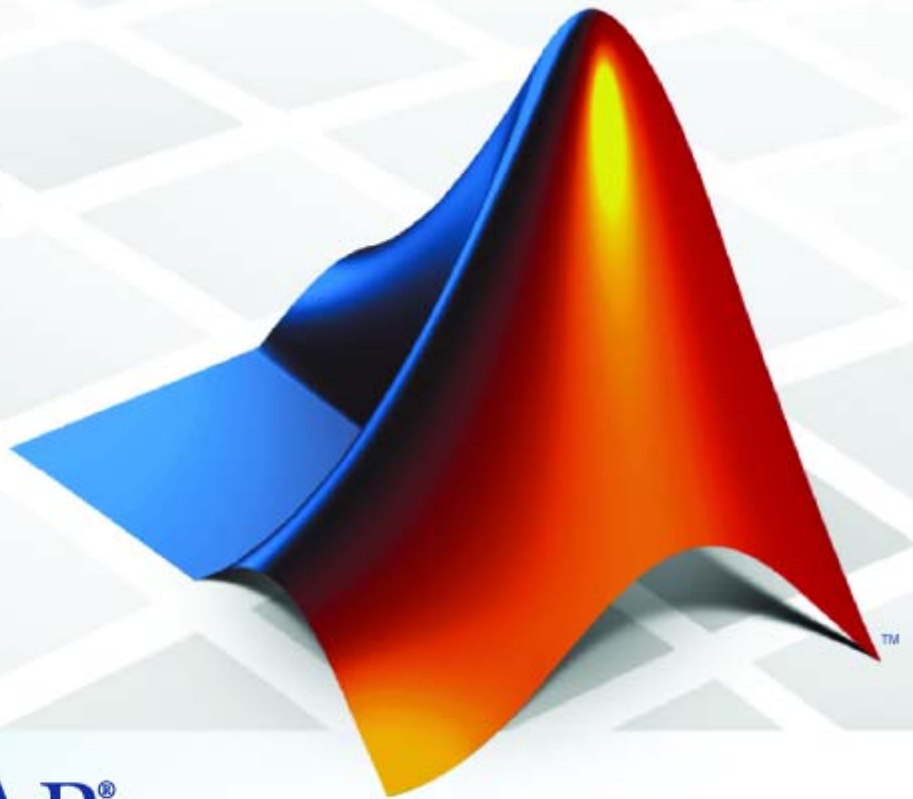


Simscape™ 3

User's Guide



MATLAB®
& SIMULINK®

How to Contact The MathWorks



www.mathworks.com
[comp.soft-sys.matlab](mailto:comp.soft-sys.matlab@mathworks.com)
www.mathworks.com/contact_TS.html

Web
Newsgroup
Technical Support



suggest@mathworks.com
bugs@mathworks.com
doc@mathworks.com
service@mathworks.com
info@mathworks.com

Product enhancement suggestions
Bug reports
Documentation error reports
Order status, license renewals, passcodes
Sales, pricing, and general information



508-647-7000 (Phone)



508-647-7001 (Fax)



The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

Simscape™ User's Guide

© COPYRIGHT 2007–2010 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

The MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

March 2007	Online only	New for Version 1.0 (Release 2007a)
September 2007	Online only	Revised for Version 2.0 (Release 2007b)
March 2008	Online only	Revised for Version 2.1 (Release 2008a)
October 2008	Online only	Revised for Version 3.0 (Release 2008b)
March 2009	Online only	Revised for Version 3.1 (Release 2009a)
September 2009	Online only	Revised for Version 3.2 (Release 2009b)
March 2010	Online only	Revised for Version 3.3 (Release 2010a)

Modeling Physical Systems

1

Basic Principles of Modeling Physical Networks	1-2
Overview of the Physical Network Approach to Modeling	
Physical Systems	1-2
Variable Types	1-4
Building the Mathematical Model	1-5
Direction of Variables	1-6
Connector Ports and Connection Lines	1-8
Connecting Simscape Diagrams to Simulink Sources and Scopes	1-10
Introducing the Simscape Block Libraries	1-11
Library Structure Overview	1-11
Using the Simulink Library Browser to Access the Block Libraries	1-11
Using the Command Prompt to Access the Block Libraries	1-12
Essential Steps to Building a Physical Model	1-14
Building Your Model	1-14
Using the Conserving Ports	1-15
Using the Physical Signal Ports	1-16
Creating a Simple Model	1-17
Building a Simscape Diagram	1-17
Modifying Initial Settings	1-25
Running the Simulation	1-26
Adjusting the Parameters	1-29
Modeling Best Practices	1-35
Grounding Rules	1-35
Avoiding Numerical Simulation Issues	1-38
Modeling Pneumatic Systems	1-42

Intended Applications	1-42
Assumptions and Limitations	1-42
Fundamental Equations	1-43
Network Variables	1-44
Connection Constraints	1-45
References	1-45

Simulating Physical Models

2

How Simscape Simulation Works	2-2
Simscape Simulation Phases	2-2
Model Validation	2-4
Network Construction	2-4
Equation Construction	2-5
Computing Initial Conditions	2-5
Performing Transient Initialization	2-6
Transient Solve	2-6
Working with Solvers	2-8
Selecting a Solver	2-8
Input Filtering	2-10
Troubleshooting Simulation Errors	2-13
Troubleshooting Tips and Techniques	2-13
System Configuration Errors	2-14
Numerical Simulation Issues	2-17
Initial Conditions Solve Failure	2-19
Transient Simulation Issues	2-20
Finding an Operating Point	2-22
What Is an Operating Point?	2-22
How to Find Operating Points	2-23
Finding Operating Points with Simscape, Simulink, and Related Products	2-24
Linearizing at an Operating Point	2-28
What Is Linearization?	2-28
How to Linearize a Model	2-30

Linearizing a Model with Simscape, Simulink, and Related Products	2-30
References	2-34
Generating Code	2-35
About Code Generation from Simscape Models	2-35
Related Simulink Code Generation Documentation	2-35
Reasons for Generating Code	2-36
Using Code-Related Products and Features	2-36
How Simscape Code Generation Differs from Simulink ...	2-37
Limitations	2-39
Sample Time and Solver Restrictions	2-39
Algebraic Loops	2-39
Restricted Simulink Tools	2-40
Unsupported Simulink Tools	2-42
Simulink Tools Not Compatible with Simscape Blocks ...	2-42
Code Generation	2-42

Logging Simulation Data

3

About Simulation Data Logging	3-2
Suggested Workflows	3-2
Limitations	3-2
How to Log Simulation Data	3-3
How to Enable Data Logging	3-3
Data Logging Options	3-4
Data Logging Example	3-6

Working with Physical Units

4

Overview	4-2
Unit Definitions	4-4
Specifying Units in Block Dialogs	4-9
Thermal Unit Conversions	4-11
About Affine Units	4-11
When to Apply Affine Conversion	4-11
How to Apply Affine Conversion	4-12
Angular Units	4-14
References	4-14

Using the Simscape Editing Mode

5

About the Simscape Editing Mode	5-2
Suggested Workflows	5-2
What You Can Do in Restricted Mode	5-3
What You Can Do in Full Mode	5-4
Switching Between Modes	5-4
Working with Block Libraries	5-7
Working with Restricted and Full Modes	5-9
Setting the Model Loading Preference	5-9
Saving a Model in Restricted Mode	5-10
Working with a Model in Restricted Mode	5-13
Switching from Restricted to Full Mode	5-21
Editing Mode Information	5-23
What Is the Current Mode?	5-23
Which Licenses Are Checked Out?	5-23

A

Examples

Getting Started

A-2

Best Practices

A-2

Editing Mode

A-2

Index

Modeling Physical Systems

- “Basic Principles of Modeling Physical Networks” on page 1-2
- “Introducing the Simscape Block Libraries” on page 1-11
- “Essential Steps to Building a Physical Model” on page 1-14
- “Creating a Simple Model” on page 1-17
- “Modeling Best Practices” on page 1-35
- “Modeling Pneumatic Systems” on page 1-42

Basic Principles of Modeling Physical Networks

In this section...
“Overview of the Physical Network Approach to Modeling Physical Systems” on page 1-2
“Variable Types” on page 1-4
“Building the Mathematical Model” on page 1-5
“Direction of Variables ” on page 1-6
“Connector Ports and Connection Lines” on page 1-8
“Connecting Simscape Diagrams to Simulink Sources and Scopes” on page 1-10

Overview of the Physical Network Approach to Modeling Physical Systems

Simscape™ software is a set of block libraries and special simulation features for modeling physical systems in the Simulink® environment. It employs the Physical Network approach, which differs from the standard Simulink modeling approach and is particularly suited to simulating systems that consist of real physical components.

Simulink blocks represent basic mathematical operations. When you connect Simulink blocks together, the resulting diagram is equivalent to the mathematical model, or representation, of the system under design. Simscape technology lets you create a network representation of the system under design, based on the Physical Network approach. According to this approach, each system is represented as consisting of functional elements that interact with each other by exchanging energy through their ports.

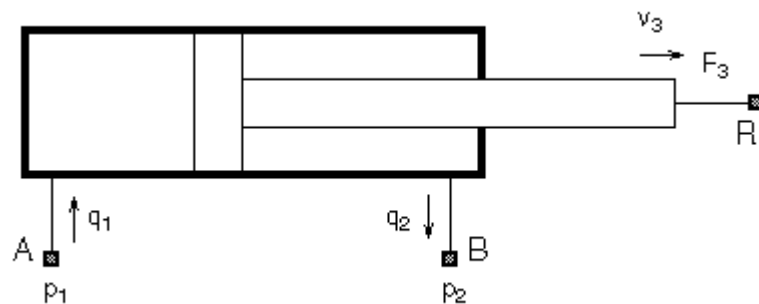
These connection ports are bidirectional. They mimic physical connections between elements. Connecting Simscape blocks together is analogous to connecting real components, such as pumps, valves, and so on. In other words, Simscape diagrams mimic the physical system layout. If physical components can be connected, their models can be connected, too. You do not have to specify flow directions and information flow when connecting Simscape blocks, just as you do not have to specify this information when you connect

real physical components. The Physical Network approach, with its Through and Across variables and bidirectional physical connections, automatically resolves all the traditional issues with variables, directionality, and so on.

The number of connection ports for each element is determined by the number of energy flows it exchanges with other elements in the system, and depends on the level of idealization. For example, a fixed-displacement hydraulic pump in its simplest form can be represented as a two-port element, with one energy flow associated with the inlet (suction) and the other with the outlet. In this representation, the angular velocity of the driving shaft is assumed constant, making it possible to neglect the energy exchange between the pump and the shaft. To account for a variable driving torque, you need a third port associated with the driving shaft.

An energy flow is characterized by its variables. Each energy flow is associated with two variables, one Through and one Across (see “Variable Types” on page 1-4 for more information). Usually, these are the variables whose product is the energy flow in watts. They are called the basic, or conjugate, variables. For example, the basic variables for mechanical translational systems are force and velocity, for mechanical rotational systems—torque and angular velocity, for hydraulic systems—flow rate and pressure, for electrical systems—current and voltage.

The following example illustrates a Physical Network representation of a double-acting hydraulic cylinder.



The element is represented with three energy flows: two flows of hydraulic energy through the inlet and outlet of the cylinder and a flow of mechanical

energy associated with the rod motion. It therefore has the following three connector ports:

- A — Hydraulic conserving port associated with pressure p_1 (an Across variable) and flow rate q_1 (a Through variable)
- B — Hydraulic conserving port associated with pressure p_2 (an Across variable) and flow rate q_2 (a Through variable)
- R — Mechanical translational conserving port associated with rod velocity v_3 (an Across variable) and force F_3 (a Through variable)

See “Connector Ports and Connection Lines” on page 1-8 for more information on connector port types.

Variable Types

Physical Network approach supports two types of variables:

- Through — Variables that are measured with a gauge connected in series to an element.
- Across — Variables that are measured with a gauge connected in parallel to an element.

The following table lists the Through and Across variables associated with each type of physical domain in Simscape software:

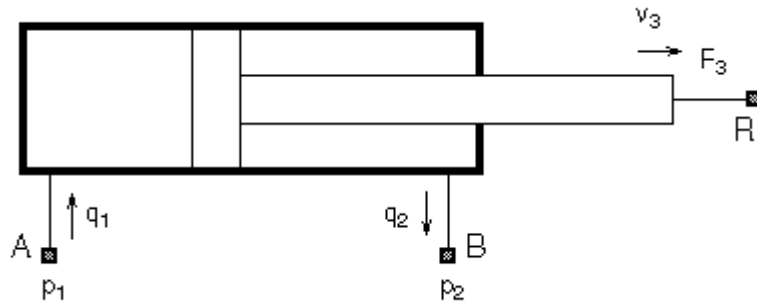
Physical Domain	Across Variable	Through Variable
Electrical	Voltage	Current
Hydraulic	Pressure	Flow rate
Magnetic	Magnetomotive force (mmf)	Flux
Mechanical rotational	Angular velocity	Torque
Mechanical translational	Translational velocity	Force

Physical Domain	Across Variable	Through Variable
Pneumatic	Pressure and temperature	Mass flow rate and heat flow
Thermal	Temperature	Heat flow

Note Generally, the product of each pair of Across and Through variables associated with a domain is power (energy flow in watts). The exceptions are pneumatic domain, where the product of pressure and mass flow rate is not power, and magnetic domain, where the product of mmf and flux is not power, but energy. These result in a pseudo-bond graph.

Building the Mathematical Model

Through and Across variables associated with all the energy flows form the basis of the mathematical model of the block.



For example, the model of a double-acting hydraulic cylinder shown in the previous illustration can be described with a simple set of equations:

$$F_3 = p_1 \cdot A_1 - p_2 \cdot A_2$$

$$q_1 = A_1 \cdot v_3$$

$$q_2 = A_2 \cdot v_3$$

where

q_1, q_2	Flow rates through ports A and B, respectively (Through variables)
p_1, p_2	Gauge pressures at ports A and B, respectively (Across variables)
A_1, A_2	Piston effective areas
F_3	Rod force (Through variable)
v_3	Rod velocity (Across variable)

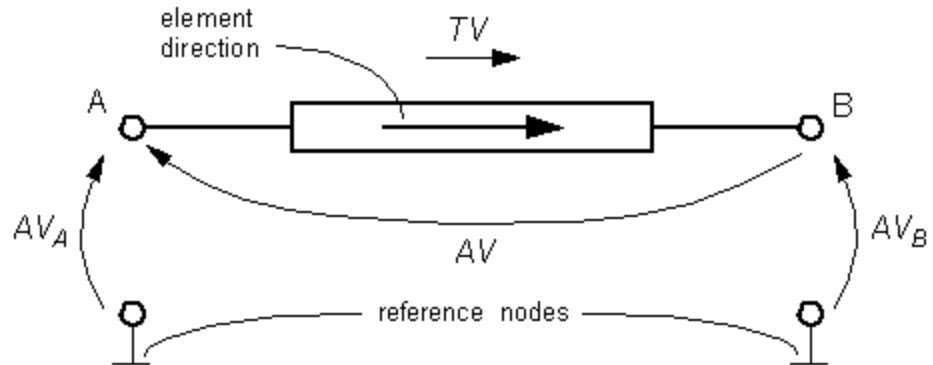
The model could be considerably more complex, for example, it could account for friction, fluid compressibility, inertia of the moving parts, and so on. For all these different mathematical models, however, the element configuration (that is, the number and type of ports and the associated Through and Across variables) would remain the same, meaning that the Physical Network approach lets you substitute models of different levels of complexity without introducing any changes to the schematic. For example, you can start developing your system by using the Resistive Tube block from the Foundation library, which accounts only for friction losses. At a later stage in development, you may want to account for fluid compressibility. You can then replace it with a Hydraulic Pipeline block, available with SimHydraulics® block libraries, or, depending on your application, even with a Segmented Pipeline block if you also need to account for fluid inertia. This modeling principle is called incremental modeling.

Direction of Variables

Each variable is characterized by its magnitude and sign. The sign is the result of measurement orientation. The same variable can be positive or negative, depending on the polarity of a measurement gauge. That is why it is very important to apply exactly the same rule to all the variables in the Physical Network.

Elements with only two ports are characterized with one pair of variables, a Through variable and an Across variable. Since these variables are closely related, their orientation is defined with one direction. For example, if an element is oriented from port A to port B, it implies that the Through variable (*TV*) is positive if it “flows” from A to B, and the Across variable is determined

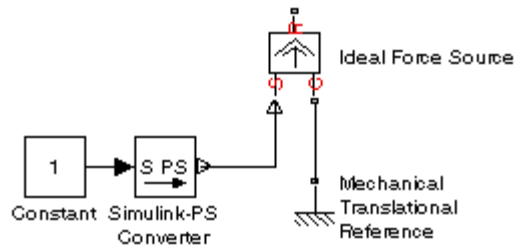
as $AV = AV_A - AV_B$, where AV_A and AV_B are the element node potentials or, in other words, the values of this Across variable at ports A and B, respectively.



This approach to the direction of variables has the following benefits:

- Provides a simple and consistent way to determine whether an element is active or passive. Energy is one of the most important characteristics to be determined during simulation. If the variables direction, or sign, is determined as described above, their product (that is, the energy) is positive if the element consumes energy, and is negative if it provides energy to a system. This rule is followed throughout the Simscape software.
- Simplifies the model description. Symbol A — B is enough to specify variable polarity for both the Across and the Through variables.
- Lets you apply the oriented graph theory to network analysis and design.

As an example of variables direction rules, let us consider the Ideal Force Source block. In this block, as in many other mechanical blocks, port C is associated with the source reference point (case), and port R is associated with the rod.



The block positive direction is from port C to port R. This means that the force is positive if it acts in the direction from C to R, and causes bodies connected to port R to accelerate in the positive direction. The relative velocity is determined as $v = v_C - v_R$, where v_R , v_C are the absolute velocities at ports R and C, respectively, and it is negative if velocity at port R is greater than that at port C. The power generated by the source is computed as the product of force and velocity, and is negative if the source provides energy to the system.

All the elements in a network are divided into active and passive elements, depending on whether they deliver energy to the system or dissipate (or store) it. Active elements (force and velocity sources, flow rate and pressure sources, etc.) must be oriented strictly in accordance with the line of action or function that they are expected to perform in the system, while passive elements (dampers, resistors, springs, pipelines, etc.) can be oriented either way.


Connector Ports and Connection Lines

Simscape blocks may have the following types of ports:

- Physical Conserving ports — Bidirectional ports (for example, hydraulic or mechanical) that represent physical connections and relate physical variables based on the Physical Network approach.
- Physical Signal ports — Unidirectional ports transferring signals that use an internal Simscape engine for computations.

Each of these ports and connections between them are described in greater detail below.

Physical Conserving Ports


Simscape blocks have special Conserving ports . You connect Conserving ports with Physical connection lines, distinct from normal Simulink lines. Physical connection lines have no inherent directionality and represent the exchange of energy flows, according to the Physical Network approach.

- You can connect Conserving ports only to other Conserving ports of the same type.
- The Physical connection lines that connect Conserving ports together are bidirectional lines that carry physical variables (Across and Through variables, as described above) rather than signals. You cannot connect Physical lines to Simulink ports or to Physical Signal ports.
- Two directly connected Conserving ports must have the same values for all their Across variables (such as pressure or angular velocity).
- You can branch Physical connection lines. When you do so, components directly connected with one another continue to share the same Across variables. Any Through variable (such as flow rate or torque) transferred along the Physical connection line is divided among the multiple components connected by the branches. How the Through variable is divided is determined by the system dynamics.

For each Through variable, the sum of all its values flowing into a branch point equals the sum of all its values flowing out.

Each type of Physical Conserving ports used in Simscape blocks uniquely represents a physical modeling domain. For a list of port types, along with the Through and Across variables associated with each type, see the table in “Variable Types” on page 1-4.

Physical Signal Ports

Physical Signal ports  carry signals between Simscape blocks. You connect them with regular connection lines, similar to Simulink signal connections. Physical Signal ports are used in Simscape block diagrams instead of Simulink input and output ports to increase computation speed and avoid issues with algebraic loops. Unlike Simulink signals, which are essentially unitless, physical signals can have units associated with them. You specify the units along with the parameter values in the block dialogs, and Simscape

software performs the necessary unit conversion operations when solving a physical network.

Simscape Foundation library contains, among other sublibraries, a Physical Signals block library. These blocks perform math operations and other functions on physical signals, and allow you to graphically implement equations inside the Physical Network.

Connecting Simscape Diagrams to Simulink Sources and Scopes

Simscape block diagrams use physical signals instead of regular Simulink signals. Therefore, you need converter blocks to connect Simscape diagrams to Simulink sources and scopes.

Use the Simulink-PS Converter block to connect Simulink sources or other Simulink blocks to the inputs of a Physical Network diagram. You can also use it to specify the input signal units. For more information, see the Simulink-PS Converter block reference page.

Use the PS-Simulink Converter block to connect outputs of a Physical Network diagram to Simulink scopes or other Simulink blocks. You can also use it to specify the desired output signal units. For more information, see the PS-Simulink Converter block reference page.

For an example of using converter blocks to connect Simscape diagrams to Simulink sources and scopes, see “Creating a Simple Model” on page 1-17.

Introducing the Simscape Block Libraries

In this section...

“Library Structure Overview” on page 1-11

“Using the Simulink Library Browser to Access the Block Libraries” on page 1-11

“Using the Command Prompt to Access the Block Libraries” on page 1-12

Library Structure Overview

Simscape block library contains two libraries that belong to the Simscape product:

- Foundation library — Contains basic hydraulic, pneumatic, mechanical, electrical, magnetic, thermal, and physical signal blocks, organized into sublibraries according to technical discipline and function performed
- Utilities library — Contains essential environment blocks for creating Physical Networks models

In addition, if you have installed any of the add-on products of the Physical Modeling family, you will see the corresponding libraries under the main Simscape library.

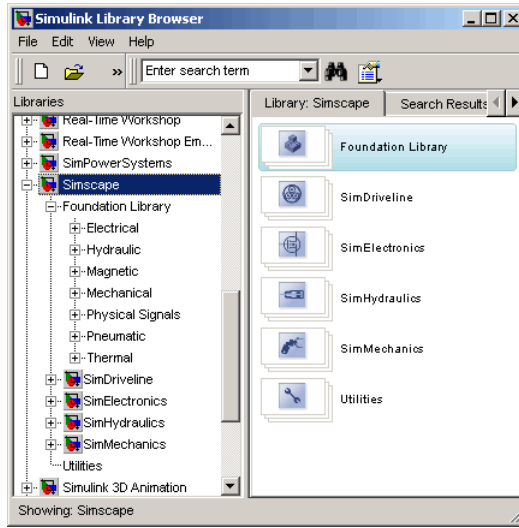
You can combine all these blocks in your Simscape diagrams to model physical systems. You can also use the basic Simulink blocks in your diagrams, such as sources or scopes. See “Connecting Simscape Diagrams to Simulink Sources and Scopes” on page 1-10 for more information on how to do this.

Using the Simulink Library Browser to Access the Block Libraries

You can access the blocks through the Simulink Library Browser. To display the Library Browser, click the **Library Browser** button in the toolbar of the MATLAB® desktop or Simulink model window:



Alternatively, you can type `simulink` in the MATLAB Command Window. Then expand the **Simscape** entry in the contents tree.



For more information on using the Library Browser, see “Library Browser” in the *Simulink Graphical User Interface* documentation.

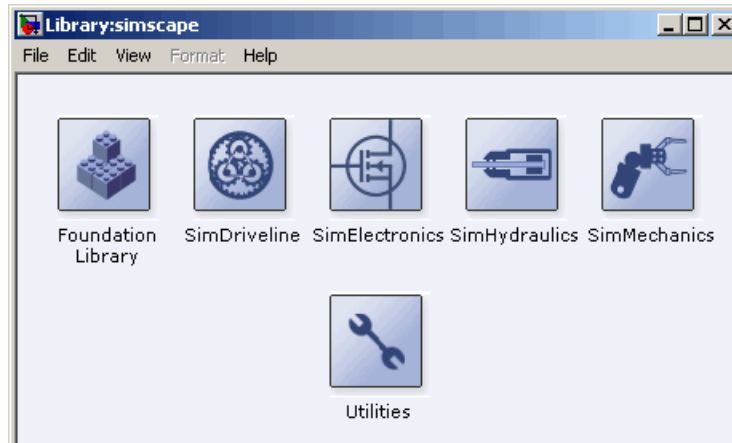
Using the Command Prompt to Access the Block Libraries

To access individual block libraries by using the command prompt:

- To open the Simscape library, type `simscape` in the MATLAB Command Window.
- To open the main Simulink library (to access generic Simulink blocks), type `simulink` in the MATLAB Command Window.

The Simscape library consists of two top-level libraries, Foundation and Utilities. In addition, if you have installed any of the add-on products of the Physical Modeling family, you will see the corresponding libraries under Simscape library, as shown in the following illustration. Some of these libraries contain second-level and third-level sublibraries. You can expand

each library by double-clicking its icon. For more details on library hierarchy and descriptions of block categories, see “Block Reference”.





Essential Steps to Building a Physical Model

Building Your Model

The rules that you must follow when building a physical model with Simscape software are described in “Basic Principles of Modeling Physical Networks” on page 1-2. This section briefly reviews these rules.

- Build your physical model by using a combination of blocks from the Simscape Foundation and Utilities libraries. Simscape software lets you create a network representation of the system under design, based on the Physical Network approach. According to this approach, each system is represented as consisting of functional elements that interact with each other by exchanging energy through their ports.
- Each Simscape diagram (or each topologically distinct physical network in a diagram) must contain a Solver Configuration block from the Simscape Utilities library.
- If you have hydraulic elements in your model, the working fluid used in the hydraulic circuit defines their global parameters, such as fluid density, fluid kinematic viscosity, fluid bulk modulus, and so on. To specify the working fluid, attach a Custom Hydraulic Fluid block (or a Hydraulic Fluid block, available with SimHydraulics block libraries) to each topologically distinct hydraulic circuit. If no Hydraulic Fluid block or Custom Hydraulic Fluid block is attached to a circuit, the hydraulic blocks use the default fluid, which is Skydrol LD-4 at 60°C and with a 0.005 ratio of entrapped air.
- If you have pneumatic elements in your model, default gas properties are for dry air and ambient conditions of 101325 Pa and 20 degrees Celsius. Attach a Gas Properties block to each topologically distinct pneumatic circuit to change gas properties and ambient conditions.
- To connect regular Simulink blocks (such as sources or scopes) to your physical network diagram, use the connector blocks, as described in “Using the Physical Signal Ports” on page 1-16.
- Use the incremental modeling approach. Start with a simple model, run and troubleshoot it, then add the desired special effects. For example, you can start developing your system by using the Resistive Tube block from the Foundation library, which accounts only for friction losses. At a later stage in development, you may want to account for fluid compressibility.

You can then replace it with a Hydraulic Pipeline block, available with SimHydraulics block libraries, or, depending on your application, even with a Segmented Pipeline block if you also need to account for fluid inertia. For all these different mathematical models, the element configuration (that is, the number and type of ports and the associated Through and Across variables) would remain the same, meaning that the Physical Network approach lets you substitute models of different levels of complexity without introducing any changes to the schematic.

Simscape blocks, in general, feature both Conserving ports  and Physical Signal inports and outputs .

