Load Selected Subsystem**.
6. (Optional) Make changes in the configuration section.
7. Click **Linearize**. The linear system object is created and the equations for the system are displayed.
8. (Optional) Create bode, nyquist, root locus, or response plots.
9. After you have analyzed and configured your system, you can create a custom component based on the system and attach it to your MapleSim model.

Linear System Analysis

You can use the tools in the **Analysis** section to analyze your linear system and to view the effects of different inputs on the outputs of your system.

For analysis, you can use the following tools:

- Bode plot
- Nyquist plot

- Root locus plot
- Response plot

In the **Response** section you can choose an input signal to apply to the system and then simulate to see the effects on the output.

Create Component

After you have analyzed and configured your system, you can create a custom component based on the system and attach it to your MapleSim model.

To create a custom component from your system:

1. In the **Create Model** section, Enter a name in the **Component Name** text box.
2. Enter a description for your component in the **Description** text box.
3. Click **Create**.

Your custom component can be found in the **Components** palette of the **Local Components** tab () of your MapleSim model.

5.4 Optimizing Parameters

You can use the **Parameter Optimization App** 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 website 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 the **Add Apps or Templates** tab ()
3. From the **Apps** palette, double-click **Optimization**. The App opens in the **Analysis** window.
4. Use the navigation tools under **Subsystem Selection** to select the subsystem for which you want to view equations. If you want to retrieve equations from the complete system, click **Main**.
5. Click **Load System**. The model simulation settings are imported.

6. In the **Parameters** section, use the combo box list to select a parameter you want to optimize and click **Add**. Do the same to select other parameters and you'll see the list box appear with the parameters you selected.

Note: When a parameter is selected, its current (nominal) value is shown in **Nom**.

7. Set the range over which the parameter may vary using the **Min** and **Max** fields.
8. Using the same process described above, set the **Min** and **Max** fields for the other parameters you want to optimize.
9. When you have defined all of the parameters, under **Objective Function** you can specify details of how to construct your objective function, and specify whether to **Minimize Objective** or **Maximize Objective**. The objective function is defined as a Maple procedure.

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

10. Now you can perform the parameter optimization. If you have the Maple Global Optimization toolbox, you can use it for this step. Click **Run Parameter Optimization** to perform the parameter optimization. The **Results** section displays the parameter values that optimize the objective function.

To test different values for the parameters, move the sliders and then click **Run Simulation**. Click **Restore Optimum Values** to restore the values computed in step 10.

11. To use the parameter values of the sliders in the linked model, click **Update Parameters in MapleSim Model**.

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 App** 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.

For a general high-level overview of the MapleSim code export process, refer to Generating Code for Export.

With the **Code Generation** app, 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 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:

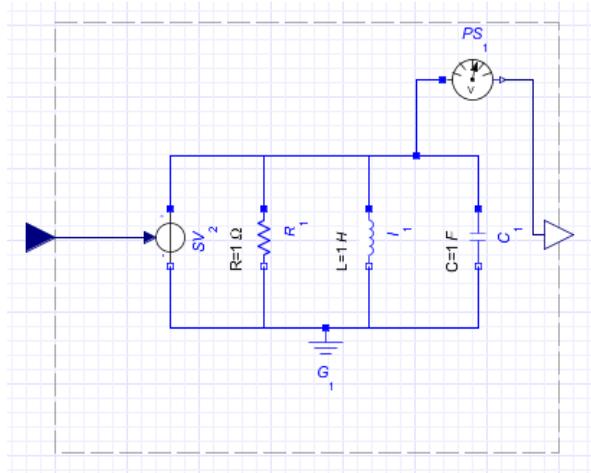
- Multibody parameters cannot be directly exported. Only user-generated parameters that the multibody parameters are assigned using Properties tab () are able to be exported.
- Dependent parameters cannot be exported. If the parameter A is a function of b ($A=b$, $A=\sin(b)$, $A=1+3/b$, etc.), then A 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:

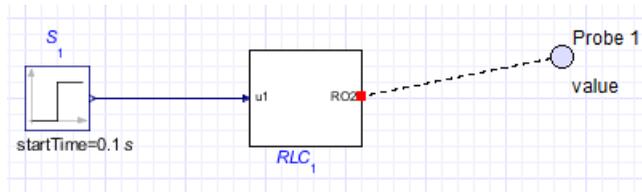
- Preparing the MapleSim model
- Opening the Code Generation app
- Loading the subsystem
- Customizing, defining, and assigning parameter values to specific ports
- Selecting the code generation options
- Generating and saving the C code

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.

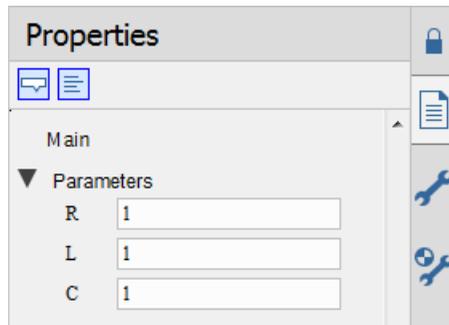


When generating code for the top-level system, there are no inputs, but all probed values are considered as outputs, as shown in the following figure.



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 can 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.



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 can 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.

Opening the Code Generation App

To perform code generation, first open the Code Generation app.

1. Click the **Add Apps or Templates** tab ()
2. From the **Apps** palette, select **Code Generation**. The C Code Generation app opens in the **Analysis** window.

Loading the Subsystem

The Subsystem Selection part of the app 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 Configuration interface lets you customize, define and assign parameter values to specific ports. Subsystem components to which you assign the parameter inherit a parameter value defined at the subsystem level.

Configuration: **Inputs** Outputs Parameters Code Export Options

Tip: If you close and reopen this app, the Apps Manager remembers the previous state of the app.

The refresh, export, and import buttons can be used to maintain or restore settings if you close the app and then reopen it:

- **Refresh** (↻): Return the app to the default settings.
- **Export** (↑): Save the current settings for the app.
- **Import** (↓): Retrieve saved settings for an app.

Inputs:

Input Variables		Change Row
1		

Inputs: 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

Outputs: Contains the model output variables.

Export: Select which variables you want to leave in the symbolic form.

Change Row: Select the equations with the specified row.

Export All/Export None: Allows you to either select or remove all of the parameters for export.

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

Parameters:

Toggle Export Column

	Parameters	Value	Export	Change Row
1	C1_C	0.4e-3		
2	EMF1_fixed_phi0	0.		
3	EMF1_k	0.1e2		
4	I1_L	.76		
5	R1_R	14.1		
6	R1_T_ref	300.15		

Parameters: Contains the model parameters.

Filter: Filter for specific parameters.

View All/Exports: Toggle the view.

Export: Select which parameters you want to export in the symbolic form.

Value: Displays the value for the system parameter.

Export All/Export None: Allows you to either select or remove all of the parameters for export.

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 Code Export 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

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.

Max projection iterations:

Error tolerance:

Apply projection during event iterations

Set the **Max 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™ website to control the drift in the result of the DAE system.

Baumgarte Constraint Stabilization

The Baumgarte constraint stabilization method stabilizes the position constraint equations by combining the position, velocity, and acceleration constraints into a single expression. By integrating the linear equation in terms of the acceleration, the Baumgarte parameters, alpha and beta, act to stabilize the constraints at the position level.

Baumgarte Constraint Stabilization:

Apply Baumgarte constraint stabilization

Export Baumgarte parameters

alpha

beta

Apply Baumgarte constraint stabilization: select this option to apply Baumgarte constraint stabilization to your model.

Export Baumgarte parameters: select this option to have constants for alpha and beta included in the generated C code. This allows you change the values of alpha and beta in the source code. You can then recompile your code and run it to see the effect on your model.

alpha: enter a value for the derivative gain that is appropriate to your model.

beta: enter a value for the proportional gain that is appropriate to your model.

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.

Event Handling Options:

Max event iterations:

Width of event hysteresis band:

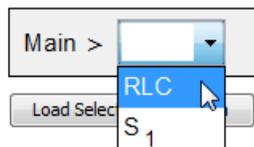
Set the **Maximum number of event iterations** to specify the maximum number of times that a projection is permitted to iterate to obtain a more accurate solution.

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 the **Add Apps or Templates** tab () .
4. From the **Apps** palette, double-click **Code Generation**. The app opens in the **Apps Manager** tab of the **Analysis** window.
5. 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.



6. 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 app.
7. Configure the inputs, outputs, and parameters.

8. Under **Code Export Options**, select 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**. The C code is saved to your specified location. After the C code is generated, the code can be viewed in the **View Code** area at the bottom of the app.

```

/*****
 * Automatically generated by Maple.
 * Created On: Tue May 04 17:14:50 2021.
 *****/
#ifdef WMI_WINNT
#define EXP __declspec(dllexport)
#else
#ifdef X86_64_WINDOWS
#define EXP __declspec(dllexport)
#else
#define EXP
#endif
#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) { ((void)(args)); kv=kv_in; return(kv->toMapleNULL()); }
#include <string.h>
#else
#include <string.h>
#include <stdarg.h>
<

```

5.6 Generating a Custom Component from External C Code/Library Definition

MapleSim can call external code directly within your model. By using the **External C Code/Library Definition** app, 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 app, 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/Library Definition app.

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/Library Definition App

Click **Add Apps or Templates** () and select the **External C/Library Block** app. The **External C/Library Block** app opens in the **Analysis** window.

Specifying the C/Library Code Location and Options

Use the External C Code/Library Definition app 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.

Source Type: C code Compiled library

Source Location: File Attachment Text Area

File:

Browse

Header File:

Browse

Attachment:

f1.c ▾

Refresh

Validate C

Specifying a Header file (optional)

If required, select **Header File** and provide the location of the existing header file.

Using an Existing C or Library File

Provide the location of the existing C or Library file.

Providing External Code into a Text Area and then Saving to File or Attachment

Enter the content directly into the app, then save to either a file or attachment before generating the component. *Tutorial 6: Using the External C Code/DLL Custom Component App (page 196)* walks you through this step.

If you save to an attachment, the attachment is saved in the **Attached Files** tab () under **Other**.

Click **Validate C** to verify the validity of the provided C code.

Defining the C/Library Code Location and Options

In the **Configuration** section, 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 output of the Modelica block (by checking passed by reference).

Use the up/down arrows to rearrange the parameters. The order must match that of the C function. If necessary click **Delete Parameter** to remove a parameter.

Configuration: Function Parameters Return ?

Function Name:

Configuration: Function Parameters Return ?

Name:

Desc:

Passed By Reference

Data Type: Array? Dim:

Configuration: Function Parameters Return ?

Return?

Return Name:

Return Type:

For a complete tutorial on how to create an external code custom component and its use, see *Tutorial 6: Using the External C Code/DLL Custom Component App (page 196)*.

Component Generation

To generate the custom component:

1. Enter the **Block Name** for the custom component under **Component Generation**.
2. Click **Generate Component**. In MapleSim, the custom component appears in the **Local Components** tab () located in the **Components** palette, on the left side of the MapleSim window. The modelica code for the component can be viewed in the Modelica Code Editor.

5.7 Working with the MapleSim API and Maple Commands

In addition to working with apps and templates to interact with and analyze a model, you can use the MapleSim application programming interface (API) in a Maple worksheet. The first step is to use the `LinkModel` command to link to a MapleSim model. The `LinkModel` command returns a 'connection module' that allows access to a MapleSim model. For more information about the MapleSim API, refer to the MapleSim help page and the examples section in the `LinkModel` help page.

Within Maple, you can also the full power of Maple to work with your model, use commands from any Maple packages, including **MapleSim** and **DynamicSystems**, to work with your model programmatically.

5.8 Working with Maple Embedded Components

Embedded Components are simple graphical interface elements that you embed into a Maple 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 **Document-Tools** 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 Components palette.

Tip: The pre-built analysis tools available in templates 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.

For more information about advanced analysis tasks, first open the **Sliding Table** example from the **Help > Examples > User's Guide Examples > Chapter 5** menu, and then open the **AdvancedAnalysis.mw** worksheet attachment (from MapleSim, under the **Attached**

Files tab () , expand the **Documents** entry).

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 **Help > Examples > User's Guide Examples** menu, in the order that they are presented in the User's Guide.

In this chapter:

- *Tutorial 1: Modeling a DC Motor with a Gearbox (page 147)*
- *Tutorial 2: Modeling a Cable Tension Controller (page 154)*
- *Tutorial 3: Modeling a Nonlinear Damper (page 158)*
- *Tutorial 4: Modeling a Planar Slider-Crank Mechanism (page 166)*
- *Tutorial 5: Using the Custom Component Template (page 174)*
- *Tutorial 6: Using the External C Code/DLL Custom Component App (page 196)*
- *Tutorial 7: Using the Equation Extraction App (page 201)*
- *Tutorial 8: Modeling Hydraulic Systems (page 206)*

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:

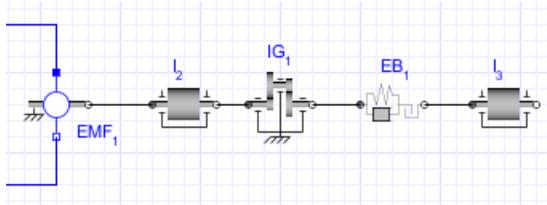
- Add a gearbox to the DC motor model
- Simulate the DC motor with gearbox model
- Group the DC motor components into a subsystem
- Assign global parameters to the model
- Add signal block components and a PI controller to the model
- Simulate the modified DC motor model using different conditions

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. From the **Help** menu, select **Examples > User's Guide Examples > Chapter 1**, and then select the **Simple DC Motor** example.
2. Delete the existing probe from the workspace.
3. Select the **Library Components** tab () and then 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 in the following figure.

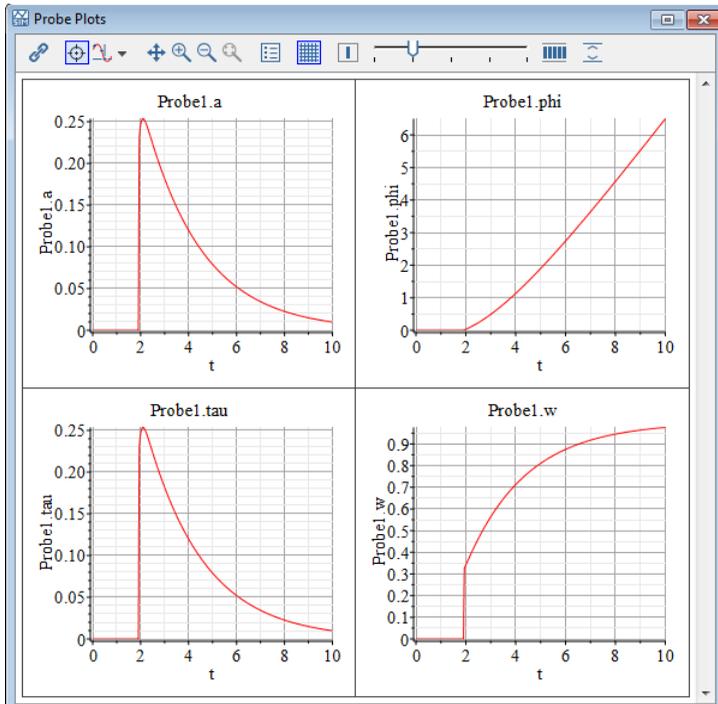


5. In the **Model Workspace**, click the **Ideal Gear** component.
6. In the **Properties** tab (), change the transmission ratio, **r**, to **10** and then press **Enter** to accept the value.
7. Specify the following parameter values for the other components:
 - For the **Elasto-Backlash** component, in the **b** field, change the total backlash value to **0.3rad**. In the **d** field, change the damping constant to $10^4 \frac{N \cdot m \cdot s}{rad}$.
 - For the first **Inertia** component (**I₂**), in the **J** field, change the moment of inertia value to **10kg·m²**.
 - For the second **Inertia** component (**I₃**), in the **J** field, change the moment of inertia value to **1kg·m²**.
 - 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 (**I₃**). 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 (ϕ), speed (ω), acceleration (a), and torque (τ) values in the simulation graphs, in the **Properties** tab () , select **Angle**, **Angular Velocity**, **Angular 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**.
9. Click **Show Simulation Results** () . The following graphs appear in the Analysis window.

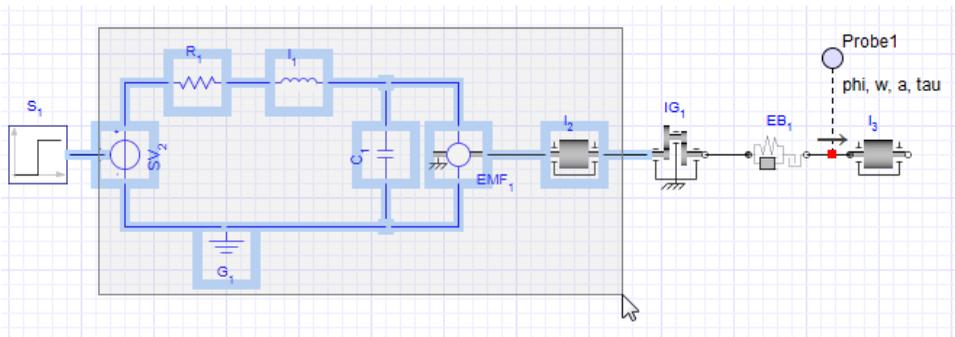


10. To verify the results, from the **Help** menu, select **Examples > User's Guide Examples > Chapter 6**, and then select the **DC Motor with Gearbox** example.

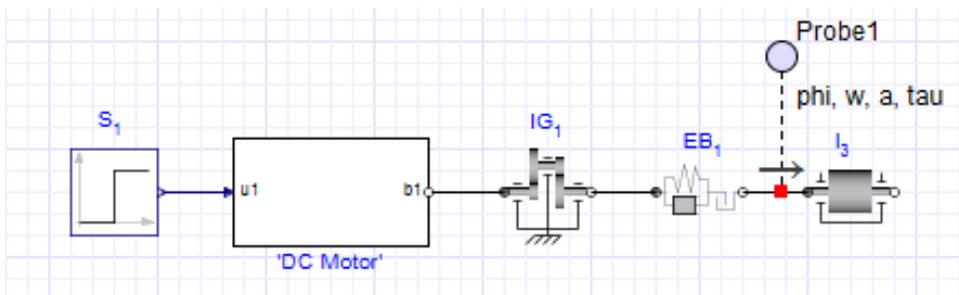
Grouping the DC Motor Components into a Subsystem

To group the DC motor components:

1. Draw a box around the electrical components and the first inertia component by dragging your mouse over these components.



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** (🏠) in the **Model Workspace 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:

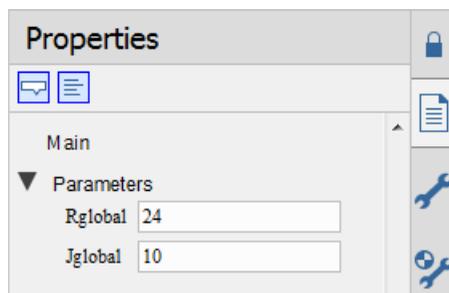
1. Click **Main** (🏠) in the **Model Workspace Toolbar** to browse to the top level of the model.
2. From the **Model Workspace Toolbar**, click **Parameters** (⚙️) to switch to the parameter editor view.

3. In the first row of the **Main subsystem default settings** table, enter **Rglobal** in the **Name** field 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, enter **Jglobal** in the **Name** field 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
<input type="text" value="Rglobal"/>	Real	<input type="text" value="24"/>		<input type="text" value="Global resistance value"/>
<input type="text" value="Jglobal"/>	Real	<input type="text" value="10"/>		<input type="text" value="Global moment of inertia value"/>
<input type="text"/>				

7. Click **Diagram View** (📐) in the **Model Workspace Toolbar** to return to the model diagram. The new **Rglobal** and **Jglobal** parameters appear in the **Properties** tab (📄). You can now assign these parameter values to other components in your model.



