Simscape™ Release Notes

How to Contact MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	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™ Release Notes

Contents

R2014a

R2013b

R2013a

R2012b

R2012a

R2011b

R2011a

R2010b

R2010a

R2009b

R2009a

R2008b

R2008a

R2007b

R2007a

R2014a

Version: 3.11

New Features: Yes

Bug Fixes: Yes

Simscape Language

`ssc_build` library build process that no longer requires C compiler

In previous releases, the library build process, whether performed using the `ssc_build` command or using the `ssc_mirror` command with the `buildmirror` flag set to `true`, required a C compiler. Before running either of these commands for the first time, you had to set up your compiler by running `mex -setup`.

In-memory execution of the library build process, implemented in this release, increases the process speed and eliminates the need for a C compiler. Therefore, you no longer need to have a C compiler available on your machine in order to build a custom block library.

`priority` attribute for setting relative priority of variable target values during initialization

The new initialization process, implemented in this release, involves block-level variable initialization. That is, a Simscape™ block dialog box now has an additional **Variables** tab, which lists all the public variables specified in the underlying component file, along with the initialization priority, target initial value, and unit of each variable. The block user can change the variable priority and target, prior to simulation, to affect the model initialization. The default values for variable priority, target value, and unit come from the variable declaration in the component file.

To enable the block author to specify default priority for a variable, a new attribute is now available. The `priority` attribute can have one of the three values: `priority.high`, `priority.low`, and `priority.none`. The default is `priority.none`, which is equivalent to leaving out the `priority` attribute entirely.

To specify a high or low default priority for a component variable, declare the variable as a field array. For example, the following declaration initializes variable `t` (spring deformation) as 0 mm, with high priority:

```
variables
    t = { value = { 0 , 'mm' }, priority = priority.high }; % Spring deform
end
```

The old way of declaring variables still works. In fact, leaving out the `priority` attribute (that is, using `priority.none`) is suitable in most cases. The block user can control the variable initialization priority, as needed, by using the **Variables** tab of the block dialog box.

For example, you can declare the same variable `t` as follows:

```
variables
    t = { 0 , 'mm' }; % Spring deformation
end
```

In this case, the **Variables** tab in the block dialog box will have the **Spring deformation** variable listed initially as `Unused` (which means its priority and target are not used in the initialization process). The block user can modify the variable priority, as well as the target value and unit, in the **Variables** tab of the block dialog box prior to simulation.

If there are no top-level public variables declared in the component file (for example, if the top-level variables are declared as hidden), the **Variables** tab does not appear in the block dialog box. The same is true for composite components, because they also have no top-level public variables.

For more information on block-level variable initialization, see “Variables tab for specifying target value and priority for new initialization process” on page 5.

Script for replacing `through` and `across` statements with `branches` and equations

The `ssc_update` script is now available to help you update the legacy component files that contain `across` and `through` statements. The scripts replaces the `through` statements with the corresponding `branches` section, and adds the equations equivalent to the `across` statements to the `equations` section of the file.

For more information on the old and new syntax, see Compatibility Considerations under "`branches` section for defining the relationship between component Through variables and nodes" on page 14.

To run the script, at the MATLAB® command prompt, type:

```
ssc_update package
```

where *package* is the name of a top-level package directory, without the leading + character. If you run the `ssc_update` command from inside the package directory structure, you can omit the argument. The script updates all the legacy component files located in the package. For more information, see `ssc_update`.

Foundation Library and Simulation

Variables tab for specifying target value and priority for new initialization process

Compatibility Considerations: Yes

New initialization process, implemented in this release, gives you more control over model initialization. Most of the Foundation library blocks now have a new **Variables** tab, which lists all the public variables specified in the underlying component file, along with priority, initial value, and unit. In most cases, the default value for each of these is `Unused`. Once you select the check box next to a variable name, you can specify its priority (`High` or `Low`), target initial value, and unit.

If the underlying component has no top-level public variables (such as a composite component, or one with the top-level variables declared as hidden), then the block dialog box does not have the **Variables** tab. For that reason, the Utilities library blocks, most of the Foundation library sensors and sources, and the absolute majority of blocks in the add-on products are not affected by this change.

The values you specify during block-level variable initialization are not the actual values of the respective variables, but rather their target values at the beginning of simulation ($t = 0$). Depending on the results of the initial conditions solve, some of these targets may or may not be satisfied. The solver tries to satisfy the high-priority targets first, then the low-priority ones:

- At first, the solver tries to find a solution where all the high-priority variable targets are met exactly, and the low-priority targets are approximated as closely as possible. If the solution is found during this stage, it satisfies all the high-priority targets. Some of the low-priority targets might also be met exactly, the others are approximated.
- If the solver cannot find a solution during the first stage, it issues a warning and enters the second stage, where `High` priority is relaxed to `Low`. That is, the solver tries to find a solution by approximating both the high-priority and the low-priority targets as closely as possible.

After you initialize the block variables and prior to simulating the model, you can open the Variable Viewer to see which of the variable targets have been satisfied. For more information on block-level variable initialization and Variable Viewer, see "Variable Initialization".

Compatibility Considerations

In previous releases, several Foundation library blocks contained parameters that let you specify an initial value for an internal block variable at the start of simulation. These parameters have now been removed. The following table lists the initialization parameters that have been removed from block dialogs and the names of the corresponding block variables:

Block Name	Parameter Name	Variable Name
Capacitor	**Initial voltage**	**Capacitor voltage**
Constant Volume Chamber (TL)	**Fluid initial pressure** **Fluid initial temperature**	**Pressure** **Temperature**
Constant Volume Hydraulic Chamber	**Initial pressure**	**Pressure (gauge)**
Constant Volume Pneumatic Chamber	**Initial pressure** **Initial temperature**	**Pressure** **Temperature**
Fluid Inertia	**Initial flow rate**	**Flow rate**
Hydraulic Piston Chamber	**Initial pressure**	**Pressure (gauge)**
Inductor	**Initial current**	**Inductor current**
Inertia	**Initial velocity**	**Rotational velocity**
Mass	**Initial velocity**	**Velocity**
Mutual Inductor	**Winding 1 initial current** **Winding 2 initial current**	**Primary current** **Secondary current**

Block Name	Parameter Name	Variable Name
Pneumatic Piston Chamber	**Initial pressure** **Initial temperature**	**Pressure** **Temperature**
Rotational Hard Stop	**Initial angular position**	**Angular position**
Rotational Spring	**Initial deformation**	**Deformation**
Rotary Pneumatic Piston Chamber	**Initial pressure** **Initial temperature**	**Pressure** **Temperature**
Thermal Mass	**Initial temperature**	**Temperature**
Translational Hard Stop	**Initial position**	**Position**
Translational Spring	**Initial deformation**	**Deformation**
Variable Hydraulic Chamber	**Initial pressure**	**Pressure (gauge)**

Legacy models using these blocks are not affected by this change. If a block had an initialization parameter, then, once you open the model in the current release, this parameter value is automatically mapped to the corresponding block variable, which assumes it as the target value with `High` priority. The simulation results stay the same.

However, if you have a custom composite component (written in Simscape language) that uses one of these Foundation library blocks and references an initialization parameter, trying to build the library or simulate a model containing the custom block produces an error, because the referenced parameter is no longer available.

For example, if you have a custom composite component that contains an Inertia block:

```
component DC_Motor
[...]
  parameters
     motor_inertia = { 0.01, 'g*cm^2' };       % Inertia
     init_velocity = { 0, 'rad/s' };           % Initial velocity
```

```
    [...]
  end
  components(Hidden=true)
    motorInertia = foundation.mechanical.rotational.inertia(inertia = motor_inertia,
                   initial_velocity = init_velocity);
    [...]
  end
[...]
end
```

`ssc_build` will produce an error similar to the following:

```
Error using ne_updatelibraryitem>lBuild (line 35)     File: C:\Work\libraries\+MyElecLibrary\DC_Motor.ssc
Line: 32 Reference to parameter     'initial_velocity' is invalid.
```

Update the custom component by removing all references to the initialization parameter:

```
component DC_Motor
[...]
  parameters
    motor_inertia = { 0.01, 'g*cm^2' };       % Inertia
    [...]
  end
  components(Hidden=true)
    motorInertia = foundation.mechanical.rotational.inertia(inertia = motor_inertia);
    [...]
  end
[...]
end
```

Variable Viewer for analyzing results of new initialization process

A new analysis tool, available for models containing Simscape blocks and blocks from add-on products, provides the back end for the block-level variable initialization by letting you view the variable targets, priority, and actual initial values prior to simulation. To open the tool, in the top menu bar of the model window, select **Analysis > Simscape > Variable Viewer**. For more information, see “Variable Viewer”.