Using the Conserving Ports

The following rules apply to Conserving ports:

- There are different types of Physical Conserving ports used in Simscape block diagrams, such as hydraulic, pneumatic, electrical, magnetic, thermal, mechanical translational, and mechanical rotational. Each type has specific Through and Across variables associated with it. For more information, see “Variable Types” on page 1-4.
- You can connect Conserving ports only to other Conserving ports of the same type.
- The Physical connection lines that connect Conserving ports together are bidirectional lines that carry physical variables (Across and Through variables, as described above) rather than signals. You cannot connect Physical lines to Simulink ports or to Physical Signal ports.
- Two directly connected Conserving ports must have the same values for all their Across variables (such as voltage or angular velocity).
- You can branch Physical connection lines. When you do so, components directly connected with one another continue to share the same Across variables. Any Through variable (such as current or torque) transferred along the Physical connection line is divided among the multiple components connected by the branches. How the Through variable is divided is determined by the system dynamics.

For each Through variable, the sum of all its values flowing into a branch point equals the sum of all its values flowing out.

Using the Physical Signal Ports

The following rules apply to Physical Signal ports:

- You can connect Physical Signal ports to other Physical Signal ports with regular connection lines, similar to Simulink signal connections. These connection lines carry physical signals between Simscape blocks.
- You can connect Physical Signal ports to Simulink ports through special converter blocks. Use the Simulink-PS Converter block to connect Simulink outputs to Physical Signal inputs. Use the PS-Simulink Converter block to connect Physical Signal outputs to Simulink inputs.
- Unlike Simulink signals, which are essentially unitless, Physical Signals can have units associated with them. Simscape block dialogs let you specify the units along with the parameter values, where appropriate. Use the converter blocks to associate units with an input signal and to specify the desired output signal units.

For examples of applying these rules when creating an actual physical model, see the following section, “Creating a Simple Model” on page 1-17.

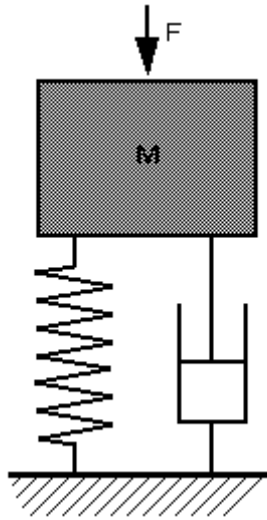
Creating a Simple Model

In this section...
“Building a Simscape Diagram” on page 1-17
“Modifying Initial Settings” on page 1-25
“Running the Simulation” on page 1-26
“Adjusting the Parameters” on page 1-29

Building a Simscape Diagram

In this example, you are going to model a simple mechanical system and observe its behavior under various conditions. This tutorial illustrates the essential steps to building a physical model, described in the previous section, and makes you familiar with using the basic Simscape blocks.

The following schematic represents a simple model of a car suspension. It consists of a spring and damper connected to a body (represented as a mass), which is agitated by a force. You can vary the model parameters, such as the stiffness of the spring, the mass of the body, or the force profile, and view the resulting changes to the velocity and position of the body.



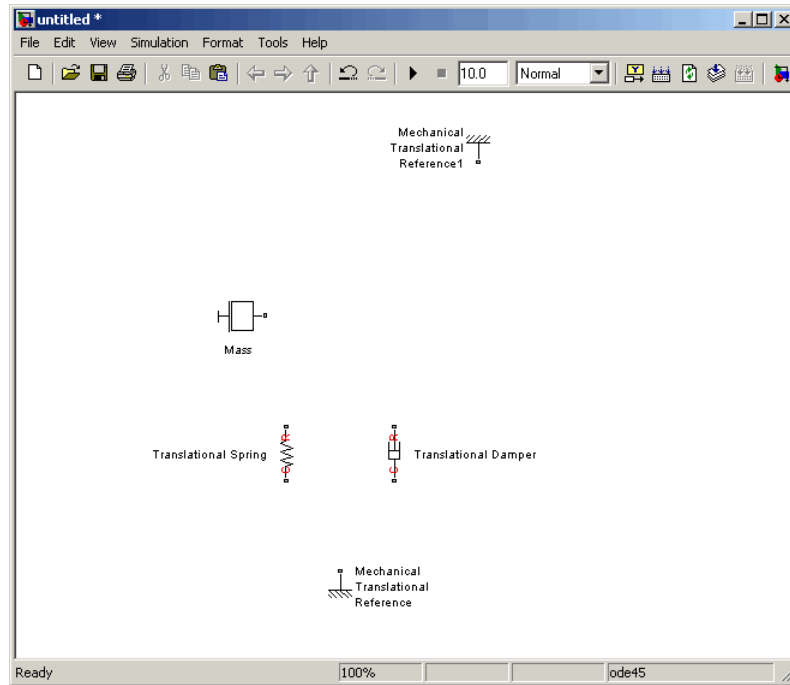
To create an equivalent Simscape diagram, follow these steps:

- 1** Open the Simscape and Simulink block libraries, as described in “Introducing the Simscape Block Libraries” on page 1-11.
- 2** Create a new model. To do this, click the **New** button on the Library Browser’s toolbar (Windows only) or choose **New** from the library window’s **File** menu and select **Model**. The software creates an empty model in memory and displays it in a new model editor window.

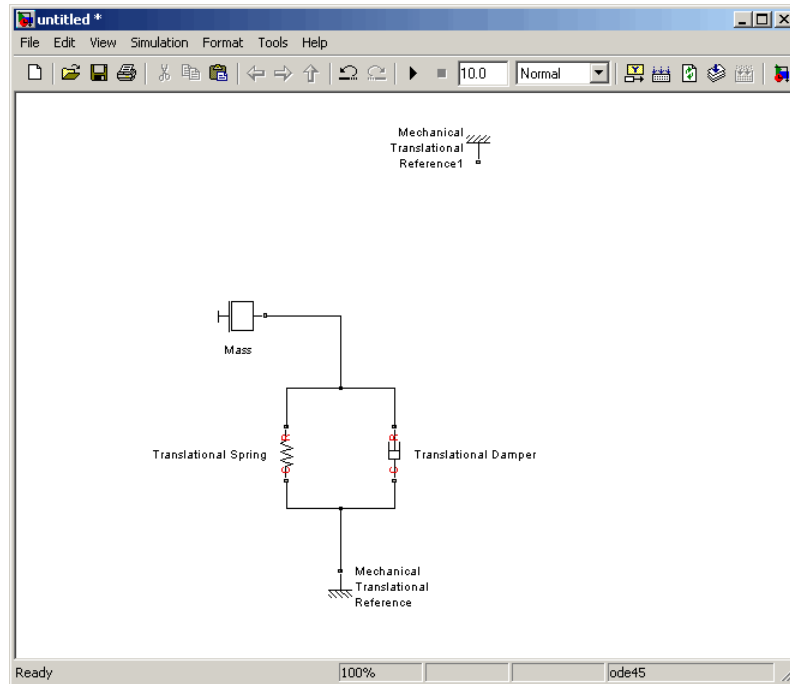
Note Alternately, you can type `ssc_new` at the MATLAB Command prompt, to create a new model prepopulated with certain required and commonly-used blocks. For more information, see “Creating a New Simscape Model”.

- 3** Open the Simscape > Foundation Library > Mechanical > Translational Elements library.
- 4** Drag the Mass, Translational Spring, Translational Damper, and two Mechanical Translational Reference blocks into the model window.

- 5 Orient the blocks as shown in the following illustration. To rotate a block, select it and press **Ctrl+R**.

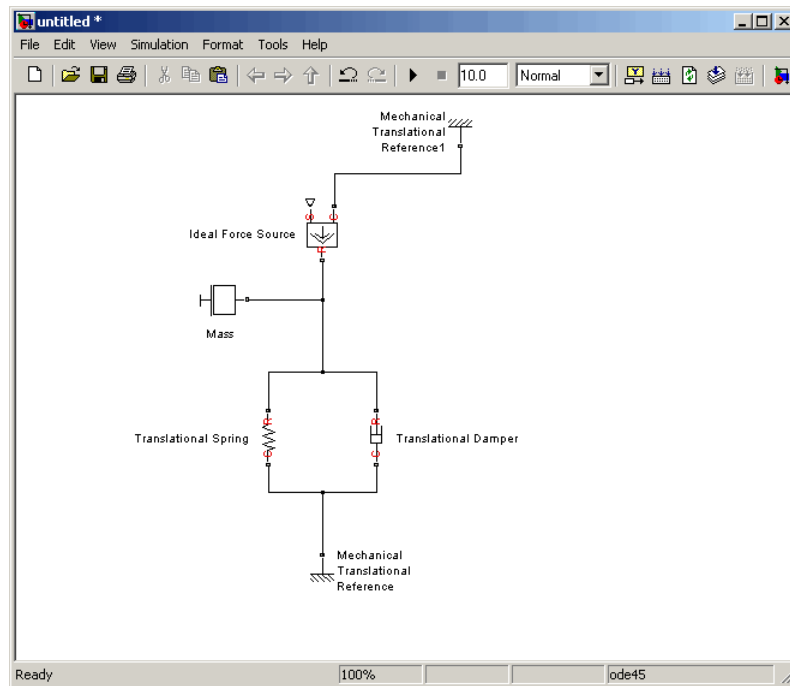


- 6 Connect the Translational Spring, Translational Damper, and Mass blocks to one of the Mechanical Translational Reference blocks as shown in the next illustration.

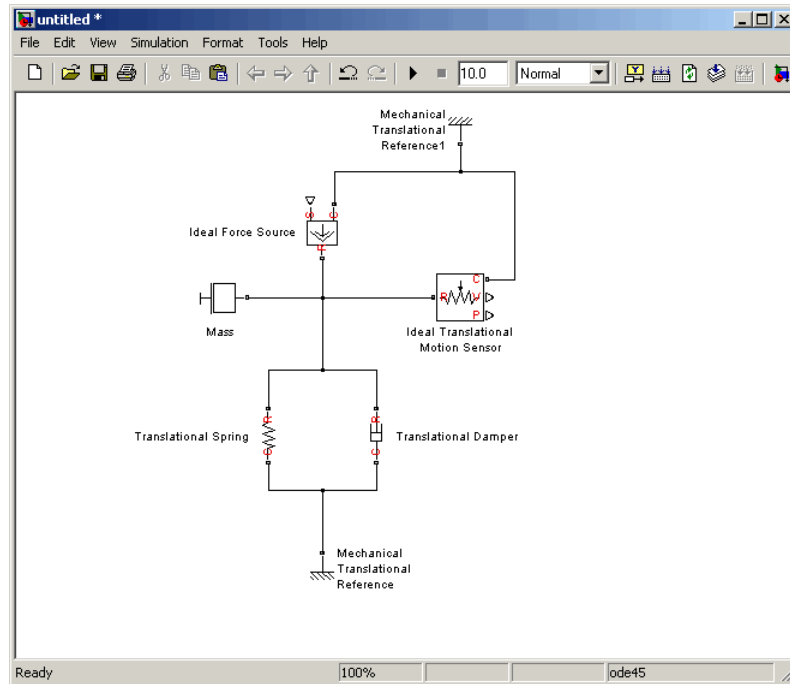


- 7 To add the representation of the force acting on the mass, open the Simscape > Foundation Library > Mechanical > Mechanical Sources library and add the Ideal Force Source block to your diagram.

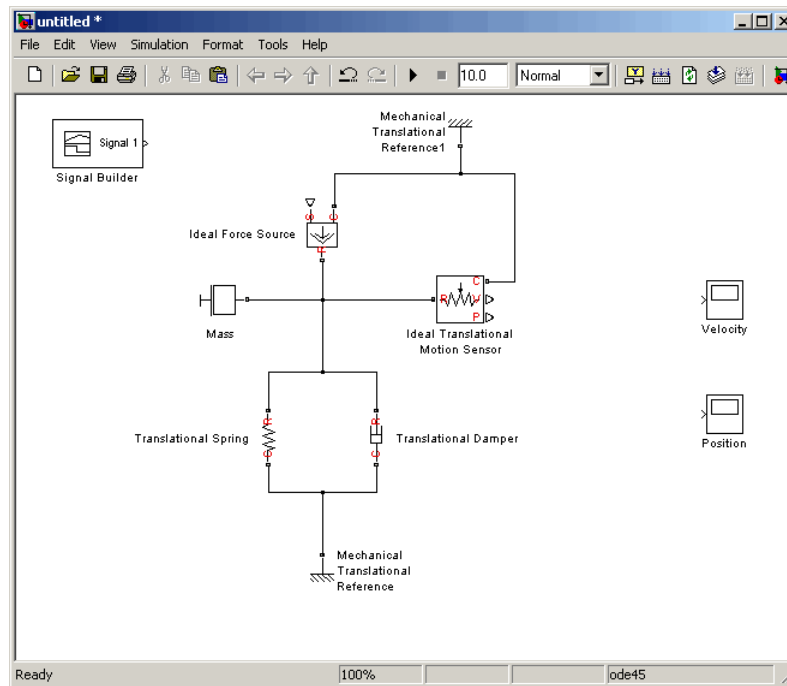
To reflect the correct direction of the force shown in the original schematic, flip the block by selecting **Format > Flip Block > Up-Down** from the top menu bar of the model window. Connect the block's port C (for "case") to the second Mechanical Translational Reference block, and its port R (for "rod") to the Mass block, as shown below.



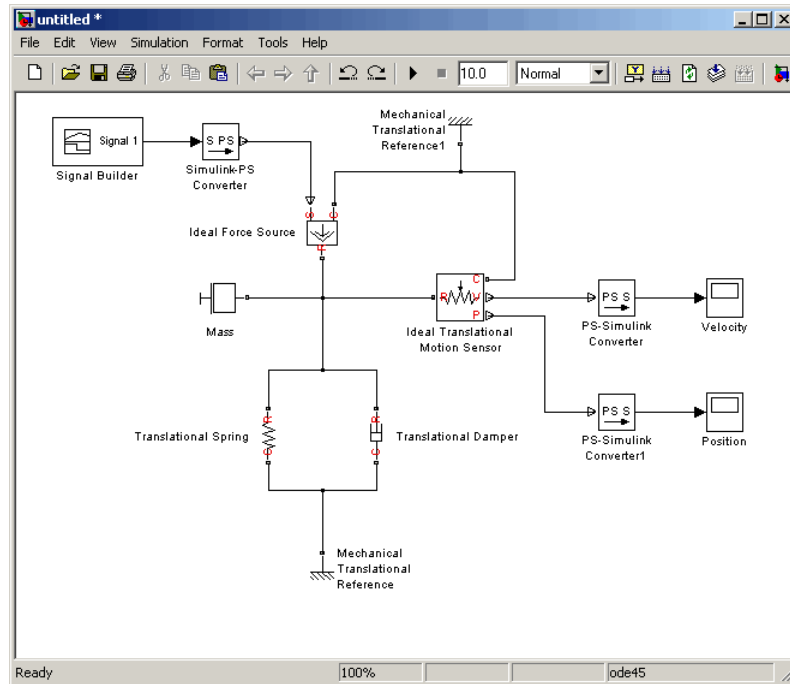
- 8** Add the sensor to measure speed and position of the mass. Place the Ideal Translational Motion Sensor block from the Mechanical Sensors library into your diagram and connect it as shown below.



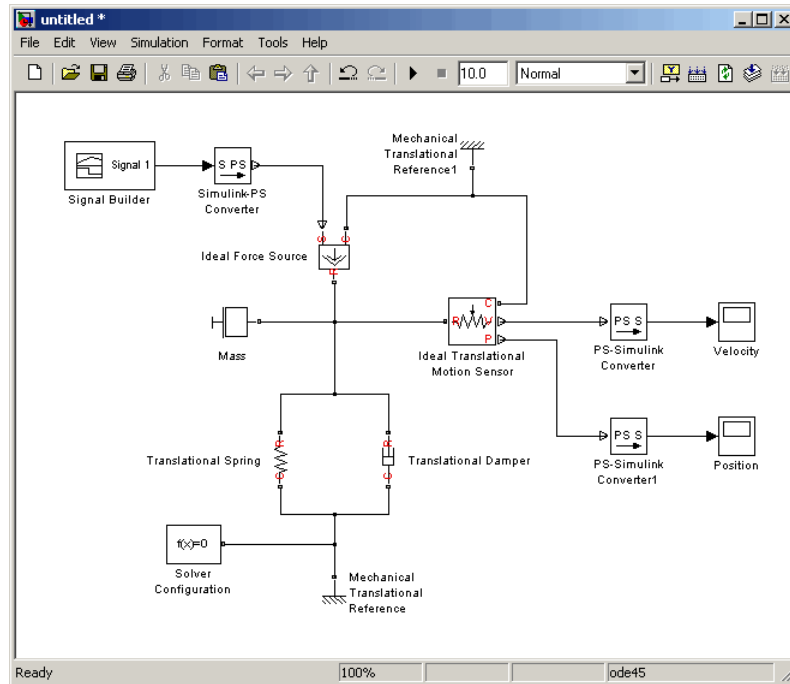
- 9 Now you need to add the sources and scopes. They are found in the regular Simulink libraries. Open the Simulink > Sources library and copy the Signal Builder block into the model. Then open the Simulink > Sinks library and copy two Scope blocks. Rename one of the Scope blocks to Velocity and the other to Position.



- 10** Every time you connect a Simulink source or scope to a Simscape diagram, you have to use an appropriate converter block, to convert Simulink signals into physical signals and vice versa. Open the Simscape > Utilities library and copy a Simulink-PS Converter block and two PS-Simulink Converter blocks into the model. Connect the blocks as shown below.



- 11 Each topologically distinct physical network in a diagram requires exactly one Solver Configuration block, found in the Simscape > Utilities library. Copy this block into your model and connect it to the circuit by creating a branching point and connecting it to the only port of the Solver Configuration block. Your diagram now should look like this.



12 Your block diagram is now complete. Save it as `simple_mech1.mdl`.

Modifying Initial Settings

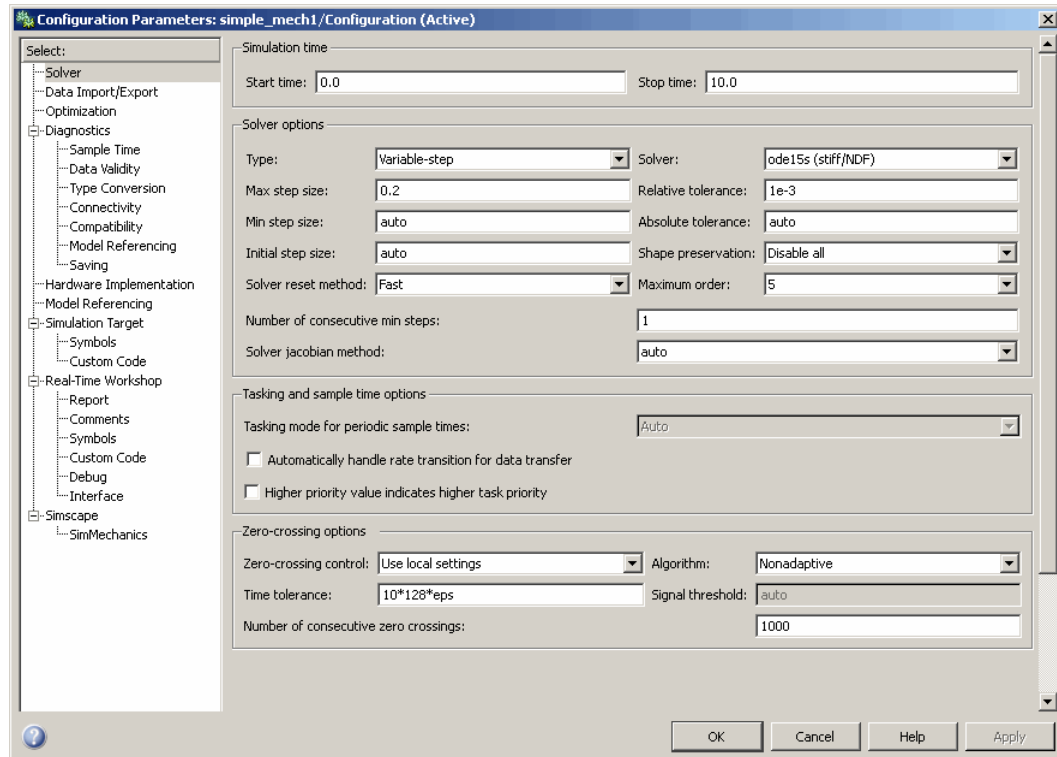
After you have put together a block diagram of your model, as described in the previous section, you need to select a solver and provide the correct values for configuration parameters.

To prepare for simulating the model, follow these steps:

- 1** Select a Simulink solver. On the top menu bar of the model window, select **Simulation > Configuration Parameters**. The Configuration Parameters dialog box opens, showing the **Solver** node.

Under **Solver options**, set **Solver** to `ode15s` (Stiff/NDF) and **Max step size** to 0.2.

Also note that **Simulation time** is specified to be between 0 and 10 seconds. You can adjust this setting later, if needed.



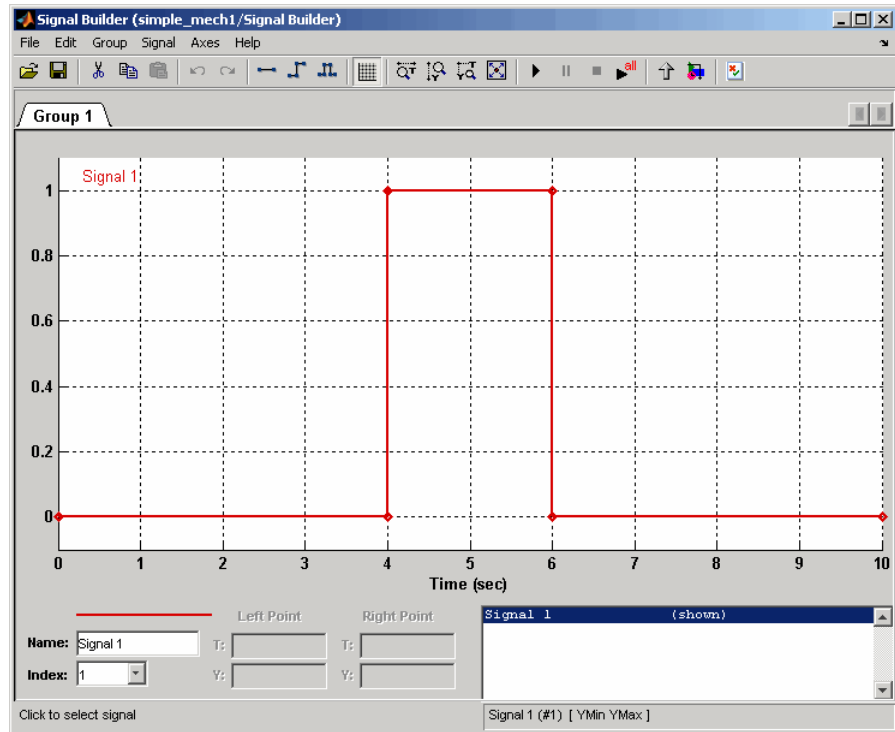
Click **OK** to close the Configuration Parameters dialog box.

2 Save the model.


Running the Simulation

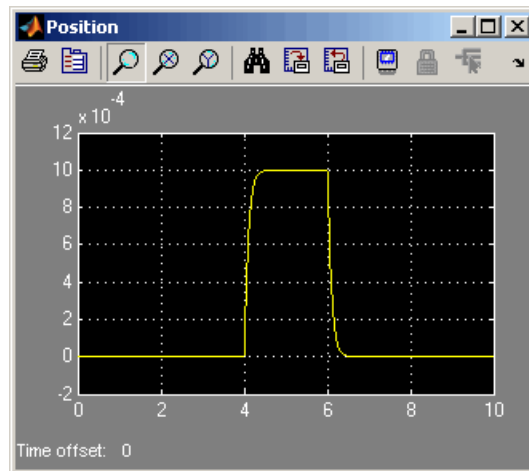
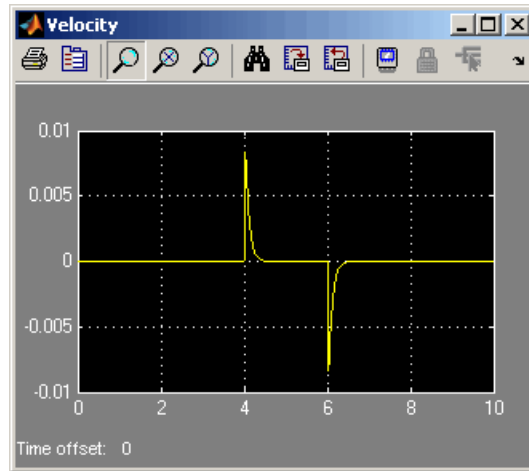
After you've put together a block diagram and specified the initial settings for your model, you can run the simulation.

1 The input signal for the force is provided by the Signal Builder block. The signal profile is shown in the illustration below. It starts with a value of 0, then at 4 seconds there is a step change to 1, and then it changes back to 0 at 6 seconds. This is the default profile.



The Velocity scope outputs the mass velocity, and the Position scope outputs the mass displacement as a function of time. Double-click both scopes to open them.

- 2 To run the simulation, click  in the model window toolbar. The Simscape solver evaluates the model, calculates the initial conditions, and runs the simulation. For a detailed description of this process, see “How Simscape Simulation Works” on page 2-2. Completion of this step may take a few seconds. The message in the bottom-left corner of the model window provides the status update.
- 3 Once the simulation starts running, the Velocity and Position scope windows display the simulation results, as shown in the next illustration.



In the beginning, the mass is at rest. Then at 4 seconds, as the input signal changes abruptly, the mass velocity spikes in the positive direction and gradually returns to zero. The mass position at the same time changes more gradually, on account of inertia and damping, and stays at the new value as long as the force is acting upon it. At 6 seconds, when the input signal changes back to zero, the velocity gets a mirror spike, and the mass gradually returns to its initial position.

You can now adjust various inputs and block parameters and see their effect on the mass velocity and displacement.

Adjusting the Parameters

After running the initial simulation, you can experiment with adjusting various inputs and block parameters.

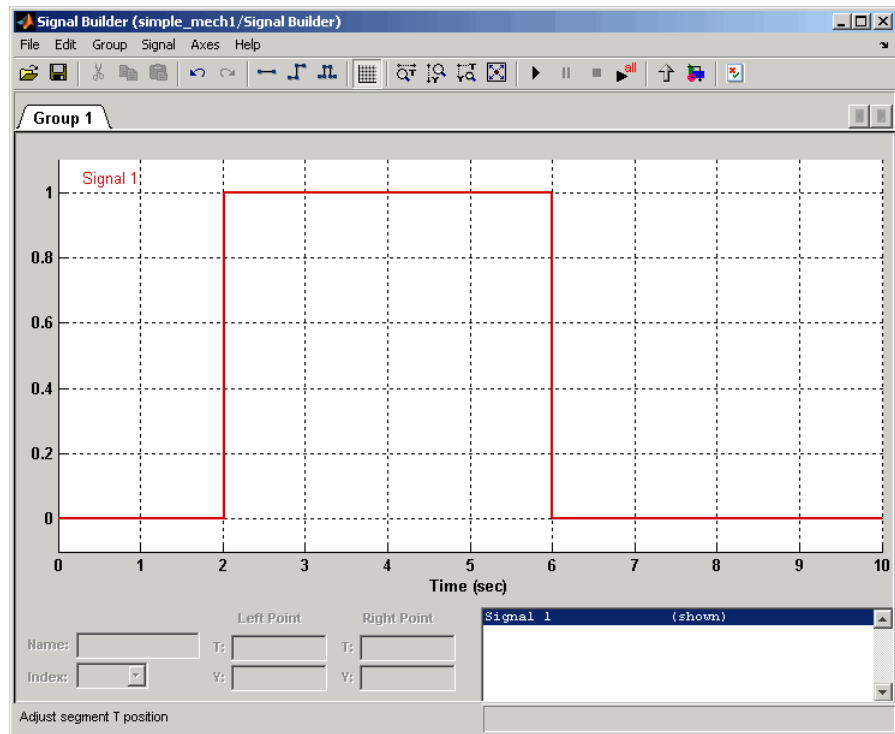
Try the following adjustments:

- 1 Change the force profile.
- 2 Change the model parameters.
- 3 Change the mass position output units.