8. In the **Model Workspace**, select the Inertia component **I₃**.
9. In the **Properties** tab (📄), 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. Double-click the '**DC Motor**' subsystem.
11. In the **Model Workspace**, select the **EMF₁** component.
12. In the **Properties** tab (📄), in the **Value** field for **k**, the transformation coefficient, enter **Rglobal·Jglobal** and press **Enter**.

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

13. Similarly, in the **Properties** tab (📄) for the **R₁** component, in the **Value** field for the resistance parameter, enter **Rglobal** and press **Enter**.

14. Click **Main** (🏠) to browse to the top level of your model.

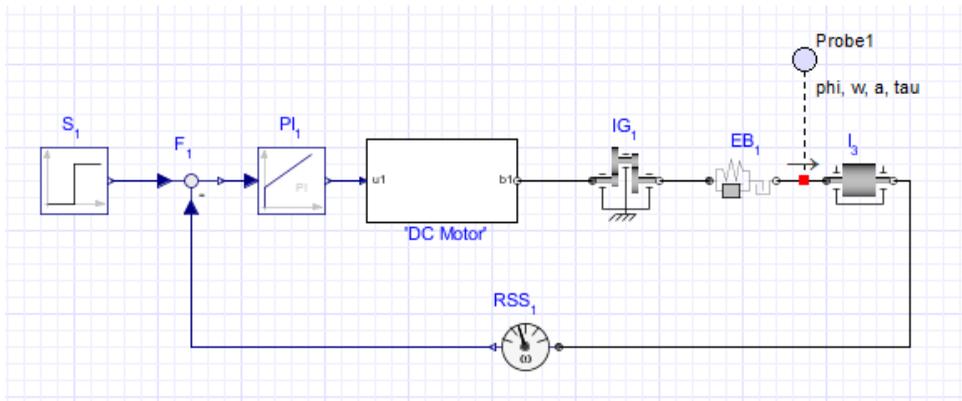
15. Save the model as **DC_Motor2.msim**.

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:

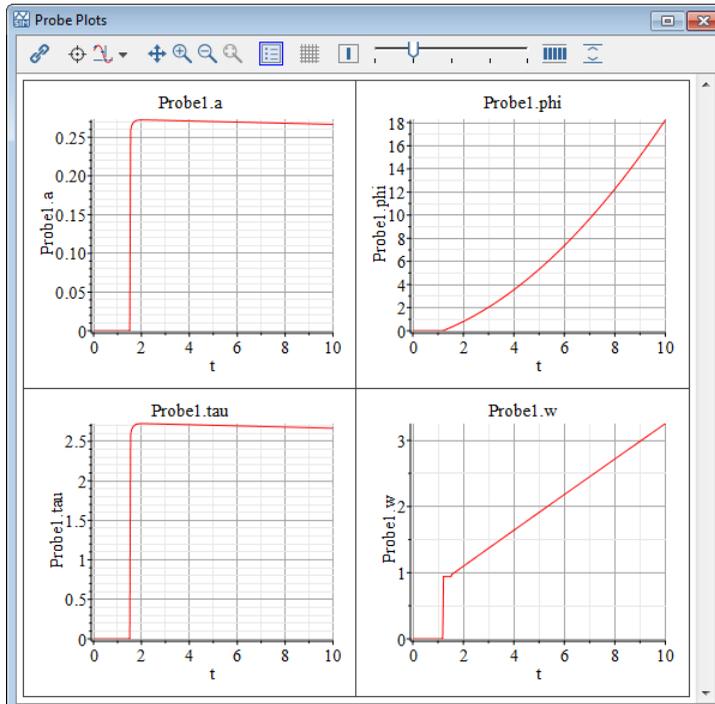
1. Under the **Library Components** tab (🔍), browse to the **1-D Mechanical > Rotational > Sensors** menu and then add the **Rotational Speed Sensor** component to the **Model Workspace** and place it below the gearbox components.
2. Right-click (Control-click for Mac) the **Rotational Speed Sensor** 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.



Tip: 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 **Properties** 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, from the **Help** menu, select **Examples > User's Guide Examples > Chapter 6**, and then select the **DC Motor Subsystem 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:

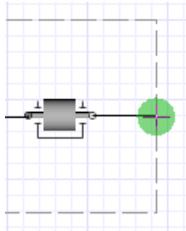
- Build a cable tension controller model
- Specify component properties
- Simulate the cable tension controller model

Building a Cable Tension Controller Model

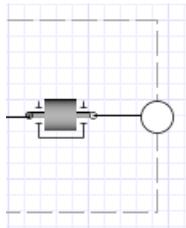
In this example, you will build the cable tension controller model using a combination of 1-D mechanical rotational and translational components. You will also group components into a **Gear** subsystem and add subsystem ports.

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 the **Rotational Speed Sensor** component and its connection lines.
 4. Select the **Ideal Gear**, **Elasto-Backlash**, and **Inertia** components and group them into a subsystem called **Gear Components**.
 5. From the **Library Components** tab () , add the following components to the **Model Workspace**:
 - From the **1-D Mechanical > Rotational > Bearings and Gears** menu, add the **Ideal Gear R 2 T** 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 Gear R 2 T** component.
 - From the **1-D Mechanical > Translational > Springs and Dampers** menu, add the **Spring** component and place it to the right of the **Force Sensor** component.
 - From the **1-D Mechanical > Translational > Common** menu, add the **Fixed** component and place it to the right of the **Spring** component.
 6. Right-click (**Control**-click for Mac) the **Fixed** component in the **Model Workspace** and select **Rotate Counterclockwise**.
 7. Delete the **Step** source and replace it with a **Constant** source from the **Library > 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.

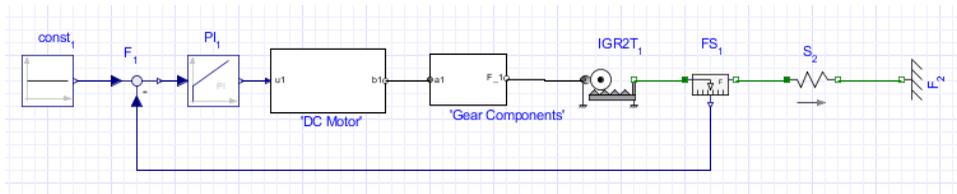


10. Click the line once. The subsystem port is added to the line.



11. Click **Main** (🏠) in the **Model Workspace 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. In the **Properties** tab (📄), specify the following parameter values for the subsystem components:
 - For the **Ideal Gear** component, change **r** to **0.01**.
 - For the **Inertia** component, change **J** to **0.1kg·m²**.
3. Click **Main** (🏠) in the **Model Workspace Toolbar** to browse to the top level of the model.

4. Specify the following parameter values for the other components:
 - For the **Spring** component, in the **c** field, change the spring constant value to $2110 \cdot 10^9 \frac{N}{m}$.
 - For the **PI** controller, change the **T** value to **0.1s**.
 - 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 **Properties** 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**.

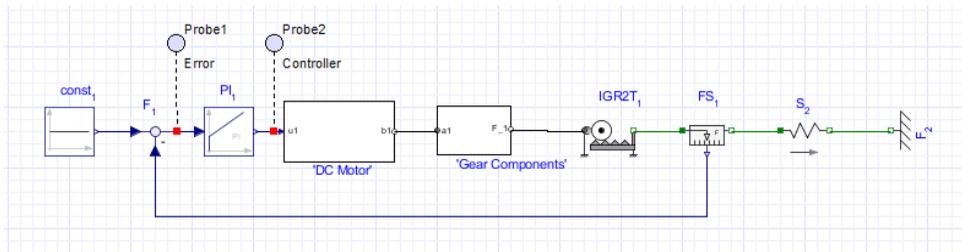
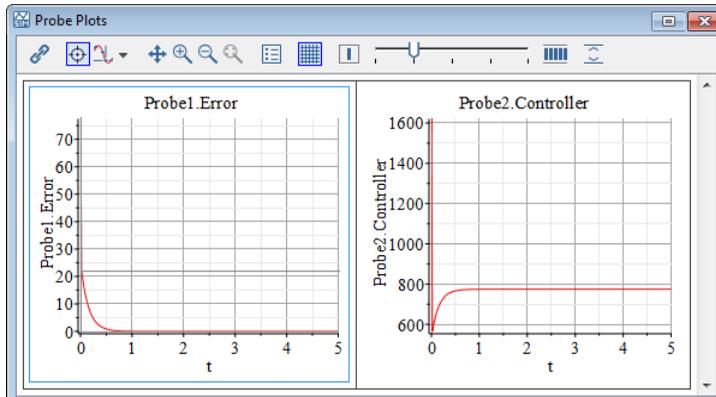


Figure 6.1: Cable Tension Controller

6. Click a blank area in the **Model Workspace**.
7. In the **Settings** tab (🔧), specify the following parameters:
 - Set the simulation duration time, t_d , to **5 S**.
 - Select **Variable** from the **Solver Type** drop-down menu.
 - Select **Rosenbrock (stiff)** from the **Solver** drop-down menu.
8. Click the **Run Simulation** (▶) in the **Main Toolbar**.

9. Click **Show Simulation Results** (). The following graphs appear in the Analysis window.



10. Save the file.

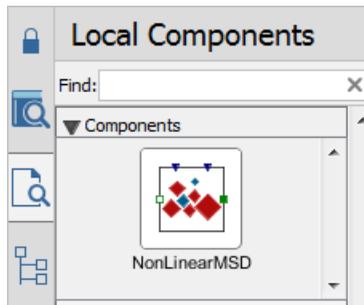
6.3 Tutorial 3: Modeling a Nonlinear Damper

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

This example uses the Nonlinear Spring Damper Custom Component created in *Example: Creating a Nonlinear Spring-Damper Custom Component (page 83)* in Chapter 3.



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 (.xlsx) 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.

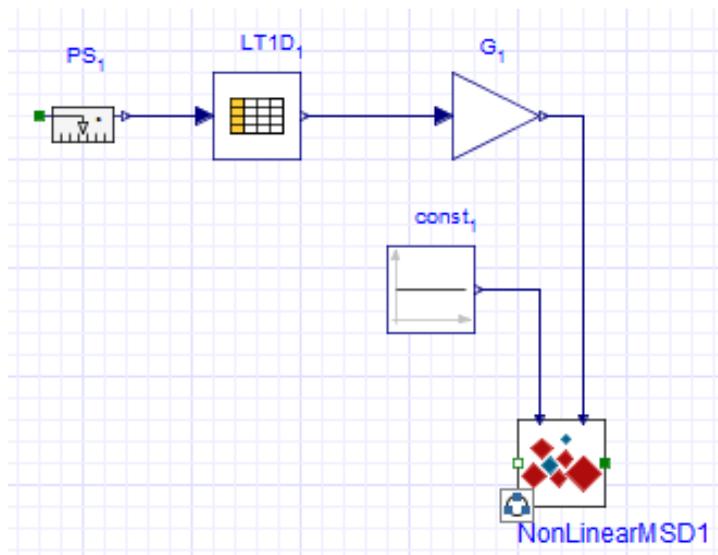
2. Save the file as **DamperCurve.xlsx** or **DamperCurve.csv**.
3. In MapleSim, open the **NonlinearSpringDamper.msimsim** model you created in *Example: Creating a Nonlinear Spring-Damper Custom Component* (page 83) in Chapter 3.
4. Select the **Attachment Files** tab ()
5. Right-click (**Control-click** for Mac) **Data Sets** and select **Attach File**.
6. 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. Select the **Local Components** tab () , and then drag the **NonLinearMSD** custom component into the **Model Workspace**.
2. Select the **Library Components** tab () , and then 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 **Lookup Table 1 D** 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 **Lookup Table 1 D** component.
3. Connect the components as shown in the following figure.



4. Add the following components to the **Model Workspace**:

- From the **1-D Mechanical > Translational > Common** menu, add **Mass** and **Force** components and place them to the left of the **Position Sensor** component.
- From the same menu, add a **Fixed** component, place it to the right of the **NonLinearMSD** component, and then rotate it **counterclockwise**.
- From the **Signal Blocks > Sources > Real** menu, add a **Step** source.

5. Connect the components as shown in the following figure.

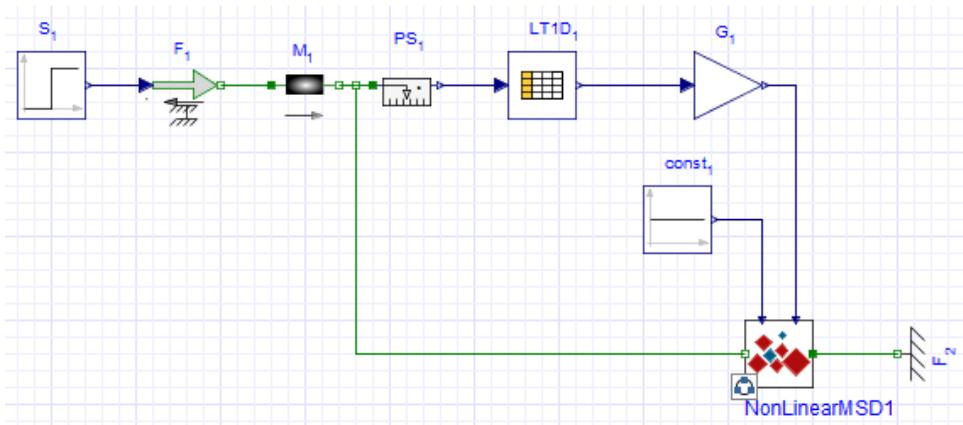
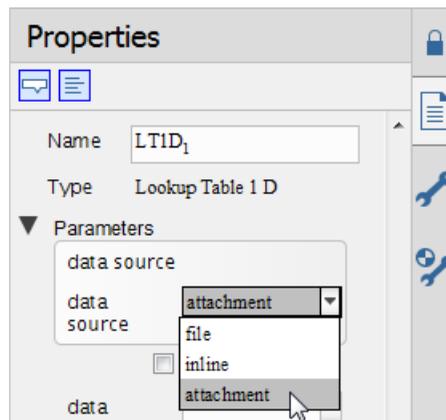


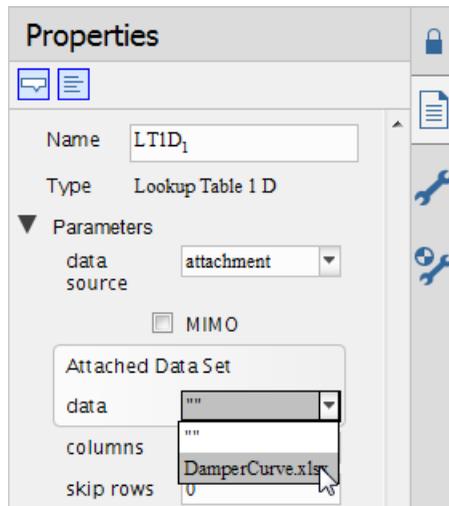
Figure 6.2: Nonlinear Damper Model

6. In the **Model Workspace**, select the **Lookup Table 1 D** component.

7. Under the **Properties** tab () , select **attachment** from the **data source** list.



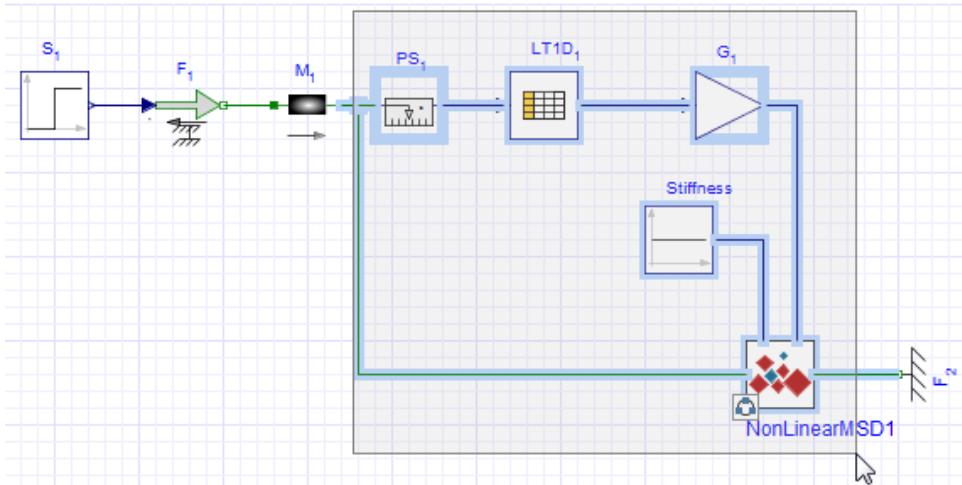
8. Select the attachment you created (either **DamperCurve.xlsx** or **DamperCurve.csv**) from the **data** list.



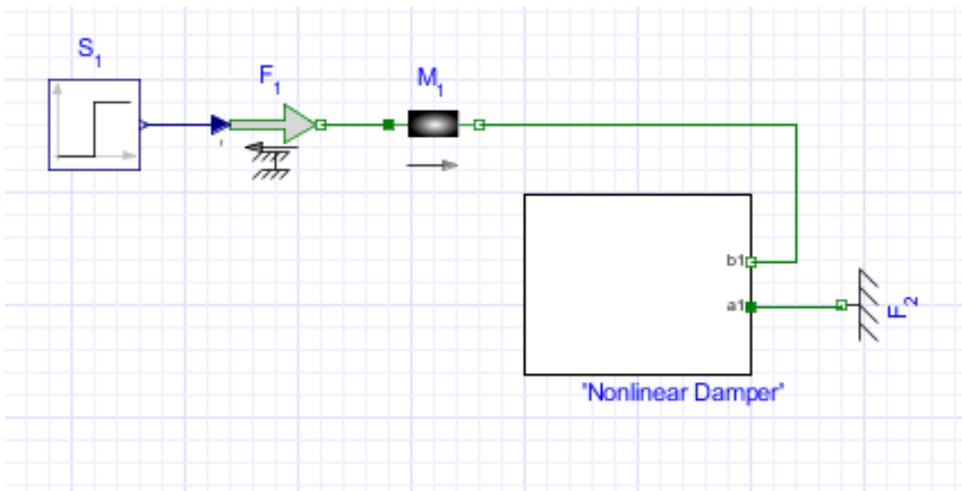
9. In the **Model Workspace**, select the **Constant** component.
10. In the **Properties** tab, in the **Name** field, change the component name to **Stiffness**.



11. Select the **Step** component, and then set **height** to **100**.
12. Select the **Mass** component, and then change the mass, **m**, to **100kg**.
13. Draw a box around all of the components in the nonlinear damper model.



14. Group the selected components into a subsystem called **Nonlinear Damper**. The complete model is shown in the following figure.



Assigning a Parameter to a Subsystem

To assign a parameter to a subsystem:

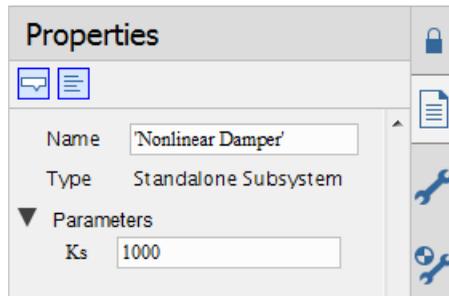
1. In the **Model Workspace**, double-click the '**Nonlinear Damper**' subsystem.
2. In the **Model Workspace Toolbar**, click **Parameters** ().
3. In the first row of the **Standalone Subsystem default settings** table, define a spring constant parameter called **Ks**, and then press **Enter**.

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 **Non-linear Damper** subsystem.