Statistics Viewer that displays variable source and number of eliminated variables

The Statistics Viewer analysis tool has the following enhancements:

- New top-level statistic, **1-D/3-D Interface**, lists the connections between the two types of physical networks. It appears only for models that connect blocks from SimMechanics™ Second Generation library to Simscape blocks, or blocks from other add-on products.
- The new **Sources** section lists variable sources for the selected statistic. If you select a connection under the **1-D/3-D Interface** statistic category, the **Sources** section lists the source and destination for this connection. If you select a statistic with a nonzero value under the **1-D Physical System** category, the **Sources** section lists all the variables that fall under this statistic.

  For each variable, the **Source** column contains the full path to the variable, starting from the top-level model, with a link to the relevant block. If you click the link in the **Source** column, the corresponding block is highlighted in the block diagram. The **Value** column contains the name of the variable, as it would appear in the **Variables** tab of the block dialog box.
- Additional statistics under the **1-D Physical System** category: **Number of eliminated variables** (further categorized as algebraic and differential variables) and **Number of dynamic variable constraints**. Eliminated variables are continuous variables that are eliminated during optimization and are not seen by the solver. Dynamic variable constraints are constraints involving only dynamic variables and inputs. Such constraints result in Index-2 differential algebraic equations and therefore can cause numerical difficulties or slow down your simulation.

For more information, see “Model Statistics”.

Fundamental Reluctance block

The new Fundamental Reluctance block in the Magnetic Elements library provides a simplified model of a magnetic reluctance, that is, a component that resists flux flow. Unlike the Reluctance block, which computes reluctance based on the geometry of the section being modeled, the Fundamental

Reluctance block lets you specify a value of reluctance directly as a block parameter.

Hydro-mechanical converter blocks with fluid compressibility option

Compatibility Considerations: Yes

The Rotational Hydro-Mechanical Converter and Translational Hydro-Mechanical Converter blocks now contain a drop-down **Compressibility** parameter, with the default value `Off`. If you select `On`, additional parameters appear in the block dialog to let you account for fluid compressibility within the converter block itself. In previous releases, you had to connect a converter to a Hydraulic Piston Chamber block to account for fluid compressibility.

Compatibility Considerations

Existing models are not affected by this change, because compressibility in converters is off by default. However, this change makes the Hydraulic Piston Chamber block obsolete. MathWorks recommends that you specify fluid compressibility directly in the converter blocks, because the new method provides more accurate results and also because the Hydraulic Piston Chamber block may be removed in a future release.

Handling of pressure or temperature below absolute zero during simulation

You can now set the models that use the hydraulic domain to either warn or stop simulating with an error when absolute pressures fall below absolute zero. The default behavior is to stop simulating with an error, which is the same as in previous releases. You can change this by using the Custom Hydraulic Fluid block, or the Hydraulic Fluid block (available with SimHydraulics® block libraries), to have the simulation continue with a warning. See the block reference pages for details.

You can also set the models that use the pneumatic domain to either warn or stop simulating with an error when pressures or temperatures fall below

absolute zero. The default behavior is to stop simulating with an error. This check was not performed during simulation in previous releases. You can change the default behavior by using the Gas Properties block, to have the simulation continue with a warning. See the block reference page for details.

Input filtering options for 1-D/3-D connections

The Solver Configuration block now lets you control whether input filtering is applied automatically for models that connect blocks from SimMechanics Second Generation library to Simscape blocks, or blocks from other add-on products. It also lets you specify a global filtering time constant value for all the 1-D/3-D connections within the network. See the block reference page for details.

Software-in-the-loop simulation for physical models

In previous releases, software-in-the-loop (SIL) simulation was not supported for models containing Simscape blocks or blocks from the add-on products. This limitation is now removed.

Change in default settings for `ssc_new`

The `ssc_new` function, which creates a new Simscape model populated by required and commonly used blocks, now uses a different default solver and absolute tolerance. Here is the summary of changes:

	Old setting	New setting
Solver	`ode15s (stiff/NDF)`	`ode23t (mod. stiff/Trapezoidal)`
Absolute tolerance	`auto`	`1e-3`

Existing models are not affected. When you create new models with `ssc_new`, they will use the new settings. For more information, see the `ssc_new` reference page.

Functionality being removed or changed

Compatibility Considerations: Yes

Simscape Language Keyword Name	What Happens When You Use the Keyword?	Use This Instead	Compatibility Considerations
`across`	Still runs	Create equivalent equations	Run `ssc_update` to update the components in an existing package. See "Script for replacing `through` and `across` statements with `branches` and equations" on page 3.
`through`	Still runs	`branches`	Run `ssc_update` to update the components in an existing package. See "Script for replacing `through` and `across` statements with `branches` and equations" on page 3.

R2013b

Version: 3.10

New Features: Yes

Bug Fixes: Yes

Simscape Language

`branches` section for defining the relationship between component Through variables and nodes

Compatibility Considerations: Yes

In previous releases, the Through and Across variables in a component file were connected to the domain Through and Across variables in the `setup` section, using the `through` and `across` statements. The syntax of these statements was nonintuitive and somewhat confusing.

Starting in this release, define the relationship between the Through variables by using the new `branches` section, located after the `setup` section in the component file. The `branches` section starts with the `branches` keyword, contains one or more branch statements, and ends with the `end` keyword.

Each branch statement has the syntax:

```
a : node1.a -> node2.a;
```

which clearly indicates direction, from `node1` to `node2`. Therefore, `a` is subtracted from the conserving equation identified by `node1.a`, and `a` is added to the conserving equation identified by `node2.a`. If the component has multiple nodes, indicate branches by writing multiple statements in the `branches` section. For syntax and examples, see the `branches` reference page.

To establish the relationship between the Across variables, use the `equations` section of the component file. Add an equation that connects the component Across variable with the respective variables at the component nodes. If there is more than one Across variable, add multiple equations, connecting each variable with its respective nodes.

For more information, see Defining Relationship Between Component Variables and Nodes.

Compatibility Considerations

You do not currently need to update the existing component files. The `across` and `through` statements compile without warnings. In a future release, they

will start to produce warnings when you attempt to build the component. Eventually they will be removed. When writing new component files, use the new syntax.

If you want to update your existing component files, the following table summarizes the old and new syntax.

Old Syntax	New Syntax
`through( a, node1.a, node2.a );`	`a :  node1.a -> node2.a ;`
`across( a, node1.a, node2.a );`	`a == node1.a - node2.a ;`

For example, suppose you have an electrical component that uses the old syntax:

```
component my_resistor
  nodes
    p = foundation.electrical.electrical;
    n = foundation.electrical.electrical;
  end
  variables
    i = { 0, 'A' };
    v = { 0, 'V' };
  end
  parameters
    R = { 1, 'Ohm' };   % Resistance
  end
  function setup
    across( v, p.v, n.v );  % voltage across
    through( i, p.i, n.i ); % current through
  end
  equations
    v == i*R;
  end
end
```

Here is the same component rewritten using the new syntax:

```
component my_resistor
```

```
  nodes
    p = foundation.electrical.electrical;
    n = foundation.electrical.electrical;
  end
  variables
    i = { 0, 'A' };
    v = { 0, 'V' };
  end
  parameters
    R = { 1, 'Ohm' };  % Resistance
  end
  branches
    i : p.i -> n.i ;   % current through
  end
  equations
    v == p.v - n.v;    % voltage across
    v == i*R;
  end
end
```

`import` statement enabling simplified access to other component classes

With the introduction of composite components in R2012b, class member declarations now include user-defined types, that is, component classes. An import mechanism provides a convenient means to accessing classes defined in different scopes, with the following benefits:

- Allows access to model class names defined in other scopes without a fully qualified reference
- Provides a simple and explicit view of dependencies on other packages

The `import` statement can have the following syntax:

```
import package_or_class;
```

or

```
import package.*
```

The first syntax is a qualified import, which imports a specific package or class. The second one is an unqualified import, which imports all subpackages and classes under the specified package.

For more information, see Importing Domain and Component Classes.

`connect` statement support for vector and matrix physical signals

Simscape language now supports nonscalar (vector-valued or matrix-valued) physical signals in `connect` statements. For more information, see Nonscalar Physical Signal Connections.

Foundation Library and Simulation

Thermal Liquid domain and block library

The Foundation library now contains a thermal liquid domain and Thermal Liquid block library. This library contains thermohydraulic elements, such as chambers, reservoirs, local restrictions, and hydro-mechanical converters. It also contains thermal liquid sources and sensors, as well as a Thermal Liquid Settings (TL) block, which controls thermal liquid domain properties for the attached circuit.