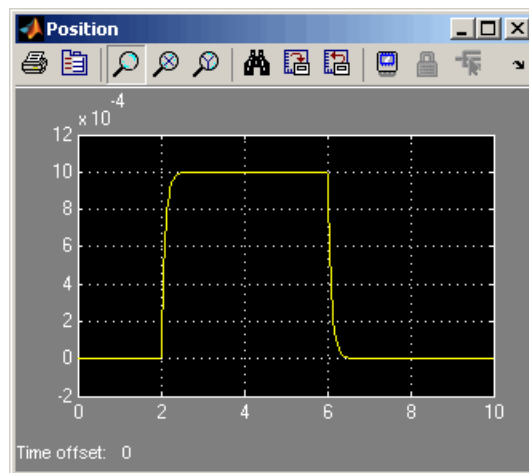
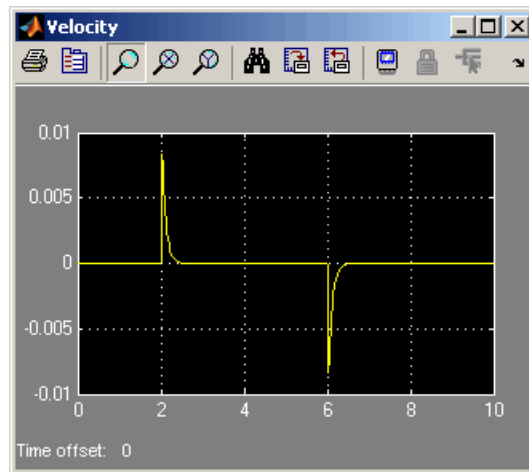
Changing the Force Profile

This example shows how a change in the input signal affects the force profile, and therefore the mass displacement.

- 1 Double-click the Signal Builder block to open it.
- 2 Click the first vertical segment of the signal profile and drag it from 4 to 2 seconds, as shown below. Close the block dialog.



3 Run the simulation. The simulation results are shown in the following illustration.

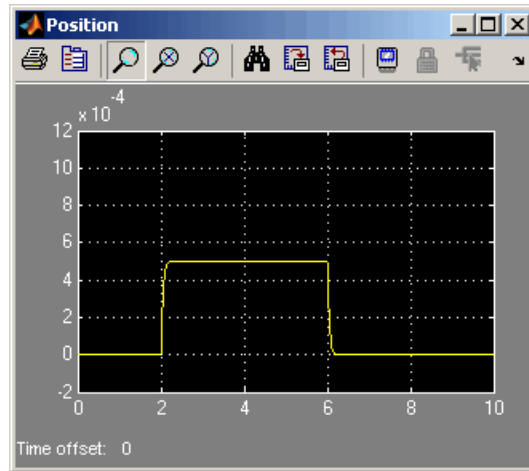


Changing the Model Parameters

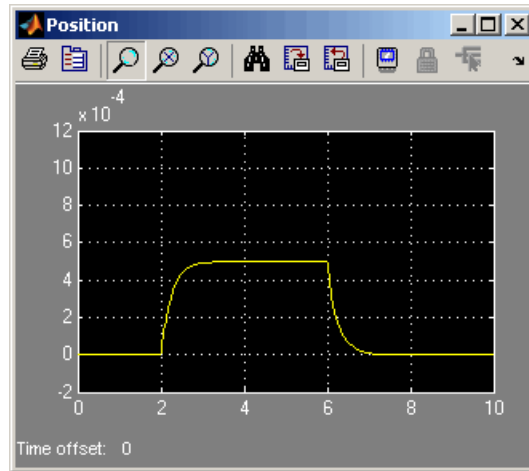
In our model, the force acts on a mass against a translational spring and damper, connected in parallel. This example shows how changes in the spring stiffness and damper viscosity affect the mass displacement.

- 1 Double-click the Translational Spring block. Set its **Spring rate** to 2000 N/m.

- 2** Run the simulation. The increase in spring stiffness results in smaller amplitude of mass displacement, as shown in the following illustration.




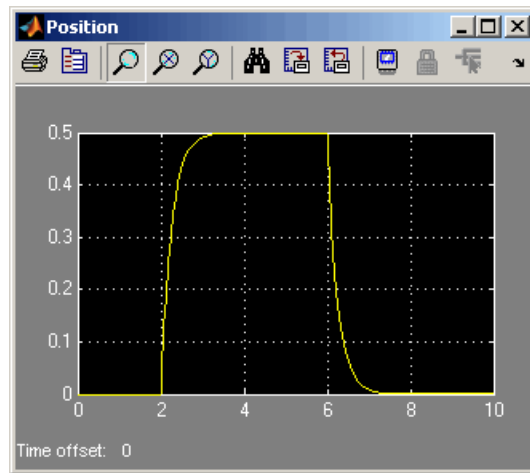
- 3** Next, double-click the Translational Damper block. Set its **Damping coefficient** to 500 N/(m/s).
- 4** Run the simulation. Because of the increase in viscosity, the mass is slower both in reaching its maximum displacement and in returning to the initial position, as shown in the following illustration.



Changing the Mass Position Output Units

In our model, we have used the PS-Simulink Converter block in its default parameter configuration, which does not specify units. Therefore, the Position scope outputs the mass displacement in the default length units, that is, in meters. This example shows how to change the output units for the mass displacement to millimeters.

- 1 Double-click the PS-Simulink Converter block. Type mm in the **Output signal unit** combo box and click **OK**.
- 2 Run the simulation. In the Position scope window, click  to autoscale the scope axes. The mass displacement is now output in millimeters, as shown in the following illustration.



Modeling Best Practices

In this section...
“Grounding Rules” on page 1-35
“Avoiding Numerical Simulation Issues” on page 1-38

Grounding Rules

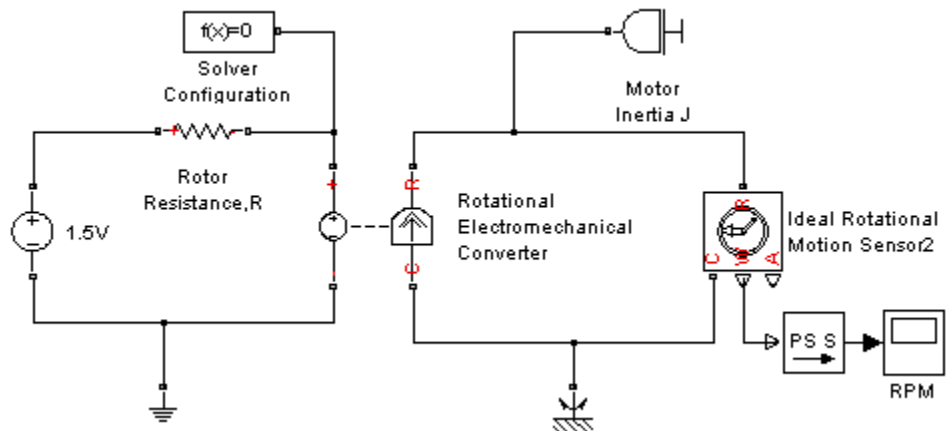
This section contains guidelines for using domain-specific reference blocks (such as Electrical Reference, Mechanical Translational Reference, and so on) in Simscape diagrams, along with examples of correct and incorrect configurations.

Add reference blocks to your models according to the following rules:

- “Each Domain Requires at Least One Reference Block” on page 1-35
- “Each Circuit Requires at Least One Reference Block” on page 1-36
- “Multiple Connections to the Domain Reference Are Allowed Within a Circuit” on page 1-37

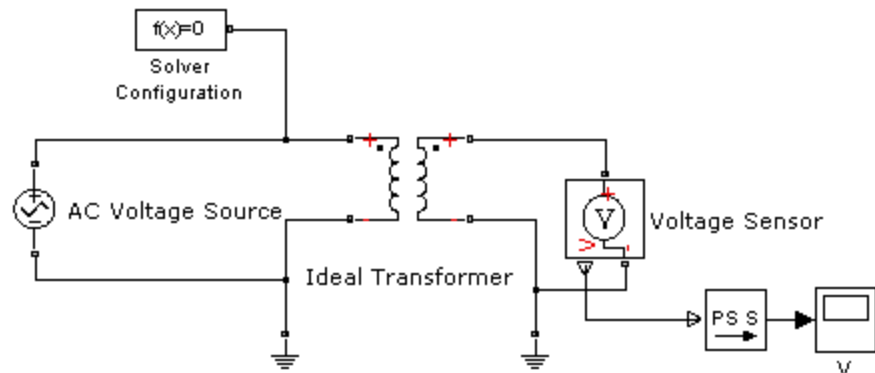
Each Domain Requires at Least One Reference Block

Within a physical network, each domain must contain at least one reference block of the appropriate type. For example, the electromechanical model shown in the following diagram has both Electrical Reference and Rotational Reference blocks attached to the appropriate circuits.

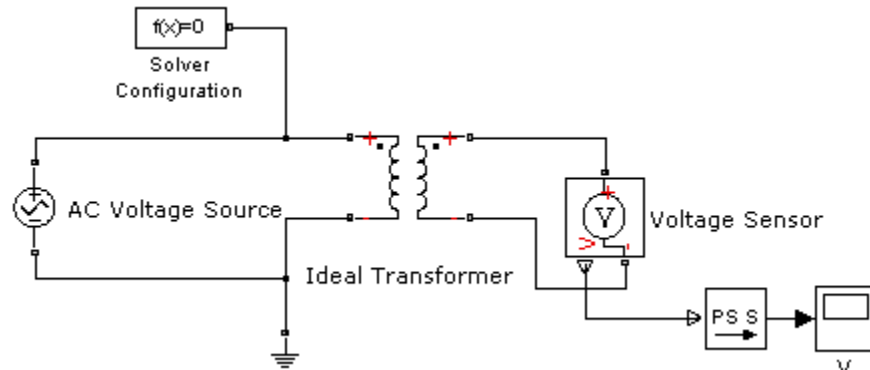


Each Circuit Requires at Least One Reference Block

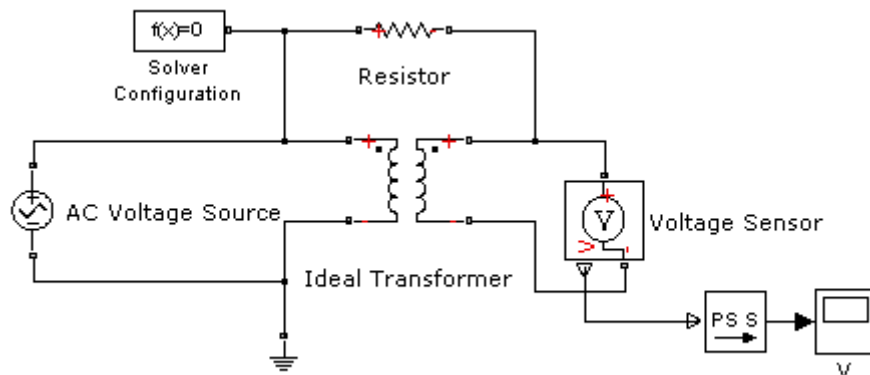
Each topologically distinct circuit within a domain must contain at least one reference block. Some blocks, such as an Ideal Transformer, interface two parts of the network but do not convey information about signal levels relative to the reference block. In the following diagram, there are two separate electrical circuits, and the Electrical Reference blocks are required on both sides of the Ideal Transformer block.



The next diagram would produce an error because it is lacking an electrical reference in the circuit of the secondary winding.



The following diagram, however, will not produce an error because the resistor defines the output voltage relative to the ground reference.

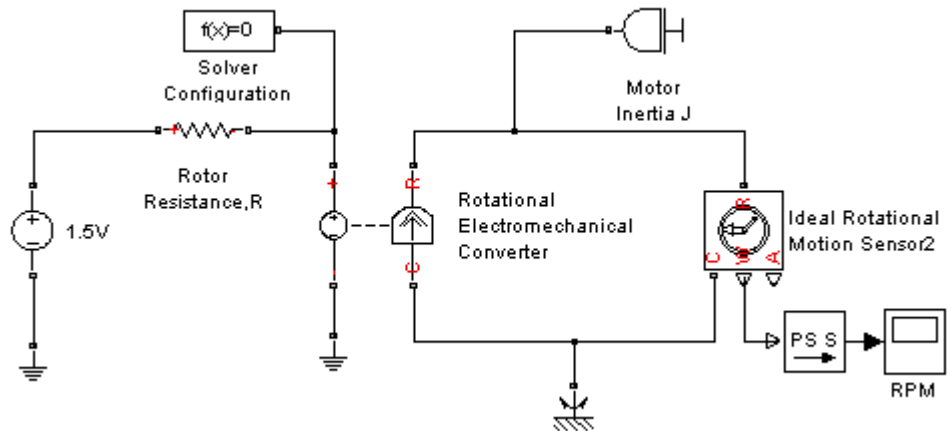


Multiple Connections to the Domain Reference Are Allowed Within a Circuit

More than one reference block may be used within a circuit to define multiple connections to the domain reference:

- Electrical conserving ports of all the blocks that are directly connected to ground must be connected to an Electrical Reference block.
- All translational ports that are rigidly clamped to the frame (ground) must be connected to a Mechanical Translational Reference block.
- All rotational ports that are rigidly clamped to the frame (ground) must be connected to a Mechanical Rotational Reference block.
- Hydraulic conserving ports of all the blocks that are referenced to atmosphere (for example, suction ports of hydraulic pumps, or return ports of valves, cylinders, pipelines, if they are considered directly connected to atmosphere) must be connected to a Hydraulic Reference block.

For example, the following diagram correctly indicates two separate connections to an electrical ground.



Avoiding Numerical Simulation Issues

Certain configurations of physical modeling blocks can cause numerical difficulties or slow down your simulation. When this happens, Simscape solver issues a warning in the MATLAB workspace and, if it fails to initialize, a Simscape error.

In electrical circuits, common examples that can cause this behavior include voltage sources connected in parallel with capacitors, inductors connected in series with current sources, voltage sources connected in parallel, and current sources connected in series. Often, the cause of the numerical difficulty is immediately apparent. For example, two voltage sources in parallel must have identical voltage values; otherwise, the ports connecting them would not be physical conserving ports. In practical circuits, topologies such as parallel voltage sources are possible, and small difference in their instantaneous voltages is possible due to parasitic series resistance.

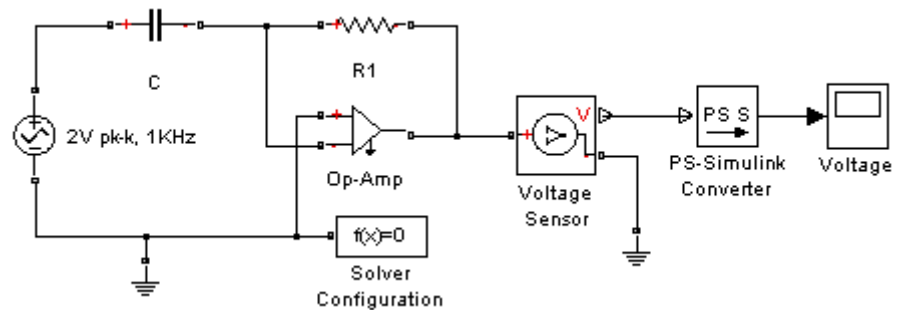
Note Mathematically, these topologies result in *Index-2 differential algebraic equations* (DAEs). Their solution requires two differentiations of the constraint equations and, as such, it is numerically better to avoid these component topologies where possible.

There are two approaches to resolving these difficulties. The first is to change the circuit to an equivalent simpler one. In the example of two parallel voltage sources, one source can be simply deleted. The same applies to two series current sources, the deleted one being replaced by a short circuit. For some circuit topologies, however, it is not possible to find an equivalent simpler one that resolves the problem, and the second approach is needed.

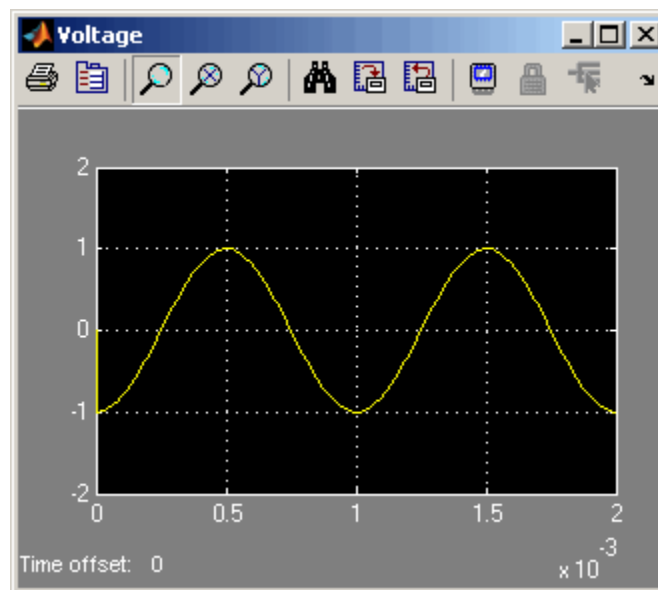
The second approach is to include small parasitic resistances in the component. In the Simscape Foundation library, the Capacitor and Inductor blocks include such parasitic terms, so that you can connect capacitances in parallel with voltage sources and inductors in series with current sources. If your circuit does not have any such topologies, then you can change the default parasitic terms to zero. Note that other blocks do not contain these parasitic terms, for example, the Mutual Inductor block. Therefore, if you wanted to connect a mutual inductor primary in series with a current source, you would need to introduce your own parasitic conductance across the primary winding.

Example of Using a Parasitic Resistance to Avoid Numerical Simulation Issues

The following diagram models a differentiator that might be used as part of a Proportional-Integral-Derivative (PID) controller. You can open this model by typing `ssc_differentiator` in the MATLAB Command Window.



Simulate the model, and you will see that the output is minus the derivative of the input sinusoid.



Now open the capacitor C block dialog, and set the series resistance to zero. The model now runs very slowly, and issues a warning:

```
Warning: problems possible for transient initialization, as well as stepsize control
for transient solve, due to equations of one or more components:
'ssc_differentiator/2V pk-k, 1KHz'
```

```
'ssc_differentiator/Op-Amp'  
'ssc_differentiator/C'
```

The cause of the warning is that the circuit effectively connects the voltage source in parallel with the capacitor. This is because an ideal op-amp satisfies $V_+ = V_-$, where V_+ and V_- are the noninverting and inverting inputs, respectively. This is an example where it is not possible to replace the circuit with an equivalent simpler one, and a parasitic small resistance has to be introduced.

Modeling Pneumatic Systems

In this section...
“Intended Applications” on page 1-42
“Assumptions and Limitations” on page 1-42
“Fundamental Equations” on page 1-43
“Network Variables” on page 1-44
“Connection Constraints” on page 1-45
“References” on page 1-45

Intended Applications

The Foundation library contains basic pneumatic elements, such as orifices, chambers, and pneumatic-mechanical converters, as well as pneumatic sensors and sources. Use these blocks to model pneumatic systems, for applications such as:

- Factory automation — basic pneumatic linear/rotational actuators, valves (variable orifices), and air supply
- Robotics — robotic arms and haptic interfaces
- Gaseous transportation systems and pipelines

You can also use these blocks to model dry air and low-pressure flows, for example, for HVAC applications.

Assumptions and Limitations

Pneumatic block models are based on the following assumptions:

- Working fluid is an ideal gas satisfying the ideal gas law.
- Specific heats at constant pressure and constant volume, c_p and c_v , are constant.

- Processes are adiabatic, that is, there is no heat transfer between components and the environment (except for components with a separate thermal port).
- Gravitational effects can be neglected, that is, underlying equations contain no head pressures due to gravity.

Fundamental Equations

The energy balance for a control volume [1] is

$$\frac{dE_{cv}}{dt} = Q_{cv} - W_{cv} + \sum_i \left(m_i \left(h_i + \frac{v_i^2}{2} + gz_i \right) \right) - \sum_o \left(m_o \left(h_o + \frac{v_o^2}{2} + gz_o \right) \right)$$

where

E_{cv}	Control volume total energy
Q_{cv}	Heat energy per second added to the gas through the boundary
W_{cv}	Mechanical work per second performed by the gas
h_i, h_o	Inlet and outlet enthalpies
v_i, v_o	Gas inlet and outlet velocities
g	Acceleration due to gravity
z_i, z_o	Elevations at inlet and outlet ports
m_i, m_o	Mass flow rates in and out of the control volume

The equation is an accounting balance for the energy of the control volume. It states that the rate of energy increase or decrease within the control volume equals the difference between the rates of energy transfer in and out across the boundary. The mechanisms of energy transfer are heat and work, as for closed systems, and the energy that accompanies the mass entering and exiting.

Pneumatic block models make several simplifying assumptions, as described previously.

The ideal gas law relates pressure, density, and temperature:

$$p = \rho RT$$

where

p	Absolute pressure
ρ	Gas density
R	Specific gas constant
T	Absolute gas temperature

Also, the specific enthalpies for an ideal gas at temperature T and constant pressure and constant volume are given by:

$$h = c_p T$$

$$h = c_v T$$

The pneumatic components also use the mass continuity equation:

$$\frac{d\rho}{dt} = \frac{1}{V}(m_i - m_o)$$

where ρ is the density of the gas within the component. For components with no internal mass of gas, the equation simplifies to:

$$G = m_i = m_o$$

where G is the mass flow rate through the component.

For specific equations used in each block, see the block reference pages.

Network Variables

The Across variables are pressure and temperature, and the Through variables are mass flow rate and heat flow. Note that these choices result in

a pseudo-bond graph, because the product of pressure and mass flow rate is not power.

Connection Constraints

Every node in a pneumatic network must have a defined temperature as well as pressure. This rule places some constraints on how you connect the pneumatic elements. In effect, every node should have a volume of fluid associated with it. When the ideal gas law is applied, this volume of fluid determines the relationship between temperature and pressure. Some elements already have a volume of fluid associated with them, and therefore having just one of these components connected to a node satisfies this condition. Such blocks include Constant Volume Pneumatic Chamber, Pneumatic Piston Chamber, Rotary Pneumatic Piston Chamber, and Pneumatic Atmospheric Reference.

An exception to the above rule (that every node must have a volume of fluid associated with it) occurs when two nodes are connected by a component for which the heat equation says that the temperatures are equal. In this case, just one of the nodes needs to be connected to a component with associated volume of fluid. Such components include the pressure and flow rate sources.

For models that represent an actual pneumatic network, these constraints should have no impact. For example, connecting two orifices in series makes no physical sense because the underlying assumption of the orifice equation is that gas is discharged into a volume of fluid. Therefore, modeling actual physical systems should automatically satisfy these constraints.

References

[1] Moran M.J. and Shapiro H.N. *Fundamentals of Engineering Thermodynamics*. Second edition. New York: John Wiley & Sons, 1992.

Simulating Physical Models

- “How Simscape Simulation Works” on page 2-2
- “Working with Solvers” on page 2-8
- “Troubleshooting Simulation Errors” on page 2-13
- “Finding an Operating Point” on page 2-22
- “Linearizing at an Operating Point” on page 2-28
- “Generating Code” on page 2-35
- “Limitations” on page 2-39

How Simscape Simulation Works

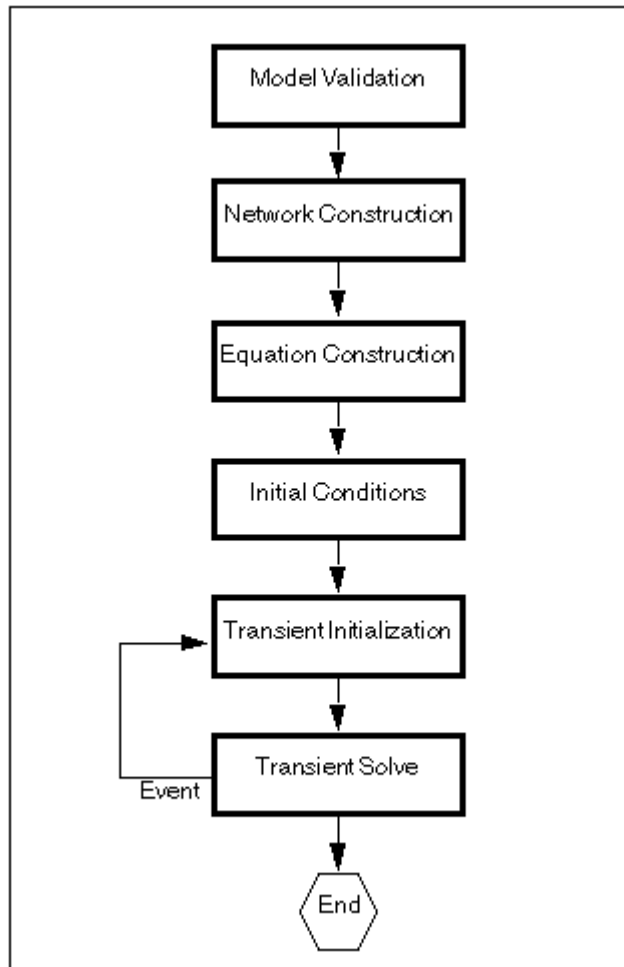
In this section...
“Simscape Simulation Phases” on page 2-2
“Model Validation” on page 2-4
“Network Construction” on page 2-4
“Equation Construction” on page 2-5
“Computing Initial Conditions” on page 2-5
“Performing Transient Initialization” on page 2-6
“Transient Solve” on page 2-6

Simscape Simulation Phases

Simscape software gives you multiple ways to simulate and analyze physical systems in the Simulink environment. Running a physical model simulation is similar to running a simulation of any other Simulink model. It entails setting various simulation options, starting the simulation, and viewing the simulation results. See the Using Simulink documentation for a general discussion of these topics. This chapter focuses on aspects of simulation specific to Simscape and SimHydraulics models. Refer to the SimMechanics™ and SimDriveline™ documentation for specifics of simulating and analyzing SimMechanics and SimDriveline models.

You might find this brief overview helpful for constructing models and understanding errors.

Simscape simulation sequence is shown in the following flow chart.