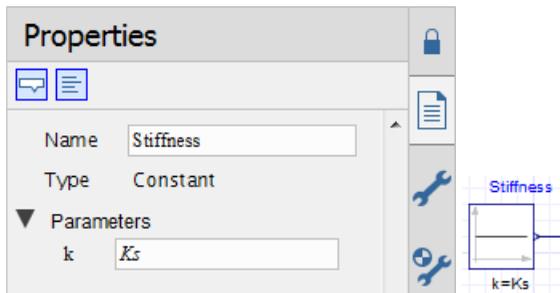
StandaloneSubsystem default settings

Name	Type	Default Value	Default Units	Description
Ks	Real	1000		Spring constant

5. In the **Model Workspace Toolbar**, click **Diagram View** (). The **Ks** parameter appears as a field in the **Properties** tab with the defined default value.



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 also inherits that change.

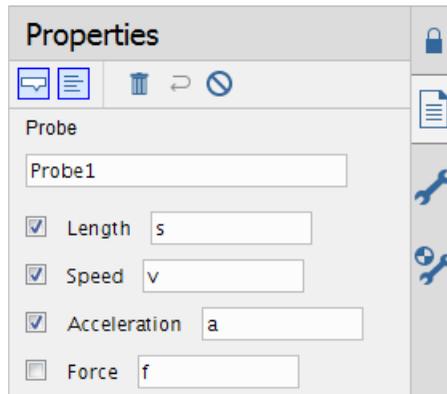


Simulating the Nonlinear Damper with Linear Spring Model

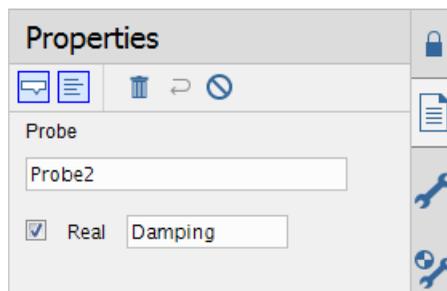
To simulate the nonlinear damper:

1. Click **Main** () in the **Model Workspace Toolbar** to browse to the top level of the model.
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 **Mass** component and the **Nonlinear Damper** subsystem and then click a spot in the workspace to anchor the probe.
4. In the **Model Workspace**, select the probe.
5. Under the **Properties** tab () , select the length, speed, and acceleration quantities.

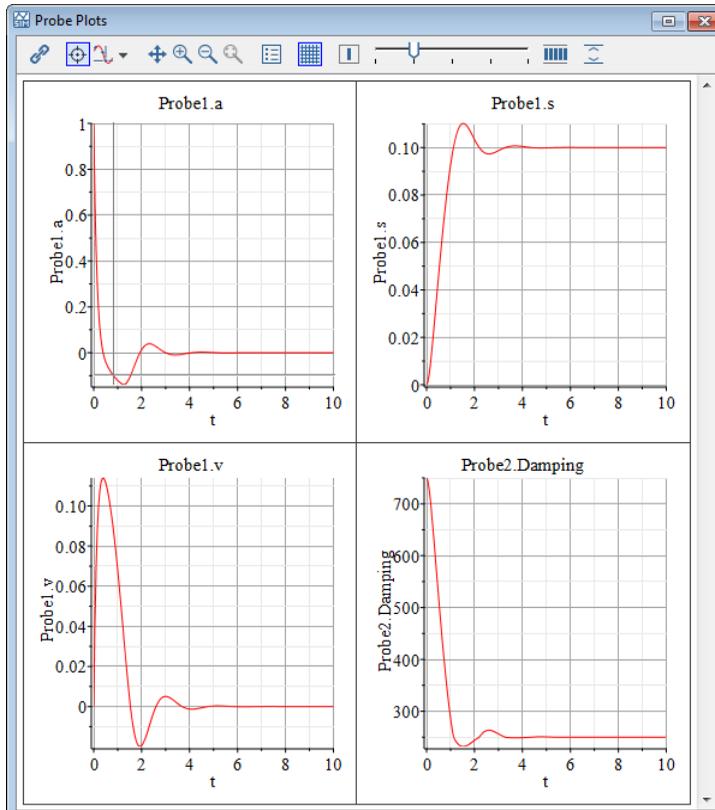


6. Click a blank area in the **Model Workspace**. The length, speed, and acceleration quantities (s, v, a) appear beside the probe.
7. Double-click the '**Nonlinear Damper**' subsystem.
8. Add a probe to the line that connects the **Gain** and the **NonLinearMSD** custom component and then click a spot in the workspace to anchor the probe.
9. In the **Model Workspace**, select the probe.
10. Under the **Properties** tab, select the **Real** quantity and change its name to **Damping**.



11. Under the **Settings** tab () , set the t_d parameter to **10** seconds.
12. Click **Run Simulation** () in the **Main Toolbar**.

13. Click **Show Simulation Results** (). The following graphs appear in the Analysis window.



14. 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.

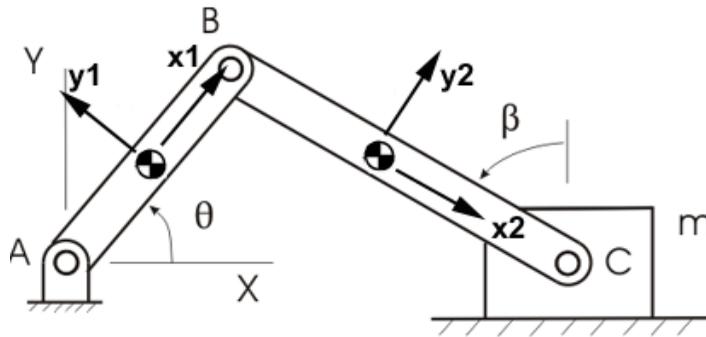


Figure 6.3: Planar Slider-Crank Mechanism

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:

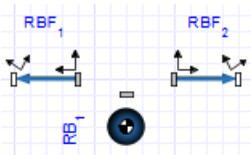
- Create a planar link subsystem
- Define and assign subsystem parameters
- Create the crank and connecting rod elements
- Add the fixed frame, sliding mass, and joint elements to the model
- Specify initial conditions
- Simulate the planar slider-crank mechanism

Creating a Planar Link Subsystem

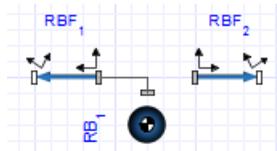
The preceding diagram shows that the slider-crank has two associated planar links: the crank (the link from point A to B) and the connecting rod (the link from B to C). In both cases, these links have their longitudinal axis along their local x -axis (\mathbf{x}_1 and \mathbf{x}_2 , respectively). Therefore, 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:

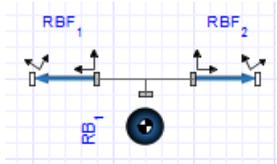
1. Open a new MapleSim document.
2. Under the **Library Components** tab (🔍), browse to the **Multibody > Bodies and Frames** menu, and then add two **Rigid Body Frame** components and a **Rigid Body** component.
3. In the **Model Workspace**, right-click (**Control-click** for Mac) one of the **Rigid Body Frame** components, and then select **Flip Horizontal**.
4. Right-click (**Control-click** for Mac) the **Rigid Body** component, and then select **Rotate Counterclockwise**.
5. Drag the components in the arrangement shown below.

**Notes:**

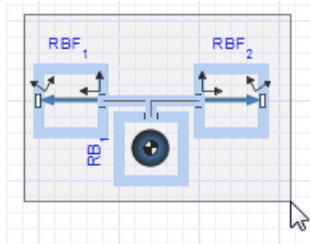
- If you cannot see the labels for your components, from the **View** menu, select **Show Labels**.
 - The labels for your components may differ from the labels in the preceding figure (that is, **RB₁**, **RBF₁**, and **RBF₂**). You can change the labels in your model by selecting the component, and then entering the new label in the **Name** field under the **Properties** 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 **RB₁** component and the right frame of the **RBF₁** component.



7. Draw another connection line between the **RB₁** component and the left frame of the **RBF₂** component.



8. Draw a box around the components by dragging your mouse over them.



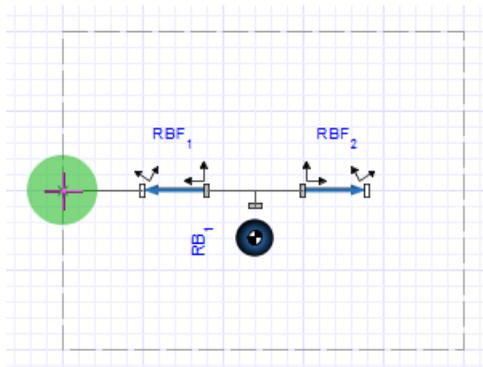
9. From the **Edit** menu, select **Create Subsystem**.

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.

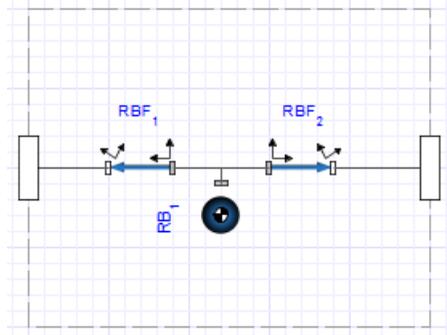
11. Double-click the **Link** subsystem.

12. Click the left frame of the **RBF₁** 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 **RBF₂** 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. If you are not already in the **Link** subsystem, click **Main** (🏠) in the **Model Workspace Toolbar**, and then double-click the **Link** subsystem.
2. In the **Model Workspace Toolbar**, click **Parameters** (⚙️), or from the **Properties** tab (📄) click **Add or Change Parameters**. The **Standalone Subsystem default settings** window appears.
3. In the first row of the **Standalone Subsystem default settings** table, enter **L** in the **Name** field, and then press **Enter**.
4. Specify a default value of **1** and enter **Length** as the description.
5. Scroll to the **RBF₁ component** section.
6. In the **Value** field for $\bar{\mathbf{r}}$, specify a position offset of $\left[-\frac{L}{2}, \mathbf{0}, \mathbf{0}\right]$, and then select **m** from the **Units** drop-down menu. To enter a fraction, use the forward slash key (/).
7. Scroll to the **RBF₂ component** section.
8. In the **Value** field for $\bar{\mathbf{r}}$, specify a position offset of $\left[\frac{L}{2}, \mathbf{0}, \mathbf{0}\right]$, and then select **m** from the **Units** drop-down menu.
9. Click **Diagram View** (🖼️).

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 **Model Workspace Toolbar** to browse to the top level of your model. The **Link** subsystem appears in the **Model Workspace**.
2. Right-click (**Control-click** for Mac) the **Link** subsystem, and then select **Convert to Shared Subsystem**. The **Create Shared Subsystem** window appears. Click **OK**. A **Link** subsystem definition is added to the **Components** palette in the **Local Components** tab (🔍) and the **Link** subsystem in the **Model Workspace** is converted to a shared subsystem.
3. Select the **Link₁** shared subsystem in the **Model Workspace** and in the **Properties** tab (📄), enter **Crank** in the **Name** field.
4. From the **Local Components** tab, drag the **Link** icon to the **Model Workspace**, placing 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 **Properties** tab (📄), change the shared subsystem name to **ConnectingRod**. See the following figure.



7. For the **ConnectingRod**, change the value of the **Length** parameter (**L**) 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. Under the **Library Components** tab (🔍), expand the **Multibody > Bodies and Frames** menu, select the **Fixed Frame** component, and then 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.

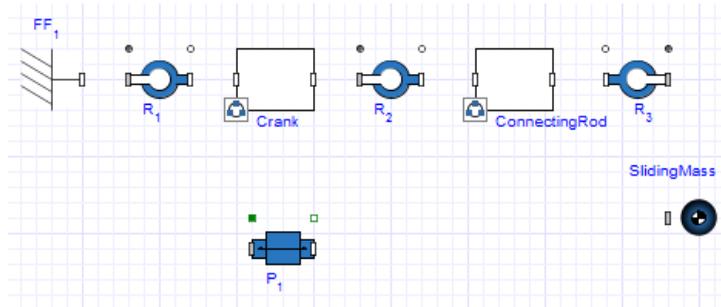
3. Add the following joints:

- 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.

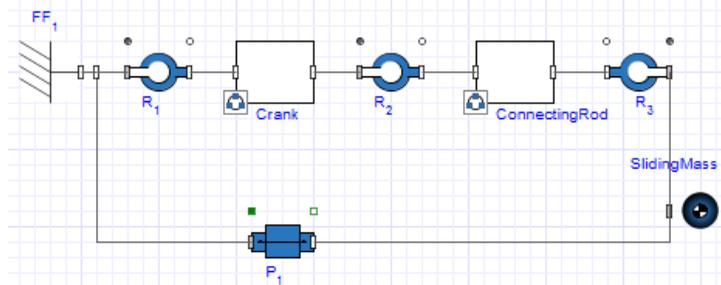
4. Select the **Rigid Body** component in the **Model Workspace** and rename it **SlidingMass**.

5. Right-click (**Control-click** for Mac) the **SlidingMass** component and select **Flip Horizontal**.

6. In the same way, right-click (**Control-click** for Mac) the **Revolute** joint that is located between the connecting rod and the rigid body and select **Flip Horizontal**. See the following figure.



7. Connect the components as shown in the following figure.



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 (\mathbf{R}_1 in the preceding figure), in the θ_0 field, change the initial angle to $\frac{\pi}{4}$ rad.

Tip: To enter π , type **pi**, press **Esc**, and then select the π symbol from the menu.

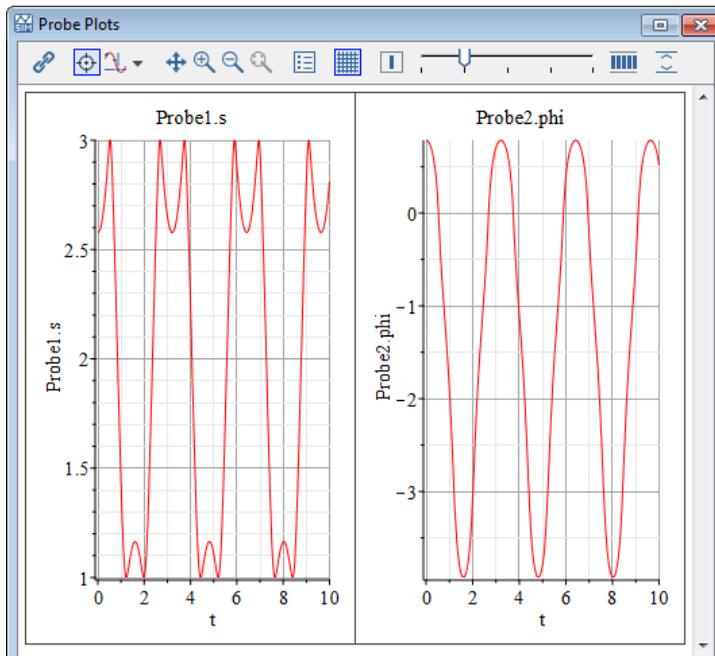
2. From the $\mathbf{IC}_{\theta,\omega}$ drop-down menu, select **Strictly Enforce**.

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 **Properties** 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 \mathbf{R}_1 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**.
9. Click **Show Simulation Results** () . The following graphs appear in the Analysis window.



10. Select the **3-D Playback Window**, and then click **Play** () in the **3-D Toolbar** to see a video of the simulation.

Tip: The quality of the visualization is affected if any open plot windows are behind the **3-D Playback Window**. If you are experiencing playback issues, try moving the **3-D Playback Window** so that it does not overlap a plot window. Alternatively, minimize or close any open plot windows.

11. 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
- 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 a custom port using the Custom Port app

For a description of the Custom Component Template see *Creating Custom Modeling Components (page 71)*.

Example: Modeling a Temperature Dependent Resistor

In this tutorial example, you will create a model of a circuit using a custom component for a temperature dependent resistor whose resistance varies as

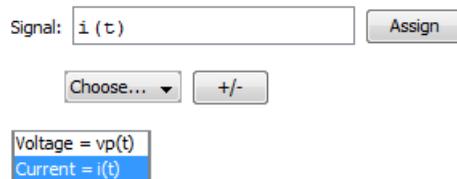
$$r(t) = RO \cdot \left(1 + \frac{(T(t) - TO)^2}{Tk^2} \right), \text{ where } RO, TO, \text{ and } Tk \text{ are parameters.}$$

To create the custom component:

1. Start a new MapleSim model and then select the **Add Apps or Templates** tab ().
2. Double-click on the **Custom Component** entry in the **Templates** palette.
3. Enter **TempResistor** and then click **Create Attachment** () . The Maple **Custom Component** template is loaded.
4. In the **Define Equations** section, enter in the following system equations to define your component. Press **Enter** at the end of the line.

$$eq := \left[v(t) = vp(t) - vn(t), r(t) = R0 \cdot \left(1 + \frac{(T(t) - T0)^2}{Tk^2} \right), \right. \\ \left. v(t) = i(t) \cdot r(t), qdot(t) = i(t) \cdot v(t) \right];$$

5. In the **Configuration** section, select **Parameters**, and then click **Refresh All**.
6. Select **Ports** in the **Configuration** section.
7. Click **Clear All Ports**.
8. Click **Add Port**. A new port appears on the left side. This will become the positive electrical pin.
9. From the **Type** drop-down list, select **Electrical**.
10. Click the **Style** radio button labeled +.
11. In the list box select **Voltage = unassigned** and then select **vp(t)** in the drop-down list under **Signal**. This assigns the across variable of the port.
12. In the list box select **Current = unassigned** and then select **i(t)** in the drop-down list under **Signal**. This assigns the through variable of the port.



13. Click **Add Port**. A new port appears on the right side. This will become the negative electrical pin.
14. From the **Type** drop-down box, select **Electrical**.
15. Click the **Style** radio button labeled -. This changes the port style to an unfilled box.
16. In the list box select **Voltage = unassigned** and then select **vn(t)** in the drop-down list under **Signal**. This assigns the across variable of the port.
17. In the list box select **Current = unassigned**, select **i(t)** in the drop-down list under **Signal**, and then click the +/- button to negate the sign of the signal.



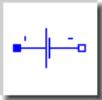
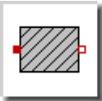
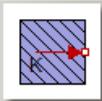
18. Click **Add Port**. A new port appears on the top edge. Drag the port to the center of the bottom edge.
19. From the **Type** drop-down box, select **Thermal**.
20. Click the **Style** radio button labeled **a**.
21. Select **T(t)** for the **Temperature** variable and **qdot(t)** for the **Heat Flow Rate** variable.
22. From the **Icon** list, select **Use default**.
23. In the **Configuration** section, select **Variables**, and then click **Refresh All**. This updates entries in the **Type** column of the **Variables** table.
24. In the **Variables** table, scroll down to see the **r(t)** and **v(t)** variables. The types for **r(t)** and **v(t)** are still listed as **real**.
25. Change the **Type** entries for **r(t)** and **v(t)** to **Resistance** and **Voltage**, respectively, and then click **Refresh All** to ensure these are accepted (if not, they revert to **real**).
26. In the **Configuration** section, select **Parameters**.
27. In the **Parameters** table, enter **Resistance** for the type of **R0**, and **Thermodynamic Temperature** for both **T0** and **Tk**. Assign the **Default** for **T0** to **300**. (The units are Kelvin.) Click **Refresh All** to ensure these are accepted.

Note: You can also find the type names used in the preceding step by expanding the Type Reference section and searching for the appropriate Domain and Type.

28. In the **Configuration** section, select **Dimensional Analysis**, and then click **Check Dimensions**. The message "*no issues found*" should appear in the text area.
29. In the **Component Generation** section, change the **Name** to **TempResistor**.
30. Click **Generate MapleSim Component** to create your component and to bring you back into the MapleSim environment. The custom component now appears under the **Local Components** tab (), in the **Components** palette.
31. Drag the custom component into your model area.
32. Create the model shown in **Figure 6.4** with the components and settings specified in **Table 6.1**.