Use these blocks for modeling applications such as:

- Transportation of heated liquid in pipeline networks
- Actuator warm-up due to viscous stresses
- Heat generation and dissipation in complex systems, such as aircraft hydraulic systems and associated heat exchangers

For more information, see the block reference pages. See also Thermal Liquid Domain for information on the thermal liquid domain definition. The Across variables are pressure and temperature, and the Through variables are mass flow rate and thermal flux. Note that the product of each pair of the Through and Across variables (pressure and mass flow rate, temperature and thermal flux) is not power, and therefore these result in a pseudo-bond graph.

Simscape model statistics viewer

A new analysis tool, available for models containing Simscape blocks and blocks from add-on products, lets you view Simscape statistics, such as the number of continuous and discrete variables, number of zero-crossing signals, and number of joints and constraints. To open the tool, in the top menu bar of the model window, select **Analysis > Simscape > Statistics Viewer**.

For more information, see Simscape Model Statistics.

Removal of laminar-turbulent zero-crossings in hydraulic blocks

Hydraulic blocks in the Foundation library no longer produce zero-crossings upon transition between the laminar and turbulent regimes during simulation. This enhancement results in increased simulation efficiency for hydraulic models.

New examples

Examples introduced in this version are:

- Hydraulic Fluid Warming due to Losses
- Optimal Pipeline Geometry for Heated Oil Transportation
- Water Hammer Effect

Functionality being removed or changed

Compatibility Considerations: Yes

Simscape Language Keyword Name	What Happens When You Use the Keyword?	Use This Instead	Compatibility Considerations
`across`	Still runs	Create equivalent equations	See Compatibility Considerations under "`branches` section for defining the relationship between component Through variables and

Simscape Language Keyword Name	What Happens When You Use the Keyword?	Use This Instead	Compatibility Considerations
			nodes" on page 14.
`through`	Still runs	`branches`	See Compatibility Considerations under "`branches` section for defining the relationship between component Through variables and nodes" on page 14.

R2013a

Version: 3.9

New Features: Yes

Bug Fixes: Yes

Simscape Language

Variable-size domain parameters

Variable-size component parameters have been implemented in Release R2012a, to support the `tablelookup` function. You can now declare variable-size domain parameters and propagate them to components. Variable-size parameters are still not allowed in the component `equations` section outside of the `tablelookup` function. For more information, see Using Lookup Tables in Equations and Propagation of Domain Parameters.

Vector and matrix physical signals

Simscape language now supports nonscalar (vector-valued or matrix-valued) physical signals in its `inputs` and `outputs` declarations. All signals in such vector or matrix should have the same units. For example, the following declaration

```
inputs
  I = {zeros(3), 'm/s'}; % :left
end
```

initializes a component input as a 3-by-3 matrix of linear velocities.

Simulink-PS Converter and PS-Simulink Converter blocks have also been enhanced to handle vector and matrix physical signals.

Foundation Library and Simulation

Random Number and Uniform Random Number blocks

Two new Physical Signal blocks have been added to the Sources library:

- The Random Number block generates normally (Gaussian) distributed random numbers.
- The Uniform Random Number block generates uniformly distributed random numbers.

The block behavior is the same as that of the respective Simulink® blocks, except that they generate a physical signal rather than a unitless Simulink signal.

Perfect Insulator block for thermal domain

The new Perfect Insulator block in the Thermal Elements library models a thermal element with no thermal mass and perfect insulation. Use this block as an insulation for thermal ports to prevent heat exchange with the environment and to model an adiabatic process.

Initialization and diagnostic improvements

When you use local solver in a model, an alternative initialization technique is automatically employed if the standard initialization techniques fail. This alternative technique attempts to find consistent states, within numerical tolerance, by taking a small finite step from the user-specified initial states. Therefore, if the alternative technique succeeds, a warning is issued to the command line that user-specified initial conditions may not be satisfied. Employing this alternative technique increases the initialization robustness, especially when there are nonlinear constraints among dynamic states of a model.

The missing reference node diagnostics have been improved to include information about the particular block and variable that needs a reference

node. This is especially helpful when multiple domains are involved in the model.

Model Advisor checks for outdated blocks and for physical unit consistency

Model Advisor user interface now contains two checks specific to Simscape models:

- **Check consistency of block parameter units** notifies you about differences between the declared and the actual settings of block parameter units. The check is triggered by default when you run Model Advisor on your model. You can also run this check individually by selecting **By Product | Simscape** or **By Task | Modeling Physical Systems**. The check detects block parameters in which the specified unit is not directly convertible into the default unit expected by the block. For example, it alerts you if a block parameter is declared with the unit of `rad/s` but the value specified in your pre-R2013a model is in `Hz`. This situation can be problematic because of the new unit definition for Hz in R2013a (see "Unit definition of Hz now consistent with SI" on page 25).

  After you run the check, a table of results appears in the right pane of the Model Advisor window. Each cell in the first column of the table contains a link to the problematic block, and the corresponding cell in the second column contains the name of parameter in question, the expected unit, and the specified unit.

  Clicking on a link highlights the corresponding block in the model. Double-click the highlighted block, verify the parameter unit setting and correct it, if necessary. Then save and reload the model.

- **Check for outdated Simscape blocks** detects a pre-R2013a version of AC Voltage Source and AC Current Source blocks in your model. The check is triggered by default when you run Model Advisor. You can also run this check individually by selecting **By Product | Simscape**.

  After you run the check, a list of links to the outdated blocks appears in the right pane of the Model Advisor window. Clicking on a link highlights the corresponding block in the model.

To update the blocks, scroll down the right pane of the Model Advisor window and click the **Update** button.

- If the automatic update is successful, the Results box displays a message that all blocks have been updated to the current Simscape version.
- If the message says that some of the blocks could not be updated automatically, rerun the check and manually replace the outdated blocks with the latest version from the block library.

For more information on using Model Advisor, see Consulting Model Advisor.

Unit definition of Hz now consistent with SI

Compatibility Considerations: Yes

The unit definition for Hz is now `1/s`, in compliance with the SI unit system. In previous releases, the unit definition for Hz was `rev/s`, consistent with the definition of frequency as cycles per second in an electrical context, or revolutions per second in a mechanical context. The old unit definition allowed you to specify frequency in angular units (such as `rad/s` or `rpm`) and write frequency-dependent equations without requiring the `2*pi` conversion factor. The main reason for changing the Hz unit definition is to give the block author responsibility for frequency units and their correct interpretation for that block. While a sinsuisoidal source might reasonably be given a frequency in units of Hz or rpm, angular frequency has no relevance when frequency refers to a nonrotational periodic signal such as the frequency of a PWM source.

As a result of the new unit definition for Hz, frequency units and angular velocity units are no longer directly convertible, and using one instead of the other may result in unexpected conversion factors applied to the numerical values by the block equations.

Drop-down lists of suggested units in block dialogs have been updated to reflect this change. For example, if your block has a **Frequency** parameter with the default unit of `Hz`, the drop-down list for this parameter now contains only units directly convertible to Hz (such as `kHz`, `MHz` and `GHz`) and does not contain the angular velocity units (such as `rpm`, `deg/s` and `rad/s`). You can still type a unit expression representing angular frequency into the units combo box, and the block will accept it as commensurate with the expected

parameter unit, but it is your responsibility to make sure that the specified unit works correctly with the block equations.

For more information, see Units for Angular Velocity and Frequency.

Compatibility Considerations

Two Foundation library blocks, AC Current Source and AC Voltage Source, have been affected by this change. In previous releases, you could specify the **Frequency** parameter for these blocks either in units of `Hz` or in angular units, such as `rad/s` or `rpm`. Starting with Release R2013a, you must specify the **Frequency** parameter in units of `Hz` or directly convertible to Hz (such as `1/s`, `kHz`, `MHz` and `GHz`) because the internal equation of the block now uses the `2*pi` conversion factor to account for the `1/s` unit definition.

If you have a pre-R2013a model that contains these blocks, update it by running the Model Advisor check, **Check for outdated Simscape blocks**, or by using the `slupdate` utility, and then save the model. A related Model Advisor check, **Check consistency of block parameter units**, notifies you about differences between the declared and the actual settings of block parameter units. For more information, see "Model Advisor checks for outdated blocks and for physical unit consistency" on page 24.

If you have custom Simscape libraries written in R2012b or earlier, and you have used `Hz`, `kHz`, `MHz`, and `GHz` as parameter units, then you will need to update your Simscape code to take account of the Hz unit change. Previously the Simscape unit manager automatically converted any value entered in units of `Hz` into units of `rad/s` before computation. Therefore, now you need to introduce a factor of `2*pi` into block equations to convert to `rad/s` and retain the old functionality.

New format for saving simulation data log objects

Compatibility Considerations: Yes

Data logging functionality lets you log simulation data to the workspace, in the form of a workspace variable. As with any workspace variable, you can save the data log to a MAT-file, and then load the file into workspace at a later date to query and analyze the data.

Starting with Release R2013a, a new format for saving the data log objects in a MAT-file has been introduced. The new format reduces the disk space usage and memory consumption, and makes it faster to save and load simulation data logs.