It consists of the following major phases:

- 1** “Model Validation” on page 2-4
- 2** “Network Construction” on page 2-4
- 3** “Equation Construction” on page 2-5
- 4** “Computing Initial Conditions” on page 2-5

5 “Performing Transient Initialization” on page 2-6

6 “Transient Solve” on page 2-6

Model Validation

Simscape solver first validates the model configuration and checks your data entries from the block dialogs. In particular:

- Each topologically distinct physical network in a diagram requires exactly one Solver Configuration block.
- If your model contains hydraulic elements, each topologically distinct hydraulic circuit in a diagram requires a Custom Hydraulic Fluid block (or Hydraulic Fluid block, available with SimHydraulics block libraries) to be connected to it. These blocks define the fluid properties that act as global parameters for all the blocks connected to the hydraulic circuit. If no hydraulic fluid block is attached to a loop, the hydraulic blocks in this loop use the default fluid. However, more than one hydraulic fluid block in a loop generates an error.

Similarly, if your model contains pneumatic elements, default gas properties for a pneumatic network are for dry air and ambient conditions of 101325 Pa and 20 degrees Celsius. Attaching a Gas Properties block to a pneumatic circuit lets you change gas properties and ambient conditions for all the blocks connected to the circuit. However, more than one Gas Properties block in a pneumatic circuit generates an error.

- Signal units specified in a Simulink-PS Converter block must match the input type expected by the Simscape block connected to it. For example, when you provide the input signal for an Ideal Angular Velocity Source block, specify angular velocity units, such as rad/s or rpm, in the Simulink-PS Converter block, or leave it unitless. Similarly, units specified in a PS-Simulink Converter block must match the type of physical signal provided by the Simscape block output.

Network Construction

After validating the model, Simscape solver constructs the physical network based on the following principles:

- Two directly connected Conserving ports have the same values for all their Across variables (such as voltage or angular velocity).

- Any Through variable (such as current or torque) transferred along the Physical connection line is divided among the multiple components connected by the branches. For each Through variable, the sum of all its values flowing into a branch point equals the sum of all its values flowing out.

Equation Construction

Based on the network configuration, the parameter values provided in the block dialogs, and the global parameters defined by the fluid properties, if applicable, Simscape solver constructs the system of equations for the model.

These equations contain variables of the following types:

- *Dynamic* — Time derivative of this variable appears in equations. Dynamic variables are the independent states for simulation.
- *Algebraic* — Time derivative of this variable does not appear in equations. Algebraic variables are always dependent (on dynamic variables, other algebraic variables, or inputs).

Computing Initial Conditions

Simscape solver computes the initial conditions only once, in the beginning of simulation ($t=0$).

Initial conditions are computed by setting all dynamic variables to 0, except those corresponding to blocks that have an initial condition field in their block dialogs, and solving for all the system variables. The blocks with initial conditions have their dynamic variables set according to the user-provided value in the block dialog. Initial conditions can only be set on dynamic variables, because these are the independent states for simulation. For example, the Translational Spring block has the **Initial deformation** parameter, so the corresponding spring position state is set to the initial offset specified in the block dialog. Refer to the block reference documentation to find which blocks have initial conditions specified through their dialogs.

It is required that the initial conditions for dependent dynamic states be set consistently. For example, the initial voltages on two parallel capacitors must be equal. When the solver detects dependent dynamic variables, it performs

a check and issues an error if the initial conditions on dynamic states are not set consistently.

This concludes the initial conditions computation when the **Start simulation from steady state** check box in the Solver block dialog box is not selected (this is the default).

When this box is selected, the solver attempts to find the steady state that would result if the inputs to the system were held constant for a sufficiently large time, starting from the initial state obtained from the initial conditions computation, described previously. Although the solver tries to find the particular steady state resulting from the given initial conditions, it is not guaranteed to do so. All that is guaranteed is that if the steady-state solve succeeds, the state found is a steady state (within tolerance). Steady state means that the system variables are no longer changing with time. Simulation then starts from this steady state.

Note If the simulation fails at or near the start time when you use the **Start simulation from steady state** option, consider clearing the check box and simulating with the plain initial conditions computation only.

Performing Transient Initialization

After computing the initial conditions, or after a subsequent event (such as a discontinuity resulting, for example, from a valve opening, or a hard stop hitting the stop), Simscape solver performs transient initialization. This is done by fixing all dynamic variables and solving for algebraic variables and derivatives of dynamic variables. The goal of transient initialization is to provide a consistent set of initial conditions for the next transient solve phase.

Transient Solve

Finally, Simscape solver performs transient solve of the system of equations. In transient solve, continuous differential equations are integrated in time to compute all the variables as a function of time.

Simscape solver continues to perform the simulation according to the results of the transient solve until it encounters an event, such as a zero crossing or

discontinuity. The event may be within the physical network or elsewhere in the Simulink model. If an event is encountered, Simscape solver returns to the phase of transient initialization, and then back to transient solve. This cycle continues until the end of simulation.

Working with Solvers

In this section...
“Selecting a Solver” on page 2-8
“Input Filtering” on page 2-10

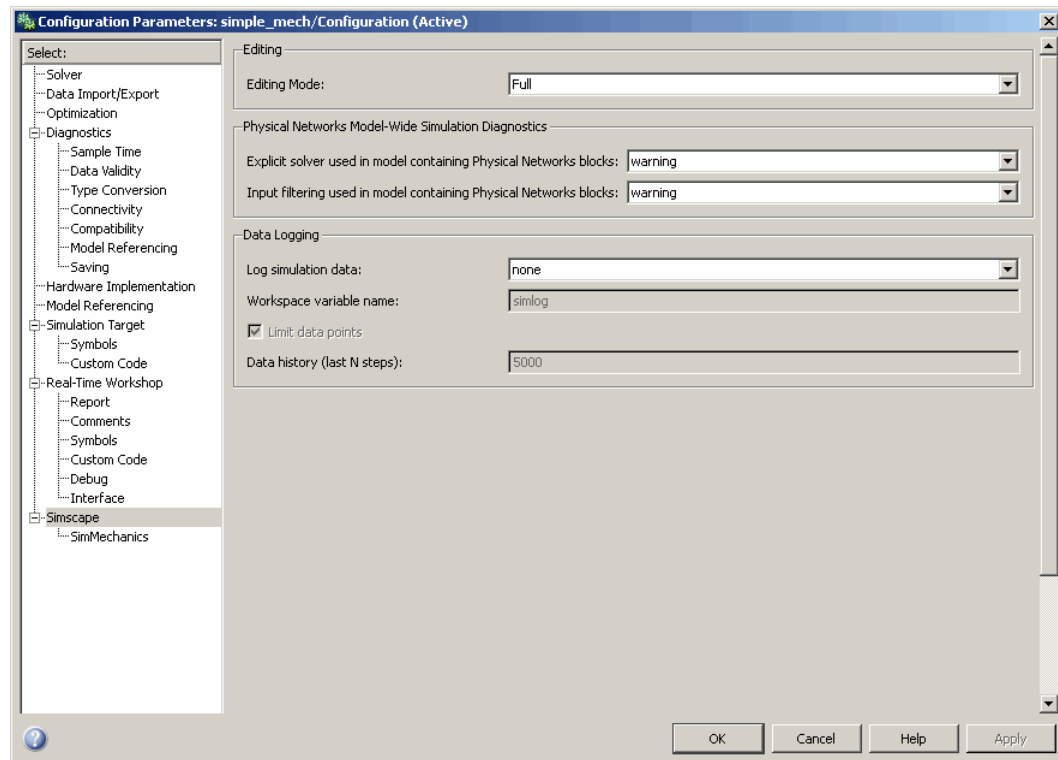
Selecting a Solver

It is possible to choose any variable-step or fixed-step solver for models containing Simscape blocks. However, implicit solvers, such as ode14x, ode23t, and ode15s, are a better choice for a typical model. In particular, for stiff systems, implicit solvers typically take many fewer timesteps than explicit solvers, such as ode45, ode113, and ode1.

When you first create a model, the default Simulink solver is ode45. To select a different solver, follow a procedure similar to that in “Modifying Initial Settings” on page 1-25.

If you do not modify the default solver and your system is stiff, your performance may not be optimal. To alert you to a potential issue, the system issues a warning when you use an explicit solver in a model containing Simscape blocks. You can turn off this default warning or changed it to an error for a particular model, by following these steps:

- 1 From the top menu bar in the model window, select **Simulation > Configuration Parameters**. The Configuration Parameters dialog box opens.
- 2 In the left pane of the Configuration Parameters dialog box, select **Simscape**.



3 Select the desired option from the **Explicit solver used in model containing Physical Networks blocks drop-down list:**

- **warning** — Makes the system issue a warning upon simulation if the model uses an explicit solver. This is the default option, designed to alert you to a potential issue if you use the default solver.
- **error** — Makes the system issue an error upon simulation if the model uses an explicit solver. If your model is stiff, and the use of explicit solvers undesirable, you may choose to select this option to avoid troubleshooting errors in the future.
- **none** — Turns off issuing a warning or error upon simulation with explicit solver. For models that are not stiff, explicit solvers can be effective, often taking fewer timesteps than implicit solvers. If you work with such models and use explicit solvers, select this option to turn off the warning upon simulation.

4 Click **OK**.

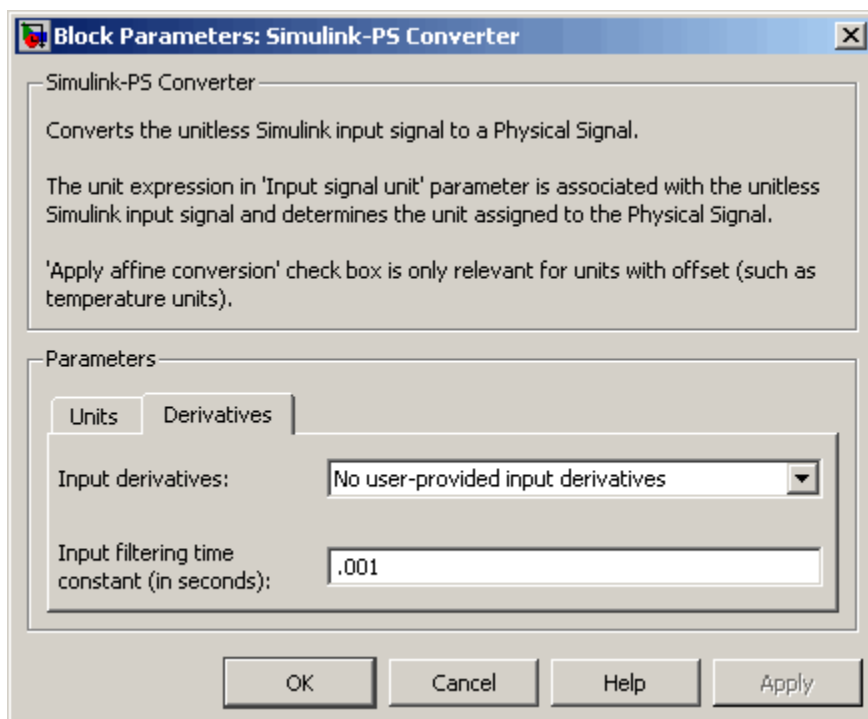
Input Filtering

If you use an explicit solver, you may need to provide time derivatives of some of the input signals. By default, needed input derivatives are provided by filtering the input through a low-pass filter. The derivative of the filtered input can then be computed by the Physical Networks simulation engine.

You can control the way you provide time derivatives for each input signal by configuring the Simulink-PS Converter block connected to it:

1 Double-click the Simulink-PS Converter block to open its dialog box.

2 Click the **Derivatives** tab.



- 3** To avoid filtering the input signal, set **Input derivatives** parameter to First derivative of input user-provided.

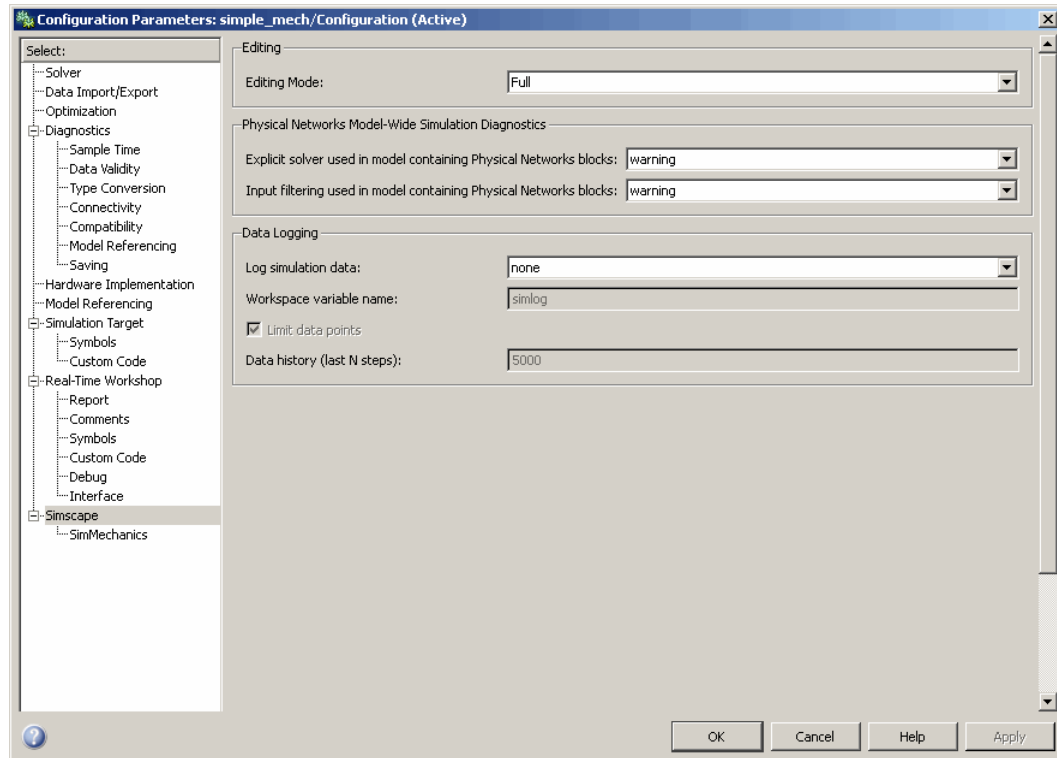
When you select this option, a second Simulink input port appears on the Simulink-PS Converter block, to let you connect the signal providing input derivatives. Input filtering is then turned off.

- 4** If you cannot provide the first derivative of the input signal as an additional input signal to the Simulink-PS Converter block, leave the **Input derivatives** parameter set to No user-input provided derivatives. In this case, however, it is important to set the appropriate **Input filtering time constant** parameter value for your model.

The filter time constant controls the filtering of the input signal. The filtered input follows the true input but is smoothed, with a lag on the order of the time constant chosen. You should set the time constant to a value no larger than the smallest time interval of interest in the system. The trade-off in choosing a very small time constant is that the filtered input signal will be closer to the true input signal, at the cost of increasing the stiffness of the system and slowing down the simulation.

Because input filtering can appreciably change the input signal and drastically affect simulation results if the time constant is too large, a warning is issued when input filtering is used. The warning indicates which Simulink-PS Converter blocks have their input signals filtered. You can turn off this warning (or change it to an error) by changing the preferences on the **Simscape** pane of the Configuration Parameters dialog box:

- 1** From the top menu bar in the model window, select **Simulation > Configuration Parameters**. The Configuration Parameters dialog box opens.
- 2** In the left pane of the Configuration Parameters dialog box, select **Simscape**.



3 Select the desired option from the **Input filtering used in model containing Physical Networks blocks drop-down list:**

- **warning** — Makes the system issue a warning upon simulation if the model uses input filtering. The warning contains a list of Simulink-PS Converter blocks that use input filtering. This is the default option.
- **error** — Makes the system issue an error upon simulation if the model uses input filtering. If you select this option and use an explicit solver, you have to provide first derivative of the input signal as an additional input signal to each Simulink-PS Converter block. For details, see the Simulink-PS Converter block reference page.
- **none** — Turns off issuing a warning or error upon simulation when the model uses input filtering.

4 Click **OK.**

Troubleshooting Simulation Errors

In this section...

- “Troubleshooting Tips and Techniques” on page 2-13
- “System Configuration Errors” on page 2-14
- “Numerical Simulation Issues” on page 2-17
- “Initial Conditions Solve Failure” on page 2-19
- “Transient Simulation Issues” on page 2-20

Troubleshooting Tips and Techniques

Simscape simulations can stop before completion with one or more error messages. This section discusses generic error types and error-fixing strategies. You might find the previous section, “How Simscape Simulation Works” on page 2-2, useful for identifying and tracing errors.

If a simulation failed:

- Review the model configuration. If your error message contains a list of blocks, look at these blocks first. Also look for:
 - Wrong connections — Verify that the model makes sense as a physical system. For example, look for actuators connected against each other, so that they try to move in opposite directions, or incorrect connections to reference nodes that prevent movement. In electrical circuits, verify polarity and connections to ground.
 - Wrong units — Simscape unit manager offers great flexibility in using physical units. However, you must exercise care in specifying the correct units, especially in the Simulink-PS Converter and PS-Simulink Converter blocks. Start analyzing the circuit by opening all the converter blocks and checking the correctness of specified units.
- Try to simplify the circuit. Unnecessary circuit complexity is the most common cause of simulation errors.
- Break the system into subsystems and test every unit until you are positive that the unit behaves as expected.

- Build the system by gradually increasing its complexity.

The MathWorks recommends that you build, simulate, and test your model incrementally. Start with an idealized, simplified model of your system, simulate it, verify that it works the way you expected. Then incrementally make your model more realistic, factoring in effects such as friction loss, motor shaft compliance, hard stops, and the other things that describe real-world phenomena. Simulate and test your model at every incremental step. Use subsystems to capture the model hierarchy, and simulate and test your subsystems separately before testing the whole model configuration. This approach helps you keep your models well organized and makes it easier to troubleshoot them.

System Configuration Errors

- “Missing Solver Configuration Block” on page 2-14
- “Extra Fluid Block or Gas Properties Block” on page 2-14
- “Missing Reference Block” on page 2-15
- “Basic Errors in Physical System Representation” on page 2-15

Missing Solver Configuration Block

Each topologically distinct Simscape block diagram requires exactly one Solver Configuration block to be connected to it. The Solver Configuration block specifies the global environment information and provides parameters for the solver that your model needs before you can begin simulation.

If you get an error message about a missing Solver Configuration block, open the Simscape Utilities library and add the Solver Configuration block anywhere on the circuit.

Extra Fluid Block or Gas Properties Block

If your model contains hydraulic elements, each topologically distinct hydraulic circuit in a diagram requires a Custom Hydraulic Fluid block (or Hydraulic Fluid block, available with SimHydraulics block libraries) to be connected to it. These blocks define the fluid properties that act as global parameters for all the blocks connected to the hydraulic circuit. If no

hydraulic fluid block is attached to a loop, the hydraulic blocks in this loop use the default fluid. However, more than one hydraulic fluid block in a loop generates an error.

Similarly, more than one Gas Properties block in a pneumatic circuit generates an error.

If you get an error message about too many domain-specific global parameter blocks attached to the network, look for an extra Hydraulic Fluid block, Custom Hydraulic Fluid block, or Gas Properties block and remove it.

Missing Reference Block

Simscape libraries contain domain-specific reference blocks, which represent reference points for the conserving ports of the appropriate type. For example, each topologically distinct electrical circuit must contain at least one Electrical Reference block, which represents connection to ground. Similarly, hydraulic conserving ports of all the blocks that are referenced to atmosphere (for example, suction ports of hydraulic pumps, or return ports of valves, cylinders, pipelines, if they are considered directly connected to atmosphere) must be connected to a Hydraulic Reference block, which represents connection to atmospheric pressure. Mechanical translational ports that are rigidly clamped to the frame (ground) must be connected to a Mechanical Translational Reference block, and so on.

If you get an error message about a missing reference block, or node, check your system configuration and add the appropriate reference block based on the rules described above. For more information and examples of best modeling practices, see “Grounding Rules” on page 1-35.

Basic Errors in Physical System Representation

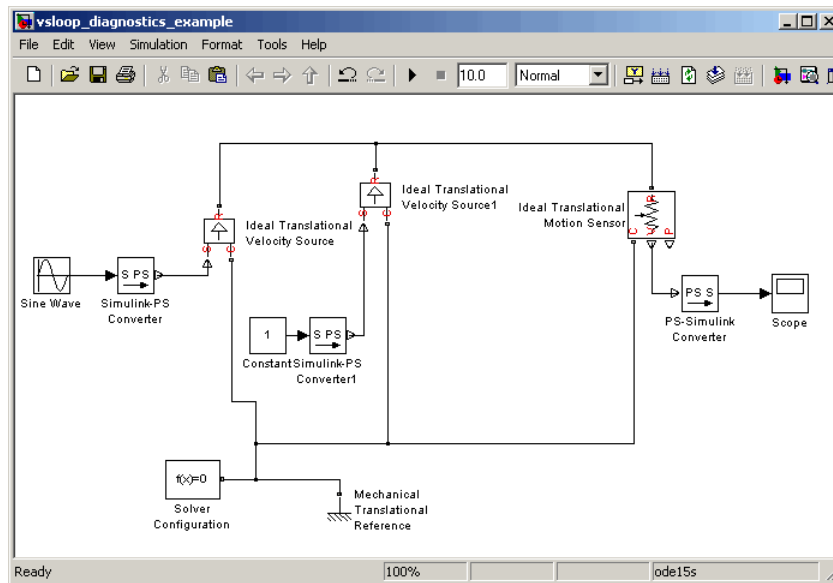
Physical systems are represented in the Simscape modeling environment as Physical Networks according to the Kirchhoff’s generalized circuit laws. Certain model configurations violate these laws and are therefore illegal. There are two broad violations:

- Sources of domain-specific Across variable connected in parallel (for example, voltage sources, hydraulic pressure sources, or velocity sources)

- Sources of domain-specific Through variable connected in series (for example, electric current sources, hydraulic flow rate sources, force or torque sources)

These configurations are impossible in the real world and illegal theoretically. If your model contains such a configuration, upon simulation the solver issues an error followed by a list of blocks, as shown in the following example.

Example. The model shown in the following illustration contains two Ideal Translational Velocity Sources connected in parallel. This produces two independent velocity loops, and the solver cannot construct a consistent system of equations for the circuit.



When you try to simulate the model, the solver issues the following error messages:

Nonlinear solver: failed to converge, residual norm too large.

Transient initialization, solving for consistent states and modes, failed to converge.

Warning: equations of one or more components may be dependent or inconsistent. This can cause problems in transient initialization. Here is the set of components involved:

'vsloop_diagnostics_example/Ideal Translational Velocity Source1'


```
'vsloop_diagnostics_example/Ideal Translational Velocity Source'
```

```
Initial conditions solve failed to converge.
```

You can simplify the system by using a single Ideal Translational Velocity Source block, with its control signal supplied by the Sine Wave block. Add the constant value from the second source to the bias of the sine wave.

Numerical Simulation Issues

- “Dependent Dynamic States” on page 2-17
- “Parameter Discontinuities” on page 2-19

Numerical simulation issues can be either a result of certain circuit configurations or of parameter discontinuities.

Dependent Dynamic States

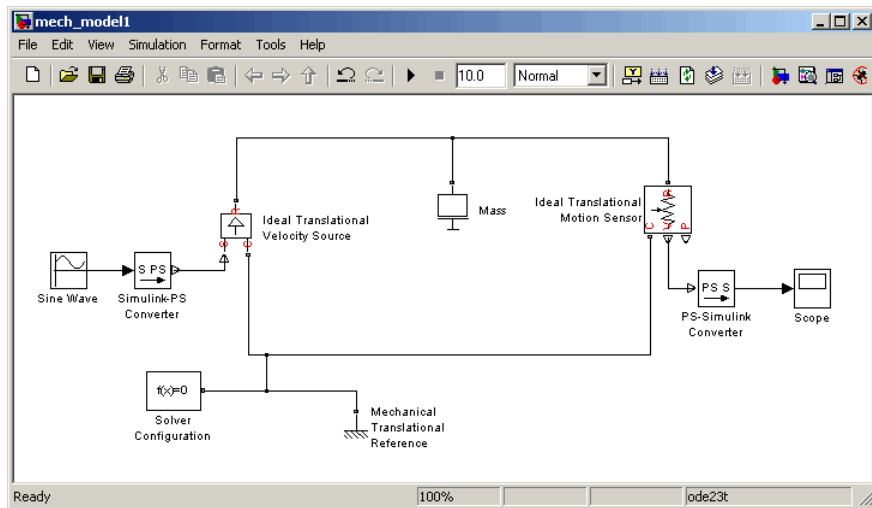
Certain circuit configurations can result in dependent dynamic states, or the so-called higher-index differential algebraic equations (DAEs). Simscape solver can handle dependencies among dynamic states that are linear in the states and independent of time and inputs to the system. For example, capacitors connected in parallel or inductors connected in series will not cause any problems. Other circuit configurations with dependent dynamic states, in certain cases, may slow down the simulation or lead to an error when the solver fails to initialize.

In electrical circuits, common examples that can cause this behavior include voltage sources connected in parallel with capacitors, inductors connected in series with current sources, and so on. To address this issue, the Capacitor and Inductor blocks include parasitic terms (parallel conductance and series resistance). For other blocks that do not contain these parasitic terms, you might need to introduce your own parasitic conductance or resistance into the circuit. For more information on best modeling practices, as well as for troubleshooting suggestions, see “Avoiding Numerical Simulation Issues” on page 1-38.

Examples in other domains include direct connections between a velocity source and a mass, a force source and a spring, a pressure source and a

hydraulic accumulator, or a flow rate source and a fluid inertia. If you encounter this error, try to either simplify the circuit or include additional blocks, such as spring-damper combinations or constant orifices, to avoid the direct connection that results in dependent states.

Example. The model shown in the following illustration contains an Ideal Translational Velocity Source directly connected to a Mass.

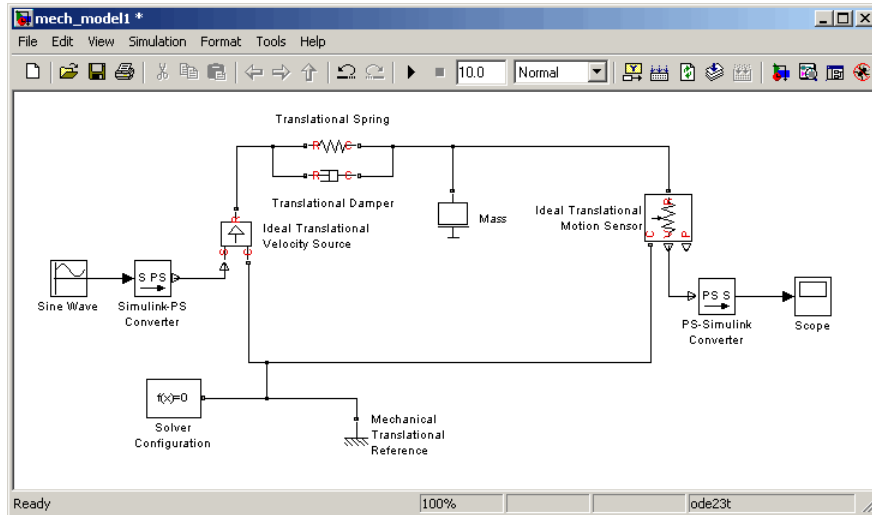


When you try to simulate the model, the solver issues the following error messages:

```
Nonlinear solver: failed to converge, residual norm too large.
Transient initialization, solving for consistent states and modes, failed to converge.
Warning: problems possible for transient initialization, as well as stepsize control
for transient solve, due to equations of one or more components:
'mech_model1/Mechanical Translational Reference'
'mech_model1//Ideal Translational Velocity Source'
'mech_model1/Mass'
```

```
Initial conditions solve failed to converge.
```

You can try adding a very stiff spring between the velocity source and the mass. To avoid the possibility of high-frequency oscillations introduced by the spring, connect it in parallel with a damper with a high damping coefficient.