Note: When you build the model, make sure to attach the probe to the custom component to measure the electrical and thermal quantities.

Table 6.1: Temperature Dependent Resistor Components

Component	Symbol	Component Location	Required Settings
TempResistor Custom Component		Local Components > Components	Use default settings
Constant Voltage		Library Components > Electrical > Analog > Common	Use default settings
Ground		Library Components > Electrical > Analog > Common	Use default settings
Thermal Resistor		Library Components > Thermal > Heat Transfer Components	Set $R = 10 \text{ K/W}$
Heat Capacitor		Library Components > Thermal > Heat Transfer Components	Set $C = 0.1 \text{ J/K}$ Set $T_0 = 280\text{K}$
Fixed Temperature		Library Components > Thermal > Sources	Set $T = 298\text{K}$

33. Right-click (**Control**-click for Mac) on the TempResistor custom component, select **Attach probe**, and then click on the workspace to place the probe. See the following figure.

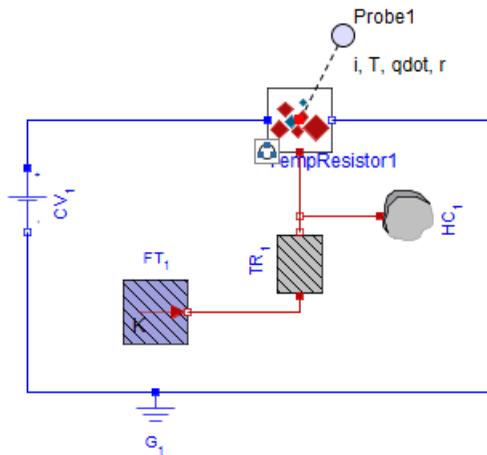


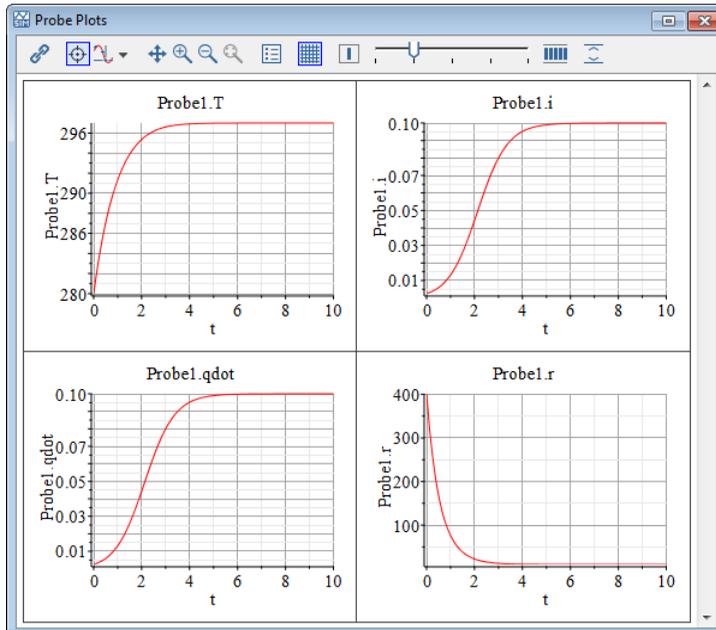
Figure 6.4: Temperature Dependent Resistor

34. Select the probe, select the **Properties** tab () , and then select the following quantities:

- Current
- ThermoDynamicTemperature
- HeatFlowRate
- Resistance

35. Click **Run Simulation** () in the **Main Toolbar**.

36. Click **Show Simulation Results** () . The following graphs appear in the Analysis window.



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.

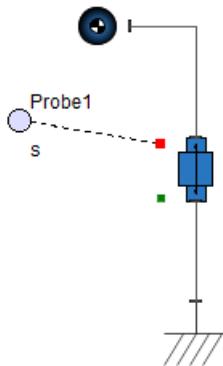


Figure 6.5: Falling Ball

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 1-D 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.

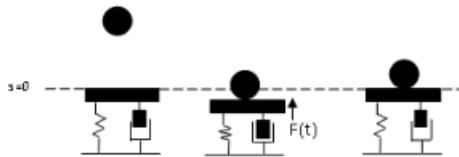


Figure 6.6: Bouncing Ball Dynamics

where

$$f(x) = K s(t) + B \frac{ds(t)}{dt}, \text{ if } s(t) < 0$$

$$f(x) = 0 \text{ if } s(t) \geq 0$$

To create the custom component:

1. Start a new MapleSim model and then select the **Add Apps or Templates** tab (🔧).
2. Double-click on the **Custom Component** entry in the **Templates** palette.
3. Enter **contact** for the name of the attachment and then click **Create Attachment** (✓). The Maple **Custom Component** template is loaded.
4. In the **Define Equations** area, enter in the following equation, parameters, and initial conditions to define your custom component. Press **Enter** at the end of the line.

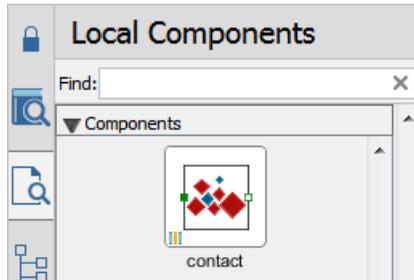
$$\begin{aligned} eq := & \left[s(t) = sa(t) - sb(t), 0 = Fa(t) + Fb(t), Fa(t) \right. \\ & \left. = \text{piecewise}\left(s(t) < 0, K \cdot s(t) + B \cdot \frac{d}{dt} s(t), 0\right) \right]; \end{aligned}$$

5. In the **Configuration** section, select **Ports**.

6. Click **Refresh All**.
7. Click **Clear All Ports**.
8. Click **Add Port**.
9. Make the left port a **Translational** type. Use style **a** (the default). Associate its **Position** variable with **sa(t)** and its **Force** variable with **Fa(t)**.
10. Click **Add Port** to add a second port to the right side.
11. Make the right port a **Translational** type. Select style **b**. Associate its **Position** variable with **sb(t)** and its **Force** variable with **Fb(t)**.
12. From the **Icon** list, select **Use default**.
13. Click **Refresh All**.
14. In the **Configuration** section, select **Variables**.
15. Click **Refresh All**.
16. In the table, change the **Type** of $s(t)$ to **Position**, and then click **Refresh All**.
17. In the Configuration section, select **Parameters**, and then click **Refresh All**.
18. Change the **Default** value for B to 10.
19. Change the **Default** value for K to 1000, and change its **Type** to **real**.
20. Click **Refresh All**.
21. In the **Configuration** section, select **Dimensional Analysis**, and then click **Check Dimensions**. The following expression appears:
$$B \frac{d}{dt} s(t) \frac{\text{m}^3 \text{kg}}{\text{s}^2 \text{A}} + K s(t) \text{ m}.$$

This indicates that the units (dimensions) are not consistent; the units associated with each element in the sum are displayed. You can choose to ignore the inconsistency, and the model will work as desired because MapleSim's engine does not use units. Checking dimensional consistency, however, is an easy way to avoid simple algebraic errors. To eliminate the inconsistency, the proper types must be added to parameters B and K.
22. Select **Parameters**, and then enter **Force/Velocity** and **Force/Distance** for the types for B and K, respectively.
23. Click **Refresh All**. The types update to the equivalent dimensional types, **TranslationalDampingConstant** and **TranslationalSpringConstant**.
24. Select **Dimensional Analysis**, and then click **Check Dimensions**. "*No issues found*" will appear.
25. In the **Component Generation** section, change the **Name** to **contact**.

26. Click **Generate MapleSim Component** to create your component and to bring you back into the MapleSim environment. The custom component now appears in the **Components** palette in the **Local Components** tab ()



27. 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.

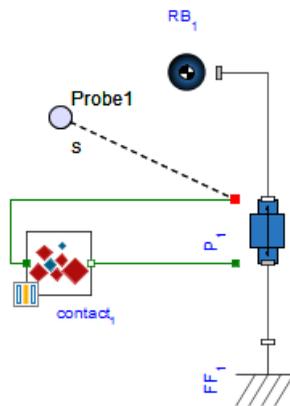
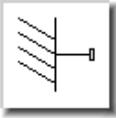
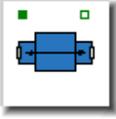


Figure 6.7: Bouncing Ball

Table 6.2: Bouncing Ball Multibody Components

Component	Symbol	Component Location	Required Settings
Custom Component		Local Components > Components	Use default settings
Rigid Body		Library Components > Multibody > Bodies and Frames	Use default settings
Fixed Frame		Library Components > Multibody > Bodies and Frames	Use default settings
Prismatic		Library Components > Multibody > Joints and Motions	<p>For the prismatic joint to translate along the y direction, set</p> $\hat{\mathbf{e}}_1 \text{ to } [0, 1, 0]$ <p>For the prismatic joint to try enforcing your translational initial conditions, set</p> $\mathbf{IC}_{s,v} \text{ to } \mathbf{Treat as Guess}$ <p>For an initial displacement, the prismatic joint requires a value > 0. Set</p> $s_0 \text{ to } 10m$

28. Click **Run Simulation** () in the **Main Toolbar**.

29. When the simulation is complete, the Simulation Results tab of the Analysis window displays the probe plot.

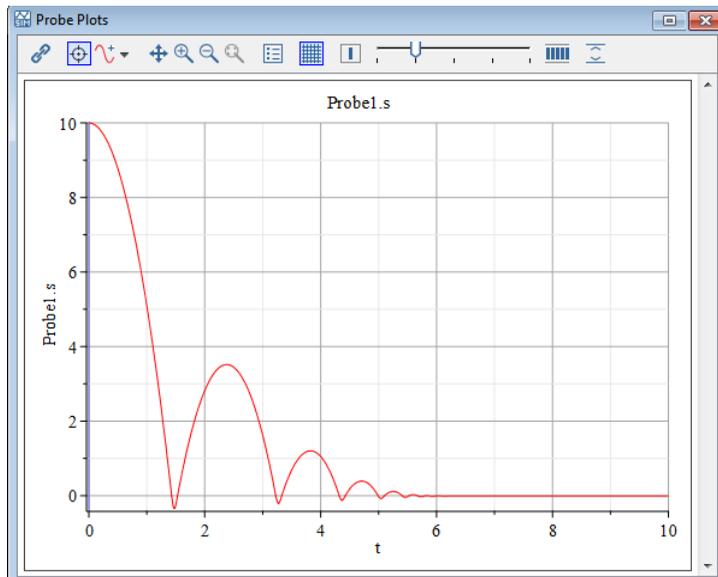


Figure 6.8: Bouncing Ball Result

30. To play the animation of the bouncing ball, select the **3-D Playback Window** in the Simulation Results tab, and then click **Play** (▶) in the **Playback Toolbar**.
31. Use the navigation controls on the toolbar to pan, zoom, or move the camera around your model to find a good view. For more information, see 3-D Toolbar.
32. To create a smoother animation, click the **Change 3-D settings** icon (⚙️) and select **Interpolate Intermediate Frames** from the drop-down menu. For more information, see Animating a 3-D Model with Interpolated Frames.

Example: Custom Ports

In this tutorial example you will create a pair of custom ports with causal signals, power, and velocity, that are generated by a dummy vehicle model and combined by a monitor block that computes the force, using the relation $f(t) = \frac{p(t)}{v(t)}$. For the dummy vehicle model we will assume the power is constant, $p(t) = P$, and the velocity is given by $v(t) = v0 + a \cdot t$, where P , $v0$, and a are parameters.

Create Custom Ports

We need to create two custom ports, an output port named *bus_out* and an input port named *bus_in*.

1. Start a new MapleSim model and then select the **Add Apps or Templates** tab ()
2. Double-click on the **Custom Port** entry in the **Apps** palette. The Maple **Custom Port Definition** app is loaded.
3. In the **Configuration** section, select **Signals**, and then click **Add Signal** to add a new signal.
4. Enter *p* as the **name** of the signal, enter *Power* as the **type**, select the **output** radio button, and enter *Vehicle power* in the **desc** column.
Tip: You can find a type by browsing the Type Reference section. If you click the name of a type, it is copied to the **type** field.
5. Click **Add Signal** again to add a second signal.
6. Enter *v* as the signal **name**, enter *Velocity* as the **type**, select the **output** radio button, and enter *Vehicle velocity* in the **desc** column.
7. In the **Generate Port** section, enter *bus_out* in the **Name** field, *vehicle bus port* as a **Description**, and select the **output** radio button.
8. Click **Generate MapleSim Port** to create the output port. You will automatically return to your MapleSim model.
9. Return to the same app to create the input port.
10. In the **Signals** section, for both signals, select the **input** radio button and click **Apply**.
11. In the **Generate Port** section, change *bus_out* to *bus_in* and click the **input** style radio button.
12. Click **Generate MapleSim Port** to create the input port. You will automatically return to your MapleSim model.
The two custom ports you have defined appear in the **Components** palette of the **Local Components** tab. See **Figure 6.9**.

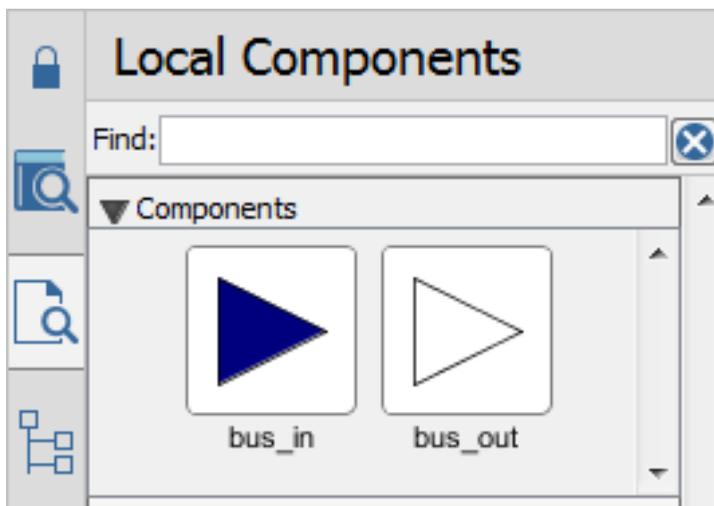


Figure 6.9: Custom Ports

Create Engine Model

We will first create the dummy vehicle model.

1. In MapleSim, select the **Add Apps or Templates** tab (.
2. Double-click on the **Custom Component** entry in the **Templates** palette, and then click **Create Attachment** () . The Maple **Custom Component** template is loaded.
3. In the **Define Equations** section, enter the following system equations to define your component, and then press **Enter** at the end of the line.

$$eq := [p(t) = P, v(t) = v0 + a \cdot t]$$

4. In the **Configuration** section, select **Parameters**, and then click **Refresh All**.
5. Change the **Type** fields for P , a , and $v0$ to *Power*, *Acceleration*, and *Velocity*, respectively.
6. In the Configuration section, select **Ports**.
7. Click **Clear All Ports**.
8. Click **Add Port**. A new port appears on the left edge. Drag it to the right edge.
9. From the **Type** drop-down menu, select **Custom**. (It is at the bottom of the list.)
10. In the text area below the **Type** drop-down menu, enter **bus_out**.
11. Select the **out** style radio button.

12. Click **Apply Custom**.

13. From the **Icon** list, select **Use default**.

14. Using the drop-down menu and list box at the bottom of this section, assign the power signal to $p(t)$ and the velocity signal to $v(t)$. **Figure 6.10** shows the completed Ports section of the template.

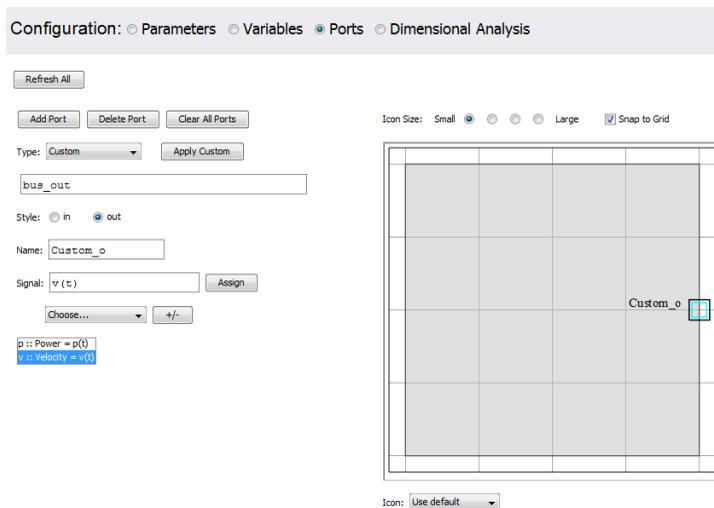


Figure 6.10: Using a Custom Port

15. In the Configuration section, select **Dimensional Analysis**, and then click **Check Dimensions**. The message *no issues found* should appear.

16. In the **Component Generation** section, change the name to *engine*, and then click **Generate MapleSim Component**.

The engine component appear in the **Components** palette of the **Local Components** tab.

Create Monitor Model

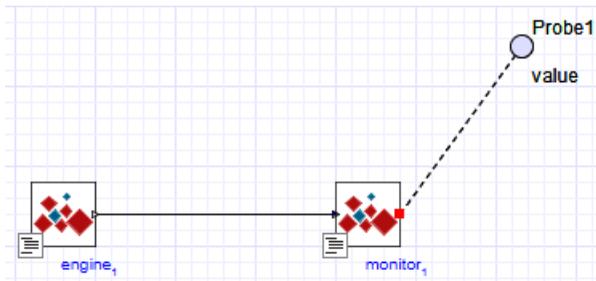
1. In MapleSim, select the **Add Apps or Templates** tab (.
2. Double-click on the **Custom Component** entry in the **Templates** palette, and then click **Create Attachment** (). The Maple **Custom Component** template is loaded.
3. In the **Define Equations** section, enter in the following system equation to define your component, and then press **Enter** at the end of the line.

$$eq := \left[f(t) = \frac{p(t)}{v(t)} \right]$$

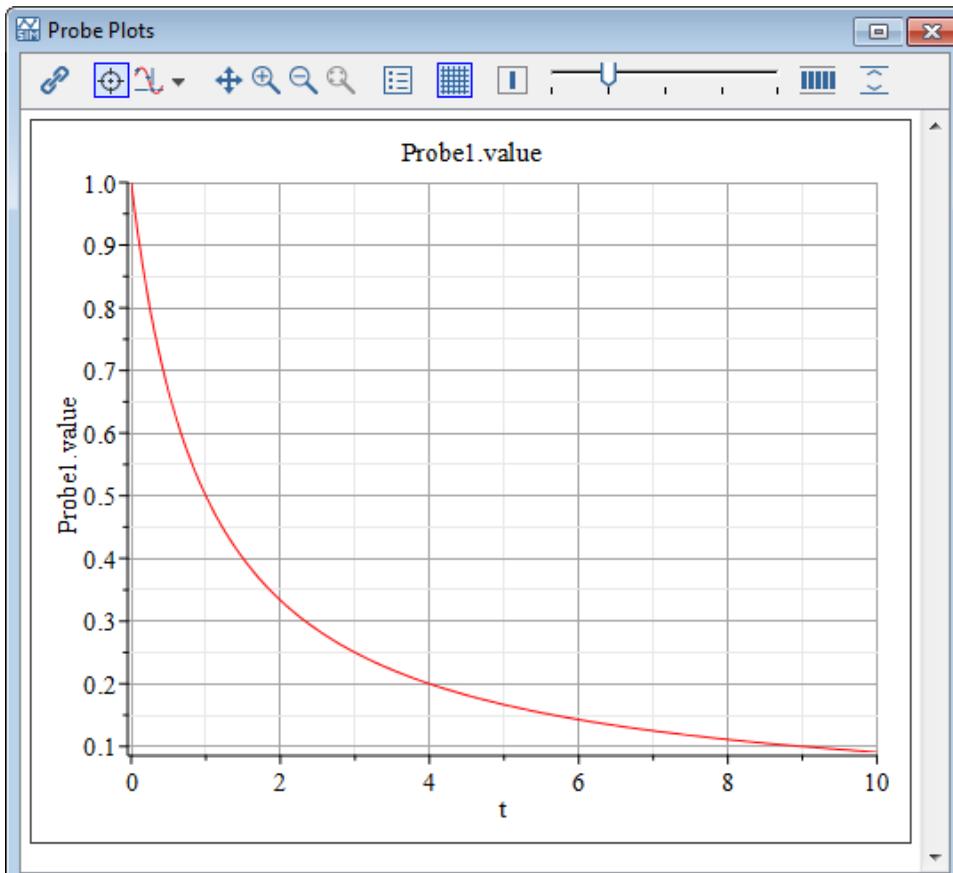
4. In the **Configuration** section, select **Parameters**, and then click **Refresh All**.
5. Select **Ports**, and then click **Clear All Ports**.
6. Click **Add Port**. A new port appears on the left edge. This will be the bus input.
7. From the **Type** drop-down menu, select **Custom** (at the bottom of the list).
8. In the text area below the **Type** drop-down menu, enter the custom type name: *bus_in*.
9. Click **Apply Custom**.
10. Using the drop-down menu and list box at the bottom of this section, assign the power signal to $p(t)$ and the velocity signal to $v(t)$.
11. Click **Add Port**. A new port appears on the right edge. This will be the computed force output.
12. From the **Type** drop-down menu, select **Real Signal**. Click the **out** style radio button. Assign the value signal to $f(t)$.
13. From the **Icon** list, select **Use default**.
14. Click **Refresh All**.
15. In the **Configuration** section, select **Variables**, and then click **Refresh All**.
16. Enter **Force** for the **Type** field of signal $f(t)$.
17. Select **Dimensional Analysis**, and then click **Check Dimensions**. The message $real_o(t) = f(t)$ N should appear. It indicates that the output port, $real_o(t)$, is a real signal but is equated to $f(t)$, which has units of force (Newtons).
18. In the **Component Generation** section, change the name to *monitor*.
19. Click **Generate MapleSim Component**.