Compatibility Considerations

There is no backward incompatibility. That is, you can load previously saved MAT-files containing simulation data with no restrictions.

However, there is a forward incompatibility. MAT-files containing simulation data log objects saved in the new format (starting with Release R2013a) cannot be opened in older versions of MATLAB software (Release R2012b or earlier).

New examples

The following example has been introduced in this version:

- Engine Cooling System

Functionality being removed or changed

Compatibility Considerations: Yes

Simscape Language Keyword Name	What Happens When you use the Keyword?	Use This Instead	Compatibility Considerations
`throughs`	The change was introduced in Release R2009a. The keyword has now been removed, and using the old syntax produces an error, instead of a warning,	`variables(Balancing = 'Sine')`	See "Simscape Language Syntax Changes" on page 69 in Release R2009a.

Simscape Language Keyword Name	What Happens When you use the Keyword?	Use This Instead	Compatibility Considerations
	when you attempt to build the component.		

R2012b

Version: 3.8

New Features: Yes

Bug Fixes: Yes

Simscape Language

Connection of components within a Simscape file

In physical modeling, there are two types of models:

- Behavioral — A model that is implemented based on its physical behavior, described by a system of mathematical equations. An example of a behavioral block implementation is the Variable Orifice block.
- Composite — A model that is constructed out of other blocks, connected in a certain way. An example of a composite, or structural, block implementation is the 4-Way Directional Valve block (available with SimHydraulics block libraries), which is constructed based on four Variable Orifice blocks.

In previous versions, Simscape language supported only behavioral models, that is, models defined by equations. To create a model containing multiple interconnected components, you had to define each component in a separate file, deploy it as a custom block, and then use masked subsystems in block diagrams to connect these blocks into a single composite model.

Now, additional language constructs let you create composite models directly in a Simscape file. A component file may now contain two additional blocks:

- A `components` declaration block, which begins with a `components` keyword and is terminated by an `end` keyword. This block contains declarations for all the constituent components. Each component is defined with its full path to the top-level package directory. Specify the required component parameters by declaring a corresponding parameter in the top-level `parameters` declaration block, and then passing this value on to the constituent component.

  For example, the following code includes a Foundation library Resistor block in your custom component file, with a default resistance of 10 Ohm:

```
component MyCompositeModel1
...
  parameters
     p1 = {10, 'Ohm'};
```

```
    ...
  end
  components(Hidden=true)
    r1 = foundation.electrical.elements.resistor(R=p1);
    ...
  end
...
end
```

- A `connections` block, located after the `setup` section, which begins with a `connections` keyword and is terminated by an `end` keyword. This block contains information on how the constituent components' ports are connected to one another, and to the external inputs, outputs, and nodes of the top-level component.

  For example, the following code includes the Foundation library Voltage Sensor and Electrical Reference blocks in your custom component file, connects the negative port of the voltage sensor to ground, and connects the physical signal output port of the voltage sensor to the external output of the composite component, located on the right side of the resulting block icon:

```
component MyCompositeModel2
...
  outputs
    Out = { 0.0, 'V' }; % V:right
    ...
  end
  components(Hidden=true)
    VoltSensor = foundation.electrical.sensors.voltage;
    Grnd = foundation.electrical.elements.reference;
    ...
  end
  connections
    connect(Grnd.V, VoltSensor.n);
    connect(VoltSensor.V, Out);
    ...
  end
end
```

For more information, see Creating Composite Components.

`floor`, `ceil`, `fix`, and `round` functions

You can now use the following MATLAB functions in the `equations` section of the Simscape file:

- `floor` performs rounding toward negative infinity.
- `ceil` performs rounding toward positive infinity.
- `fix` performs rounding toward zero.
- `round` performs rounding toward the nearest integer.

The PS Floor, PS Ceil, and PS Fix blocks in the Physical Signals/Nonlinear Operators sublibrary of the Foundation library have been reimplemented using the Simscape language and the corresponding function. The PS Round block has been added to the Nonlinear Operators sublibrary. It performs rounding toward the nearest integer.

Foundation Library and Simulation

Speed and efficiency improvements for simulation of switched linear systems

Switched linear systems are systems that have multiple configurations during simulation, but each configuration is linear. The changes in configuration during simulation may be due to deployment of switches or other elements. A new specialized simulator, implemented for switched linear systems, reduces the number of states and accelerates simulation. The specialized simulator is automatically employed based on the system structure; you do not have to select or enable it explicitly.

Zero-crossing statistics for Simscape logging

If you log simulation data for a Simscape model, you now have an option to log simulation statistics, including zero-crossing data. By default, the zero-crossing data is not logged. If you select the **Log simulation statistics** check box on the **Simscape** pane of the Configuration Parameters dialog box, the simulation log variable contains an additional `SimulationStatistics` node for each block that can produce zero crossings. You can then plot and analyze this data similar to other data logged to the workspace during simulation.

Counter and Repeating Sequence blocks that facilitate discrete sampling

Two new blocks in the Physical Signals/Sources library facilitate discrete sampling in physical modeling:

- The Counter block repeatedly increments the output signal by 1 with every time step, in the range between the minimum (reset) value and the maximum value. You can optionally specify an initial signal value, different from the reset value, and an initial time offset. Use this block, in conjunction with other physical signal blocks, to model discrete behaviors.

- The Repeating Sequence block outputs a periodic piecewise-linear signal. You can optionally specify an initial signal value and an initial time offset. The repeating sequence consists of a number of linear segments, connected to each other. Use this block to generate various types of physical signals, such as pulse, sawtooth, stair, and so on.

Open-circuit terminator blocks for electrical, hydraulic, and mechanical domains in Foundation library

Physical Network block diagrams do not allow unconnected Conserving ports. Previously, if you wanted to leave a port unconnected (open-circuit) you had to add an extra sensor, which cluttered the block diagram.

Now the following blocks represent domain-specific open-circuit terminators:

- The Open Circuit block represents an electrical terminal that draws no current. Use this block to terminate electrical ports that you want to leave open-circuit.
- The Hydraulic Cap block represents a hydraulic plug, that is, a hydraulic port with zero flow through it. Use this block to terminate hydraulic ports that you want to cap.
- The Rotational Free End block represents a mechanical rotational port that rotates freely, without torque. Use this block to terminate mechanical rotational ports that you want to leave unconnected.
- The Translational Free End block represents a mechanical translational port that moves freely, without force. Use this block to terminate mechanical translational ports that you want to leave unconnected.

Viewable and customizable source files for additional Foundation library blocks

Compatibility Considerations: Yes

In R2009a, many blocks in the Foundation library were implemented using the Simscape language. In 2012b, most of the remaining blocks have been converted.

You can now view the source files for most of the Foundation library blocks. When you open the block dialog box, it contains a link:

`View source for` *`BlockName`*

Click this link to open the Simscape source file for the block in the MATLAB Editor. To customize the block for your application, edit the source file and save it in a package directory. Some of the features, such as drop-down lists in block dialog boxes, are not yet available for custom blocks. For more information on packaging Simscape source files, see Simscape File Deployment.

Compatibility Considerations

The PS Math Function block now issues a simulation-time error when the input falls out of the expected domain for the particular function used. For example, if set to `sqrt`, the PS Math Function block now issues an error if it receives negative input during simulation.

New examples

Examples introduced in this version are:

- Lithium Battery Cell - One RC-Branch Equivalent Circuit
- Lithium Battery Cell - Two RC-Branch Equivalent Circuit
- Asynchronous PWM Voltage Source
- Discrete-Time PWM Voltage Source
- Turbojet Engine

Also, the existing Pneumatic Motor example has been modified to show how a pneumatic motor can be modeled using the Simscape language.

R2012a

Version: 3.7

New Features: Yes

Bug Fixes: Yes

Modeling Delays in Simscape Language

The new `delay` construct in Simscape language lets you refer to past values of expressions in the `equations` section of the Simscape file. For more information, see the delay reference page, and the new demo, Variable Transport Delay.

New Blocks for Modeling Delays

The new Delays sublibrary of the Physical Signals library contains two blocks:

- PS Constant Delay block returns the input signal delayed by a specified time, which is constant throughout the simulation. You specify the delay time as a block parameter.
- PS Variable Delay block delays the input signal by a variable time. You specify the delay time and the input history as additional inputs.

Use these blocks to model idealized transport phenomena at system level.

1D and 2D Interpolation Available in Simscape Language

The new `tablelookup` function in Simscape language lets you interpolate expressions in the `equations` section of the Simscape file. It computes an output value by interpolating the input value against a set of data points in a one-dimensional or two-dimensional table, and supports three interpolation and two extrapolation options. This functionality is similar to that of the Simulink and Simscape Lookup Table blocks. It allows you to incorporate table-driven modeling directly in your custom block, without the need of connecting an external Lookup Table block to your model. For more information, see the tablelookup reference page.