Parameter Discontinuities

Nonlinear parameters, dependent on time or other variables, may also lead to numerical simulation issues as a result of parameter discontinuity. These issues usually manifest themselves at the transient initialization stage (see “Transient Simulation Issues” on page 2-20).

Initial Conditions Solve Failure

The initial conditions solve, which solves for all system variables (with initial conditions specified on some system variables), may fail. This has several possible causes:

- **System configuration error.** In this case, the Simulation Diagnostics window usually contains additional, more specific, error messages, such as a missing reference node, or a warning about the component equations, followed by a list of components involved. See “System Configuration Errors” on page 2-14 for more information.

- Dependent dynamic state. In this case, the Simulation Diagnostics window also may contain additional, more specific, error messages, such as a warning about the component equations, followed by a list of components involved. See “Dependent Dynamic States” on page 2-17 for more information.
- The constraint residual tolerance may be too tight to produce a consistent solution to the algebraic constraints at the beginning of simulation. You can try to increase the **Constraint Residual Tolerance** parameter value (that is, relax the tolerance) in the Solver Configuration block.

If the Simulation Diagnostics window has other, more specific, error messages, address them first and try rerunning the simulation. See also “Troubleshooting Tips and Techniques” on page 2-13.

Transient Simulation Issues

- “Transient Initialization Not Converging” on page 2-20
- “Step-Size-Related Errors” on page 2-21

Transient initialization happens at the beginning of simulation (after computing the initial conditions) or after a subsequent event, such as a discontinuity (for example, when a hard stop hits the stop). It is performed by fixing all dynamic variables and solving for algebraic variables and derivatives of dynamic variables. The goal of transient initialization is to provide a consistent set of initial conditions for the next transient solve step.

Transient Initialization Not Converging

Error messages stating that transient initialization failed to converge, or that a set of consistent initial conditions could not be generated, indicate transient initialization issues. They can be a result of parameter discontinuity. Review your model to find the possible sources of discontinuity. See also “Troubleshooting Tips and Techniques” on page 2-13.

You can also try to decrease the **Constraint Residual Tolerance** parameter value (that is, tighten the tolerance) in the Solver Configuration block.

Step-Size-Related Errors

A typical step-size-related error message may state that the system is unable to reduce the step size without violating the minimum step size for a certain number of consecutive times. This error message indicates numerical difficulties in solving the Differential Algebraic Equations (DAEs) for the model. This might be caused by dependent dynamic states (higher-index DAEs) or by the high stiffness of the system. You can try the following:

- Tighten the solver tolerance (decrease the **Relative Tolerance** parameter value in the Configuration Parameters dialog box)
- Specify a value, other than auto, for the **Absolute Tolerance** parameter in the Configuration Parameters dialog box. Experiment with this parameter value.
- Tighten the residual tolerance (decrease the **Constraint Residual Tolerance** parameter value in the Solver Configuration block)
- Increase the value of the **Number of consecutive min step size violations allowed** parameter in the Configuration Parameters dialog box (set it to a value greater than the number of consecutive step size violations given in the error message)
- Review the model configuration and try to simplify the circuit, or add small parasitic terms to your circuit to avoid dependent dynamic states. For more information, see “Numerical Simulation Issues” on page 2-17.

Finding an Operating Point

In this section...
“What Is an Operating Point?” on page 2-22
“How to Find Operating Points” on page 2-23
“Finding Operating Points with Simscape, Simulink, and Related Products” on page 2-24

What Is an Operating Point?

An *operating point* of a system is a dynamic configuration that satisfies design and use requirements called *operating specifications*. You can express such operating specifications as requirements on the system state \mathbf{x} and inputs \mathbf{u} . It is not always possible to find a dynamic state that satisfies all operating conditions. Also, a system might have multiple operating points satisfying the same requirements.

Operating points are essential for designing and implementing system controllers. You can optimize a system at an operating point for performance, stability, safety, and reliability.

The most important and common type of operating point is a *steady state*, where some or all of the system dynamic variables are constant.

Using Operating Points for Linearization

An important motive for finding operating points is *linearization*, which determines the system response to small disturbances at an operating point. Linearization results influence the design of feedback controllers to govern dynamic behavior near the operating point. A full linearization analysis requires one or more system outputs, \mathbf{y} , in addition to inputs.

See “Linearizing at an Operating Point” on page 2-28.

Example

A pilot flying an aircraft wants to find, for a given environment, a state of the aircraft engine and control surfaces that produces level, constant-velocity, and

constant-altitude flight relative to the ground. The requirements of "level," "constant velocity," "constant altitude," and "relative to the ground" constitute operating specifications. This operating point is a steady state of the aircraft velocity, altitude, and orientation in space.

How to Find Operating Points

You can provide predefined state and input vectors, \mathbf{x}_0 and \mathbf{u}_0 , to specify an operating point. If you do not know an operating point in advance, you have two methods of identifying an operating point that satisfies operating specifications.

- “Time-Based Search” on page 2-23: Observing the actual or simulated behavior of the system in time is more general, but less precise, and usually requires a trial-and-error process to find a precise operating point.
- “State-Based Search” on page 2-23: If you know the system dynamics, you can solve for steady states, at least in principle.

Time-Based Search

You can sometimes find operating points and steady states by trial and error while operating or simulating over some length of time and varying the system parameters, inputs, and initial conditions. In such a time-based approach, you isolate and study instants or intervals of time when a system satisfies the operating specifications. The system state and inputs under those conditions constitute the operating point, which you can also specify by an operating or simulation time.

State-Based Search

The alternative to trial-and-error searching for steady states is *trimming*. In this state-based approach, you bypass time-based simulation and find solutions for inputs, outputs, states, and state derivatives satisfying an operating specification. Trimming specifies inputs and part of a state and solves the system dynamics for the rest of the state. The resulting full state and input vectors, \mathbf{x}_0 and \mathbf{u}_0 , constitute the operating point.

There is no general guarantee that such solutions, \mathbf{x}_0 , exist for given operating specifications and inputs, \mathbf{u}_0 .

Checking Discrete System States

An operating point includes the state of discrete system variables that change in a discontinuous way. In general, you cannot find these states by small, continuous changes of system variables. Such states usually require systematic exploration of the discrete variables over the full range of their possible values.

Finding Operating Points with Simscape, Simulink, and Related Products

You have a number of ways to find an operating point in a Simscape model. You can impose operating specifications and isolate operating points using Simscape and Simulink features and, in some cases, with such related products as Simulink® Control Design™ and Control System Toolbox™ software. The full state vector of your model contains:

- Simulink components, which can be continuous or discrete.
- Simscape components, which are continuous.

Tip Whichever method that you choose to find an operating point, if you want to use it for linearization, you must save the operating point information in the form of a simulation time t_0 , a state vector \mathbf{x}_0 , and an input vector \mathbf{u}_0 , or an operating point object.

Simulating in Time to Search for an Operating Point

One way to identify operating points is to simulate your model and inspect its state \mathbf{x} and output \mathbf{y} as a time series.

- 1 In your Simscape model, set up sensor outputs for whatever block outputs you want to observe.
- 2 Connect Scope blocks, To Workspace blocks, or both, to your Simscape block outputs to observe and record simulation behavior.
- 3 In the **Data Import/Export** pane of your model Configuration Parameters settings, select the **Time**, **States**, and **Output** check boxes to record this simulation information in your workspace.

Using the Simscape Steady-State Solver

Most commonly, the operating point that you want is a steady state. You have a precise method for identifying steady states with the Simscape steady-state solver, which allows you to isolate steady states more exactly than you can with ordinary simulation. It is particularly recommended for isolating steady states of a strongly nonlinear character. You can search for multiple steady states with the steady-state solver by varying the model inputs, parameters, and initial conditions.

The Simscape steady-state solver initializes the system and inputs first, then determines a steady state. In general, the system does not remain in this initial steady state $\mathbf{x}(t=0) = \mathbf{x}_0$ during simulation, because the system inputs \mathbf{u} change independently, and the system has to respond by changing its state $\mathbf{x}(t)$.

To implement the steady-state solver:

- 1** In each, some, or all of the physical networks in your Simscape model, open the Solver Configuration block.
- 2** In each block dialog box, select the **Start simulation from steady state** check box.
- 3** In the model Configuration Parameters settings, on the **Data Import/Export** pane, select the **States** check box to record the time series of \mathbf{x} values in your workspace.

If you also have input signals \mathbf{u} in the model, you can capture those inputs by connecting To Workspace blocks to the input Simulink signal lines.

- 4** Close these dialog boxes and start simulation.

The first vector of values $\mathbf{x}(t=0)$ that you capture during simulation reflects the steady state \mathbf{x}_0 that the Simscape solver identified.

Using Simulink Control Design Techniques

Note The techniques described in this section require the Simulink Control Design product. You must use the features of this product on the Simulink lines in your model, not directly on Simscape physical network lines or blocks.

The following methods are state-based, allowing you to impose operating specifications, and work well for simple to moderately complex Simscape models. The MathWorks™ does not recommend these methods for highly complex Simscape models.

To find operating points, use the `operspec` and `findop` functions, customizing where necessary. Create an operating specification object with `operspec`, then compute an operating point object with `findop`. The `findop` function attempts to find an operating point that satisfies the operating specifications and reports on its success or failure. If the search is successful, `find_op` returns state values satisfying the operating specifications.

You have several choices for operating specifications for the components of the state vector.

Assumed Operating Condition	Operating Specification
Default	Request that all state component derivatives be zero. This is a steady-state for the whole model, not just a Simscape network within the model.
Nondefault	Request any value you want independently for each state component.
Nondefault	Request that a particular state component derivative be zero. This is a steady-state condition for that state component.

Additional Simulink Control Design Methods. Instead of the command line interface, you can use the associated graphical user interface, through the model menu bar: **Tools > Control Design > Linear Analysis**. For more details on the use of operating point specification objects, related functions, and the graphical interface, see the Simulink Control Design documentation.

Using Sources to Find Operating Points Not Recommended

You can impose an operating specification on part of a Simscape model by inserting source blocks from the Simscape Foundation Library. These impose specified values of system variables in parts of the model. You can simulate and save the state vector.

However, you cannot obtain an operating point for the original system (without the source blocks) by saving the state values from the model and then removing the source blocks. In general, the number, order, and identity of state components change after adding and removing Simscape blocks in a model.

Simulink `trim` Function Not Supported with Simscape Models

The Simulink `trim` function is not supported for models containing Simscape components.

Linearizing at an Operating Point

In this section...
“What Is Linearization?” on page 2-28
“How to Linearize a Model” on page 2-30
“Linearizing a Model with Simscape, Simulink, and Related Products” on page 2-30
“References” on page 2-34

What Is Linearization?

Determining the response of a system to small perturbations at an operating point is a critical step in system and controller design. Once you find an operating point, you can linearize the model about that operating point to explore the response and stability of the system. To find an operating point in a Simscape model, see “Finding an Operating Point” on page 2-22.

Choosing a Good Operating Point for Linearization

Although steady-state and other operating points (state \mathbf{x}_0 and inputs \mathbf{u}_0) might exist for your model, that is no guarantee that such operating points are suitable for linearization. The critical question is: how good is the linearized approximation compared to the exact system dynamics?

- When perturbed slightly, a problematic operating point might exhibit strong asymmetries, with strongly nonlinear behavior when perturbed in certain ways and smoother behavior in other ways.
- Small perturbations might change discrete states in a large, discontinuous way that violates the linear approximation.

Operating points with a strongly nonlinear or discontinuous character are not suitable for linearization. You should analyze them in full simulation and perturb the system by varying its inputs, parameters, and initial conditions.

Tip A reliable way to check for such an unsuitable operating point is to linearize at several nearby operating points. If the results differ greatly, the operating point is strongly nonlinear.

What Is Linear Response?

Near an operating point, you can express the system state \mathbf{x} , inputs \mathbf{u} , and outputs \mathbf{y} relative to that operating point in terms of $\mathbf{x} - \mathbf{x}_0$, $\mathbf{u} - \mathbf{u}_0$, and $\mathbf{y} - \mathbf{y}_0$. For convenience, shift the vectors by subtracting the operating point: $\mathbf{x} - \mathbf{x}_0 \rightarrow \mathbf{x}$, etc.

If the system dynamics does not explicitly depend on time and the operating point is a steady state, the system response to state and input perturbations near the steady state is approximately governed by a *linear time-invariant* (LTI) state space model:

$$d\mathbf{x}/dt = \mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{u}$$

$$\mathbf{y} = \mathbf{C} \mathbf{x} + \mathbf{D} \mathbf{u}.$$

The matrices \mathbf{A} , \mathbf{B} , \mathbf{C} , \mathbf{D} have components and structures that are independent of the simulation time. A system is stable to changes in state at an operating point if the eigenvalues of \mathbf{A} are negative.

If the operating point is not a steady state or the system dynamics depends explicitly on time, the linearized dynamics near the operating point is more complicated. The matrices \mathbf{A} , \mathbf{B} , \mathbf{C} , \mathbf{D} are not constant and depend on the simulation time t_0 , as well as the operating point \mathbf{x}_0 and \mathbf{u}_0 [1].

Tip While you can linearize a closed system with no inputs or outputs and obtain a nonzero \mathbf{A} matrix, obtaining a nontrivial linearized input-output model requires at least one input component in \mathbf{u} and one output component in \mathbf{y} .

Example

A pilot is flying, or simulating, an aircraft in level, constant-velocity, and constant-altitude flight relative to the ground, with a known environment. A crucial question for the aircraft pilot and designers is: will the aircraft return to the steady state if perturbed from it by a disturbance, such as a wind gust — in other words, is this steady state stable? If the operating point is unstable, the aircraft trajectory can diverge from the steady state, requiring human or automatic intervention to maintain steady flight.

How to Linearize a Model

Once you know an operating point, you have three practical methods for investigating the system response to small disturbances.

Full Simulation- or Operation-Based Perturbations

You can experiment with the system or a system simulation by making repeated, different, and slight changes to the system parameters, inputs, and initial conditions, while operating at a steady state. This method requires costly trial and error and generates uncontrolled and imprecise approximations.

Analytic Approximations to Known State Dynamics

If you know the system state dynamics and an operating point \mathbf{x}_0 and \mathbf{u}_0 in analytic form, you can apply standard approximation techniques to derive an analytic form for the A , B , C , D matrices.

Numerical Approximations to Known State Dynamics

If you have a controlled numerical approximation to your system state dynamics and operating point, you can apply standard computational techniques to generate numerical approximations to the A , B , C , D matrices.

Linearizing a Model with Simscape, Simulink, and Related Products

Use the following methods to create numerical linearized state-space models from a model containing Simscape components.

Independent Versus Dependent States

An important difference from normal Simulink models is that the states in a physical network are not independent in general, because some states have dependencies on other states through constraints.

- The independent states are a subset of system variables and consist of Simscape dynamic variables and Simulink states.
- The dependent states consist of Simscape algebraic variables and dependent (constrained) dynamic variables.

For more information on Simscape dynamic and algebraic variables, see “How Simscape Simulation Works” on page 2-2.

The only components of the A , B , C , D matrices that can be nonzero are those corresponding to independent states.

- The A matrix, of size n_states by n_states , is all zeros except for a submatrix of size n_ind by n_ind , where n_ind is the number of independent states.
- The B matrix, of size n_states by n_inputs , is all zeros except for a submatrix of size n_ind by n_inputs .
- The C matrix, of size $n_outputs$ by n_states , is all zeros except for a submatrix of size $n_outputs$ by n_ind .
- The D matrix, of size $n_outputs$ by n_inputs , can be nonzeros everywhere.

Obtaining the Independent Subset of States. A complete linearized solution uses only an independent subset of the system states. From the matrices A , B , C , D , you can obtain a minimal input-output linearized model with the `minreal` and `sminreal` functions from Control System Toolbox software.

Linearizing with the Simulink `linmod` and `dlinmod` Functions

The Simulink functions `linmod` and `dlinmod` provide a way of linearizing a Simscape model. There are several ways that you can use `linmod` and `dlinmod`, and the linearization results differ depending on the method chosen. To use these functions, you do not have to open the model, just have the model file on your MATLAB path.

For more information about Simulink linearization, see “Linearizing Models” in the Simulink documentation.

Note If your model has continuous states, use `linmod`. (Continuous states are the Simscape default.) If your model has mixed continuous and discrete states, or purely discrete states, use `dlinmod`.

Linearizing a model with the local solver enabled (in the Solver Configuration block) is not supported.

Linearizing with Default State and Input. You can call `linmod` without specifying state or input. Enter `linmod('modelName')` at the command line.

With this form of `linmod`, Simulink linearization solves for consistent initial conditions in the same way it does on the first step of any simulation. In particular, any initial conditions, such as initial offset from equilibrium for a spring, are set as if the simulation were starting from the initial time.

`linmod` allows you to change the time of externally specified signals (but not the internal system dynamics) from the default. For this and more details, see the function reference page.

Linearizing with the Steady-State Solver at an Initial Steady State.

If you have previously used the Simscape steady-state solver to find an operating point, you can linearize at that operating point:

- 1 Open one or more Solver Configuration blocks in your model.
- 2 Select the **Start simulation from steady state** check box for the physical networks that you want to linearize.
- 3 Close the Solver Configuration dialog boxes and save the modified model.
- 4 Enter `linmod('modelName')` at the command line.

`linmod` linearizes at the first step of simulation. In this case, the initial state is also an operating point, a steady state.

For more about setting up the steady-state solver, see the Solver Configuration block reference page. For more details on its use, see “Using the Simscape Steady-State Solver” on page 2-25.

Linearizing with Specified State and Input – Ensuring Consistency of States. You can call `linmod` and specify state and input. Enter `linmod('modelName',x0,u0)` at the command line. The extra arguments specify, respectively, the steady state \mathbf{x}_0 and inputs \mathbf{u}_0 for linearizing the simulation. When you specify a state to `linmod`, ensure that it is self-consistent, within solver tolerance.

With this form of `linmod`, Simulink linearization does not solve for initial conditions. Because not all states in the model have to be independent, it is possible, though erroneous, to provide `linmod` with an inconsistent state to linearize about.

If you specify a state that is not self-consistent (within solver tolerance), the Simscape solver issues a warning at the command line when you attempt linearization. The Simscape solver then attempts to make the specified \mathbf{x}_0 consistent by changing some of its components, possibly by large amounts.

Tip You most easily ensure a self-consistent state by taking the state from some simulated time. For example, by selecting the **States** check box on the **Data Import/Export** pane of the model Configuration Parameters dialog box, you can capture a time series of state values in a simulation run.

Linearizing with Simulink Linearization Blocks

You can generate linearized state-space models from your Simscape model by adding a Timed-Based Linearization or Trigger-Based Linearization block to the model and simulating. These blocks combine time-based simulation up to specified times or internal trigger points, with state-based linearization at those times or trigger points.

For complete details about these blocks, see their respective block reference pages.

Linearizing with Simulink Control Design Software

Note The techniques described in this section require the Simulink Control Design product. You must use the features of this product on the Simulink lines in your model, not directly on Simscape physical network lines or blocks.

The following methods require that you start with an operating point object saved from trimming the model to an operating specification, as explained in “Using Simulink® Control Design Techniques” on page 2-26.

To linearize a model with an operating point object, use the `linearize` function, customizing where necessary. The resulting state-space object contains the matrices A , B , C , D .

Additional Simulink Control Design Methods. Instead of the command line interface, you can use the associated graphical user interface, through the model menu bar: **Tools > Control Design > Linear Analysis**. For more details on linearization, operating points and state-space objects, related functions, and the graphical interface, see the Simulink Control Design documentation.

References

- [1] Brogan, W. L., *Modern Control Theory*, 2nd Ed., Englewood Cliffs, New Jersey, Prentice-Hall, 1985.
- [2] The MathWorks, *Control System Toolbox User Guide*, www.mathworks.com/access/helpdesk/help/toolbox/control/.

Generating Code

In this section...
“About Code Generation from Simscape Models” on page 2-35
“Related Simulink Code Generation Documentation” on page 2-35
“Reasons for Generating Code” on page 2-36
“Using Code-Related Products and Features” on page 2-36
“How Simscape Code Generation Differs from Simulink” on page 2-37

About Code Generation from Simscape Models

You can use Real-Time Workshop® software to generate stand-alone C code from your Physical Networks models and enhance simulation speed and portability. Certain features of Simulink software also make use of generated or external code. This section explains code-related tasks you can perform with your Simscape models.

Code versions of Simscape models typically require fixed-step Simulink solvers, which are discussed in the Simulink documentation. Some features of Simscape software are restricted when you translate a model into code. See “How Simscape Code Generation Differs from Simulink” on page 2-37, as well as “Limitations” on page 2-39.

Note Code generated from Simscape models is intended for rapid prototyping and hardware-in-the-loop applications. It is not intended for use as production code in embedded controller applications.

Add-on products based on the Simscape platform also support code generation, with some variations and exceptions described in their respective User’s Guides. Consult those User’s Guides for more information.

Related Simulink Code Generation Documentation

Consult Simulink Acceleration modes, Real-Time Workshop, and xPC Target™ documentation for general information on code generation.

Reasons for Generating Code

Code generation has many purposes and methods. There are two essential rationales:

- Compiled code versions of Simulink and Simscape models run faster than the original block diagram models. The time savings can be dramatic.
- An equally important consideration for Simscape models is the stand-alone implementation of generated and compiled code. Once you convert part or all of your model to code, you can deploy the stand-alone executable program on virtually any platform, independently of MATLAB application.

Converting a model or subsystem to code also hides the original model or subsystem.

Using Code-Related Products and Features

With Simulink, Real-Time Workshop, and xPC Target software, using several code-related technologies, you can link existing code to your models and generate code versions of your models.

Code-Related Task	Component or Feature
Link existing code written in C or other supported languages to Simulink models	Simulink S-functions to generate customized blocks
Speed up Simulink simulations	Accelerator mode Rapid Accelerator mode
Generate stand-alone fixed-step code from Simulink models	Real-Time Workshop software
Generate stand-alone variable-step code from Simulink models	Real-Time Workshop Rapid Simulation Target (RSim)
Convert Simulink model to code and compile and run it on a target PC	Real-Time Workshop and xPC Target software

How Simscape Code Generation Differs from Simulink

In general, using the code generated from Simscape models is similar to using code generated from regular Simulink models. However, there are certain differences.

Simscape and Simulink Code Are Generated Separately

Real-Time Workshop software generates code from the Simscape blocks separately from the Simulink blocks in your model. The generated Simscape code does not pass through `model.rtw` or the Target Language Compiler. All the code generated from a single model resides in the same directory, however.

Compiler Support and Precompiled Libraries

Simscape software and its add-on products provide static runtime libraries precompiled for compilers supported by Real-Time Workshop software. These are the standard UNIX compilers for UNIX operating systems, LCC and Microsoft® Visual Studio® for 32-bit Windows®, and Microsoft Visual Studio for 64-bit Windows.

For all other compilers, the static runtime libraries needed by code generated from Simscape models are compiled once per model during the code generation build process.

Simscape Code Reuse Is Not Supported

Reusable subsystems in Simulink reuse code that is generated once from the subsystem. You cannot generate reusable code from subsystems containing Simscape blocks.

Tunable Parameters Are Not Supported

A tunable parameter is a Simulink run-time parameter that you can change while the simulation is running. Simscape blocks do not support tunable parameters in either simulations or generated code.

Simscape Run-Time Parameter Inlining Ignores Global Exceptions

If you choose to enable parameter inlining for code generated from a Simscape model, the software inlines all its run-time parameters. If you choose to make some of the global Simscape block parameters exceptions to inlining, the exceptions are ignored. You can change global tunable parameters only by regenerating code from the model.

Limitations

In this section...
“Sample Time and Solver Restrictions” on page 2-39
“Algebraic Loops” on page 2-39
“Restricted Simulink Tools” on page 2-40
“Unsupported Simulink Tools” on page 2-42
“Simulink Tools Not Compatible with Simscape Blocks” on page 2-42
“Code Generation” on page 2-42

Sample Time and Solver Restrictions

The default sample times of Simscape blocks are continuous. You cannot simulate Simscape blocks with discrete solvers using the default sample times.

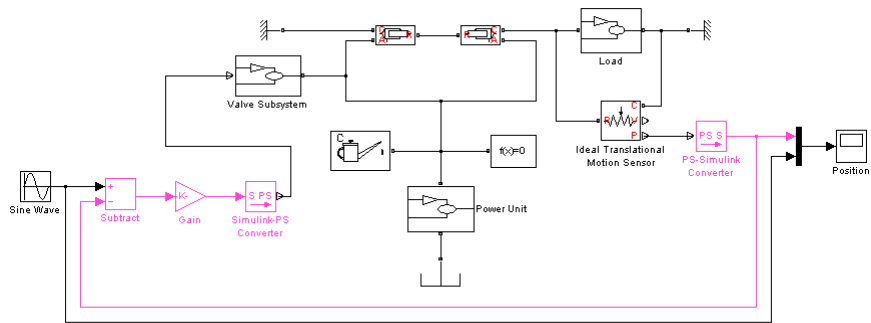
If you switch to a local solver in the Solver Configuration block, the associated states become discrete. If there are no continuous Simulink or Simscape states elsewhere in the model, you should use a discrete solver to simulate such a model.

You cannot override the sample time of a nonvirtual subsystem containing Simscape blocks.

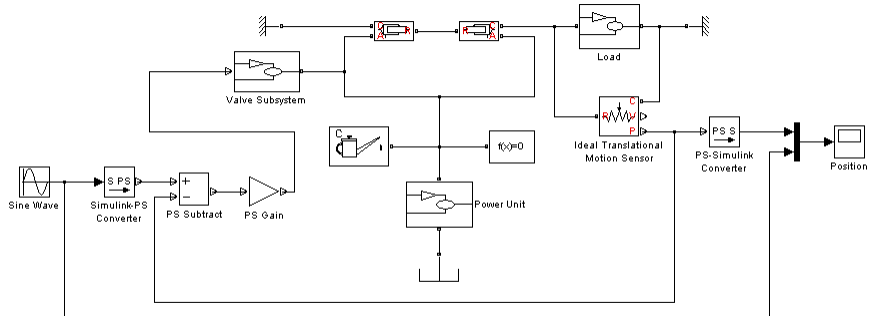
Algebraic Loops

A Simscape physical network should not exist within a Simulink algebraic loop. This means that you should not directly connect an output of a PS-Simulink Converter block to an input of a Simulink-PS Converter block of the same physical network.

For example, the following model contains a direct feedthrough between the PS-Simulink Converter block and the Simulink-PS Converter block (highlighted in magenta). To avoid the algebraic loop, you can insert a Transfer Function block anywhere along the highlighted loop.



A better way to avoid an algebraic loop without introducing additional dynamics is shown in the modified model below.