Complete Model

1. In MapleSim, drag the **engine** and **monitor** components from the **Components** palette in the **Local Components** tab () into the workspace.
2. Connect the output of the **engine** block to the input of the **monitor** block. Then, attach a probe to the output of the **monitor** block as shown below.



3. Run the simulation. The Simulation Results tab in the Analysis window shows the following probe plot:



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 provides a sample advanced application.

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

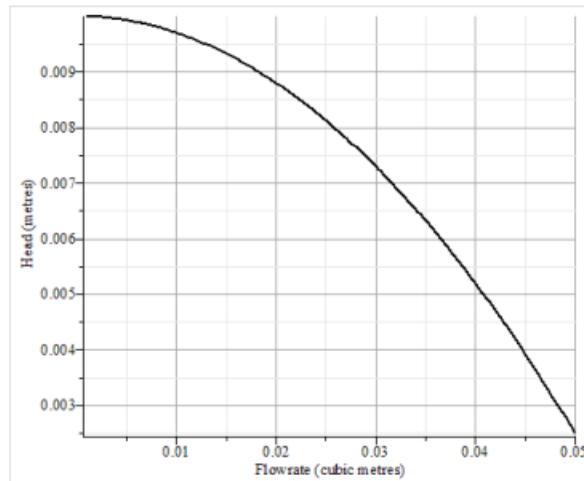


Figure 6.11: Centrifugal Pump Head Flow Rate Curve

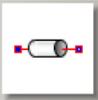
Table 6.3: Centrifugal Pump Data

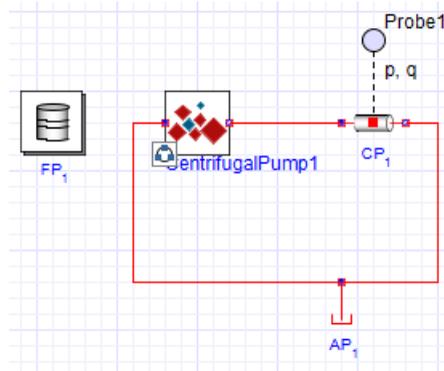
Flow Rate (cubic meters)	Pressure Head (meters)
0.01	0.0098
0.02	0.00874
0.03	0.00725
0.04	0.005
0.05	0.0025

Table 6.4: Circular Pipe Parameters

Symbol	Description	Value
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	Maximum Reynolds number in laminar regime	2000
ReT	Minimum Reynolds number in turbulent regime	4000

Table 6.5: Centrifugal Pump Components

Component	Symbol	Component Location	Its Use	Value
Tank		Library Components > Hydraulics > 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		Library Components > Hydraulics > 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: <ul style="list-style-type: none"> • rhoFluid: liquid density • K: Bulk Modulus defines the fluid compressibility • nuFluid: Kinematic Viscosity defined as dynamic viscosity divided by liquid density 	Use default settings : rhoFluid : $850 \frac{kg}{m^3}$ K : 8000 bar nuFluid : $0.000018 \frac{m^2}{s}$
Circular Pipe		Library Components > Hydraulics > Restrictions	The circular pipe defines a pressure drop in the hydraulic line. The pressure drop is given by the Darcy equation.	Use default settings (see Table 6.4)
Custom Component		Local Components > Components	This custom component defines hydraulic pressure and flow rate properties for the hydraulic line.	User defined

**Figure 6.12: Centrifugal Pump Custom Component**

To create the custom component:

1. Start a new MapleSim model and then select the **Add Apps or Templates** tab ()
2. Double-click on the **Custom Component** entry in the **Templates** palette.
3. Click **Create Attachment** (). The Maple **Custom Component** template is loaded.
4. In the **Define Equations** section, place your cursor in the first Maple command line (that is, the one containing *eq*), and then insert two document blocks (from the main menu, select **Edit > Document Blocks > Create Document Block** twice).
5. Replace the contents of the first line with the following Maple command, and then press **Enter** at the end of the line. This command places the values from **Table 6.3** into the list *L*.

$$L := [[0.01, 0.0098], [0.02, 0.00874], [0.03, 0.00725], [0.04, 0.005], [0.05, 0.0025]]$$

6. Enter the following Maple command in the second document block, and then press **Enter** at the end of the line. This command fits a quadratic curve to the data points.

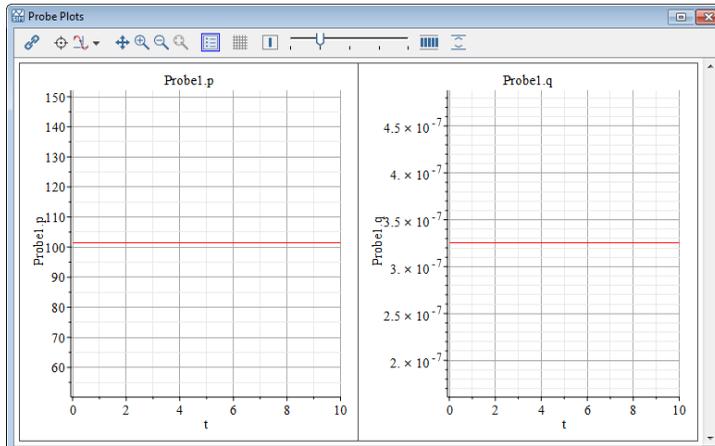
$$f := \text{unapply}(\text{CurveFitting}[\text{LeastSquares}](L, x, \text{curve} = a \cdot x^2 + b \cdot x + c), x)$$

7. Enter the following system of equations, parameters, and initial conditions to define your component in the third document block, and then press **Enter** at the end of the line. This command implements the polynomial in a custom component by defining your equations in terms of the regression curve and parameters for the block.

$$\text{eq} := [P(t) = f(Q(t)) \cdot \rho \cdot g, P(t) = Pr(t) - Pl(t), \rho = 1000, g = 9.81]$$

8. In the **Configuration** section, select **Ports**.
9. Click **Clear All Ports**.
10. Click **Add Port** to add a port on the left edge. Select the **Hydraulic** type, and assign the **Pressure** variable to $Pl(t)$ and the **Volume Flow Rate** variable to $Q(t)$.
11. Click **Add Port** to add a port on the right edge. Select the **Hydraulic** type, choose **style b**, assign the **Pressure** variable $Pr(t)$ and the **Volume Flow Rate** variable $-Q(t)$ [click the +/- button to change the sign].
12. From the **Icon** list, select **Use default**, and then click **Refresh All**.
13. In the **Configuration** section, select **Parameters**, and then click **Refresh All**.

14. Change the default for g to 9.81 and assign its type to **Acceleration**. For rho, change its default to 1000 and its type to **Density**. Click **Refresh All**.
15. Select **Variables**, and then click **Refresh All**.
16. Change the type for $P(t)$ to **Pressure**.
17. Select **Dimensional Analysis**, and then click **Check Dimensions**. The following expression appears: $\frac{517}{50000} - \frac{13 Q(t)^2}{5} \frac{\text{m}^6}{\text{s}^2} - \frac{137 Q(t)}{5000} \frac{\text{m}^3}{\text{s}}$. This indicates a dimensional inconsistency, however, because it is benign we can leave it as is. If you prefer to remove the inconsistency, you could replace $Q(t)$ in the original expressions with $\frac{Q(t)}{Q1}$, assign the parameter $Q1$ the default value 1 with type **VolumeFlowRate**, and protect it (add an **X** to the **Protected** column).
18. In the **Component Generation** section, change the **Name** to **CentrifugalPump**.
19. Click **Generate MapleSim Component** to create your component and to bring you back into the MapleSim environment. The custom component now appears in the **Components** palette in the **Local Components** tab (.
20. Drag the custom component into the **Model Workspace** and create the model shown in **Figure 6.12** 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 Mac) 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 **VolumeFlowRate** quantities from the **Properties** tab.
21. Click **Run Simulation** () in the **Main Toolbar**.
22. Click **Show Simulation Results** (). The following graphs appear in the Analysis window.



6.6 Tutorial 6: Using the External C Code/DLL Custom Component App

In this tutorial, you will use the **External C/Library Block** app 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. Select the **Add Apps or Templates** tab ()
3. Double-click on the **External C/Library Block** entry in the Component Creation category of the **App** palette. The **External C Code/Library Definition** app opens.
4. In the **Code/Library Location** section, for **Source Location**, select the **Text Area** radio button. Doing so opens a text area in which C code is to be entered. For a Windows platform, the initial content is the code shown in **Figure 6.13**. For a Unix platform, the code is shown in **Figure 6.14**.

```

#ifdef WMI_WINNT
#define EXP __declspec(dllexport)
#define M_DECL __stdcall
#else
#ifdef X86_64_WINDOWS
#define EXP __declspec(dllexport)
#define M_DECL
#endif
#endif

EXP double M_DECL f1(double a, double *b, double *c)
{
    c[0] = a*a;
    c[1] = b[0] + b[1];
    c[2] = c[0]*c[1];
    return c[1];
}

```

Figure 6.13: External C Code Definition for Windows

```

/*****
double f1(double a, double *b, double *c)

```

Figure 6.14: External C Code Definition for Unix

5. Select the **Attachment** radio button just above the code edit region.
6. Enter a name for the attachment, say **fl.c**, in the text area to the right of the **Attachment** radio button, then click the **Attach** button to attach the file to the MapleSim file.
7. Select the **Attachment** radio button for the **Source Location**, near the top of the app and ensure that **fl.c** is selected in the drop-down menu.
8. Click **Validate C** to validate the code.
9. In the **Configuration** section, select **Function**, and then enter **fl** for the **Function Name**.

Function Name:

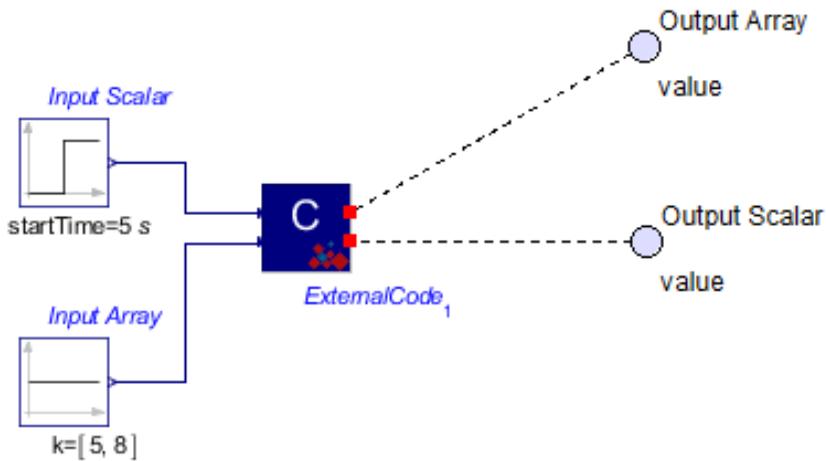
10. In the **Configuration** section, select **Parameters**.
11. Click **Add Parameter**, set the **Name** to **a** and click **Apply** to apply the changes to the selected parameter.
12. Click **Add Parameter**, set the **Name** to **b**, select the **Array?** box, set **Dim** to **2** and click **Apply**.
13. Click **Add Parameter**, set the **Name** to **c**, select **Passed by Reference**, select the **Array?** box, set **Dim** to **3** and click **Apply**.
14. In the **Configuration** section, select **Return**.
15. Use the following values for the C function return parameter.

Return?
Return Name:
Return Type:

16. In the **Component Generation** section, enter **ExternalCode** in the **Block Name** field.
17. Click **Generate Component**. In MapleSim, the custom component appears in the **Local Components** tab () located in the **Components** palette, on the left side of the MapleSim window.

To use the external code custom component:

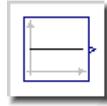
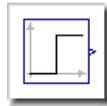
1. Using the specified model components and their settings from **Table 6.6**, drag the components into the **Model Workspace** and set their values.



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 **Properties** tab () located on the right side of the MapleSim window.

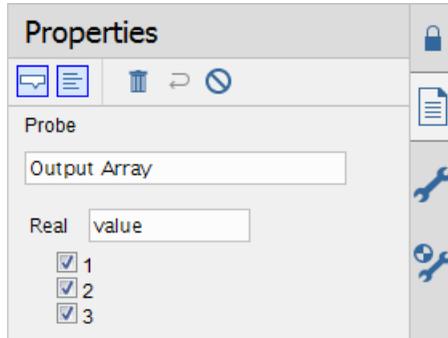
Table 6.6 shows the required components and their settings.

Table 6.6: External C Code DLL Custom Components and Required Settings

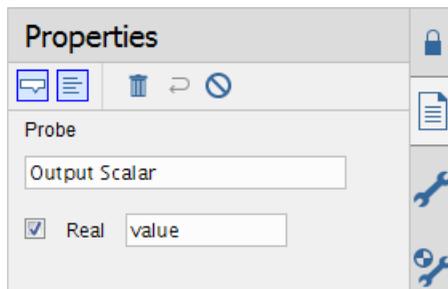
Component	Symbol	Component Location	Required Settings
Custom Component		Local Components > Components	Use default settings
Constant Vector		Library Components > Signal Blocks > Sources > Real	Constant output value , set K to [5, 8]
Step		Library Components > Signal Blocks > Sources > Real	Height: 4 Offset: 0 T₀: 5

2. Connect the **Step** component to custom component **input port a**.

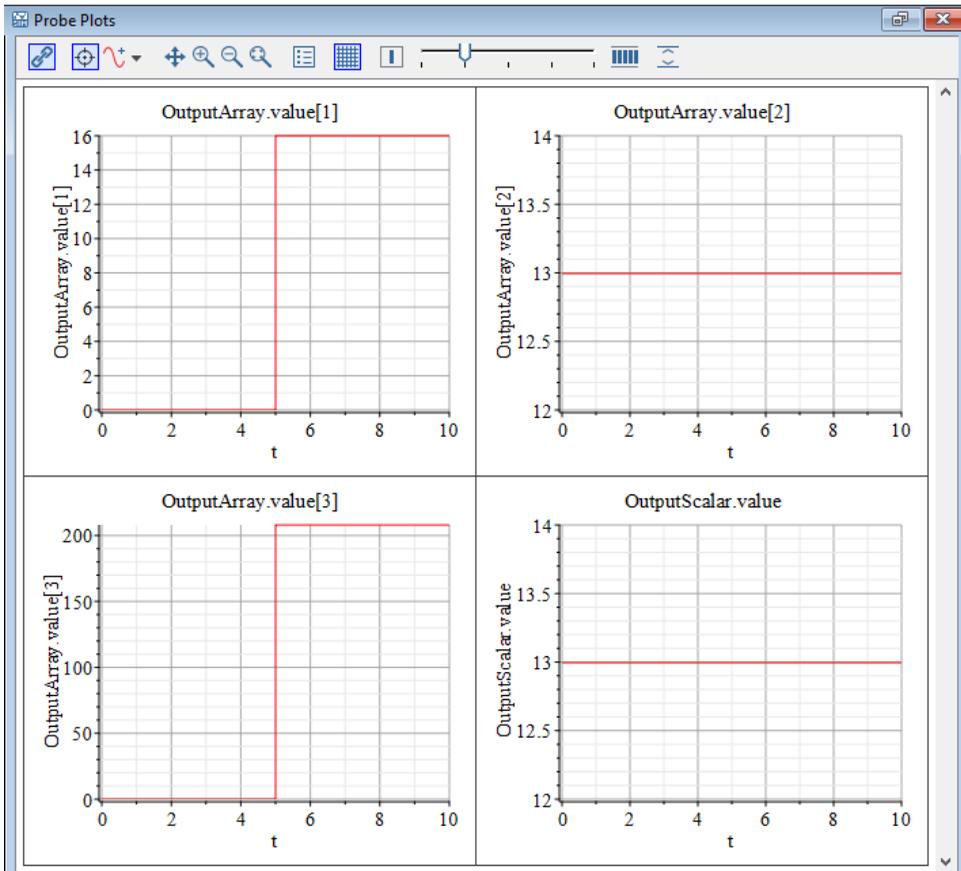
3. Connect the **Constant Vector** component to custom component **input port b**.
4. Attach a probe to the custom component **output port c** and enter the following values:



5. Attach a probe to the custom component **output port r** and enter the following values:



6. Click **Run Simulation** () in the **Main Toolbar**.
7. Click **Show Simulation Results** (). The following graphs appear in the Analysis window.



8. To verify the results, from the **Help** menu, select **Examples > User's Guide Examples > Chapter 6**, and then select the **Simple External C Code Function** example.

6.7 Tutorial 7: Using the Equation Extraction App

In this tutorial, you will use the Equation Extraction App to extract the equations for a model by performing the following tasks:

- Open the model
- Open the Equation Extraction App
- View, manipulate, and reassign equations

The Equation Extraction App contains pre-built embedded components that lets you extract, manipulate, and analyze the symbolic system equations generated by any MapleSim model.

You can select variables and parameters of interest and assign them user-definable names. These features are useful in generating reusable equations when there is more than one subsystem.

App Description

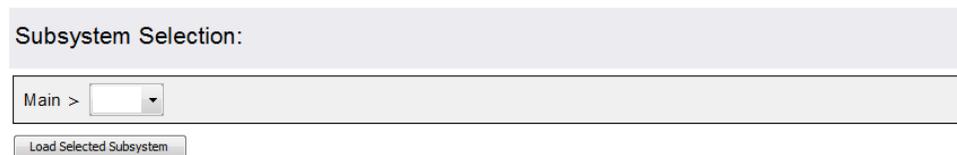
The Equation Extraction App is a collection of pre-built controls and procedures associated with specific Maple commands to easily generate equations from MapleSim models.