The PS Lookup Table (1D) and PS Lookup Table (2D) blocks in the Foundation library now use the `tablelookup` function in the Simscape language. The block functionality and user interface remain the same.

Input Filtering Usability Enhancements

Compatibility Considerations: Yes

In previous releases, input filtering was automatically turned on whenever you used an explicit solver in a Simscape model. To turn off input filtering when using an explicit solver, you had to supply first derivative of the input signal as an additional input signal to the Simulink-PS Converter block. For models using other types of solvers, input filtering was not available.

Now input filtering is independent of the solver used in the model. You can control whether you filter input or provide time derivatives for each input signal individually, by configuring the Simulink-PS Converter block connected to that input signal. You can:

- Set the **Filtering and derivatives** parameter to `Filter input`, and select whether you want to use the first-order or second-order filter. Input filtering makes the input signal smoother and generally improves model performance. The additional benefit is that the Simscape engine computes the time derivatives of the filtered input. The first-order filter provides one derivative, while the second-order filter provides the first and second derivatives. If you use input filtering, it is very important to select the appropriate value for the filter time constant.
- Set the **Filtering and derivatives** parameter to `Provide input derivative(s)`, and provide either just the first time derivative, or the first and the second time derivatives, through additional input ports on the Simulink-PS Converter block.

By default, input signals are used as is, without performing input filtering or otherwise providing time derivatives of the input signal. If you use an explicit solver, MathWorks recommends that you provide input derivatives by selecting one of the options listed above. If you do not provide input derivatives, and the solver you use requires them, you get an error message indicating how many input derivatives you need to provide. For more information, see Harmonizing Simulink and Simscape Solvers and the Simulink-PS Converter block reference page.

Compatibility Considerations

Input filtering is no longer automatically turned on for models using explicit solvers. Therefore if you have an existing model that uses an explicit solver,

its performance and simulation results may be different from the previous version.

Because Simscape solver no longer automatically provides the required input derivatives, you may get an error message indicating how many input derivatives you need to provide.

To preserve the old behavior, open each Simulink-PS Converter block and set the **Filtering and derivatives** parameter to `Filter input`, while keeping the value of the **Input filtering time constant** parameter unchanged. If any of the Simulink-PS Converter blocks in your model had input derivatives provided as additional signals, set the **Filtering and derivatives** parameter for such block to `Provide input derivative(s)`.

Zero Damping Allowed for Resistive Elements

In previous releases, resistive blocks in the Foundation library required positive damping coefficients. Negative damping values are nonphysical. However, zero damping values are useful for model checking and testing, For example, if you use a Rotational Damper block as part of a customized gear model, it is convenient to be able to set the **Damping coefficient** parameter to `0` temporarily, to compute undamped responses.

The following blocks now allow zero damping values, to support model testing:

- Resistor (Electrical Elements library)
- Linear Hydraulic Resistance (Hydraulic Elements library)
- Rotational Damper (Rotational Elements library)
- Translational Damper (Translational Elements library)

Changes to Simscape Demos

The following demo has been added in Version 3.7:

Demo Name	Description
Variable Transport Delay (`ssc_transport_delay`)	Provides an example of modeling a variable transport delay using Simscape language. The Transport Delay subsystem models signal propagation through media moving between the Input and the Output terminals. The media velocity may vary, thus it is specified through the block port. The distance between the terminals is constant and it is specified as a block parameter. To see the implementation details, look under mask of the Transport Delay subsystem, then right-click the Variable Transport Delay block and select **View Simscape source**.

R2011b

Version: 3.6

New Features: Yes

Bug Fixes: Yes

Simscape Language Enables User-Defined Diagnostics During Simulation

The new `assert` construct in Simscape language lets you implement run-time error messages when the custom block is used in a model. In the component file, you specify the condition to be evaluated, as well as the error message to be output if this condition is violated. When the custom block based on this file is used in a model, it will output this error if the condition is violated during simulation. For more information, see Programming Run-Time Errors and Warnings.

New Block for Modeling Discrete Delays

The Asynchronous Sample & Hold block, in the new Discrete sublibrary of the Physical Signals library, sets output Y equal to input U when the rising edge of the trigger input becomes greater than zero. Use this block, in conjunction with other physical signal blocks, to model discrete and event-based behaviors.

Specialized Simulator for Linear Systems

The new specialized simulator, implemented for linear systems, reduces number of states and accelerates simulation. The specialized linear simulator is automatically employed based on the system structure, you do not have to select or enable it explicitly.

Rebuilding of Custom Block Libraries Now Required

Due to further scalability improvements, as well as other internal enhancement to facilitate add-on product functionality, you have to rebuild your custom block libraries once you upgrade to Version 3.6 (R2011b). It is required that you rebuild your custom block libraries for use with each new version of Simscape software. For more information, see When to Rebuild the Custom Library.

To rebuild the libraries, run `ssc_build` on the component Simscape files. If you try to use the custom blocks without rebuilding the libraries, you will get an error message.

Running `ssc_clean` before `ssc_build` is strongly recommended but not required. If you run into errors after running `ssc_build`, run `ssc_clean` and then try running `ssc_build` again.

R2011a

Version: 3.5

New Features: Yes

Bug Fixes: Yes

Additional Hydraulic Sources

Two new blocks have been added to the Hydraulic Sources library:

- Hydraulic Constant Flow Rate Source block represents an ideal source of hydraulic energy that is powerful enough to maintain specified flow rate at its outlet regardless of the pressure differential across the source. The **Source flow rate** parameter specifies the flow rate through the source.
- Hydraulic Constant Pressure Source block represents an ideal source of hydraulic energy that is powerful enough to maintain the specified pressure differential between its inlet and outlet regardless of the flow rate through the source. The **Pressure** parameter specifies the pressure differential across the source.

Use these blocks for models where flow rate or pressure remain constant throughout simulation.

Improved Simulation Diagnostics

The following improvements in simulation diagnostics have been implemented, to aid debugging:

- Equation dependency diagnostics now point to the specific equations (with line number and file location info) within the component Simscape files.
- Equation dependency diagnostics have been extended to include switched-linear and nonlinear equations in the analysis.
- Equation dependency diagnostics are now triggered on nonlinear solver failures that occur after the start of simulation.
- Zero-crossing related warnings and error messages now point to the specific equations (with line number and file location info) within the component Simscape files.

Improved Scalability

Compatibility Considerations: Yes

Various scalability improvements have been implemented in this release, accelerating simulation of larger systems.

Compatibility Considerations

Due to this change, you have to rebuild your custom block libraries once you upgrade to Version 3.5 (R2011a). To rebuild the libraries, run `ssc_build` on the component Simscape files. If you try to use the custom blocks without rebuilding the libraries, you will get an error message.

Running `ssc_clean` before `ssc_build` is strongly recommended but not required. If you run into errors after running `ssc_build`, run `ssc_clean` and then try running `ssc_build` again.

Improved Algorithms for Algebraic Loop Detection and Zero-Crossing Robustness

The following simulation algorithm improvements have been implemented in this release:

- False algebraic loop detection and prevention—The improved algorithm can now recognize false algebraic loops and prevent them from affecting simulation results.
- Performance and zero-crossing robustness improvements—Zero-crossing detection algorithm has been optimized to ignore zero-crossings that do not result in model behavior changes during simulation.

Change in Evaluating Unit Expressions

Compatibility Considerations: Yes

In the Unit Manager parser, multiplication (`*`) used to mistakenly have higher precedence than division (`/`). This issue is now fixed. When evaluating unit expressions, `*` and `/` now have the same precedence and are evaluated based on left associativity.

Compatibility Considerations

Due to this change, custom units specified in the unit registry may now evaluate differently. For example, if you have added a unit `m/s*s`, in previous releases it was evaluated as `m/(s*s) = m/s^2`. It will now evaluate to `(m/s)*s = m`. You can use parentheses to preserve the old behavior.

R2010b

Version: 3.4

New Features: Yes

Bug Fixes: Yes

Simulation Data Logging Enhancements

Compatibility Considerations: Yes

The following data logging enhancements have been implemented in this version:

- `plot` and `plotxy` commands have been added. The `plot` command lets you plot logged data against time, while the `plotxy` command plots two sets of data against each other. Also, `plot` and `plotxy` methods are now available for `simscape.logging.Node` and `simscape.logging.Series` objects. For more information, see the respective reference pages.
- New configuration parameter, **Decimation**, lets you downsample logged data by skipping time steps. For more information, see Data Logging Options in the *Simscape User's Guide*.
- Simulation data is now logged according to the value specified for the **Output options** parameter in the **Data Import/Export** pane of the Configuration Parameters dialog box.