Restricted Simulink Tools

Certain Simulink tools are restricted for use with Simscape software:

- You can use the Simulink `set_param` and `get_param` commands to set or get Simscape block parameters. The MathWorks does not recommend that you use these commands for this purpose.

If you make changes to block parameters at the command line, run your model first before saving it. Otherwise, you might save invalid block parameters. Any block parameter changes that you make with `set_param` are not validated unless you run the model.

- Enabled subsystems can contain Simscape blocks. Always set the **States when enabling** parameter in the Enable dialog to **held** for the subsystem's Enable port.

Setting **States when enabling** to **reset** is not supported and can lead to fatal simulation errors.

- You can place Simscape blocks within nonvirtual subsystems that support continuous states. Nonvirtual subsystems that support continuous states include Enabled subsystems and Atomic subsystems. However, physical connections and physical signals must not cross nonvirtual boundaries. When placing Simscape blocks in a nonvirtual subsystem, make sure to place all blocks belonging to a given Physical Network in the same nonvirtual subsystem.
- Simulink configurable subsystems work with Simscape blocks only if all of the block choices have consistent port signatures.
- For Iterator, Function-Call, Triggered, and While Iterator nonvirtual subsystems cannot contain Simscape blocks.
- An atomic subsystem with a user-specified (noninherited) sample time cannot contain Simscape blocks.
- You can use the Simulink **Save As** command only to rename Simscape models within the current version. Saving in a previous version format is not supported for models containing Simscape blocks.
- Linearization with the Simulink `linmod` function or with equivalent Simulink Control Design functions and graphical interfaces is not supported with Simscape models if you use local solvers.
- Model referencing is supported, with some restrictions:
 - All Physical connection lines must be contained within the referenced model. Such lines cannot cross the boundary of the referenced model subsystem in the referencing model.
 - The referencing model and the referenced model must use the same solver.

For further information, consult the Simulink documentation on referencing models.

Unsupported Simulink Tools

Certain Simulink tools and features do not work with Simscape software:

- The Simulink Profiler tool does not work with Simscape models.
- Physical signals and physical connection lines between conserving ports are different from Simulink signals. Therefore, the Signal and Scope Manager tool and the signal label functionality are not supported.

Simulink Tools Not Compatible with Simscape Blocks

Some Simulink tools and features do not work with Simscape blocks:

- Execution order tags do not appear on Simscape blocks.
- Simscape blocks do not invoke user-defined callbacks.
- You cannot set breakpoints on Simscape blocks.
- Reusable subsystems cannot contain Simscape blocks.
- You cannot use the Simulink Fixed-Point Tool with Simscape blocks.
- The Report Generator reports Simscape block properties incompletely.

Code Generation

Code generation is supported for Simscape physical modeling software and its family of add-on products. However, there are restrictions on code generated from Simscape models.

- Code reuse is not supported.
- C++ code generation is not supported.
- Tunable parameters are not supported.
- Run-time parameter inlining ignores global exceptions.
- Simulation of Simscape models on fixed-point processors is not supported.
- Block diagnostics in error messages are not supported. This means that if you get an error message from simulating generated code, it does not contain a list of blocks involved.
- Conversion of models or subsystems containing Simscape blocks to S-functions is not supported.

- Simulation-in-the-loop (SIL) is not supported.
- Code generation respects the **Linear Algebra** setting in the Solver Configuration block, but only when you choose the local solver option. Otherwise, code generation uses full linear algebra.

“Generating Code” on page 2-35 describes Simscape code generation features. “Restricted Simulink Tools” on page 2-40 describes limitations on model referencing.

There are variations and exceptions as well in the code generation features of the add-on products based on Simscape platform. For details, see the User’s Guides for individual add-on products.

Code Generation and Fixed-Step Solvers

Most code generation options for Simscape models require the use of fixed-step Simulink solvers. This table summarizes the available solver choices, depending on how you generate code.

Code Generation Option	Solver Choices
Accelerator mode Rapid Accelerator mode	Variable-step or fixed-step
Real-Time Workshop software: RSim Target*	Variable-step or fixed-step
Real-Time Workshop software: Targets other than RSim	Fixed-step only

* For the RSim Target, Simscape software supports only the Simulink solver module. In the model Configuration Parameters dialog box, see the **Real-Time Workshop: RSim Target: Solver selection** menu. The default is automatic selection, which might fail to choose the Simulink solver module.

Logging Simulation Data

- “About Simulation Data Logging” on page 3-2
- “How to Log Simulation Data” on page 3-3
- “Data Logging Example” on page 3-6

About Simulation Data Logging

In this section...
“Suggested Workflows” on page 3-2
“Limitations” on page 3-2

Suggested Workflows

You can log simulation data to the workspace for debugging and verification. Data logging lets you analyze how internal block variables change with time during simulation. For example, you may want to see that the pressure in a hydraulic cylinder is above some minimum value, or compare it against the pump pressure. If you log simulation data to the workspace, you can later query, plot, and analyze it without rerunning the simulation.

Simulation data logging can also replace connecting sensors and scopes to track simulation data. These blocks increase the model complexity and slow down simulation. The “Data Logging Example” on page 3-6 shows how you can log and plot simulation data instead of adding sensors to your model. It also shows how you can print the complete logging tree for a model, and plot simulation results for a selected variable.

For additional information, see the reference pages for the classes `simscape.logging.Node`, `simscape.logging.Series`, and their associated methods.

Limitations

Simulation data logging is not supported for:

- Model reference
- Generated code
- Accelerator mode
- Rapid Accelerator mode

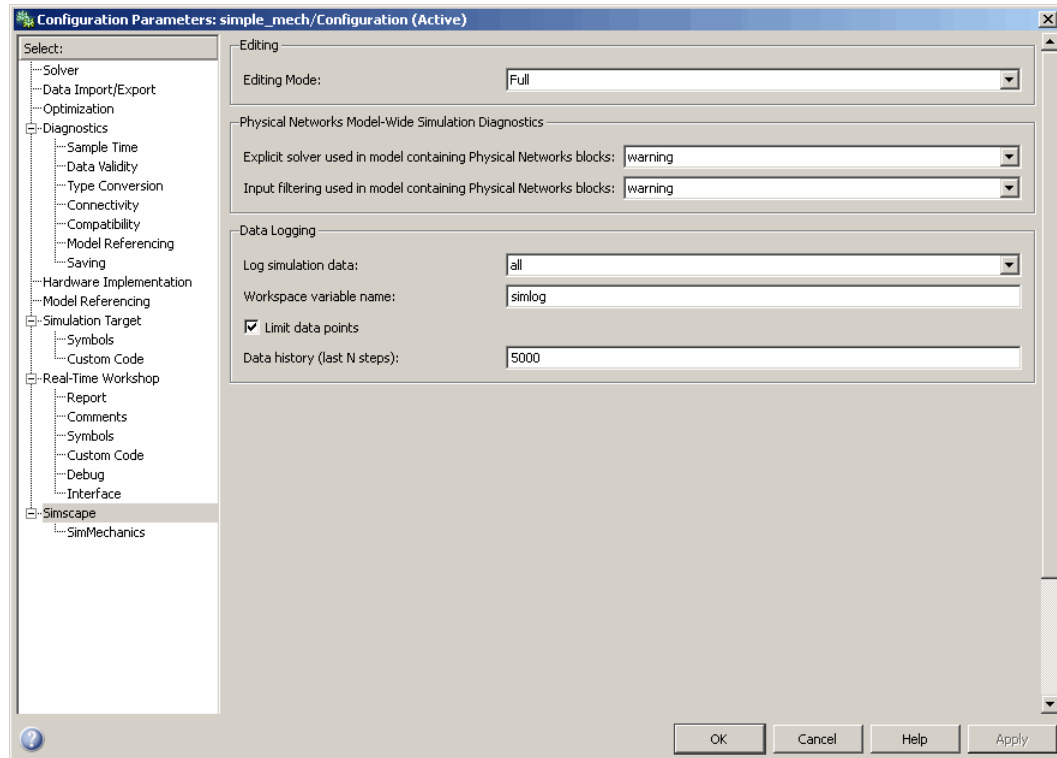
How to Log Simulation Data

In this section...
“How to Enable Data Logging” on page 3-3
“Data Logging Options” on page 3-4

How to Enable Data Logging

By default, simulation data is not logged. To turn on the data logging for a model, use the **Log simulation data** configuration parameter.

- 1 In the model window, from the top menu bar, select **Simulation > Configuration Parameters**. The Configuration Parameters dialog box opens.
- 2 In the Configuration Parameters dialog box, on the left pane, select **Simscape**. The right pane displays the **Log simulation data** option, which is set to none, by default.
- 3 From the drop-down list, select **all**, then click **OK**.



- 4 Simulate the model. This creates a workspace variable named `simlog` (as specified by the **Workspace variable name** parameter), which contains the simulation data.

For information on how to access and use the data stored in this variable, see “Data Logging Example” on page 3-6.

Data Logging Options

When you set the **Log simulation data** configuration parameter to `all`, other options in the Data Logging group box become available.

- **Workspace variable name** — Specifies the name of the workspace variable that stores the simulation data. Subsequent simulations overwrite the data in the simulation log variable. If you want to compare data from

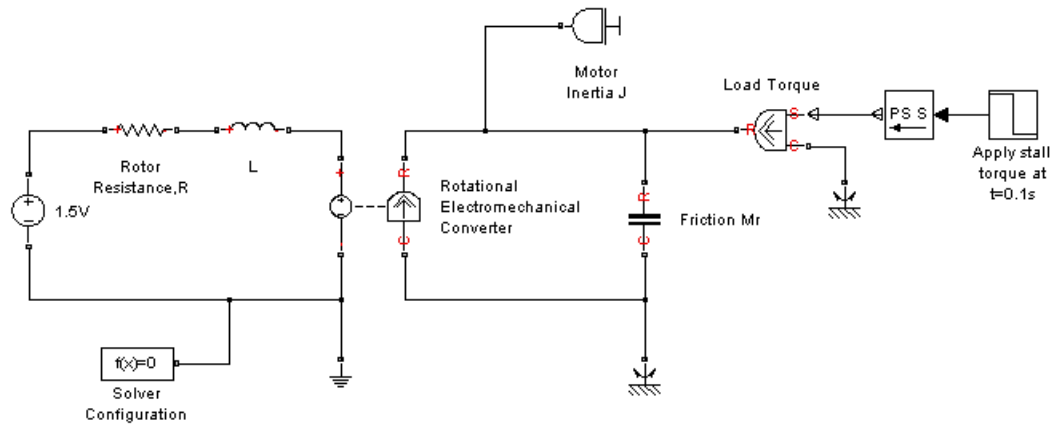
two models or two simulation runs, use different names for the respective log variables. The default variable name is `simlog`.

- **Limit data points** — Saving data to the workspace can slow down the simulation and consume memory. Use this checkbox in conjunction with the **Data history (last N steps)** parameter to limit the number of data points saved. The checkbox is selected by default. If you clear it, the simulation log variable contains the data points for the whole simulation, at the price of slower simulation speed and heavier memory consumption.
- **Data history (last N steps)** — Specify the number of simulation steps to limit the number of data points output to the workspace. The simulation log variable contains the data points corresponding to the last N steps of the simulation, where N is the value that you specify for the **Data history (last N steps)** parameter. You have to select the **Limit data points** checkbox to make this parameter available. The default value logs simulation data for the last 5000 steps. You can specify any other positive integer number. If the simulation contains fewer steps than the number specified, the simulation log variable contains the data points for the whole simulation.

After changing your data logging preferences, rerun the simulation to generate a new data log.

Data Logging Example

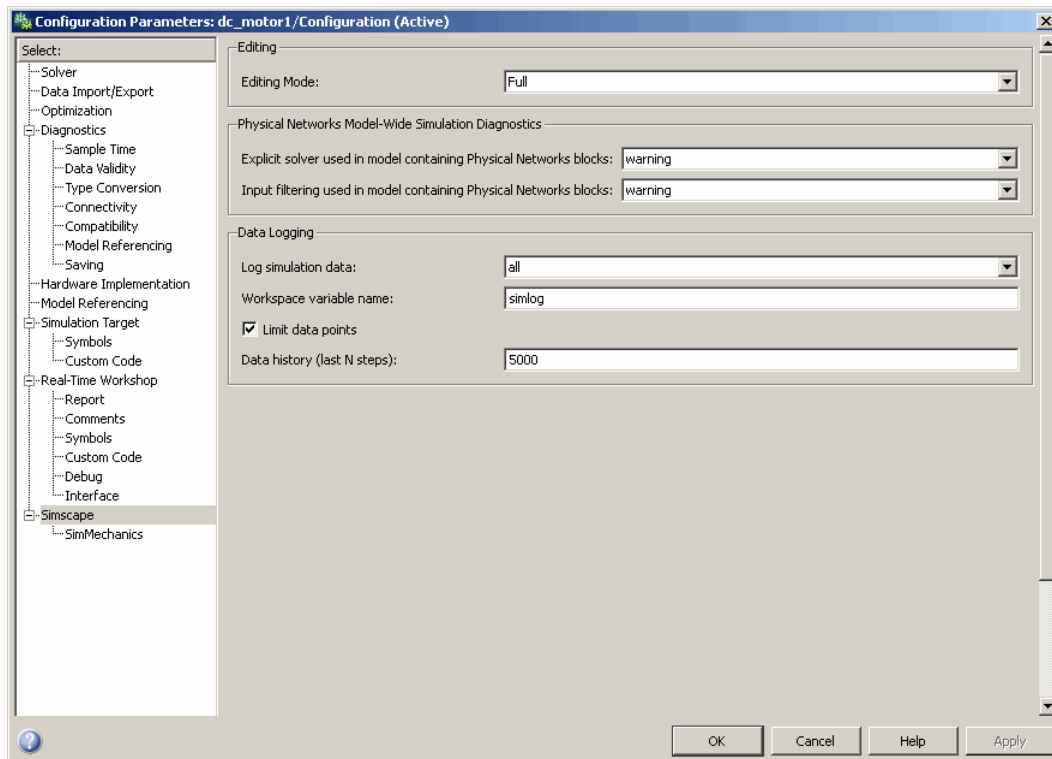
The model shown represents a permanent magnet DC motor.



This model is very similar to the Permanent Magnet DC Motor demo, but, unlike the demo model, it does not include the Ideal Rotational Motion Sensor and the Current Sensor blocks, along with the respective PS-Simulink Converter blocks and scopes. For a detailed description of the Permanent Magnet DC Motor demo, see “Working with a Simscape Demo Model” in the *Simscape Getting Started Guide*.

This example shows how you can log and plot simulation data instead of adding sensors to your model.

- 1 Build the model, as shown in the preceding illustration.
- 2 To enable data logging, open the Configuration Parameters dialog box, in the left pane, select **Simscape**, then set the **Log simulation data** parameter to **all** and click **OK**.



- 3 Simulate the model. This creates a workspace variable named `simlog` (as specified by the **Workspace variable name** parameter), which contains the simulation data.
- 4 The `simlog` variable has the same hierarchy as the model. To see the whole variable structure, at the command prompt, type:

```
simlog.print
```

This command prints the whole data tree.

```
dc_motor1
+-Electrical_Reference2
| +-V
| | +-v
| +-i
```

```
+Friction_Mr
| +-C
| | +-w
| +-R
| | +-w
| +-t
| +-w
+-L
| +-i
| +-i_L
| +-n
| | +-v
| +-p
| | +-v
| +-v
+-Load_Torque
| +-C
| | +-w
| +-R
| | +-w
| +-S
| +-t
| +-w
+-Mechanical_Rotational_Reference
| +-W
| | +-w
| +-t
+-Mechanical_Rotational_Reference1
| +-W
| | +-w
| +-t
+-Motor_Inertia_J
| +-I
| | +-w
| +-t
+-Rotational_Electromechanical_Converter
| +-C
| | +-w
| +-R
| | +-w
```

```

| +-i
| +-n
| | +-v
| +-p
| | +-v
| +-t
| +-v
| +-w
+-Rotor_Resistance_R
| +-i
| +-n
| | +-v
| +-p
| | +-v
| +-v
+-Simulink_PS_Converter
+-x1_5V
| +-i
| +-n
| | +-v
| +-p
| | +-v
| +-v

```

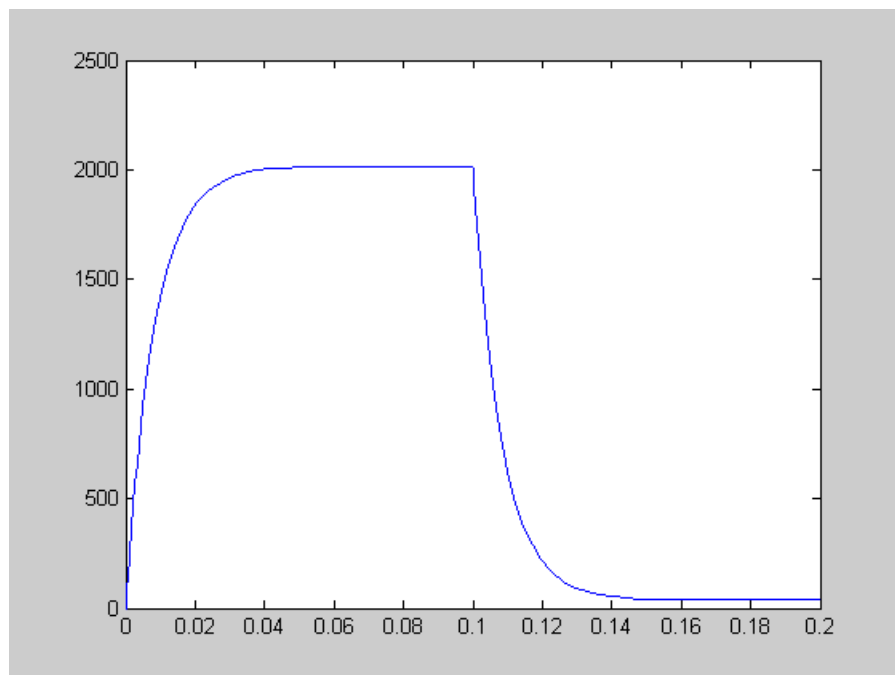
- 5** Every node that represents an Across, Through, or internal block variable contains series data. To get to the series, you have to specify the complete path to it through the tree, starting with the top-level variable name. For example, to get a handle on the series representing the angular velocity of the motor, type:

```
s1 = simlog.Rotational_Electromechanical_Converter.R.w.series;
```

- 6** To see how the motor speed changes with time, type:

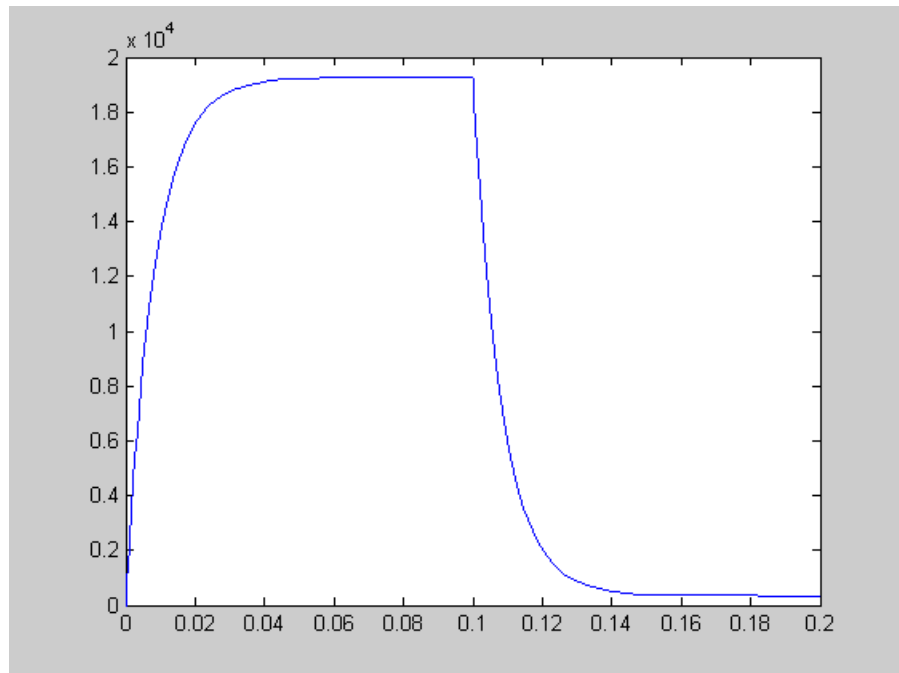
```
plot(s1.time, s1.values)
```

This plots the motor speed in default units (radian/s).



7 To get a plot in revolutions per minute, type:

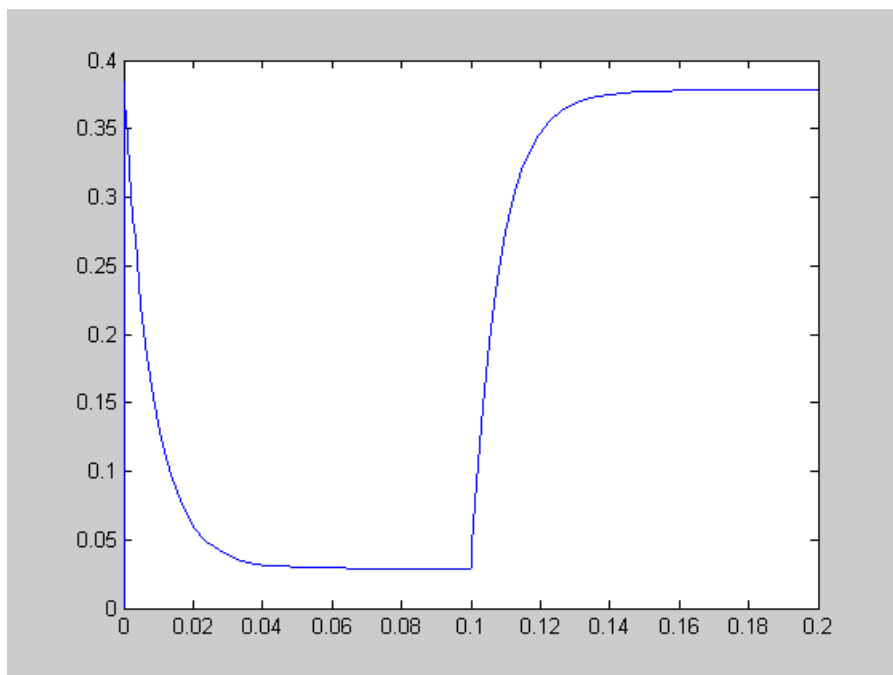
```
plot(s1.time, s1.values('rpm'))
```



8 Compare this figure to the RPM scope display in the Permanent Magnet DC Motor demo. The results are exactly the same.

9 To plot the current through the motor, type:

```
s2 = simlog.Rotational_Electromechanical_Converter.i.series;  
plot(s2.time, s2.values)
```



- 10** Compare the resulting figure with the Motor Current scope display in the demo.

Working with Physical Units

- “Overview” on page 4-2
- “Unit Definitions” on page 4-4
- “Specifying Units in Block Dialogs” on page 4-9
- “Thermal Unit Conversions” on page 4-11
- “Angular Units” on page 4-14

Overview

Unlike Simulink signals, which are essentially unitless, physical signals can have units associated with them. You specify the units along with the parameter values in the block dialogs, and Simscape unit manager performs the necessary unit conversion operations when solving a physical network. Simscape blocks support standard measurement systems. The default block units are meter-kilogram-second or MKS (SI).

Simscape software comes with a library of standard units, and you can define additional units as needed (see “Unit Definitions” on page 4-4). You can use these units in your block diagrams:

- To specify the units of an input physical signal, type a unit name, or a mathematical expression with unit names, in the **Input signal unit** field of the Simulink-PS Converter block dialog. You can also select a unit from a drop-down list, which is prepopulated with some common input units. Signal units that you specify in a Simulink-PS Converter block must match the input type expected by the Simscape block connected to it. For example, when you provide the input signal for an Ideal Angular Velocity Source block, specify angular velocity units, such as rad/s or rpm, in the Simulink-PS Converter block, or leave it unitless. If you leave the block unitless, with the **Input signal unit** parameter set to 1, then the physical signal units are inferred from the destination block.
- Simscape block dialogs have drop-down combo boxes of units next to a parameter value, letting you either select a unit from the drop-down list, or type a unit name (or a mathematical expression with unit names) directly into the box. These drop-down lists are automatically populated by those units that are commensurate with the unit of the parameter, based on the current list of unit definitions. For example, if a parameter is specified, by default, with the units of meters per second, m/s, the drop-down list of units contains commensurate units, such as mm/s, in/s, fps (feet per second), fpm (feet per minute), and so on, including any other linear velocity units currently defined in your unit registry.
- To specify the units of an output physical signal, type a unit name, or a mathematical expression with unit names, in the **Output signal unit** field of the PS-Simulink Converter block dialog. You can also select a unit from a drop-down list, which is prepopulated with some common output units. The system compares the units you specified with the actual units

of the input physical signal coming into the converter block and applies a gain equal to the conversion factor before outputting the Simulink signal. The default value is 1, which means that the unit is not specified. If you do not specify a unit, or if the unit matches the actual units of the input physical signal, no gain is applied.

For more information, see “Specifying Units in Block Dialogs” on page 4-9.

Unit Definitions

Simscape unit names are defined in the `pm_units.m` file, which is shipped with the product. You can open this file to see how the physical units are defined in the product, and also as an example when adding your own units. This file is located in the directory `matlabroot\toolbox\physmod\unit_manager\unit_manager`.

Default registered units and their abbreviations are listed in the following table. Use the `pm_getunits` command to get an up-to-date list of units currently defined in your unit registry. Use the `pm_adddimension` and `pm_addunit` commands to define additional units.

Physical Unit Abbreviations Defined by Default in the Simscape Unit Registry

Quantity	Abbreviation	Unit
Amount of substance	mol	Mole
Angle	rad	Radian
	deg	Degree
	rev	Revolution
Angular velocity	rpm	Revolutions/minute
	Hz	Revolutions/second
Capacitance	F	Farad
	pF	Picofarad
	nF	Nanofarad
	uF	Microfarad
Charge	c	Coulomb
Conductance	S	Siemens
	nS	Nanosiemens
	uS	Microsiemens
	mS	Millisiemens

Physical Unit Abbreviations Defined by Default in the Simscape Unit Registry (Continued)

Quantity	Abbreviation	Unit
Current	A	Ampere
	pA	Picoampere
	nA	Nanoampere
	uA	Microampere
	mA	Milliampere
	kA	Kiloampere
Energy	J	Joule
	Btu	British thermal unit
	eV	Electronvolt
Flow rate	lpm	Liter/minute
	gpm	Gallon/minute
Force	N	Newton
	dyn	Dyne
	lbf	Pound-force
	mN	Millinewton
Inductance	H	Henry
	uH	Microhenry
	mH	Millihenry

Physical Unit Abbreviations Defined by Default in the Simscape Unit Registry (Continued)

Quantity	Abbreviation	Unit
Length	m	Meter
	cm	Centimeter
	mm	Millimeter
	km	Kilometer
	um	Micrometer
	in	Inch
	ft	Foot
	mi	Mile
	yd	Yard
Magnetic flux	Wb	Weber
Magnetic flux density	T	Tesla
	G	Gauss
Mass	kg	Kilogram
	g	Gram
	mg	Milligram
	lbm	Pound mass
	oz	Ounce
	slug	Slug
Pressure	Pa	Pascal
	bar	Bar
	atm	Atmosphere
	psi	Pound/inch^2
Power	W	Watt
	HP	Horsepower

Physical Unit Abbreviations Defined by Default in the Simscape Unit Registry (Continued)

Quantity	Abbreviation	Unit
Resistance	Ohm	Ohm
	kOhm	Kiloohm
	MOhm	Megaohm
	GOhm	Gigaohm
Temperature	K	Kelvin
	C	Celsius
	Fh	Fahrenheit
	R	Rankine
Time	s	Second
	min	Minute
	hr	Hour
	ms	Millisecond
	us	Microsecond
	ns	Nanosecond
Velocity	mph	Miles/hour
	fpm	Feet/minute
	fps	Feet/second
Viscosity absolute	Poise	Poise
	cP	Centipoise
	reyn	Reyn
Viscosity kinematic	St	Stokes
	cSt	Centistokes
	Newt	Newt

Physical Unit Abbreviations Defined by Default in the Simscape Unit Registry (Continued)

Quantity	Abbreviation	Unit
Volume	l	Liter
	gal	Gallon
Voltage	V	Volt
	mV	Millivolt
	kV	Kilovolt

Note This table lists the unit abbreviations defined in the product. See the following section, “Specifying Units in Block Dialogs” on page 4-9, for information on how to use the abbreviations above, or mathematical expressions with these abbreviations, to specify units for the parameter values in the block dialogs.