The Equation Extraction App consists of two main areas, **Equation Details** and **View Equations**.

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.

Load Selected Subsystem: Loads the subsystem parameters and variables. If no subsystem is selected, equations for the whole model will be loaded when you click **Load Selected Subsystem**.



Equation Details

In this area you can customize and define ports, DAE variables, and parameters for the generated equations.

Ports

For acausal ports, you can configure a port so that either the flow-variable or the across-variable is considered an input (known).

One of the two signals must be selected as input. Select each port and then select either **Flow** or **Across** to specify which signal is the input.

DAE Variables

Select and rename DAE variables of interest.

Variables: Contains the model DAE variables.

New Name: Rename a DAE variable by selecting the variable and specifying a new variable name in the **New Name** field.

Keep: Mark a variable of interest by selecting the variable and then selecting **Keep**. If you use this feature, **Extract Equations** only displays equations of variables marked with **Keep**. If no variables or parameters are marked as keep, all equations are displayed.

Use Subscripts: Use subscripted variable names.

Reset Substitutions: Restores the original names of all variables.

Parameters

Select and rename parameters of interest.

Parameters: Contains the model parameters.

New Name: Rename a model parameter by selecting the parameter and specifying a new name in the **New Name** field.

Keep: Mark a parameter of interest by selecting the parameter and then selecting **Keep**. If you use this feature, **Extract Equations** only displays equations of variables marked with **Keep**. If no variables or parameters are marked as keep, all equations are displayed.

Symbolic: Specify which parameters are left in symbolic form. By default, parameters are evaluated in the equations. (Or, use **Toggle Symbolic** to toggle this setting for all parameters.)

Use Subscripts: Use subscripted parameter names.

Reset Substitutions: Restores the original names of all 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). For more information about equation types, see `GetEquations`.

The code edit region shows Maple code that can be used in a Worksheet template to access the equations.

View Equations:

Simplify returned equations (may take time for large equations)

Extract Equations

Auto Update

Typeset Math

Equations Types: DAEs Definitions Relations Events ODEs AEs

```

1 A := MapleSim:-LinkModel():
2 A:-SetSubsystemName("sub"):
3 A:-SetSubstitutions(['Main.sub.contact1.B' = B__Cont1, `
4 eqs := A:-GetEquations('output' = 'all', 'inputs' = {'Ma
5 daes := eqs[1];

```

$$\left\{ \begin{array}{l} \frac{981}{100} + \frac{d^2}{dt^2} y_{RB1}(t) = 0, - \left(\left(\frac{d}{dt} \zeta_{RB1}(t) \right) \cos(\zeta_{RB1}(t)) \left(\frac{d}{dt} \xi_{RB1}(t) \right) \cos(\right. \\ \left. + \left(\frac{d}{dt} \zeta_{PR1}(t) \right) \left(\frac{d}{dt} \eta_{PR1}(t) \right) \sin(\zeta_{PR1}(t)) + \left(\frac{d^2}{dt^2} \xi_{PR1}(t) \right) \cos(\eta_{PR1}(t)) \right) \end{array} \right.$$

Generating the Equations

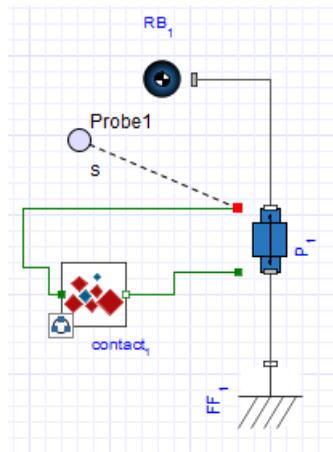
For this example, we will use the model from the *Example: Compliant Contact and Piecewise Functions* (page 180), part of *Tutorial 5: Using the Custom Component Template* (page 174). This model is also found in the **Help > Examples > User's Guide Examples > Chapter 6** menu. In this model, a bouncing ball is analyzed using a prismatic joint to model a falling ball and a custom component to model the compliant ground contact using a spring-damper arrangement.

You will group these components into a subsystem to use in the Equation Extraction App.

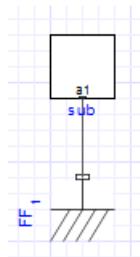
Generating the System Equations

To generate the system equations:

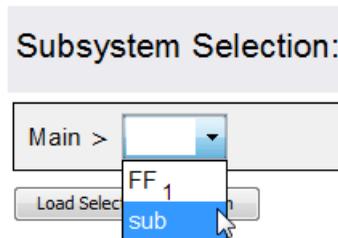
1. From the **Help** menu, select **Examples > User's Guide Examples > Chapter 6**, and then select **Compliant Contact and Piecewise Functions**.



2. Place the **Prismatic** joint, **contact** custom component, probe, and **Rigid Body** in a subsystem called **sub**. This allows the Equation Extraction app to generate equations specifically for the selected subsystem.



3. Select the **Add Apps or Templates** tab ().
4. Double-click on the **Equation Extraction** entry in the **Apps** palette. The **Equation Extraction** App opens in the Analysis window. The toolbar in the **Subsystem Selection** window shows all of the subsystems in your model.
5. From the toolbar, select the **sub** subsystem.



6. Click **Load Selected Subsystem**. The subsystem's component ports, DAE variables, and parameters load automatically in the **Ports**, **DAE Variables**, and **Parameters** areas.

7. Click **Extract Equations** in the **View Equations** section.

The system equations appear in the **View Equations** area.

View Equations:

Simplify returned equations (may take time for large equations)

Extract Equations

Auto Update

Typeset Math

Equations Types: DAEs Definitions Relations Events ODEs AEs

```

1 A := MapleSim:-LinkModel():
2 A:-SetSubsystemName("sub"):
3 A:-SetSubstitutions(['Main.sub.contact1.B' = B__Cont1, `
4 eqs := A:-GetEquations('output' = 'all', 'inputs' = {'Ma
5 daes := eqs[1];

```

$$\left\{ \begin{array}{l} \frac{981}{100} + \frac{d^2}{dt^2} y_{RB1}(t) = 0, - \left(\left(\frac{d}{dt} \zeta_{RB1}(t) \right) \cos(\zeta_{RB1}(t)) \left(\frac{d}{dt} \xi_{RB1}(t) \right) \cos(\right. \\ \left. + \left(\frac{d}{dt} \zeta_{DR1}(t) \right) \left(\frac{d}{dt} \eta_{DR1}(t) \right) \sin(\zeta_{DR1}(t)) + \left(\frac{d^2}{dt^2} \xi_{DR1}(t) \right) \cos(\eta_{DR1}(t)) \right) \end{array} \right.$$

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

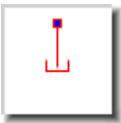
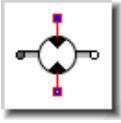
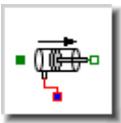
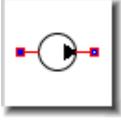
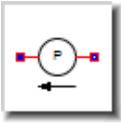
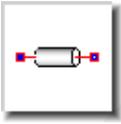
Computational Issues

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 6.7: Basic Hydraulic Library Components

Component	Symbol	Library Location	Its Use
Tank		Hydraulics > 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		Hydraulics > 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: <ul style="list-style-type: none"> • rhoFluid: fluid density • K: Bulk Modulus is the fluid compressibility • nuFluid: Kinematic Viscosity is the dynamic viscosity divided by liquid density
Hydraulic Motor		Hydraulics > 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 Rate		Hydraulics > 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		Hydraulics > Restrictions	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.

$$\frac{p}{\rho \cdot g} + \frac{V^2}{2 \cdot g} + z = \text{constant}$$

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.

$$\Delta P = f \cdot \frac{L \cdot V^2}{D \cdot 2 \cdot g}$$

Hence

$$\frac{p}{\rho \cdot g} + \frac{V^2}{2 \cdot g} + z + f \cdot \frac{L \cdot V^2}{D \cdot 2 \cdot g} = \text{constant}$$

Table 6.8: 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	<i>dimensionless</i>

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:

$$f = \frac{64}{\text{Re}}$$

$$\text{Re} = \frac{D \cdot V}{\nu}$$

where

f is the internal friction

Re is the Reynolds number

D is the pipe diameter

V is the fluid velocity and

ν is the dynamic viscosity

In turbulent flow, the frictional effects of the surface roughness of the pipe are characterized by the **Haaland Equation**.

$$f = \frac{1}{\left(1.8 \log_{10} \left(\frac{6.9}{\text{Re}} + \left(\frac{\epsilon}{3.7 \cdot D} \right)^{1.11} \right)\right)^2}$$

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 6.9** gives the Reynolds number for laminar (**ReL**) and turbulent (**ReT**) flow. Between these two parameters, the friction factor is determined by linear interpolation.

Table 6.9: Circular Pipe Parameters

Symbol	Description	Values
L	Pipe length	5 m
ϵ	Height of internal pipe roughness	$1.5 \cdot 10^{-5}$ m
ReL	Maximum Reynolds number in laminar regime	2000
ReT	Minimum Reynolds number in turbulent regime	4000
D	Pipe hydraulic diameter	0.01 m

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 6.15 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 6.10**.

Tip: To attach the probe on the **Circular Pipe** component, right-click (**Control**-click for Mac) on the component, select **Attach probe**, and then position the probe by clicking on the workspace.

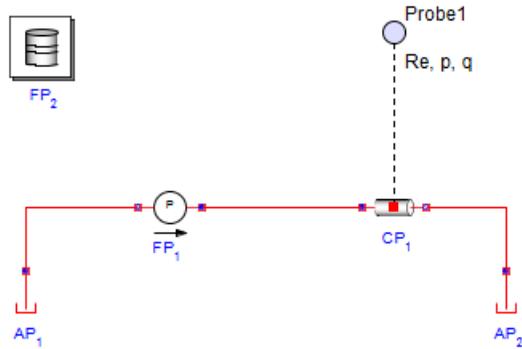
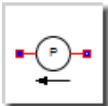
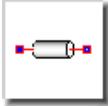
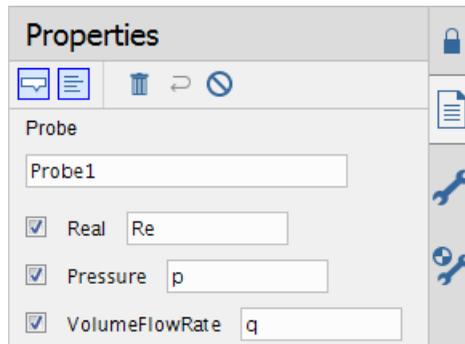


Figure 6.15: Flow Through a Pipe

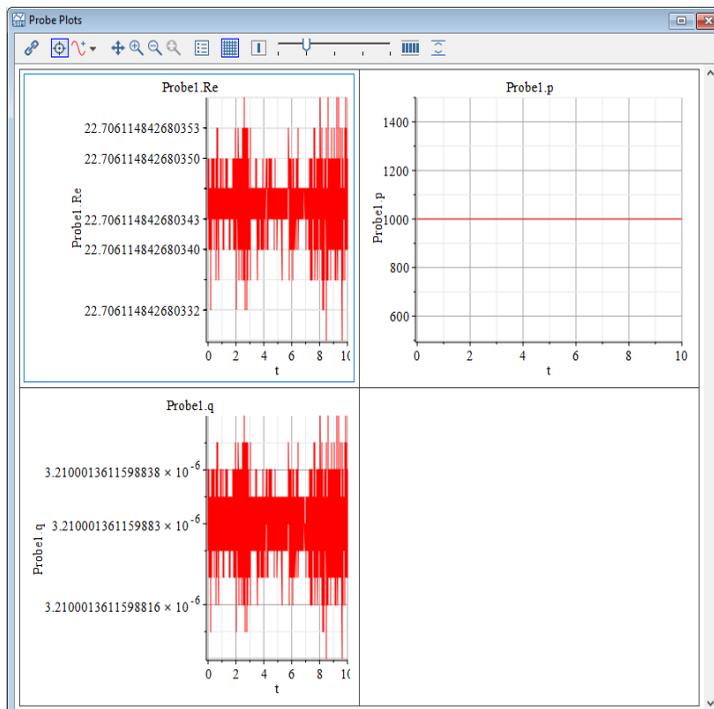
Table 6.10: Hydraulic Components

Component	Quantity	Symbol	Library Location	Values
Tank	2		Hydraulics > Reference Components	Use default settings
Hydraulic Fluid Properties	1		Hydraulics > Reference Components	Use default settings: rhoFluid: $850 \frac{kg}{m^3}$ K: 8000 bar nuFluid: $0.000018 \frac{m^2}{s}$
Fixed Pressure Source	1		Hydraulics > Sources	Set P = $1 \cdot 10^3 Pa$
Circular Pipe	1		Hydraulics > Restrictions	Use default settings

- Click the probe and select the **Real** (the instantaneous Reynolds number), **Pressure**, and **VolumeFlowRate** probe parameters.



3. Click **Run Simulation** (▶) in the **Main Toolbar**.
4. Click **Show Simulation Results** (📊). The following graph appears showing the predicted flow rate of $Q = 3.21 \cdot 10^{-6} \frac{m^3}{s}$.



Confirming the Modeling Results from First Principles

When analyzing the system shown in **Figure 6.15**, apply the Darcy equation.

$$\frac{\Delta P}{\rho \cdot g} = f \cdot \frac{L \cdot V^2}{D \cdot 2 \cdot g}$$

Assuming that the system is in laminar flow, then

$$f = \frac{64}{\text{Re}}$$

Hence

$$\frac{1 \cdot 10^3}{850 \times 9.81} = \frac{64}{\frac{0.01 \times V}{0.000018}} \times \frac{5}{0.01} \times \frac{V^2}{2 \times 9.81}$$

$$0.1199256461 = 2.935779816 V$$

Where

$$V = 0.0408 \frac{m}{s}$$

Using **V** in the flow rate equation yields the following result:

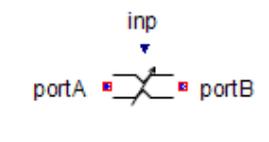
$$Q = \frac{1}{4} V \pi D^2 = 0.0408 \times \frac{\pi \times 0.01^2}{4} = 3.2 \times 10^{-6} \frac{m^3}{s}$$

This is the same value given by MapleSim. Using the calculated value of **V** gives **Re= 22.7**. This is far less than the critical value of 2000, and hence the system is in laminar flow.

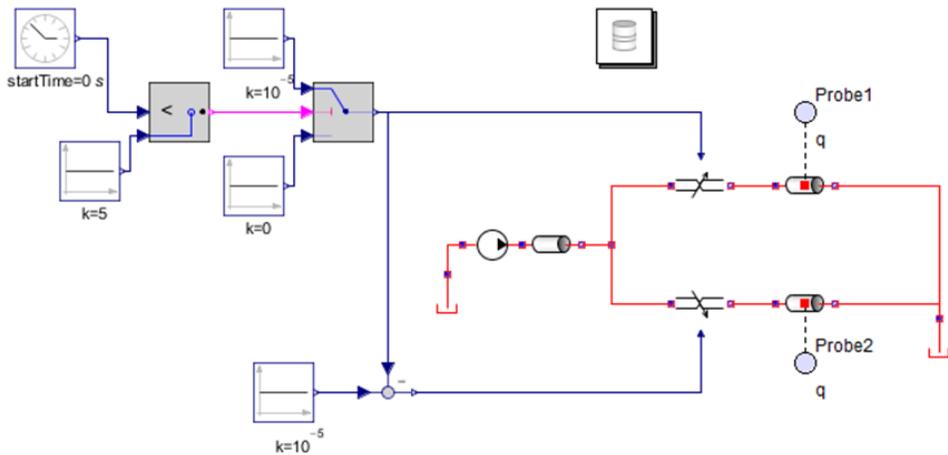
Overview of Controlling Hydraulic Flow Path

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 6.11: Spool Valve

Spool Valve	Name	Description	ID
	<i>PortA</i>	Upstream port	portA
	<i>PortB</i>	Downstream port	portB
	<i>Area</i>	Real input; area of orifice in selected units	Area

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 5 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 6.16: 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 6.12**.

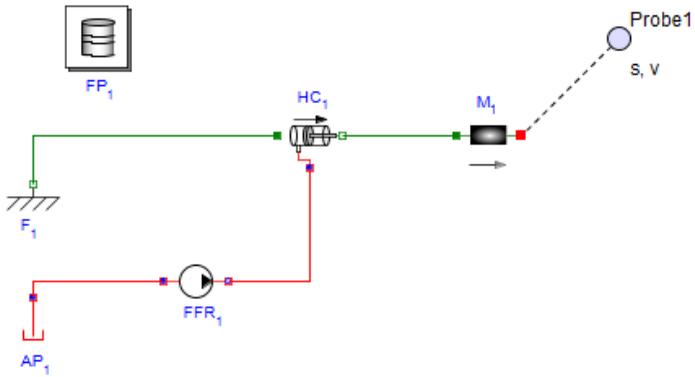
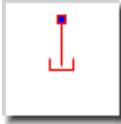
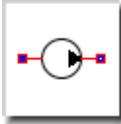
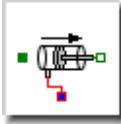
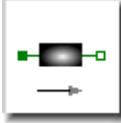
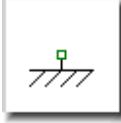


Figure 6.17: Fixed Flow Rate Source

Table 6.12: Translational Motion with Fixed Flow Rate Sources

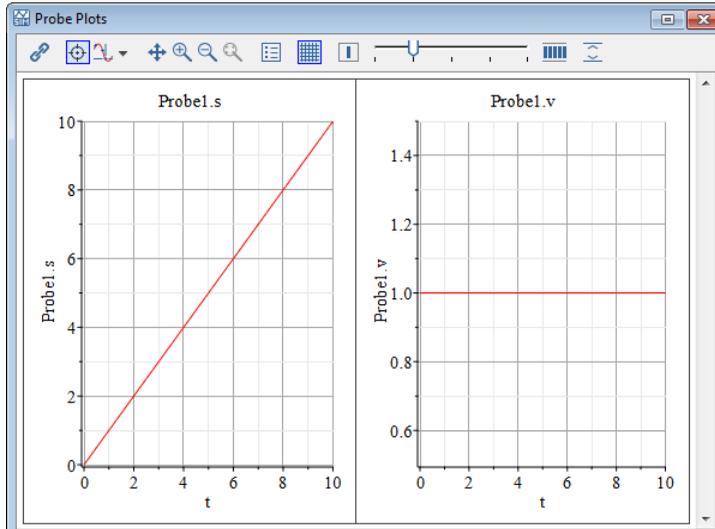
Component	Quantity	Symbol	Library Location	Values
Tank	1		Hydraulics > Reference Components	Use default settings
Hydraulic Fluid Properties	1		Hydraulics > Reference Components	Use default settings: rhoFluid: $850 \frac{kg}{m^3}$ K: 8000 bar nuFluid: $0.000018 \frac{m^2}{s}$
Fixed Flow Rate	1		Hydraulics > Sources	Use default settings
Hydraulic Cylinder	1		Hydraulics > 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 Rate has a flow Q of $1 \frac{\text{m}^3}{\text{s}}$.

The cylinder pushes the sliding mass at a speed of

$$V = \frac{Q}{A} = \frac{1 \text{ m}^3 \text{ s}^{-1}}{1 \text{ m}^2} = 1 \frac{\text{m}}{\text{s}}$$

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 Rate with the Fixed Pressure Source component shown in **Figure 6.18** and **Table 6.13**. The following model converts flow from a fixed pressure source to translational motion.