Compatibility Considerations

For models that do not use the default value of the **Output options** parameter in the **Data Import/Export** pane of the Configuration Parameters dialog box, logged simulation data may change compared to the previous release.

Zero-Crossing Handling Enhancements

The creation and detection of zero-crossing conditions in Simscape models have been improved.

New Implementation of Relational Functions Without Creating Simulink Zero-Crossing Conditions

The Simscape language now implements relational functions in a different way from how relational operators are implemented. Unlike relational operators, these relational functions now do not create Simulink zero-crossing conditions.

For more information about zero crossings in Simscape models, see Creating and Detecting Zero Crossings in Simscape Models in the Simscape User's Guide.

For more information about creating models in the Simscape language, see the Simscape Language Guide and the `equations` syntax.

Improved Zero-Crossing Detection Diagnostics

If your model generates a Simulink warning or error about zero crossings involving Simscape blocks, the warning or error message now specifies which blocks are generating this zero-crossing diagnostic message.

If you globally disable zero-crossing detection in a Simulink model containing Simscape blocks, and if you are using a variable-step solver without a local solver, you now receive a diagnostic warning or error. You can control which message that you receive in the **Simscape** pane of the model Configuration Parameters dialog box, through the **Zero-crossing control is globally disabled in Simulink** drop-down list. This option supports context-sensitive or "What's This?" help, by default accessed through right-clicking the item.

See Harmonizing Simulink and Simscape Solvers in the Simscape User's Guide.

C++ Code Generation Support

You can now generate C++ code from Simscape models. Encapsulated C++ code generation is not supported.

For more information, see Code Generation and Limitations in the Simscape User's Guide.

Sparse Solver Enhancement

The implementation and control of matrix linear algebra in the Solver Configuration block have been improved and simplified. Whether you choose the sparse solver or the full solver, your linear algebra choice is now implemented in both model simulation and code generated from the model.

For more information, see the Solver Configuration block reference page.

Component Descriptor Is No Longer Inherited from the Base Class

Compatibility Considerations: Yes

In Simscape Language, the name of the block built from a component file generally corresponds to the component file name. You can provide a more descriptive name for the block by adding a comment line immediately following the component declaration. This comment line is called the descriptor. For more information, see How to Customize the Block Name.

If you use subclassing, the subclass inherits all of the members (parameters, variables, nodes, inputs and outputs) from the base class (for more information, see Subclassing and Inheritance). In previous releases, descriptor was one of the inherited properties. That is, if you had a subclass component without a descriptor, the resulting block name was derived from the descriptor of the base class, instead of the component file name. This could create issues with library building if both the subclass component and the base class component were in the same sublibrary.

Starting with this release, descriptor is no longer inherited from the base class. In other words, if a component file does not contain a comment line immediately following the component declaration, the component name is always used for the block name.

Compatibility Considerations

For existing subclass components, if the component does not have a descriptor line, upon rebuilding the library the block name will change compared to the previous release. To preserve the old name, add a descriptor line to the subclass component file.

Documentation Enhancements

The Model Simulation chapter has been expanded and improved with revised and new sections, including:

- How Simscape Models Represent Physical Systems
- How Simscape Simulation Works
- Setting Up Solvers for Physical Models
- Customizing Solvers for Physical Models
- Real-Time Simulation
- Finding an Operating Point
- Linearizing at an Operating Point
- Finding and Using Operating Points in an Electronic Circuit

Changes to Simscape Demos

The following demo has been added in Version 3.4:

Demo Name	Description
Electrical Transformer (`ssc_transformer`)	Presents a view inside a transformer core using the electromagnetic blocks from the Magnetic block library.

R2010a

Version: 3.3

New Features: Yes

Bug Fixes: Yes

Magnetic Blocks Added to Foundation Library

Foundation library now contains magnetic domain and Magnetic block library. This library contains electromagnetic elements, such as reluctances, actuators, and electromagnetic converters, as well as magnetic sensors and sources.

Use these blocks to model magnetic circuits that can be represented by a one-dimensional flux flow, for example, solenoids and transformers.

Magnetic block models are based on the following assumptions:

- The magnetic system is assumed lossless. You can model losses in interconnecting systems instead – in electric systems using resistors and in mechanical systems using friction.
- Modeling of superconductors (with zero relative permeability) is not supported.

For more information see the block reference pages. See also Magnetic Domain for information on the magnetic domain definition. The Across variable is magnetomotive force (mmf), and the Through variable is flux. Note that these result in a pseudo-bond graph, because the product of mmf and flux is energy, not power.

Simulation Data Logging Now Available

You can now log simulation data to workspace for debugging and verification purposes. 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 workspace, you can later query, plot, and analyze it without rerunning the simulation. For more information, see Data Logging in the *Simscape User's Guide*.

Simscape Diagnostics Improvements

Simscape error messages triggered by initial dynamic state inconsistencies and by nonlinear solver convergence failures are now more detailed. These

messages report specific components of your models that may have caused the error.

Sparse Jacobian Support

You can now use a Jacobian method with an implicit Simulink solver in your Simscape models. You can choose a method yourself or allow Simulink to determine an appropriate Jacobian method for you. Depending on the sparsity pattern and number of states of your model, your simulation may be more efficient.

See Choosing a Jacobian Method for an Implicit Solver in the Simulink documentation.

Ability to Generate Simscape Language Equations from Symbolic Expressions

If you have Symbolic Math Toolbox™ software, you can use the `simscapeEquation` function to generate Simscape language equations from symbolic expressions. For more information, see Generating Simscape Equations in the Symbolic Math Toolbox documentation.

Placing Simscape Blocks in Nonvirtual Subsystems

Compatibility Considerations: Yes

Nonvirtual subsystems that support continuous states include Enabled subsystems and Atomic subsystems. These subsystems can contain Simscape blocks. However, physical connections and physical signals must not cross nonvirtual boundaries. For more information, see Restricted Simulink Tools in the *Simscape User's Guide*.

Compatibility Considerations

Simscape solver no longer permits physical connections and physical signals to cross nonvirtual subsystem boundaries, because the semantics of these types of connections are unclear. If either a physical signal or a physical connection crosses a nonvirtual boundary, the solver issues an error upon

simulation. To resolve the issue, place all blocks belonging to a given Physical Network in the same nonvirtual subsystem.

Trimming and Linearization Documentation Enhancements

The documentation on Simscape model trimming and linearization has been revised and expanded. See Finding an Operating Point and Linearizing at an Operating Point.

Changes to Simscape Demos

The following demos have been added in Version 3.3:

Demo Name	Description
Circuit Breaker (`ssc_circuitbreaker`)	Implements a simple circuit breaker model.
Solenoid with Magnetic Blocks (`ssc_solenoid_magnetic`)	Shows how to model a solenoid using the electromagnetic blocks from the new Magnetic block library.

R2009b

Version: 3.2

New Features: Yes

Bug Fixes: Yes

Pneumatic Blocks Added to Foundation Library

Compatibility Considerations: Yes

Foundation library now contains pneumatic domain and Pneumatic block library. This library contains 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.

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 are neglected.

For more information see the block reference pages, as well as Modeling Pneumatic Systems in the *Simscape User's Guide*.

Compatibility Considerations

To avoid duplicate block names in different Simscape domains and increase naming consistency across domains, the following hydraulic blocks have been renamed:

Old Block Name	New Block Name
Constant Area Orifice	Constant Area Hydraulic Orifice
Constant Volume Chamber	Constant Volume Hydraulic Chamber
Piston Chamber	Hydraulic Piston Chamber
Resistive Tube	Hydraulic Resistive Tube
Variable Area Orifice	Variable Area Hydraulic Orifice
Variable Chamber	Variable Hydraulic Chamber
Ideal Hydraulic Flow Rate Sensor	Hydraulic Flow Rate Sensor
Ideal Hydraulic Pressure Sensor	Hydraulic Pressure Sensor
Ideal Hydraulic Flow Rate Source	Hydraulic Flow Rate Source
Ideal Hydraulic Pressure Source	Hydraulic Pressure Source

Old models containing any of these blocks will be updated automatically once you open and save them.

New and Enhanced Switches

The following switching capability enhancements have been implemented in Foundation libraries:

- New Physical Signal PS Switch block has been added to the Nonlinear Operators library. It contains three physical signal input ports, a physical signal output port, and one parameter, **Threshold**. If the second input is greater than or equal to the threshold, then the output is connected to the first input. Otherwise, the output is connected to the third input. The second input never connects to the output.
- Electrical Switch block has been enhanced to use a value specified in the **Threshold** parameter (rather than zero) for opening and closing the switch.

Intermediate Terms in Simscape Language Equations

You can now introduce intermediate terms in Simscape Language equations by using the `let` and `in` keywords. This functionality helps increase the equation readability, as well as avoid duplicating information by defining an intermediate term once and then using it in multiple equations. For more information, see the Simscape Language Guide.