Specifying Units in Block Dialogs

Simscape block dialogs have drop-down combo boxes for units next to a parameter value. For example, in the Constant Volume Chamber block, the drop-down list for the **Chamber volume** parameter contains 1, gal, in³, ft³, mm³, cm³, m³, and km³, and the drop-down list for the **Initial pressure** parameter contains Pa, bar, psi, and atm.

You can either select a unit from the drop-down list, or type a commensurate unit name (or a mathematical expression with unit names) directly into the unit combo box of the block dialog. You can use the abbreviations for the units defined in your registry, or any valid mathematical expressions with these abbreviations. For example, you can specify torque in newton-meters (N*m) or pound-feet (lbf*ft). To specify velocity, you can use one of the defined unit abbreviations (mph, fpm, fps), or an expression based on any combination of the defined units of length and time, such as meters/second (m/s), millimeters/second (mm/s), inches/minute (in/min), and so on.

Note Affine units (such as Celsius or Fahrenheit) are not allowed in unit expressions. For more information, see “About Affine Units” on page 4-11.

The following operators are supported in the unit mathematical expressions:

- * Multiplication
- / Division
- ^ Power
- +, - Plus, minus — for exponents only
- () Brackets to specify evaluation order

Metric unit prefixes, such as *kilo*, *milli*, or *micro*, are not supported. For example, if you want to use milliliter as a unit of volume, you have to add it to the unit registry:

```
pm_addunit('ml', 0.001, 'l');
```

The drop-down lists next to parameter names are automatically populated by those units that are commensurate with the unit of the parameter. If you specify the units by typing, it is your responsibility to enter units that are commensurate with the unit of the parameter. The unit manager performs error checking when you click **Apply** or **OK** in the block dialog box, and issues an error if you type an incorrect unit.

In the Simulink-PS Converter and the PS-Simulink Converter block dialogs, the drop-down lists are prepopulated with some common input and output units, and it is your responsibility to select or type a unit expression commensurate with the expected input or output units. The error checking for the converter blocks is performed at the time of simulation. See “Model Validation” on page 2-4 for details.

Note Currently, physical units are not propagated through the blocks in the Physical Signals library, such as PS Add, PS Gain, and so on.

Thermal Unit Conversions

In this section...

“About Affine Units” on page 4-11

“When to Apply Affine Conversion” on page 4-11

“How to Apply Affine Conversion” on page 4-12

About Affine Units

Thermal units often require an affine conversion, that is, a conversion that performs both multiplication and addition. To convert from the old value T_{old} to the new value T_{new} , we need a linear conversion coefficient L and an offset O :

$$T_{new} = L * T_{old} + O$$

For example, to convert a temperature reading from degrees Celsius into degrees Fahrenheit, the linear term equals 9/5, and the offset equals 32:

$$T_{Fahr} = 9 / 5 * T_{Cels} + 32$$

Simscape unit manager defines kelvin (K) as the fundamental temperature unit. This makes Celsius (C) and Fahrenheit (Fh) affine units because they are both related to kelvin with an affine conversion. Rankine (R) is defined in terms of kelvin with a zero linear offset and, therefore, is not an affine unit.

The following are the default Simscape unit registry definitions for temperature units:

```
pm_adddimension('temperature', 'K'); % defines kelvin as fundamental temperature unit
pm_addunit('C', [1 273.15], 'K'); % defines Celsius in terms of kelvin
pm_addunit('Fh', [5/9 -32*5/9], 'C'); % defines Fahrenheit in terms of Celsius
pm_addunit('R', [5/9 0], 'K'); % defines rankine in terms of kelvin
```

When to Apply Affine Conversion

In dealing with affine units, sometimes you need to convert them using just the linear term. Usually, this happens when the value you convert represents relative, rather than absolute, temperature, $\Delta T = T_1 - T_2$.

$$\Delta T_{new} = L * \Delta T_{old}$$

In this case, adding the affine offset would yield incorrect conversion results.

For example, the outdoor temperature rose by 18 degrees Fahrenheit, and you need to input this value into your model. When converting this value into kelvin, use linear conversion

$$\Delta T_{kelvin} = 5 / 9 * \Delta T_{Fahr}$$

and you get 10 K, that is, the outdoor temperature changed by 10 kelvin. If you apply affine conversion, you will get a temperature change of approximately 265 kelvin, which is incorrect.

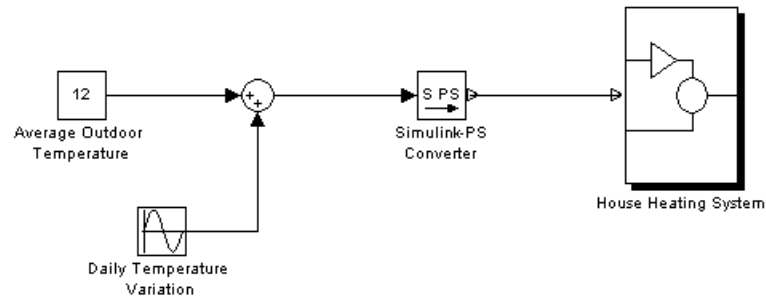
This is even better illustrated if you use degrees Celsius for the input units because the linear term for conversion between Celsius and kelvin is 1:

- If the outdoor temperature *changed* by 10 degrees Celsius (relative temperature value), then it changed by 10 kelvin (do not apply affine conversion).
- If the outdoor temperature *is* 10 degrees Celsius (absolute temperature value), then it is 283 kelvin (apply affine conversion).

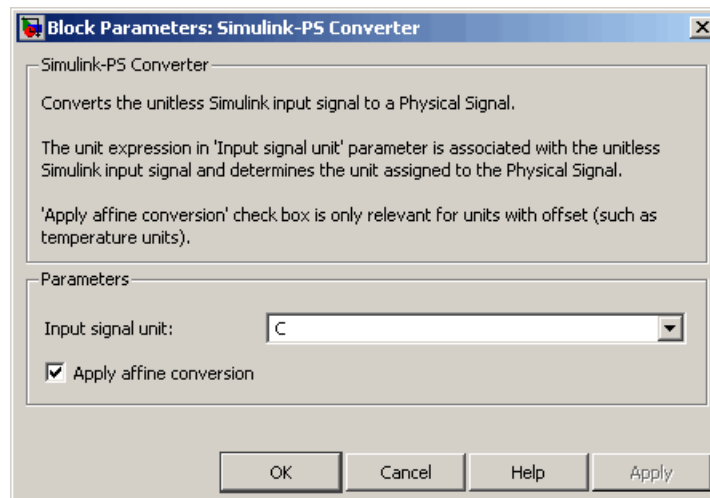
How to Apply Affine Conversion

When you specify affine units for an input temperature signal, it is important to consider whether you need to apply affine conversion. Usually this decision depends on whether the signal represents absolute or relative temperature (see “When to Apply Affine Conversion” on page 4-11).

For example, you model a house-heating system, and you need to input the outdoor temperature. In the following diagram, the Constant source block represents the average outdoor temperature, in degrees Celsius, and the Sine source block adds the daily temperature variation. The average outdoor temperature, in this case, is 12 degrees Celsius. Daily variation with an amplitude of 8 makes the input outdoor temperature vary between 4 and 20 degrees Celsius.



This signal is an absolute temperature reading. Therefore, when the signal converts into kelvin for further computations, you need to specify that it should use affine conversion. Double-click the Simulink-PS Converter block, type **C** in the **Input signal unit** field, and select the **Apply affine conversion** check box.



As a result, the Simulink-PS Converter block outputs a value varying between 277 K and 293 K.

Angular Units

Simscape implementation of angular units relies on the concept of angular units, specifically radians, being a unit but dimensionless. The notion of angular units being dimensionless is widely held in the metrology community. The fundamental angular unit, radian, is defined in the Simscape unit registry as:

```
pm_addunit('rad', 1, 'm/m');
```

which corresponds to the SI and NIST definition [1]. In other words, Simscape unit manager does not introduce a separate dimension, 'angle', with a fundamental unit of 'rad' (similar to dimensions for length or mass), but rather defines the fundamental angular unit in terms of meter over meter or, in effect, 1.

The additional angular units, degree and revolution, are defined respectively as:

```
pm_addunit('deg', pi/180, 'rad');  
pm_addunit('rev', 2*pi, 'rad');
```

As a result, forward trigonometric functions, such as `sin`, `cos`, and `tan`, work directly with arguments expressed in angular units. For example, `cosinus` of 90 degrees equals the `cosinus` of $(\pi/2)$ radians and equals the `cosinus` of $(\pi/2)$. Expansion of forward trigonometric functions works in a similar manner.

Another effect of dimensionless implementation of angular units is the convenience of the work-energy conversion. For example, torque (in $\text{N}\cdot\text{m}$) multiplied by angle (in rad) can be added directly to energy (in J, or $\text{N}\cdot\text{m}$). If you specify other commensurate units for the components of this equation, Simscape unit manager performs the necessary unit conversion operations and the result is the same.

References

[1] *The NIST Reference on Constants, Units, and Uncertainty*,
<http://physics.nist.gov/cuu/Units/units.html>

Using the Simscape Editing Mode

- “About the Simscape Editing Mode” on page 5-2
- “Working with Restricted and Full Modes” on page 5-9
- “Editing Mode Information” on page 5-23

About the Simscape Editing Mode

In this section...
“Suggested Workflows” on page 5-2
“What You Can Do in Restricted Mode” on page 5-3
“What You Can Do in Full Mode” on page 5-4
“Switching Between Modes” on page 5-4
“Working with Block Libraries” on page 5-7

Suggested Workflows

The Simscape Editing Mode functionality is implemented for customers who perform physical modeling and simulation using Simscape platform and its add-on products: SimDriveline, SimElectronics®, SimHydraulics, and SimMechanics. It allows you to open, simulate, and save models that contain blocks from add-on products in Restricted mode, without checking out add-on product licenses, as long as the products are installed on your machine. It is intended to provide an economical way to distribute simulation models throughout a team or organization.

Note Unless your organization uses concurrent licenses, see the Simscape product page on the MathWorks Web site for specific information on how to install add-on products on your machine, to be able to work in Restricted mode.

The Editing Mode functionality supports widespread use of Physical Modeling products throughout an engineering organization by making it economical for one user to develop a model and provide it to many other users.

Specifically, this feature allows a user, *model developer*, to build a model that uses Simscape platform and one or more add-on products and share that model with other users, *model users*. When building the model in Full mode, the model developer must have a Simscape license and the add-on product licenses for all the blocks in the model. For example, if a model combines Simscape, SimHydraulics, and SimDriveline blocks, the model developer needs to check out licenses for all three products to work with it in Full mode.

Once the model is built, model users need only to check out a Simscape license to simulate the model and fine-tune its parameters in Restricted mode. As long as no structural changes are made to the model, model users can work in Restricted mode and do not need to check out add-on product licenses.

Another workflow, available with concurrent licenses only, lets multiple users, who all have Simscape licenses, share a small number of add-on product licenses by working mostly in Restricted mode, and temporarily switching models to Full mode only when they need to perform a specific design task that requires being in Full mode.

Note The MathWorks recommends that you save all the models in Full mode before upgrading to a new version of Simulink or Simscape software.

If you have saved a model in Restricted mode and, upon upgrading to a new product version, open the model and it does not run, switch it to Full mode and save. You can then again switch to Restricted mode and work without problem.

What You Can Do in Restricted Mode

When your model is open in Restricted mode, you can:

- Simulate the model.
- Inspect parameters.
- Change certain block parameters. In general, you can change numerical parameter values, but cannot change the block parameterization options. See the block reference pages for specifics.
- Generate code.
- Make data logging or visualization changes.
- Add or delete regular Simulink blocks (such as sources or scopes) and appropriate connections.

For other types of changes, listed in the following section, your model has to be in Full mode. Some of these disallowed changes are impossible to make in Restricted mode (for example, Restricted parameters are grayed out in block

dialog boxes). Other changes, like changing the physical topology of a model, are not explicitly disallowed, but if you make these changes in Restricted mode, the software will issue an error message when you try to run, compile, or save such a model.

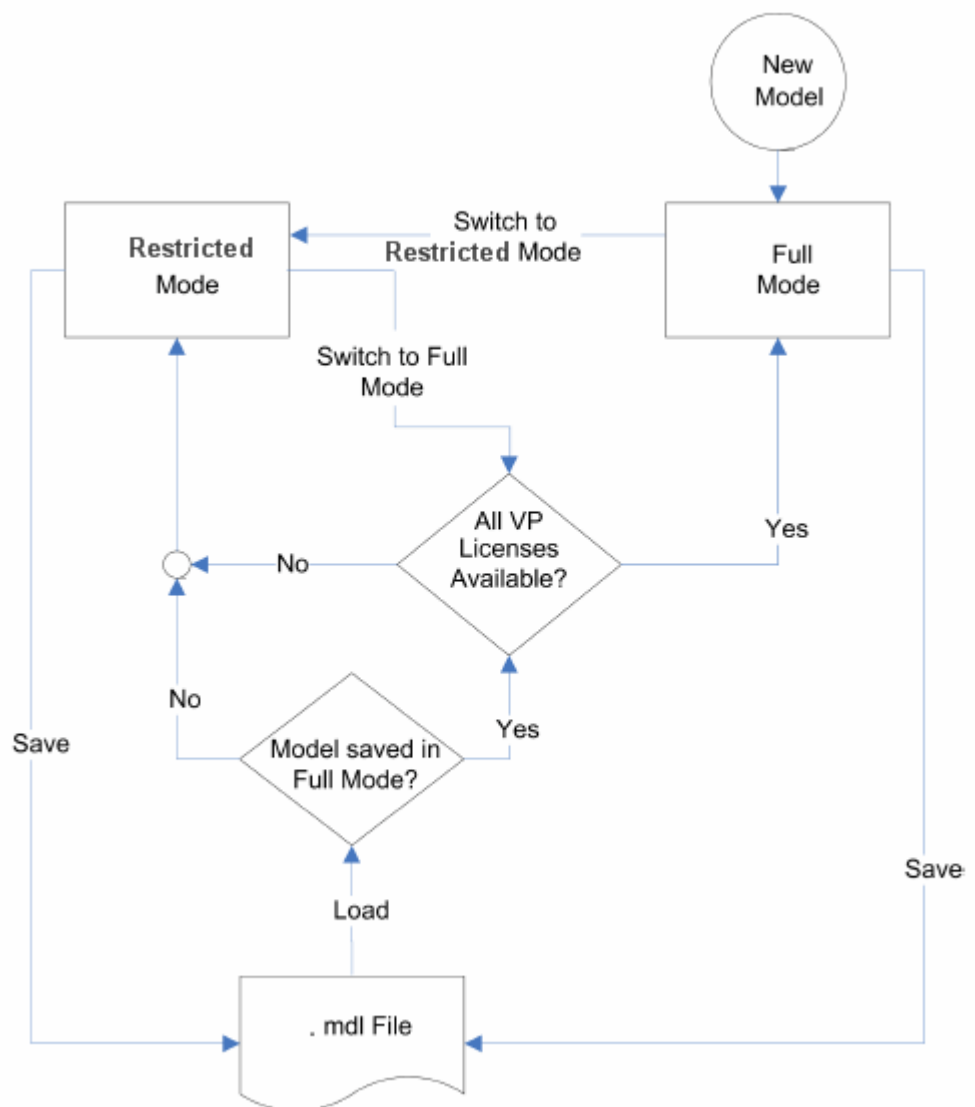
What You Can Do in Full Mode

You need to open a model in Full mode if you need to do any of the following:

- Add or delete Physical Modeling blocks (that is, Simscape blocks or blocks from the add-on product libraries).
- Make or break Physical connections (between Conserving or Physical Signal ports).
- Change the types of signals going into actuators or out of sensors (for example, from velocity to torque).
- Change configuration parameters.
- Change block parameterization options and other restricted parameters.
- Change physical units of parameters.

Switching Between Modes

The following flow chart shows what happens when you switch between modes.



New models are always created in Full mode. You can then either save the model in Full mode, or switch to Restricted mode and save the model in Restricted mode.

When you load an existing model, the license manager checks whether it has been saved in Full or Restricted mode.

- If the model has been saved in Restricted mode, it opens in Restricted mode.
- If the model has been saved in Full mode, the license manager checks whether all the add-on product licenses for this model are available and, if so, opens it in Full mode. If a add-on product license is not available, the license manager issues an error message and opens the model in Restricted mode. See also “Example with Multiple Add-On Products” on page 5-6.

Note You can set a Simulink preference to specify that the models are always to open in Restricted mode, regardless of the way they have been saved.

When a model is open, you can transition it between Full and Restricted modes at any time, in either direction:

- When you try to switch from Restricted to Full mode, the license manager checks whether all the add-on product licenses for this model are available. If a add-on product license is not available, the license manager issues an error message and the model stays in Restricted mode. See also “Example with Multiple Add-On Products” on page 5-6.
- No checks are performed when switching from Full to Restricted mode.

Note If a add-on product license has been checked out to open a model in Full mode, it remains checked out for the remainder of the MATLAB session. Switching to Restricted mode does not immediately return the license.

Example with Multiple Add-On Products

When you try to open a model in Full mode or to switch from Restricted to Full mode, the license manager scans the model and attempts to check out the required add-on product licenses as it encounters them in the model. If a license is not available, the license manager issues an error message and the model stays in Restricted mode. The licenses are checked out sequentially. As a result, if a model uses blocks from multiple add-on products, some of the

add-on product licenses may have already been checked out by the time the license manager encounters an unavailable license. In this case, these add-on product licenses stay checked out until you quit the MATLAB session, even though the model is in Restricted mode.

For example, consider a model that uses blocks from SimHydraulics and SimDriveline libraries, but the user who tries to open it has only the SimDriveline license available. It may happen that the license manager checks out a SimDriveline license first, and then tries to check out a SimHydraulics license, which is not available. The license manager then issues an error message and opens the model in Restricted mode, but the SimDriveline license stays checked out until the end of the MATLAB session.

Working with Block Libraries

This section describes the specifics of working with block libraries while using the Editing Mode functionality. These rules are applicable to any physical modeling blocks, that is, blocks from all Simscape libraries, including the add-on products. In general, you need to work in Full mode when you modify a library block. However, when you open a model that references the modified block, you may work in Restricted mode, under certain conditions. The following summary details the Editing Mode rules for modifying and using library blocks:

- To add physical modeling blocks to a library block, you need to work in Full mode.
 - If this library block had not previously contained physical modeling blocks, you need to work in Full mode to load a preexisting model that uses this library block or to drag this block to a model.
 - If this library block had previously contained physical modeling blocks, you can work in Restricted mode when loading a preexisting model that uses this library block. However, you have to work in Full mode to drag this block from the library to a model.
- To add external physical ports to a library block, you need to work in Full mode.
 - You can work in Restricted mode when loading a preexisting model that uses this library block.

- However, to connect these additional ports, you need to work in Full mode because you are changing the model topology.
- To delete external physical ports from a library block, you need to work in Full mode. If these ports were connected in a model saved in Restricted mode, loading the model causes the topology to change, so you need to switch to Full mode to save or compile the model.

Resolving Block Library Links

All Simscape blocks in your models, including the add-on products' blocks, must have resolved block library links. You can neither disable nor break these library links. This is a global requirement of Simscape platform, which is necessary to enforce the Editing Mode rules for modifying and using library blocks, listed above. A model with broken library links will neither compile nor save. You must restore all the broken block library links for your model to be valid.

If you want to customize certain blocks and use them in your models, you must add these modified blocks to your own custom library, then copy the block instances that you need to your model.

Working with Restricted and Full Modes

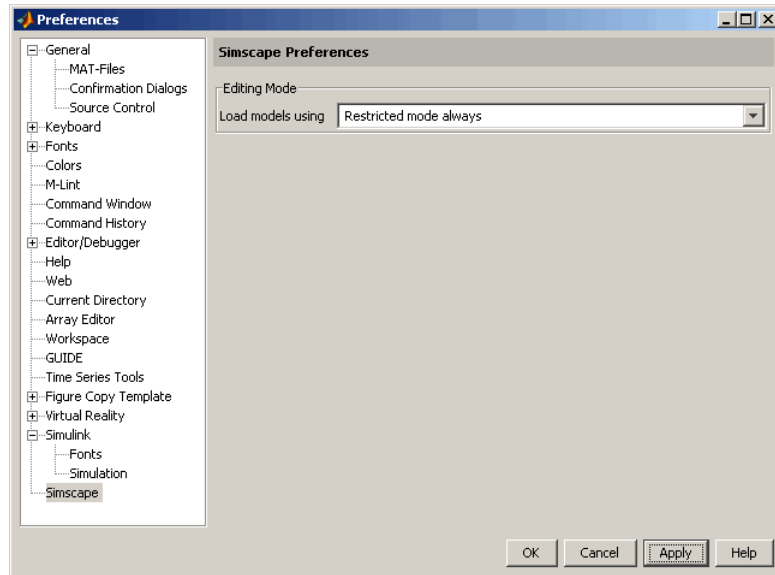
In this section...

- “Setting the Model Loading Preference” on page 5-9
- “Saving a Model in Restricted Mode” on page 5-10
- “Working with a Model in Restricted Mode” on page 5-13
- “Switching from Restricted to Full Mode” on page 5-21

Setting the Model Loading Preference

By default, when you load an existing model, the license manager checks whether it has been saved in Full or Restricted mode and tries to open it in this mode. However, you can set your preferences so that the models are always open in Restricted mode, regardless of the way they have been saved.

- 1 From the MATLAB top menu bar, select **File > Preferences**. The Preferences dialog box opens.
- 2 In the left pane of the Preferences dialog box, select **Simscape**. The right pane displays the **Editing Mode** group box. By default, the **Load models using** option is set to Editing mode specified in models.
- 3 Select **Restricted mode** always from the drop-down list, as shown, and click **OK**.

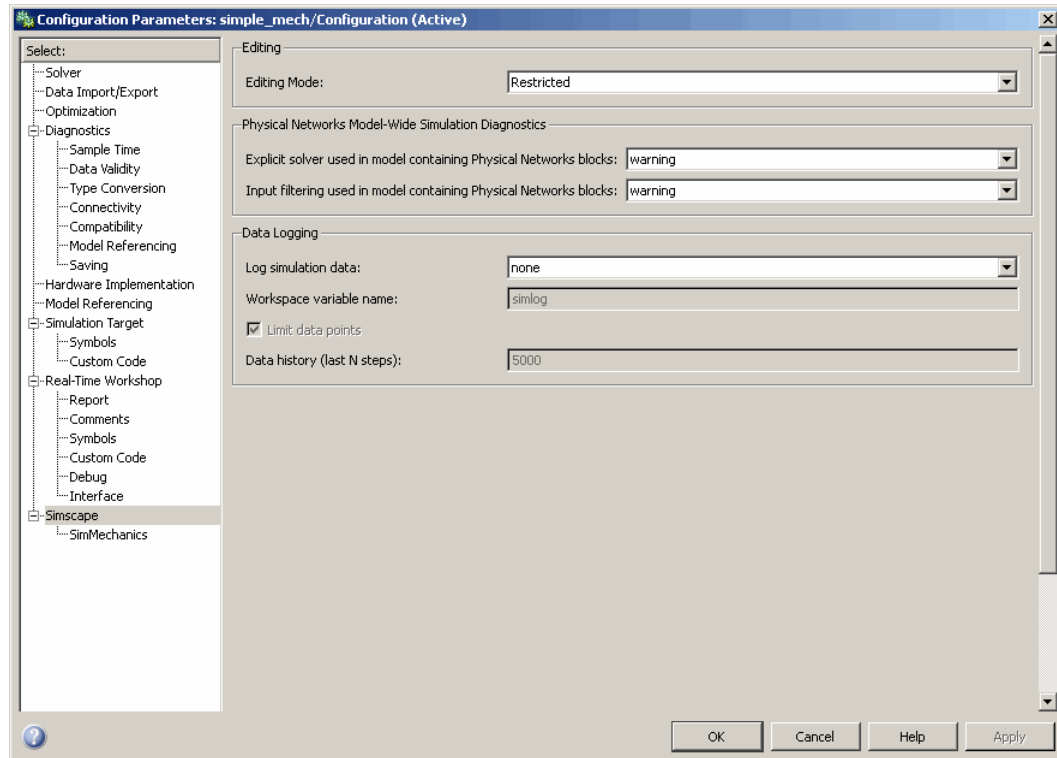


Now, when you open a model, the license manager does not attempt to check out add-on product licenses and always opens the model in Restricted mode.

Saving a Model in Restricted Mode

Rather than setting your preferences so that all the models always open in Restricted mode, you can switch an individual model to Restricted mode before saving it. Such a model will then, by default, open in Restricted mode.

- 1 From the top menu bar in the model window, select **Simulation > Configuration Parameters**. The Configuration Parameters dialog box opens.
- 2 In the left pane of the Configuration Parameters dialog box, select **Simscape**. The right pane displays the **Editing Mode** option, which is by default set to Full.
- 3 Select **Restricted** from the drop-down list, as shown, and click **OK**.



4 Save the model.

Note The **Simscape** entry does not appear in the left pane of the Configuration Parameters dialog box until you add at least one Physical Modeling block to your model. If you create an additional configuration set for a model, the **Simscape** entry does not appear in it until you either activate it or perform a Physical Modeling operation, such as adding or deleting a Physical Modeling block or connection, opening a Physical Modeling block dialog box, and so on.