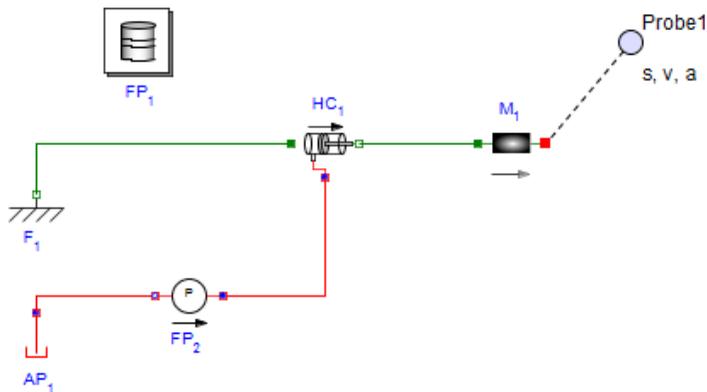
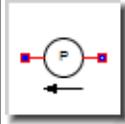


Figure 6.18: Translational Motion with Fixed Pressure Source

Table 6.13: Translational Motion with a Fixed Pressure Source

Component	Symbol	Library Location	Value
Fixed Pressure Source		Hydraulics > Sources	Use default settings

The force on the Sliding Mass is equal to the cross-sectional area of the hydraulic cylinder A multiplied by the pressure P of the hydraulic fluid.

$$F = A \cdot P = 1 \text{ m}^2 \cdot 1 \text{ Pa} = 1 \text{ N}$$

The acceleration of the Sliding Mass is given by:

$$F = m \cdot a = 1 \text{ m}^2 \cdot 1 \text{ Pa} = 1 \text{ N}$$

$$1 \text{ N} = 1 \text{ kg} \cdot a$$

Therefore,

$$a = 1 \frac{\text{m}}{\text{s}^2}$$

By probing the acceleration, speed, and displacement of the Sliding Mass, these values are confirmed with the results in **Figure 6.19**.

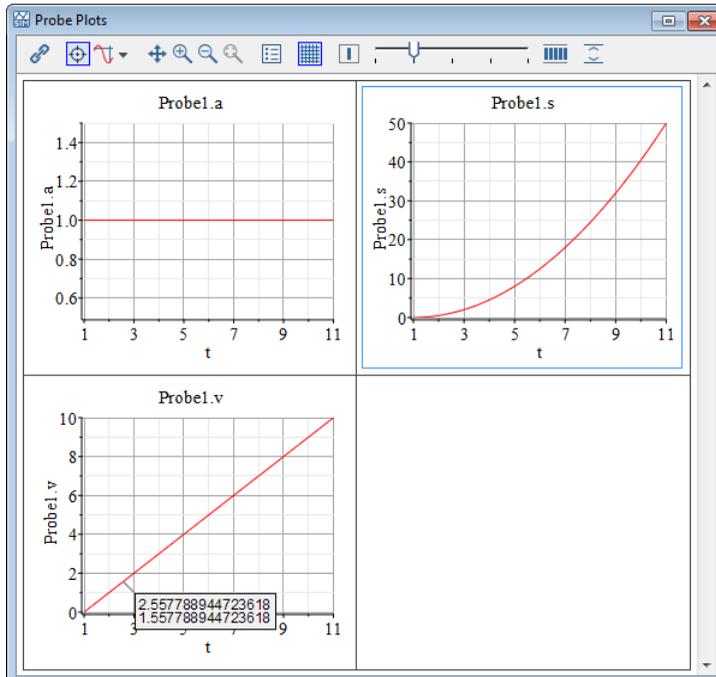


Figure 6.19: Fixed Pressure Source Results

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 6.14**. Use the default settings for each component.

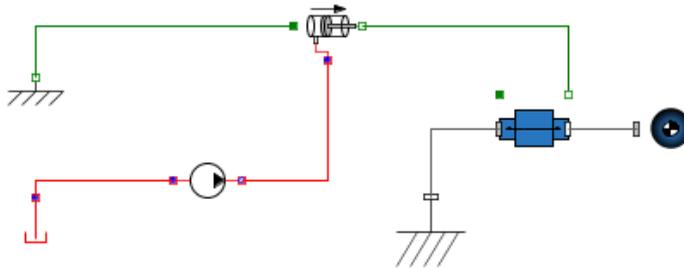


Figure 6.20: Translational Fixed Flange Hydraulic component

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 6.14**. Use the default settings for each component.

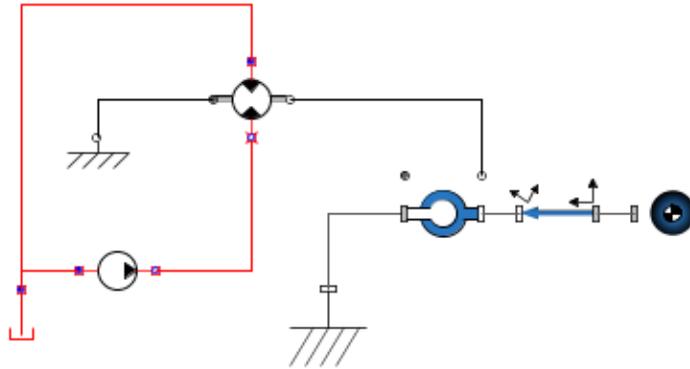
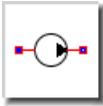
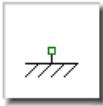
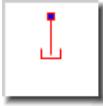
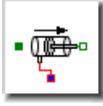
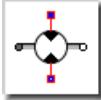
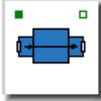
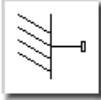


Figure 6.21: Rotational Fixed Flange Hydraulic component

Table 6.14: Actuating Multibody Components

Component	Symbol	Library Location
Fixed Flow Rate		Hydraulics > Sources
Rotational Fixed Flange		1-D Mechanical > Rotational > Common
Translational Fixed Flange		1-D Mechanical > Translational > Common
Tank		Hydraulics > Reference Components
Hydraulic Cylinder		Hydraulics > Actuators

Component	Symbol	Library Location
Hydraulic Motor		Hydraulics > Actuators
Prismatic		Multibody > Joints and Motions
Rigid Body Frame		Multibody > Bodies and Frames
Rigid Body		Multibody > Bodies and Frames
Revolute		Multibody > Joints and Motions
Fixed Frame		Multibody > Bodies and Frames

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 6.22** 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 1 kg 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 $\frac{1 \text{ m}^2}{0.1 \text{ m}^2}$.

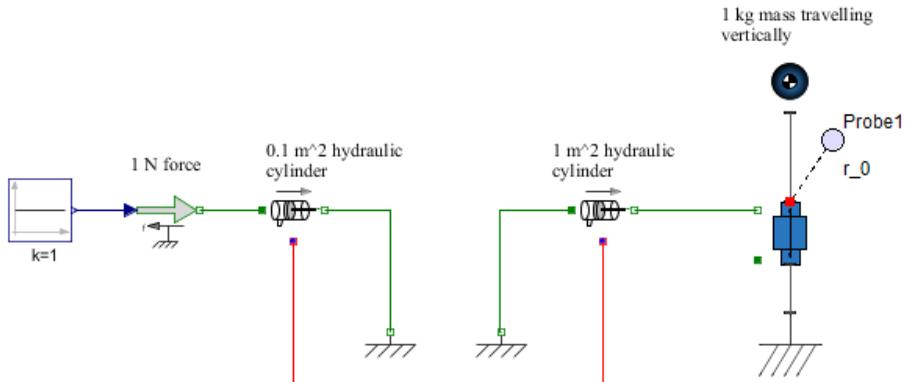


Figure 6.22: Pascal's Principle Example

Overview of Compressibility of Hydraulic Liquids

The compliant cylinder and constant volume chamber components (**Table 6.16**) 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 6.23**.

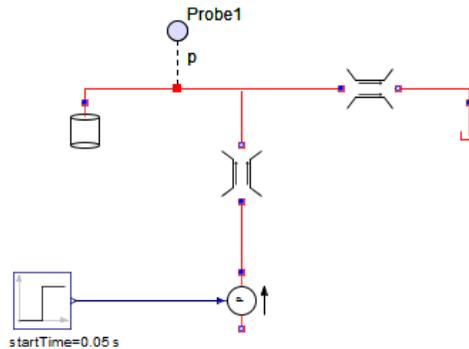


Figure 6.23: Hydraulic Liquids Compressibility

Table 6.15: Hydraulic Liquids Compressibility Components

Component	Symbol	Library Location	Value
-----------	--------	------------------	-------

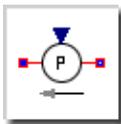
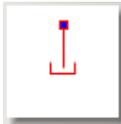
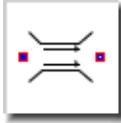
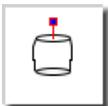
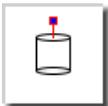
Signal Pressure		Hydraulics > Sources	Use default settings
Tank		Hydraulics > Reference Components	Use default settings
Linear Resistance		Hydraulics > Restrictions	Use default settings

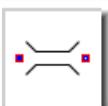
Table 6.16: Confined Hydraulic System Components

Component	Quantity	Symbol	Library Location	Value
Compliant Cylinder	1		Hydraulics > Chambers	Use default settings
Constant Volume	1		Hydraulics > 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 6.17: Fluid Inertia

Component	Quantity	Symbol	Library Location	Value
Fluid Inertia	1		Hydraulics > Restrictions	Use default settings

System without Fluid Inertia

Figure 6.24 shows a system without fluid inertia.

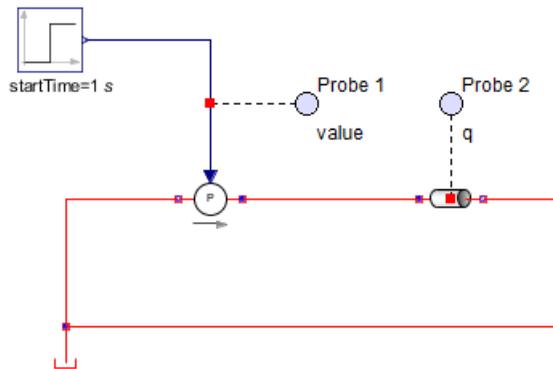


Figure 6.24: System without Fluid Inertia

System with Fluid Inertia

Figure 6.24 shows a system with fluid inertia.

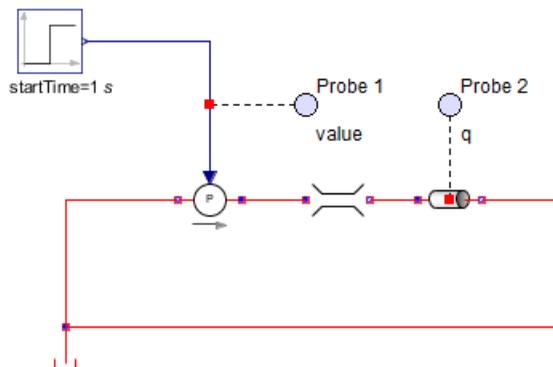


Figure 6.25: System with Fluid Inertia

Figure 6.26 shows typical system flow rates with (green) and without (red) fluid inertia. Introducing fluid inertia adds a lag into the system.

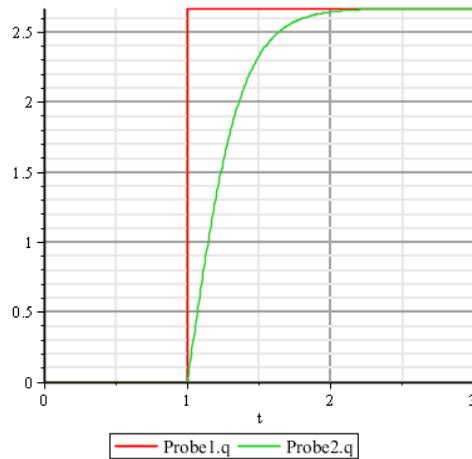


Figure 6.26: System with and without Fluid Inertia

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.

$$\frac{dV(x, t)}{dt} + \frac{1}{\rho} \frac{dP(x, t)}{dt} + \frac{fV(x, t)|V(x, t)|}{2D}$$

$$\frac{dV(x, t)}{dx} + \frac{1}{Ks} \frac{dP(x, t)}{dt} = 0$$

$$Ks = \frac{1}{\frac{1}{K} + \frac{D}{Et}}$$

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 6.27: Water Hammer

Each subsystem consists of a compliant cylinder, a pipe, and a fluid inertia component as shown in **Figure 6.28**. 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 constant volume chambers, each with a volume,

$$\frac{V}{N}$$

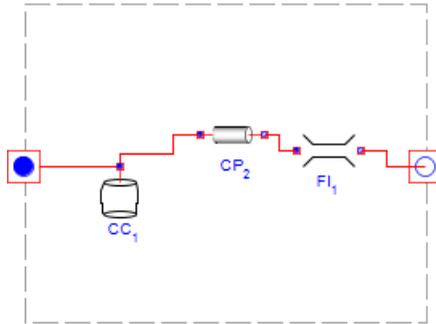


Figure 6.28: Discretized Pipeline Segment

To build the water hammer model:

1. Build the model with the components and connections shown in **Figure 6.27** and **Figure 6.28**.
2. Configure the **Fluid Properties** component with the values specified in **Table 6.18**.

Table 6.18: Fluid Properties Values

Parameter Name	Symbol	Description	Value
rhoFluid	ρ	Fluid density	1000 kg m^{-3}
K	K	Fluid bulk modulus	$200 \cdot 10^6 \text{ Pa}$
nuFluid	ν	Fluid kinematic viscosity	$\frac{10^{-3} \text{ m}^2}{\text{rhoFluid} \text{ s}}$

3. Configure the parameter block with the parameters and values shown in **Table 6.19**.

Table 6.19: Water Hammer Parameters

Parameter Name	Description	Value
N	Number of pipe segments	20
Dia	Pipe hydraulic diameter (D)	0.1 m
Len	Pipe Length	25 m
Em	Pipe Young's modulus (E)	$70 \cdot 10^9 Pa$
ef	Pipe internal roughness	0.0001 m
$thickness$	Pipe wall thickness (t)	0.001 m

- For the **Fixed Pressure** component (**FP₂** in **Figure 6.27**), set the pressure (P) to 500kPa.
- Configure the **Circular Pipe** components in **Main** and the **HydraulicPipeline** shared subsystem with the following settings.

Parameter Name	Setting
D	Dia
L	$\frac{Len}{N+1}$
ϵ	ef
Re_L	2000
Re_T	4000

- Configure the **Fluid Inertia** components in **Main** and the **HydraulicPipeline** shared subsystem with the following settings.

Parameter Name	Setting
A	$\frac{1}{4} \cdot \pi \cdot Dia^2$
L	$\frac{Len}{N+1}$
ρ	ρ_{Fluid}

- Configure the **Compliant Cylinder** component in the **HydraulicPipeline** shared subsystem with the following settings.

Parameter Name	Setting
α	0
k	1.4
L	$\frac{Len}{N}$
D	Dia
D_o	$Dia + 2*thickness$
Em	Em

8. For the **First Order** component, under the Properties tab () , set **T** to **0.01s** and **y0** to **0.01**.
9. Configure the **Step** component with the following settings.
 - For **height**, enter **-0.009999**.
 - For **offset**, enter **0.01**.
 - For **T₀**, enter **2s**.
10. Under the Settings tab, configure the following Simulation parameters.
 - For **t_d**, enter **3s**.
 - For **Solver Type**, select **Variable**.
 - For **Solver**, select **Rosenbrock (stiff)**.

Figure 6.29 plots the pressure and flow rate at the end of a pipe for a valve that rapidly closes after 2 seconds.

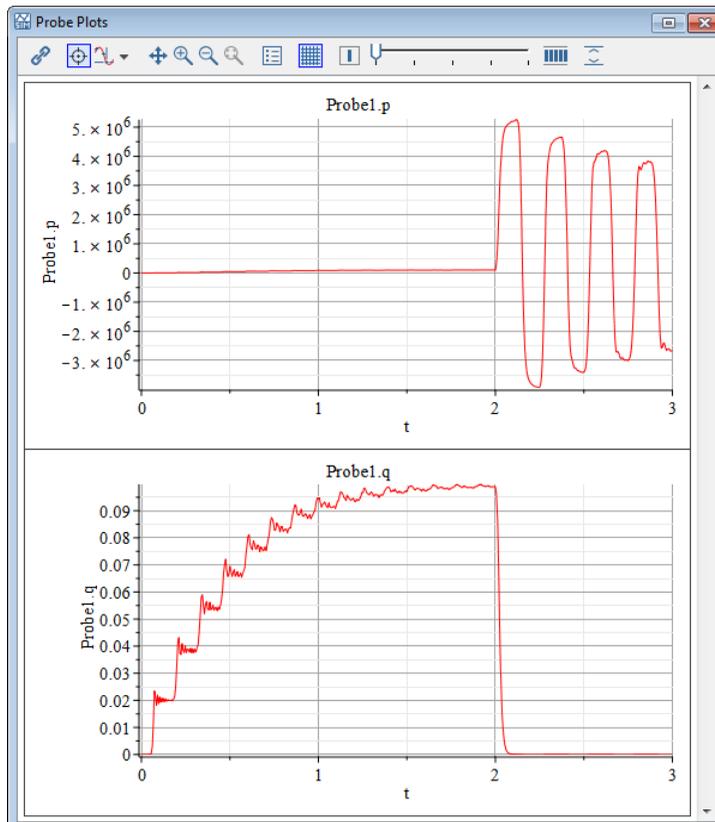


Figure 6.29: Water Hammer Pressure Flow Rate

The maximum pressure is about $5 \times 10^6 \text{ Pa}$, with the liquid reaching a flow rate of $0.099 \frac{\text{m}^3}{\text{s}}$.
 The maximum pressure can also be calculated using the Joukowsky equation,

$$\Delta P = \rho c \frac{\Delta Q}{A}$$

$$c = \sqrt{\frac{Ke}{\rho}}$$

$$Ke = \frac{1}{\frac{1}{K} + \frac{D}{Et}}$$

If you substitute the parameters from **Table 6.18** and **Table 6.19** into the preceding equations, and then assume:

$$\Delta Q = 0.099 \frac{m^3}{s}$$

you get

$$\Delta P \approx 5 \times 10^6 Pa$$

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 71).

$$eq := \left[\begin{array}{l} q(t) = \dot{V}F(t), VF(t) \\ \\ = \left\{ \begin{array}{ll} Ks p(t) & p(t) \leq ppr \\ Vpr + (k \cdot (p(t) - ppr)) & ppr < p(t) < pm \\ Vmax + (Ks (p(t) - pmax)) & pmax \leq p(t) \end{array} \right. \\ \\ , k = \frac{(Vmax - Vpr)}{(pmax - ppr)} \end{array} \right]$$

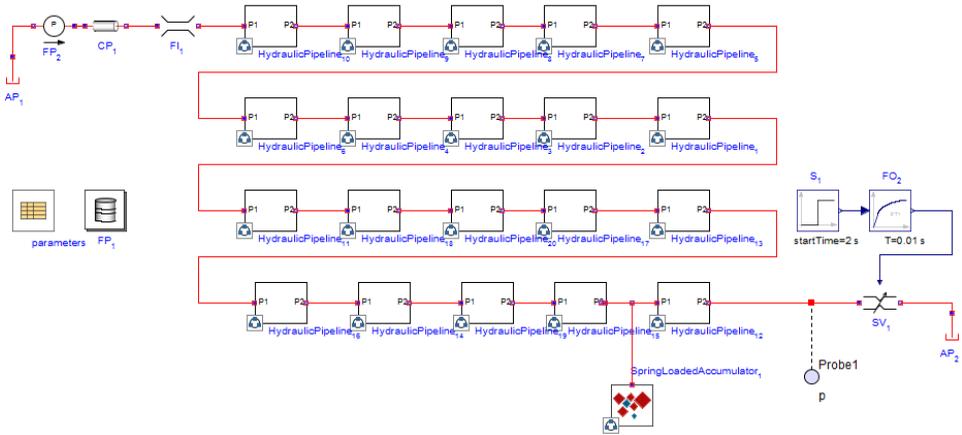
$$params := [Vmax = 0.1, ppr = 10^5, pmax = 3 \cdot 10^6, Ks = 4 \cdot 10^{-10}, Vpr = 0]$$

$$initialconditions := [VF(0) = 0]$$

Table 6.20: Accumulator Parameters Custom Component

Description	Values
Vmax	0.1 m
ppr	100000
Pmax	3000000
Ks	10×10^{-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 6.30** shows the pressure surge at the end of a pipeline with an accumulator.