Local Solver Support in Physical Networks

The Solver Configuration block now lets you use sample-based local solver with a specific sample time. In sample-based simulation, all the Physical Network states, otherwise represented as continuous, become discrete states. The solver updates the states once per time step. This option is especially useful for code generation, or hardware-in-the-loop (HIL) simulations. For more information, see the Solver Configuration reference page.

Simulink Manifest Tool Support

Dependency analysis tools for Simscape files have been added in this release. They consist of the following command-line options:

- `simscape.dependency.file` — Perform dependency analysis for a single Simscape file.
- `simscape.dependency.lib` — Perform dependency analysis for a Simscape custom library.
- `simscape.dependency.model` — Perform dependency analysis on a model containing Simscape and Simulink blocks.

Manifest reports generated using Simulink Manifest Tools now also include model dependencies for the Simscape blocks. For more information, see Checking File and Model Dependencies in the *Simscape Language Guide*.

SimState Support

Simscape software now supports Simulink SimState feature, introduced in R2009a. This feature allows you to save all runtime data necessary for

restoring the simulation state of a model. For more information, see Saving and Restoring the Simulation State as the SimState in the *Simulink User's Guide*.

Note When using SimState to save and restore simulations of models involving Simscape blocks, please ensure both `'DstWorkSpace'` and `'SrcWorkSpace'` to be `'base'` by using:

```
simset('DstWorkspace', 'base', 'SrcWorkspace', 'base')
```

Model Reference Accelerator Mode Support

Simscape and its add-on products now fully support Model Reference Accelerator Mode, both for model simulation and for code generation.

Physical Port Rotation for Simscape Blocks

Compatibility Considerations: Yes

When you rotate a regular Simulink block, its ports are by default reordered after rotation, to maintain the left-right and top-down block diagram orientation convention used in control system modeling applications. This convention is not applicable to physical modeling and is potentially confusing, because it results in effectively rotating and flipping the block at the same time.

Therefore, starting with Version 3.2 (R2009b), when you rotate a Simscape block (including blocks from add-on products), its ports are not reordered. This behavior is similar to that of the masked blocks with **Port Rotation** set to `Physical`. For illustration of differences between the *default port rotation* type and the *physical port rotation* type, see Changing a Block's Orientation in the *Simulink User's Guide*.

Compatibility Considerations

This change in the behavior of the ports after block rotation may result in visually crossed connection lines in some of your existing block diagrams with

rotated blocks. The effect is purely cosmetic and has no impact on actual model connections or simulation.

Changes to Simscape Demos

The following demos have been added in Version 3.2:

Demo Name	Description
Pneumatic Actuation Circuit (`ssc_pneumatic_actuator`)	This demo shows how the Foundation Library pneumatic components can be used to model a controlled pneumatic actuator. The Directional 5-way valve, Double-acting pneumatic actuator and Pipe blocks are masked subsystems created from Foundation Library blocks.
Pneumatic Motor (`ssc_pneumatic_motor`)	This demo shows how the Rotational Pneumatic-Mechanical Converter block can be used to approximate the behavior of a pneumatic vane motor.
Pneumatic Motor and Directional Control Valve (`ssc_pneumatic_motor_and_valve`)	This model shows the pneumatic vane motor, as defined in the Pneumatic Motor demo, deployed in a typical pneumatic circuit.

R2009a

Version: 3.1

New Features: Yes

Bug Fixes: Yes

Simscape Language Source Protection

Simscape language files can be protected to enable model sharing without disclosing the component or domain source. You can then share the protected (executable) files without disclosing the file content (similar to P-code vs M-code). While Simscape source files have the extension `.ssc`, Simscape protected files have the extension `.sscp`.

Use the `ssc_protect` command to protect individual files and directories.

Use the `ssc_mirror` command to create a protected copy of a whole package, along with a custom block library built from it.

For more information, see Using Source Protection for Simscape Files.

Expanded MATLAB Support of Simscape Files

MATLAB support of Simscape files has been expanded:

- If you issue the `open` command on a Simscape file, the file will open in the MATLAB Editor. The Simscape file must be on the MATLAB path, or in a package residing in a directory on the MATLAB path. For more information on packaging Simscape files, see Organizing Your Simscape Files.

  If you issue the `open` command on a Simscape protected file (`*.sscp`), the corresponding Simscape source file (`*.ssc`) will open, provided it exists in the same directory as the Simscape protected file.
- Issuing the `help` command on a Simscape file displays the domain or component description, that is, all the comments immediately following the domain or component declaration, in the MATLAB Command Window.
- MATLAB Editor now supports syntax highlighting of Simscape files, similar to M-files. For more information, see Adjust Editor Appearance.

Viewable and Customizable Source Files for Foundation Library Blocks

Compatibility Considerations: Yes

You can now view the source files for many Foundation library blocks. When you open the block dialog box, it contains a link:

`View source for` *`BlockName`*

Click this link to open the Simscape source file for this block in the MATLAB Editor. To customize the block for your application, edit the source file and save it in a package directory. For more information, see Simscape File Deployment.

Compatibility Considerations

The block source has been optimized, with some previously defined but unused variables eliminated. Therefore, when you load an old model containing Foundation blocks, you might get warnings, for example:

```
Warning: In instantiating linked block 'model/R1' : Resistor block (mask) does not
have a parameter named 'current_Log'.
```

You can safely ignore these warnings. Once you save the model, the warnings will disappear.

Simscape Language Syntax Changes

Compatibility Considerations: Yes

The following changes have been implemented in Simscape language:

- The `throughs` keyword has been obsoleted. Use `variables(Balancing=true)` to declare Through variables in a domain.
- The `equation` keyword has been changed to `equations`.
- The name of a Simscape file must match the name of the component or domain it defines. If this is not the case, you will get an error when trying to build a library or use the block in a model.

For more information, see the Simscape Language Guide.

Compatibility Considerations

The changes are relatively minor, but may require modifying your existing Simscape files. The following table summarizes the old and new syntax.

Old Syntax	New Syntax
`throughs`	`variables(Balancing=true)`
`equation`	`equations`

Increased Efficiency of Simscape Language Equations Processing

Simscape language equations are now processed more efficiently, reducing the time required to process equations with multiple `if` statements.

New Physical Signal Blocks to Facilitate Rounding

Three new Physical Signal blocks have been added to the Nonlinear Operators library:

- PS Ceil block performs rounding of the signal toward positive infinity
- PS Floor block performs rounding of the signal toward negative infinity
- PS Fix block performs rounding of the signal toward zero

Model Reference Accelerator Mode Support

Simscape and its add-on products now support Model Reference Accelerator Mode for model simulation, but not for code generation. Model Reference Accelerator Mode for code generation is supported only by SimMechanics and SimDriveline™ software.

Changes to Simscape Demos

The following demo has been added in Version 3.1:

Demo Name	Description
Creating A New Circuit (`ssc_new_elec`)	Use this demo as a template for creating a new electrical model. Open the demo and use **File > Save As** to save it under the desired model name. Then delete the unwanted components and add new ones. This demo also opens an Electrical Starter Palette, which contains links to the most often used electrical components.

R2008b

Version: 3.0

New Features: Yes

Bug Fixes: Yes

Simscape Language

New Simscape language extends the Simscape modeling environment by enabling you to create new components that do not exist in the Foundation library or in any of the add-on products. It is a dedicated textual physical systems modeling language with the following characteristics:

- Derives from MATLAB and familiar to those who use MATLAB
- Contains additional constructs specific to physical modeling and excludes constructs that have nothing to do with physical modeling
- Incorporated into the Simscape modeling interface
- Not focused on algorithm development

The Simscape language is intended to make modeling physical systems easy and intuitive. It lets you create new physical domains and components as textual files and then use them in Simscape block diagrams to model the desired physical effects. For more information, see the Simscape Language Guide.

Explicit Solvers

It is now possible to choose any variable-step or fixed-step solver for models containing Simscape blocks. Note, however, that implicit solvers, such as ode14x, ode23t, and ode15s, are still 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.

By default, you will get a warning when using an explicit solver for a model containing Simscape blocks. For models that are not stiff, however, explicit solvers can be effective, often taking fewer timesteps than implicit solvers. Depending on the type of your model, you can configure your preferences to either turn off this warning (if your model is not stiff) or even change it into an error (to avoid inadvertent use of explicit solvers), by using the **Simscape** pane of the Configuration Parameters dialog box.

If you use an explicit solver, it requires time derivatives 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. The new **Derivatives** tab in the Simulink-PS Converter dialog box lets you turn off input filtering and instead provide the first derivative of input as an additional input signal to the Simulink-PS Converter block. For more information, see the Simulink-PS Converter block reference page.

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. This warning can also be turned off (or changed to an error) by changing the preferences on the **Simscape** pane of the Configuration Parameters dialog box.