Once you have switched a model to Restricted mode, working with it follows the rules described in “Working with a Model in Restricted Mode” on page

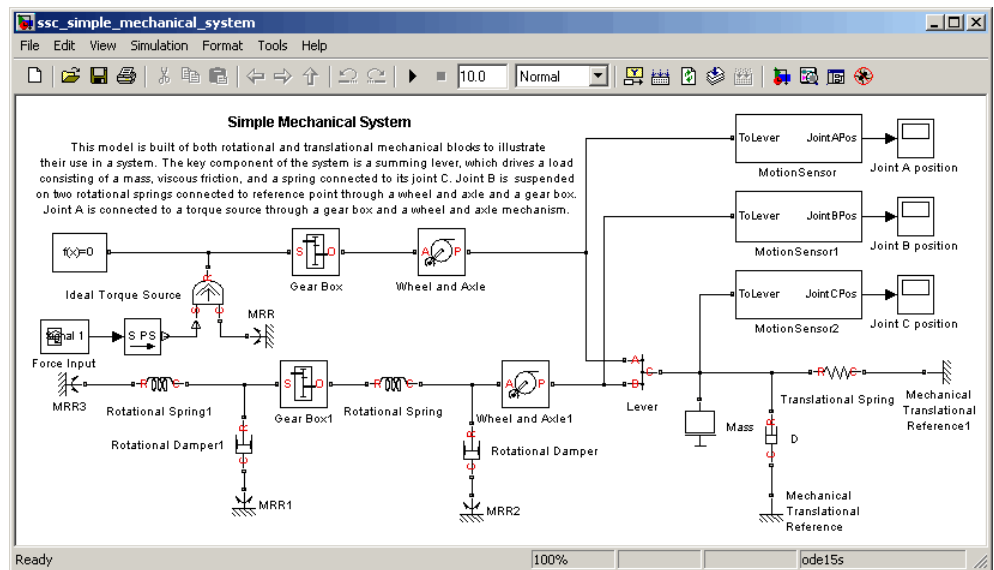
5-13. Note, however, that the add-on product licenses for this model stay checked out until you quit the MATLAB session.

When you open a model that has been saved in Restricted mode, the license manager opens it in Restricted mode and does not check out the add-on product licenses.

Example of Saving a Model in Restricted Mode

In this example, you switch a model to Restricted mode and save it.

- 1 Open the Simple Mechanical System demo model (ssc_simple_mechanical_system).



- 2 From the top menu bar in the model window, select **Simulation > Configuration Parameters**. The Configuration Parameters dialog box opens.
- 3 In the left pane of the Configuration Parameters dialog box, select **Simscape**. The right pane displays the **Editing Mode** option, which is set to Full by default.

4 Select **Restricted** from the drop-down list and click **OK**.

5 Save the model as `test_edit_mode.mdl`.

Working with a Model in Restricted Mode

When you open a model in Restricted mode, you can perform a variety of tasks: simulate the model, inspect and fine-tune block parameters, add and delete basic Simulink blocks, and so on. For a complete list of allowed operations, see “What You Can Do in Restricted Mode” on page 5-3.

When you open a block dialog box in Restricted mode, some of the block parameters may be grayed out. These are the so-called *restricted* parameters that can be modified only in Full mode. In general, you can change numerical parameter values in Restricted mode, but you cannot change the block parameterization options. See the block reference pages for specifics. Note also that when a restricted parameter defines the block parameterization schema, nonrestricted parameters available for fine-tuning in Restricted mode depend on the value of this restricted parameter. For example, in a Constant Volume Chamber block, the **Chamber specification** parameter is restricted. If, at the time the model entered Restricted mode, this parameter was set to **By volume**, then the nonrestricted parameters available for fine-tuning would be **Chamber volume**, **Specific heat ratio**, and **Initial pressure**. If, however, it was set to **By length and diameter**, you will have a different set of parameters available in Restricted mode.

You cannot change physical units in Restricted mode. When you open a block dialog box in Restricted mode, the drop-down lists of units next to a parameter name and value are grayed out. When you open a PS-Simulink Converter or Simulink-PS Converter block dialog box, the **Unit** parameter is grayed out.

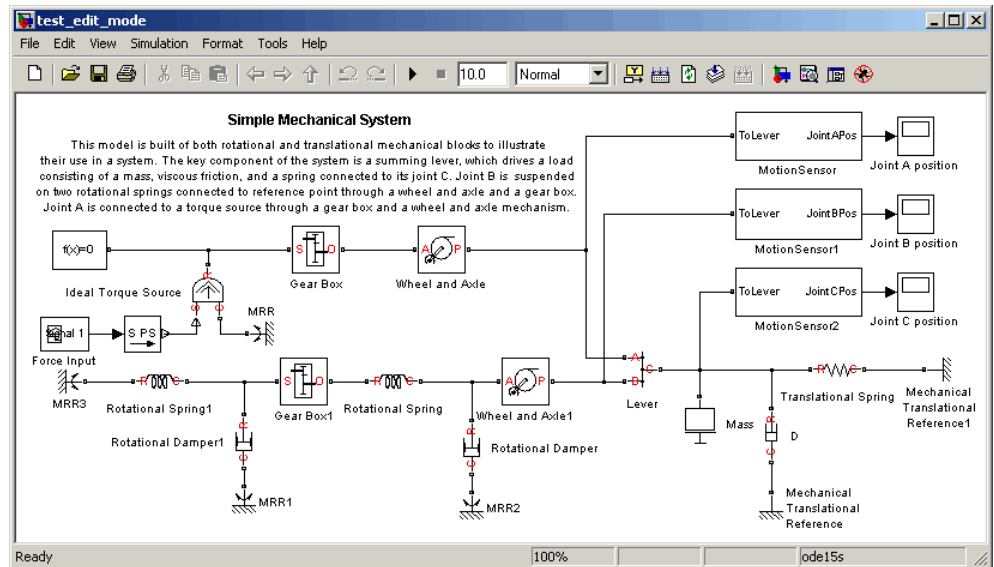
The following examples illustrate operations allowed and disallowed in Restricted mode:

- “Simulating and Fine-Tuning a Model in Restricted Mode” on page 5-14
- “Adding and Deleting Simulink Blocks in Restricted Mode” on page 5-17
- “Performing an Operation Disallowed in Restricted Mode” on page 5-19

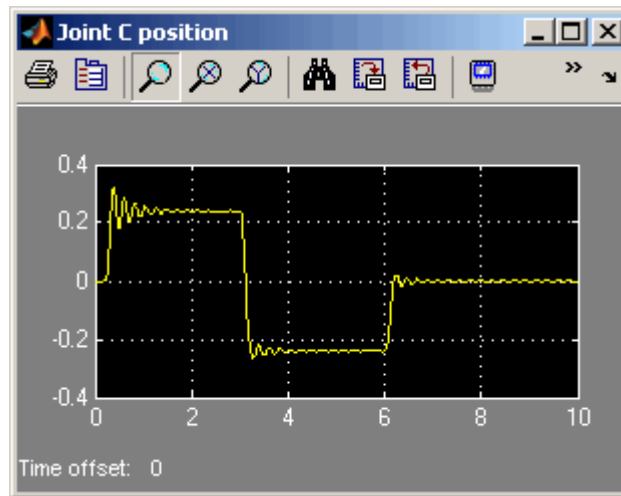
Simulating and Fine-Tuning a Model in Restricted Mode

This example shows how you can work with a model in Restricted mode by changing certain parameter values and observing the simulation results.

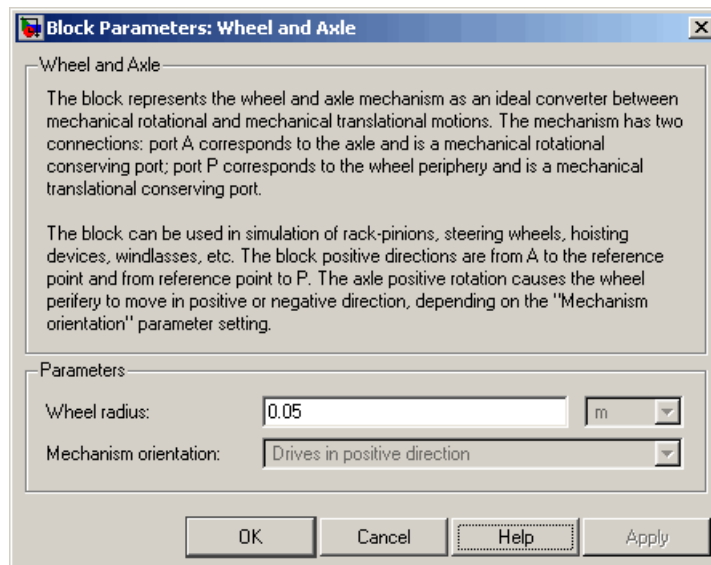
- 1 Open the `test_edit_mode` model, which you saved in Restricted mode in “Example of Saving a Model in Restricted Mode” on page 5-12. The model opens in Restricted mode.



- 2 Open the Joint C Position scope and simulate the model. The models runs and simulates in Restricted mode.

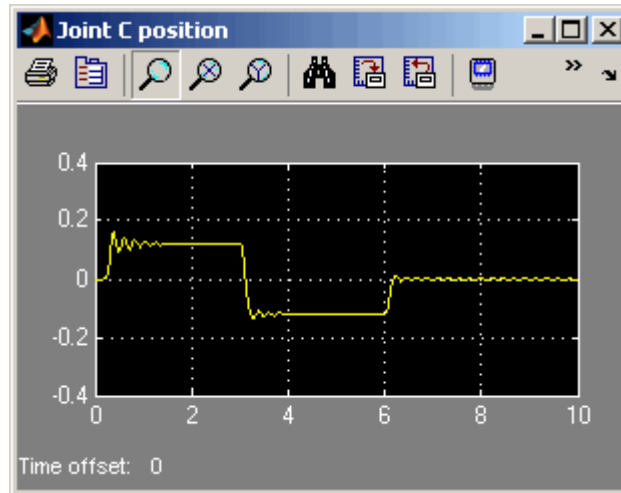


- 3 Double-click the Wheel and Axle block to open its dialog box. Notice that the **Mechanism orientation** parameter is grayed out, because you cannot modify the block driving direction in Restricted mode.

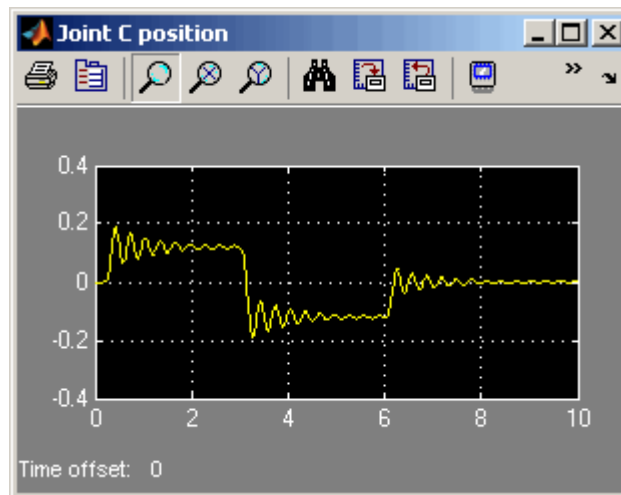


- 4 Change the **Wheel radius** parameter value to 0.1.

- 5** Simulate the model again. Notice that the motion amplitude of node C became smaller as a result of the wheel radius change.



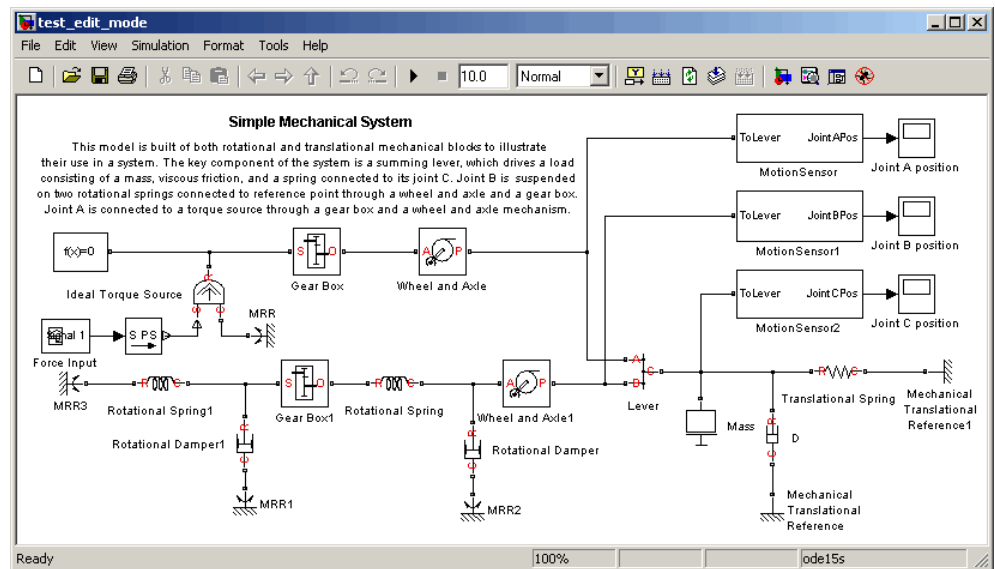
- 6** Double-click the Mass block and change the **Mass** parameter value to 24.
- 7** Simulate the model. Notice that doubling the mass resulted in increased vibrations.



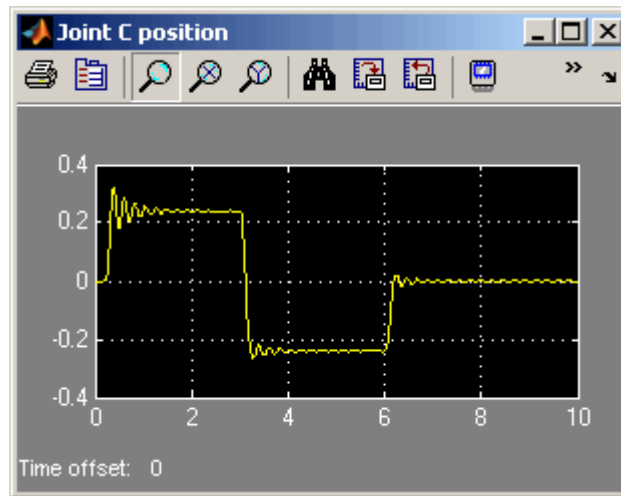
Adding and Deleting Simulink Blocks in Restricted Mode

This example shows how you can change the model input signal in Restricted mode by adding and deleting basic Simulink blocks.

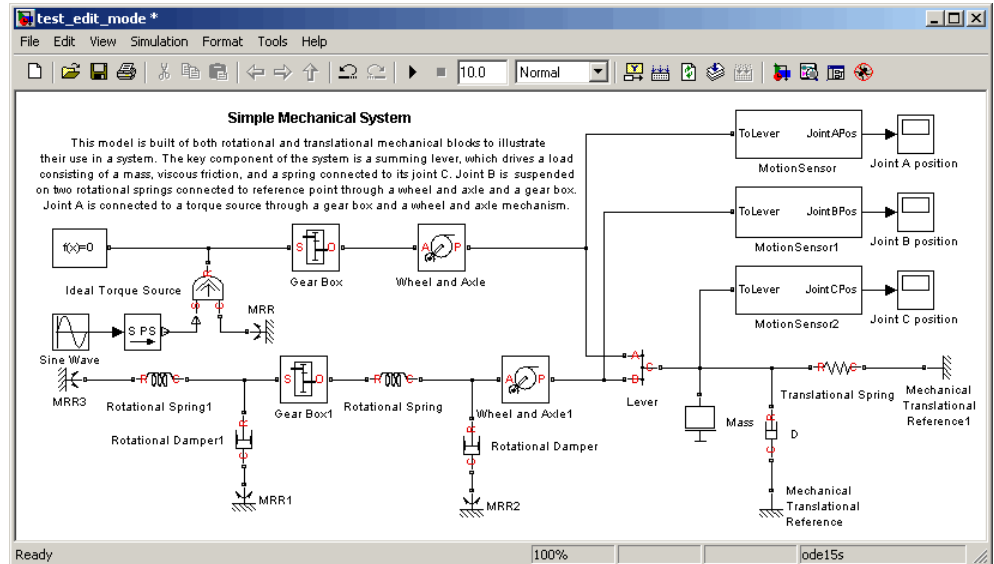
- 1 Open the `test_edit_mode` model, which you saved in Restricted mode in “Example of Saving a Model in Restricted Mode” on page 5-12. The model opens in Restricted mode.



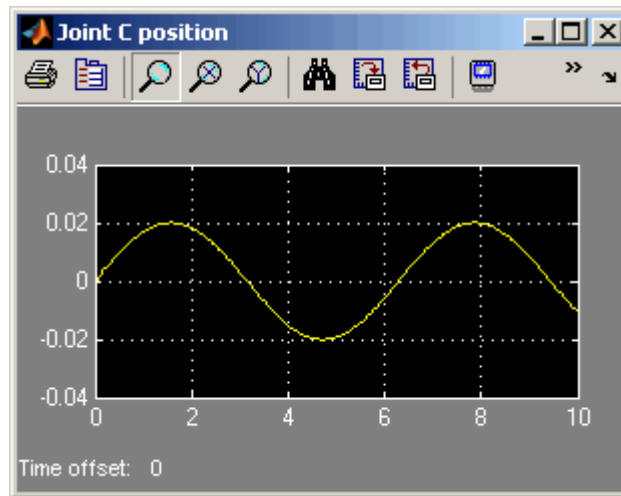
- 2 Open the Joint C Position scope and simulate the model.



- 3 Delete the Signal Builder block named Force Input. Replace it with a Sine Wave block from the Simulink Sources library, as shown below.



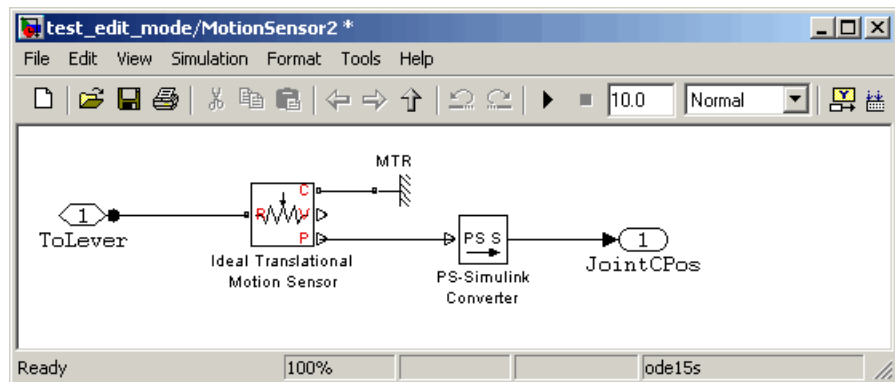
- 4 Simulate the model again. The model successfully compiles and simulates in Restricted mode.



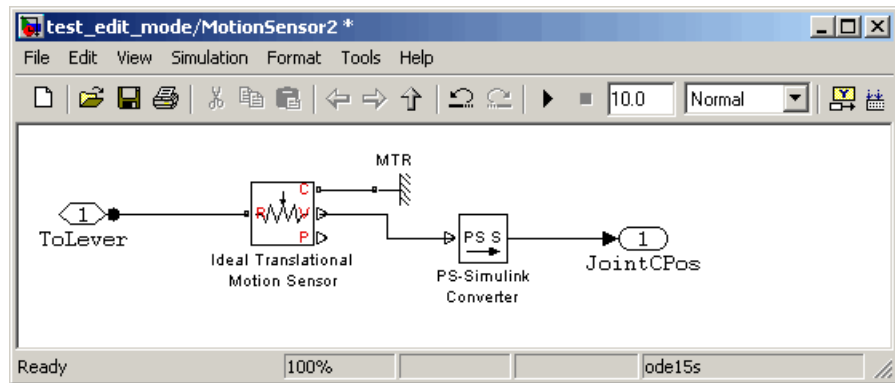
Performing an Operation Disallowed in Restricted Mode

This example shows what happens when you perform an operation that is disallowed in Restricted mode.

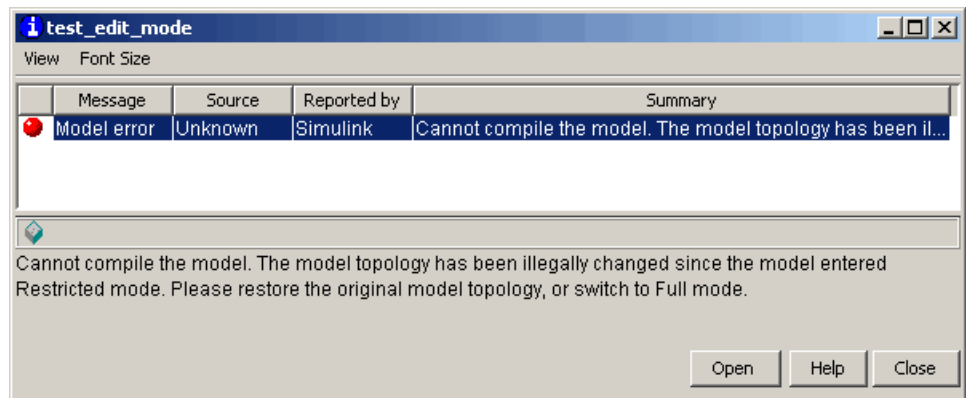
- 1 Open the `test_edit_mode` model, which you saved in Restricted mode in “Example of Saving a Model in Restricted Mode” on page 5-12. The model opens in Restricted mode.
- 2 Double-click the `MotionSensor2` block to open the subsystem.



- 3 Delete the connection line between port P of the Ideal Translational Motion Sensor block and the PS-Simulink Converter block. Instead, connect port V of the Ideal Translational Motion Sensor block to the input port of the PS-Simulink Converter block, to measure the velocity on node C of the lever.



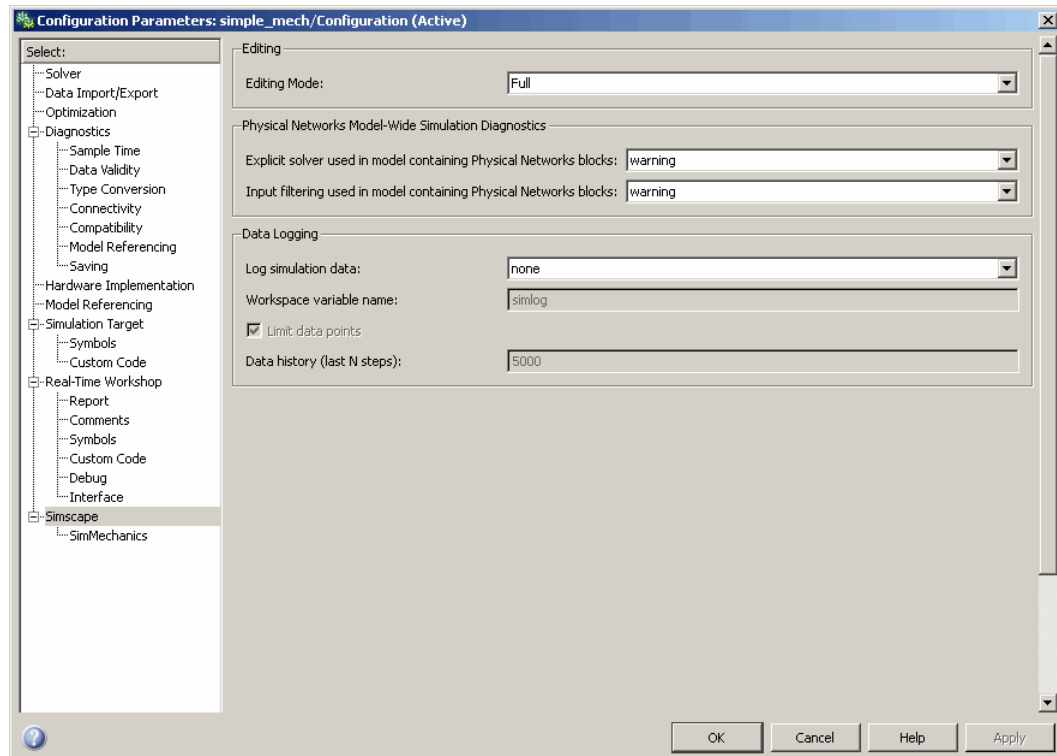
- 4 Try to simulate the model. An error message appears saying that the model cannot be compiled because its topology has been changed while in Restricted mode. You can either undo the changes, or switch to Full mode, as described in “Switching from Restricted to Full Mode” on page 5-21.



Switching from Restricted to Full Mode

If you need to perform a task that is disallowed in Restricted mode, you can try to switch the model to Full mode.

- 1 From the top menu bar in the model window, select **Simulation > Configuration Parameters**. The Configuration Parameters dialog box opens.
- 2 In the left pane of the Configuration Parameters dialog box, select **Simscape**. The right pane displays the **Editing Mode** option.
- 3 Select **Full** from the drop-down list, as shown, and click **OK**.



The license manager checks whether all the add-on product licenses for this model are available. If yes, it checks out the add-on product licenses

and switches the model to Full mode. If a add-on product license is not available, the license manager issues an error message and the model stays in Restricted mode.

Note If the switch to Full mode fails but some of the add-on product licenses have already been checked out, they stay checked out until you quit the MATLAB session. For more information, see “Example with Multiple Add-On Products” on page 5-6.

Once the model is switched to Full mode, you can perform the needed design and simulation tasks, and then either save it in Full mode, or switch back to Restricted mode and save it in Restricted mode.

Editing Mode Information

In this section...
“What Is the Current Mode?” on page 5-23
“Which Licenses Are Checked Out?” on page 5-23

What Is the Current Mode?

If you are unsure whether the model is currently open in Restricted or Full mode, you can check by following these steps.

- 1 From the top menu bar in the model window, select **Simulation > Configuration Parameters**. The Configuration Parameters dialog box opens.
- 2 In the left pane of the Configuration Parameters dialog box, select **Simscape**. The right pane displays the **Editing Mode** option, which is either Full or Restricted.
- 3 At this point, you can either try switching the mode by selecting a different option from the drop-down list, or click **Cancel** to stay in the current mode.

Which Licenses Are Checked Out?

Use the MATLAB `license` command to get a list of all the licenses currently in use. In the MATLAB Command Window, type

```
license('inuse')
```

This command returns a list of licenses checked out in the current MATLAB session. In the list, products are listed alphabetically by their license feature names.

Examples

Use this list to find examples in the documentation.

Getting Started

“Creating a Simple Model” on page 1-17

Best Practices

“Grounding Rules” on page 1-35

“Example of Using a Parasitic Resistance to Avoid Numerical Simulation Issues” on page 1-39

Editing Mode

“Example of Saving a Model in Restricted Mode” on page 5-12

“Simulating and Fine-Tuning a Model in Restricted Mode” on page 5-14

“Adding and Deleting Simulink Blocks in Restricted Mode” on page 5-17

“Performing an Operation Disallowed in Restricted Mode” on page 5-19

E

- electrical ground
 - specifying 1-35

L

- linearizing
 - Simscape™ models 2-28

N

- numerical simulation issues
 - avoiding 1-38

O

- operating points
 - finding in Simscape™ models 2-22
 - linearizing Simscape™ models at 2-28

P

- ports
 - physical conserving 1-9
 - physical signal 1-9

S

- Simscape Editing Mode 5-2
 - Full mode 5-4

- information 5-23
- Restricted mode 5-3
- saving in Restricted mode 5-10
- switching between modes 5-4
- workflows 5-2
- working in Restricted mode 5-13
- working with block libraries 5-7

- Simscape software
 - block library structure 1-11
 - editing modes 5-2
 - logging simulation data 3-2

T

- trimming
 - Simscape™ models 2-22

U

- units
 - defining physical units 4-4

V

- variables
 - across 1-4
 - direction 1-6
 - through 1-4
 - using in model equations 1-5