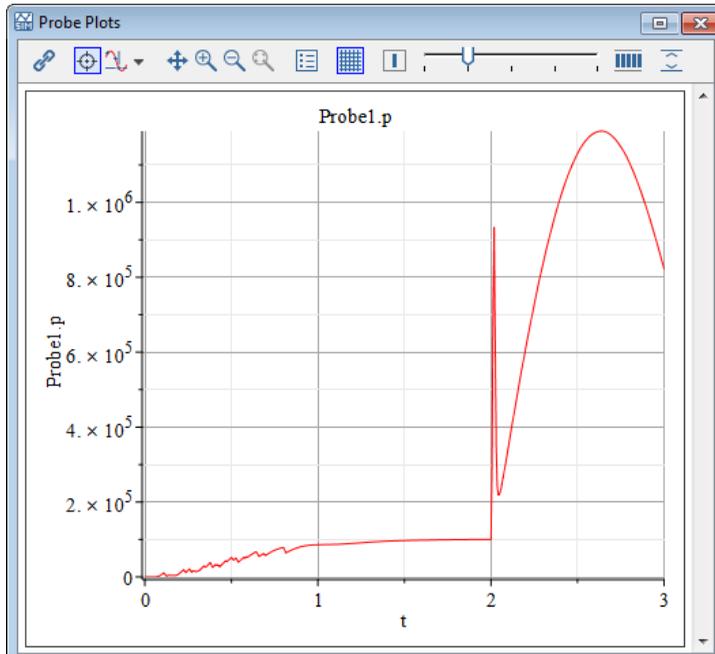


Figure 6.30: Pressure Surge with an Accumulator

Overview of Hydraulic Custom Components

Two examples of hydraulic custom components are centrifugal pumps and vertical pipes. For a complete description on how to create custom components, see *Creating Custom Modeling Components* (page 71).

Centrifugal Pumps

Typically, manufacturers provide head flow rate charts for centrifugal pumps, as shown in **Figure 6.31**.

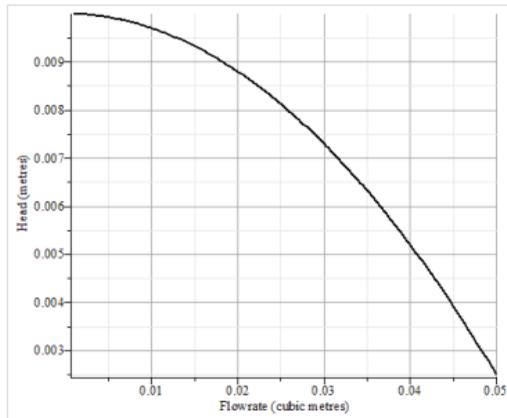


Figure 6.31: Head Flow Rate

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 6.32**, 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.

$$\begin{aligned}
 eq &:= [P(t) = (d + c \cdot Q(t) + b \cdot Q(t)^2 + a \cdot Q(t)^3) \cdot \rho \cdot g, P(t) = Pr(t) - Pl(t)] : \\
 params &:= [a = -3.3521 \cdot 10^{-8}, b = -0.0000039589, c = 0.009948, d = 35.04, \rho = 1000, g = 9.81] : \\
 initialconditions &:= [] :
 \end{aligned}$$

Figure 6.32: Centrifugal Pump Custom Component Equations

Note: Assigning the value rhoFluid to the Density parameter sets the density to be the value defined in the Hydraulic Fluid Properties component.

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 6.33** shows the custom component equations to simulate gravity head.

```
eq := [  $dP(t) = \text{rho} \cdot g \cdot z$ ,  $P_{in}(t) - P_{out}(t) = dP(t)$ ,  $Q_{in}(t) = Q_{out}(t)$ , ] :  
params := [  $z = 0$ ,  $\text{rho} = 1000$ ,  $g = 9.81$  ] :  
initialconditions := [ ] :
```

Figure 6.33: Gravity Head Custom Component Equations

7 Reference: MapleSim Keyboard Shortcuts

Table 7.1: Opening, Closing, and Saving a Model

Task	Windows and Linux	Mac
Create a new model	Ctrl + N	Command + N
Open an existing model	Ctrl + O	Command + O
Close the active document	Ctrl + F4 (Windows) Ctrl + W (Linux)	Command + W
Save the model as an .msim file	Ctrl + S	Command + S

Table 7.2: Building a Model in the Block Diagram View

Task	Windows and Linux	Mac
Cancel operation while drawing a connection	Esc	Esc
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 horizontally	Ctrl + H	Command + K
Flip the selected modeling component vertically	Ctrl + F	Command + F
Group the selected modeling components into a subsystem	Ctrl + G	Command + G
Reroute the selected connections	Ctrl + D	Command + D
Reroute all connections	Ctrl + A, Ctrl + D	Command + A, Command + D

Table 7.3: Browsing a Model in the Block Diagram View

Task	Windows and Linux	Mac
View the selected modeling component or subsystem in detail	Ctrl + M, or Ctrl + Down Arrow	Command + M, or Command + Down Arrow
View the parent level of the current subsystem	Ctrl + Up Arrow	Command + Up Arrow
Return to Main	Home	Home
Navigate to parent component	Ctrl + Up Arrow	Command + Up Arrow

Task	Windows and Linux	Mac
Zoom into the model workspace	Ctrl + numeric keypad plus key, or Ctrl and move the mouse wheel forward	Command + numeric keypad plus key, or Command and move the mouse wheel forward
Zoom out from the model workspace	Ctrl + numeric keypad minus key, or Ctrl and move the mouse wheel backward	Command + numeric keypad minus key, or Command and move the mouse wheel backward
Scale the model diagram to fit in the model workspace	Ctrl + T	Command + T
Reset zoom factor to the default zoom factor (100%)	Ctrl + 0 (zero)	Command + 0 (zero)

Table 7.4: Browsing a Model in the 3-D View

Task	Windows and Linux	Mac
Connect ports	Ctrl + Shift + C	Command + Shift + C
Delete the currently selected item	Backspace or Delete	Delete
Fit scene	V	V
Select mode	Esc	Esc
Pan mode	F2	F2
Zoom mode	F3	F3
Rotate mode	F4	F4
Move the camera around a 3-D model in the perspective view	Left mouse button click and drag or Ctrl + left mouse button click and drag	Mouse click and drag or Command + mouse click and drag
Panning a 3-D model	Right mouse button click and drag or Shift + left mouse button click and drag	Right mouse button click and drag or Shift + mouse click and drag
Zoom into or out from the 3-D workspace	Mouse scroll wheel forward (zoom in), or backward (zoom out) or middle mouse click and drag or Alt + left mouse button click and drag	Mouse wheel scroll forward (zoom in), or backward (zoom out) or middle mouse click and drag or Alt + mouse click and drag, or mouse wheel
Change perspective view to look down the negative X, Y, or Z axis	X	X
	Y	Y
	Z	Z
Change perspective view to look down the positive X, Y, or Z axis	Shift + X	Shift + X
	Shift + Y	Shift + Y
	Shift + Z	Shift + Z
Change perspective view to look at the right side of the scene	R or Ctrl + 4	R or Command + 4

Task	Windows and Linux	Mac
Change perspective view to look at the left side of the scene	L or Ctrl + 3	L or Command + 3
Change perspective view to look at the top side of the scene	T or Ctrl + 5	T or Command + 5
Change perspective view to look at the bottom side of the scene	M or Ctrl + 6	M or Command + 6
Change perspective view to look at the Front side of the scene	F or Ctrl + 1	F or Command + 1
Change perspective view to look at the Back side of the scene	B or Ctrl + 2	B or Command + 2
Save the current camera pose. (Only one camera pose can be saved.)	S	S
Return to the last saved camera pose.	H	H
Refresh 3-D view	Shift + F5	Shift + F5

Table 7.5: Simulating a Model

Task	Windows and Linux	Mac
Run simulation	F5	F5
View Simulation Results. If the Simulation Results is currently in focus, bring MapleSim window into focus.	F6	F6
View 3-D Workspace. If the 3-D Workspace is currently in focus, bring MapleSim window into focus.	F7	F7
View Apps Manager. If the Apps Manager is currently in focus, bring MapleSim window into focus.	F8	F8
Toggle enable/disable selected components or connections to exclude selection from the next simulation	Ctrl + E	Command + E

Table 7.6: Navigating the Console Pane

Task	Windows and Linux	Mac
Expands a section <ul style="list-style-type: none"> • Press the Right Arrow key to expand a section. • Press the Left Arrow key to collapse a section. • Press the Down Arrow key to move to the following section. • Press the Up Arrow key to move to the preceding section. 	Right Arrow	Right Arrow
Collapses a section	Left Arrow	Left Arrow
Moves cursor to the following or previous section	Up or Down Arrow	Up or Down Arrow

Table 7.7: Modifying the Plot Window Layout

Task	Windows and Linux	Mac
Tile plot windows	Shift + T	Shift + T
Cascade plot windows	Shift + C	Shift + C
Fit plots in plot window. Automatically adjust row heights to fit all plots in a plot window in the available vertical space.	Shift + F	Shift + F

Table 7.8: Editing a Modelica Custom Component

Task	Windows and Linux	Mac
View the Modelica Code Editor. If the Modelica Code Editor is currently in focus, bring MapleSim window into focus.	F9	F9
New Modelica custom component	Ctrl + N	Command + N
Save Modelica custom component	Ctrl + S	Command + S
Find and replace	Ctrl + F	Command + F
Go to a line	Ctrl + G	Command + G
Autocomplete on part of a keyword or component name	Ctrl + Space	Command + Spacebar
Insert a Modelica syntax template after typing a typing a term like <i>block</i> , <i>if</i> , <i>ife</i> , or <i>for</i> . See Using MapleSim > Building a Model > Modelica	Ctrl + Shift + Space	Command + Shift + Spacebar

Task	Windows and Linux	Mac
Custom Components > Modelica Code Editor > Syntax Templates for a complete list of terms.		
Toggle left navigation pane (if unlocked)	Ctrl + Tab	Command + Tab

Table 7.9: Miscellaneous

Task	Windows and Linux	Mac
Perform a search	Alt+S	Alt+S
Start Maple from MapleSim	F10	F10
Toggle Palettes Pane (if unlocked)	Ctrl + Tab	Command + Tab

For keyboard shortcuts for 2-D math notation, refer to **Using MapleSim > Building a Model > Annotating a Model > Key Combinations for 2-D Math Notation**.

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 playback window 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 <i>subsystem definition</i> , which defines the configuration.
Standalone subsystem	A subsystem that is not linked to a <i>subsystem definition</i> 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 <i>shared subsystems</i> .

Index

Symbols

- 2-D math notation, 62
 - 3-D animation, 125
 - Enable, 107
 - 3-D display controls
 - 3-D manipulators, 115
 - adding a trace, 112
 - Attached shapes, 112, 117
 - Implicit geometry, 111
 - Initial conditions, 124
 - 3-D model construction, 114
 - 3-D Playback Window, 106
 - 3-D view navigation, 109
 - 3-D views
 - Orthographic, 109
 - Perspective, 109
 - 3-D workspace, 108
 - axis designation, 109
- ## A
- Acausal Mapping, 79
 - Acausal modeling, 2, 5, 8
 - Across Variables, 3
 - Custom components, 80
 - Adding a Probe, 12
 - Advanced Simulation Settings, 94
 - Alpha, 95
 - Analyzing Models
 - Using the API, 144
 - With Apps and Templates, 129
 - Animating the 3-D Model, 125
 - Annotations, 60
 - API, 144
 - Apps
 - Code Generation, 133
 - Equation Extraction, 130
 - Parameter Optimization, 132
 - Apps and Templates
 - Analyzing Your Model with, 129

- Arrow convention, 65, 67
- Attaching Files to a Model, 57
- Attachments palette, 57

B

- Baumgarte, 95
 - Alpha, 95
 - Beta, 95
- Best Practices, 69
 - Building 1-D Translational Models, 67
 - Building Electrical Models, 65
 - Building Hydraulic Models, 69
 - Building Multibody Models, 68
 - Enforcing Initial Conditions, 70
 - Laying Out and Creating Subsystems, 64
 - Simulating and Visualizing a Model, 126
- Beta, 95
- Building a Model
 - Adding and Moving Objects in the 3-D Workspace, 118
 - Assembling a 3-D Model, 116
 - Displaying Attached Shapes as You Build a 3-D Model, 117
 - Moving Objects in the 3-D Workspace, 115
 - Using Do Not Enforce Constraints, 116

C

- CAD Geometry, 117
- Causal modeling, 2, 5, 8
- Code generation
 - C code, 133
 - initialization, 136
 - subsystem, 134
- Compile optimized, 97
- Compiler, 97
- Connection lines, 24
 - Colors, 24
- Connection ports, 24
- Conserved Quantity Flow
 - arrow convention, 16, 67, 92
- Constraint Handling Options, 138

- Constraint projection, 95, 138
 - During event iterations, 95
 - Iterations, 95
 - Tolerance, 95
- Constraint stabilization, 95
- Construct mode, 114
- Custom components
 - Defining equations, 84
 - Defining ports, 85
 - Editing, 83
 - External C Code/DLL, 143, 199
 - modeling from extrapolated data, 191
 - template, 195
 - Understanding Custom Components, 73
- Custom libraries, 58
- Custom plot window, 100

D

- DAE Variables, 202
- Data sets
 - creating in Maple, 63
 - generating for model, 57
- Debugging console, 39
- Diagnostic Messages, 7, 65
- Differential algebraic equations, 1
- DLL
 - custom component, 143
- Drawing, 60
- DynamicSystems package, 144

E

- Embedded components, 144
- EMI Component Options, 138
- Equations
 - App, 205
 - Retrieving, 130
- Error tolerance, 93
- Event hysteresis, 95
- Event Iterations, 95
- Event Projection, 95
- Examples

- Adding Attached Shapes to a Double Pendulum Model, 112
- Adding Text Annotation to a Model, 61
- Assigning a Subsystem Parameter to a Shared Subsystem, 44
- Building a Double Pendulum Model in the 3-D Workspace, 118
- Copying and Pasting a Standalone Subsystem, 40
- Creating a Custom Library from an Existing Model, 58
- Creating a Data Set in Maple, 63
- Creating a Parameter Override, 53
- Creating a Subsystem, 29
- Creating and Using a Parameter Block, 46
- Defining and Assigning a Global Parameter, 42
- Editing Shared Subsystems that are Linked to the Same Subsystem Definition, 34
- Nonlinear Spring-Damper Custom Component, 87
- Plotting Multiple Quantities in Individual Graphs, 101
- Plotting One Quantity Versus Another, 103
- Removing the Link Between a Shared Subsystem and its Subsystem Definition, 37
- Resolving Warning Messages in the Debugging Console, 39

- External C Code/DLL
 - custom component, 143, 199

F

- Fixed time step, 93
- Flow direction, 3

G

- Global parameters, 151
- Grid

3-D grid, 117
 using CAD geometry, 117

H

Help pane, 20
 Hydraulic Systems, 235
 basic hydraulic equations, 209
 basic hydraulic library components, 207, 211
 Bernoulli Equation, 209
 compressibility, 223
 custom accumulator component, 232
 custom component, 234
 Darcy Equation, 209
 fluid inertia, 224
 Friction Factor, 210
 Joukousky equation, 231
 multibody hydraulic, 220
 multidomain, 215
 Pascal's principle, 222
 spool valve, 214
 translational motion, 218
 water hammer, 226

I

Implicit geometry, 111
 Index 1 error control, 95
 Index 1 tolerance, 96
 Initial conditions, 52
 Best practices for enforcing initial conditions, 70
 overrides, 56
 specifying, 26
 Specifying How Initial Conditions Are Enforced, 27
 Initial hysteresis, 95
 Initialization, 136
 Interpolation tables, 159

J

Jacobian, 95

K

Keyboard Shortcuts, 237
 Kinematic Constraints, 116

L

Latest results
 From simulation, 99
 Linear systems
 Analyzing, 131
 Linearization, 131
 LinkModel, 144

M

MapleSim component library, 5, 8, 19
 MapleSim Model
 Embedded components, 144
 MapleSim Window, 7
 Minimize events, 96
 Model tree, 21
 Model Workspace, 6
 Modelica, 89
 Modelica Custom Component, 73
 Modeling components
 Connecting, 11
 Models
 Building, 8
 Multibody
 Best Practices for building multibody models, 68
 Settings, 106
 Multibody Parameter Values, 107

P

Palettes, 7, 19
 Parameter block, 45
 Parameter optimization, 132
 Parameter Override, 53
 Parameters
 Advanced Parameter Settings, 44, 51
 Advanced Variable Settings, 51
 Global parameters, 42
 Parameter sets, 50, 97

- Parameter values, 12, 25
- Subsystem parameters, 44
- Physical Components, 81
- Physical models
 - Analyzing, 127
 - Navigating, 23
- Plot events, 96
- Plot points, 94
- Plot Windows, 105
- Plots
 - Add Second Variable to Graph, 101
 - Set X-Axis Variable, 103
- Port and Parameter Management, 137
- Probes, 91
 - Adding, 12
 - arrow convention, 16, 67, 92
- Probes palette, 97
- Progress information messages, 98
- Projection, 95
- Projection Iterations, 95
- Projection Tolerance, 95

S

- Scaling, 96
- Settings
 - Advanced Simulation, 94
 - Simulation, 92
- Sign convention, 3
- Signal Flow, 79
- Simulating, 97
- Simulation
 - Duration time, 93
 - Initial Conditions, 99
 - Settings, 92
 - Start time, 94
- Simulation Graphs, 105
- Simulation parameters
 - Compiler, 97
 - Settings, 92
- Simulation results
 - Clear message console, 99
 - Comparing, 99
 - Exporting graph data, 101

- Managing, 99
- Progress Messages, 98
- Snapshots, 99
 - Storing, 99
 - Viewing, 99
- Snapshots, 99
 - Using, 94
- Solver, 93
- Solver diagnostics, 96
- Solver type, 93
- Specifying Component Properties, 12
- Standalone subsystem, 38
- State, 99
- Step size, 94
- Stored results
 - From simulation, 99
- Subsystem(s)
 - Adding a Port, 155
 - Adding Multiple Copies of a Subsystem to a Model, 31
 - Adding Subsystem Definitions and Shared Subsystems to a Model, 32
 - code generation, 134
 - Creating and Managing, 28, 150
 - definition, 31
 - Editing multiple instances, 34
 - linking, 32
 - parameters, 44
 - shared, 31
 - standalone, 38

T

- Templates, 129
 - Custom Component, 84
 - Understanding Custom Components, 74
- Through Variables, 3
 - arrow convention, 16, 67, 92
 - custom components, 80
- Time
 - Simulation duration, 93
 - Simulation end, 94
 - Simulation start, 94
- Time step, 93

Trace lines, 106, 112
 example, 112, 114
Tutorials, 147
 Basic Tutorial: Modeling an RLC Circuit
 and DC Motor, 17

U

Units
 Specifying Parameter Units, 25

V

Variable scaling, 96
Variable time step, 93
Visualization, 106
 transparency, 106
Visualization parameters
 Settings, 106