New Ways to Model Variable Chambers

Compatibility Considerations: Yes

There are now two blocks that let you model fluid compressibility in variable chambers:

- Piston Chamber block lets you model fluid compressibility in a chamber created by the piston in a cylinder. It replaces the Variable Volume Chamber block, available in previous releases.
- Variable Chamber block lets you model fluid compressibility in variable volume chambers of any shape. The instantaneous value of the chamber volume is provided by using a physical signal port.

Compatibility Considerations

The Variable Volume Chamber block, available in previous releases, has been deprecated. It has been replaced by the Piston Chamber block in other (structural) blocks and in demos shipped with the product. If you have used the Variable Volume Chamber block in your models, it will continue to work. Going forward, however, use the Piston Chamber block to model fluid compressibility in cylinder chambers.

Model Reference Support

Simscape software now supports the Simulink model referencing functionality in Normal mode. Other Simulink models can now reference Simscape models in normal (non-code-generation) execution. Simscape models continue to be able to reference Simulink models (that do not contain Simscape blocks) in normal execution. See Limitations for more details.

More Solver Performance and Robustness Enhancements

Version 3.0 contains multiple further enhancements to simulation algorithms, resulting in improved robustness and reliability.

R2008a

Version: 2.1

New Features: Yes

Bug Fixes: Yes

Trimming Now Available for Simscape Models

Finding and managing operating points by trimming has been implemented for models that include Simscape and SimHydraulics blocks. Simulink Control Design™ product is required for using this functionality. For more information, see Finding an Operating Point in the Simscape documentation.

Thermal Unit Conversions Now Supported

You can now specify temperature for your thermal models in a variety of units, including degrees Celsius, Fahrenheit, and Rankine. The unit manager automatically handles conversions between thermal units.

Thermal units sometimes require an affine conversion, that is, a conversion that performs both multiplication and addition. In situations when you deal with a relative, rather than absolute, temperature, you need to convert using just the linear term. Thermodynamic variables in block dialogs are automatically tagged as appropriate and handled by the unit manager. However, when an input or output signal is related to thermodynamic variables and contains units of temperature, you must decide whether affine conversion needs to be applied. The Simulink-PS Converter and PS-Simulink Converter block dialogs now contain the **Apply affine conversion** checkbox. If you select it, the unit manager uses the affine conversion, otherwise it applies the default linear conversion.

For more information, see Thermal Unit Conversions in the Simscape documentation, as well as the Simulink-PS Converter and PS-Simulink Converter block reference pages.

The `pm_addunit` command has also been modified to support affine conversions. Its second argument, `conversion`, may now be either a positive real scalar or a 1x2 array. If this argument has two elements, then it is specifying an affine conversion, with the first element (a positive real number) being the linear conversion coefficient, and the second being the offset.

Enhancement to Specifying Units

Simscape block dialogs have drop-down combo boxes for units next to a parameter value. 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 units combo box of the block dialog. For more information, see How to Specify Units in Block Dialog Boxes in the Simscape documentation.

Similarly, the Simulink-PS Converter and PS-Simulink Converter block dialogs now contain a drop-down list, which is prepopulated with some common input or output units. You can either select a unit from the list or type a unit name, or a mathematical expression with unit names. Note that you must still match the unit type:

- For a PS-Simulink Converter block, these units must be commensurate with the units of the input physical signal coming into the block.
- Signal units that you specify in a Simulink-PS Converter block must match the input type expected by the Simscape block connected to it.

New Blocks

Version 2.1 contains two new blocks:

- Gyrator block in the Electrical Elements library simulates an ideal gyrator, which can be used to implement an inductor with a capacitor.
- PS Abs block in the Physical Signals library returns absolute value of input signal.

Enhancements to Simulation Algorithms

Version 2.1 contains multiple enhancements to simulation algorithms, resulting in improved robustness and reliability.

"What's This?" Context-Sensitive Help Available for Simulink Configuration Parameters Dialog

R2008a introduces "What's This?" context-sensitive help for parameters that appear in the Simulink Configuration Parameters dialog, including those on the **Simscape** pane. This feature provides quick access to a detailed description of the parameters, saving you the time it would take to find the information in the Help Browser.

To use the "What's This?" help, do the following:

1. Place your cursor over the label of a parameter.
2. Right-click. A **What's This?** context menu appears.

   For example, the following figure shows the **What's This?** context menu appearing after a right-click on the **Start time** parameter in the **Solver** pane.

3. Click **What's This?**. A context-sensitive help window appears showing a description of the parameter.

New Simscape Demo

The following demo has been added in Version 2.1:

Demo Name	Description
House Heating System (`ssc_house_heating_system`)	The demo represents a simple house heating system consisting of a heater, thermostat, and a house structure with four thermally distinguishable parts: inside air, house walls, windows, and roof. You can investigate system behavior with the heating system turned on or off, and plot the heat cost and indoor versus outdoor temperatures.

R2007b

Version: 2.0

New Features: Yes

Bug Fixes: Yes

Code Generation Now Available for Simscape Models

Code generation has been implemented for models that include Simscape and SimHydraulics blocks. For more information, see Code Generation in the Simscape documentation.

New Thermal Block Libraries

Version 2.0 contains new block libraries of fundamental thermal elements, sensors, and sources:

- Conductive Heat Transfer
- Convective Heat Transfer
- Radiative Heat Transfer
- Thermal Mass
- Thermal Reference
- Ideal Heat Flow Source
- Ideal Heat Flow Sensor
- Ideal Temperature Source
- Ideal Temperature Sensor

Additional Physical Signal Blocks

The new Physical Signal blocks introduced in Version 2.0 are listed below:

- PS Constant
- PS Math Function
- PS Max
- PS Min
- PS Sign

Improved Simulation Performance

In Version 2.0, various solver improvements have led to improved simulation performance:

- Enhanced handling of dependent dynamic states (higher-index DAEs)

  Simscape can now handle dependencies among the dynamic states as long as they are linear in the states and independent of time and inputs. This allows you, for example, to connect capacitors in parallel (even with their parasitic series resistances set to 0), inductors in series, and so on.

- Significant reduction of the number of equations, which substantially increased simulation speed

  The typical speedup of your models is between 5 and 10 times. There are some models that are below and above this range. Also, the number of states and equations changed between releases. This means that you will have to reset any calculations that relied on the states (such as initial state setting).

The changes to the simulation technology are significant. You may find that some of your models may require different or tighter tolerances to converge, while others will require no change. Refer to the troubleshooting section in the User's Guide for help in finding the cause of a problem if simulation failed.

New Simscape Demos

The following demos have been added in Version 2.0:

Demo Name	Description
DC Motor Thermal Circuit (`ssc_dc_motor_thermal_circuit`)	The demo illustrates how the thermal behavior of a motor can be simulated in lumped parameters.
Round Rod Heat Conduction (`ssc_round_rod_heat_conduction`)	The demo illustrates the usage of thermal blocks for developing a model of a long iron rod that is heated with a heat source through its left face. The right face and the outer

cylindrical surface are open to atmosphere, with a force heat convection.

R2007a

Version: 1.0

New Features: Yes

Bug Fixes: No

Product Introduction

Simscape software extends the Simulink product line with tools for modeling and simulating multidomain physical systems. It enables you to describe multidomain physical systems containing mechanical, hydraulic, and electrical components as physical networks.

Simscape key features are:

- Single modeling environment for modeling and simulating physical systems, such as mechanical, electrical, and hydraulic systems
- Foundation library of physical modeling building blocks and fundamental mathematical elements
- Connection blocks to bridge modeling domains
- Full simulation and limited editing capabilities for models built with SimMechanics, SimDriveline, or SimHydraulics blocks (no license for these products required as long as the products are installed)
- Ability to specify units of parameters and variables, with all unit conversion handled automatically

Simscape software can be used for a variety of automotive, aerospace, defense, and industrial equipment applications. Together with SimMechanics, SimDriveline, SimHydraulics, and SimPowerSystems™ (all available separately), Simscape lets you model complex interactions in electromechanical and hydromechanical systems.

Block Libraries Moved from SimHydraulics to Simscape

Compatibility Considerations: Yes

The Foundation and Utilities block libraries that used to be included in SimHydraulics (V1.0 and V1.1) are now part of Simscape product.

Compatibility Considerations

Several blocks that used to be in SimHydraulics V1.1 and are now part of Simscape software have undergone changes that have compatibility impact. These blocks are:

- Fluid Inertia
- Inertia
- Mass
- PS Integrator
- Rotational Spring
- Translational Spring

Each of these blocks has a parameter that specifies the initial condition for use in computing the block's initial state at the beginning of a simulation run. In this version, there is a difference in the way these initial conditions are computed, and as a result, the blocks work differently than they used to in the previous version. For details, see the block reference pages.