

Simscape™

Reference



MATLAB® & SIMULINK®

R2020b



How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



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

Simscape™ Reference

© COPYRIGHT 2007–2020 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

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)
September 2010	Online only	Revised for Version 3.4 (Release 2010b)
April 2011	Online only	Revised for Version 3.5 (Release 2011a)
September 2011	Online only	Revised for Version 3.6 (Release 2011b)
March 2012	Online only	Revised for Version 3.7 (Release 2012a)
September 2012	Online only	Revised for Version 3.8 (Release 2012b)
March 2013	Online only	Revised for Version 3.9 (Release 2013a)
September 2013	Online only	Revised for Version 3.10 (Release 2013b)
March 2014	Online only	Revised for Version 3.11 (Release 2014a)
October 2014	Online only	Revised for Version 3.12 (Release 2014b)
March 2015	Online only	Revised for Version 3.13 (Release 2015a)
September 2015	Online only	Revised for Version 3.14 (Release 2015b)
October 2015	Online only	Rereleased for Version 3.13.1 (Release 2015aSP1)
March 2016	Online only	Revised for Version 4.0 (Release 2016a)
September 2016	Online only	Revised for Version 4.1 (Release 2016b)
March 2017	Online only	Revised for Version 4.2 (Release 2017a)
September 2017	Online only	Revised for Version 4.3 (Release 2017b)
March 2018	Online only	Revised for Version 4.4 (Release 2018a)
September 2018	Online only	Revised for Version 4.5 (Release 2018b)
March 2019	Online only	Revised for Version 4.6 (Release 2019a)
September 2019	Online only	Revised for Version 4.7 (Release 2019b)
March 2020	Online only	Revised for Version 4.8 (Release 2020a)
September 2020	Online only	Revised for Version 5.0 (Release 2020b)

1 | **Blocks**

2 | **Functions**

3 | **Configuration Parameters**

Simscape Pane: General	3-2
Simscape Pane Overview	3-3
Editing Mode	3-3
Explicit solver used in model containing Physical Networks blocks	3-4
Zero-crossing control is globally disabled in Simulink	3-5
Normalize using nominal values	3-5
Log simulation data	3-6
Log simulation statistics	3-7
Record data in Simulation Data Inspector	3-7
Open viewer after simulation	3-8
Workspace variable name	3-8
Decimation	3-9
Limit data points	3-9
Data history (last N steps)	3-10
Enable operating point initialization	3-11
Model operating point	3-11

4 | **Model Advisor Checks**

Simscape Model Advisor Checks	4-2
Simscape Checks Overview	4-2
Modeling Physical Systems Checks Overview	4-3
Check consistency of block parameter units	4-3
Check for outdated AC source blocks	4-3
Check for dry hydraulic nodes	4-4
Simscape Checks Overview	4-5
Check Simscape Solver Configuration block settings	4-5
Check Fluid dynamic compressibility option	4-6

Electrical Checks Overview	4-7
Check Model dynamics option	4-7
Check Noise mode option	4-8
Check Parasitic ground conductance	4-9
Check Resolver parameterization option	4-10
Check Simulation mode option	4-11
Check Transmission Line blocks	4-12
Check Zero sequence	4-13
Fluids Checks Overview	4-14
Check Valve opening dynamics option	4-14
Driveline Checks Overview	4-15
Check Gear friction model option	4-15
Check Tire compliance option	4-16
Check Engine time constant option	4-17
Check Dog clutch model option	4-17
Check Losses model option	4-18
Check Model transmission lag option	4-19
Check Hard stop model option	4-20
Check and update outdated Simscape Physical Signal blocks	4-21
Check usage of Simscape event variables with unspecified priority	4-22

Bibliography

A

Blocks

2-Port Constant Volume Chamber (2P)

Chamber with two ports and fixed volume of two-phase fluid

Library: Simscape / Foundation Library / Two-Phase Fluid / Elements



Description

The 2-Port Constant Volume Chamber (2P) block models the accumulation of mass and energy in a chamber containing a fixed volume of two-phase fluid. The chamber has two inlets, labeled **A** and **B**, through which fluid can flow. The fluid volume can exchange heat with a thermal network, for example one representing the chamber surroundings, through a thermal port labeled **H**.

The mass of the fluid in the chamber varies with density, a property that in a two-phase fluid is generally a function of pressure and temperature. Fluid enters when the pressure upstream of an inlet rises above that in the chamber and exits when the pressure gradient is reversed. The effect in a model is often to smooth out sudden changes in pressure, much like an electrical capacitor does with voltage.

The flow resistance between each inlet and the interior of the chamber is assumed to be negligible. The pressure in the interior is therefore equal to that at the inlets. Similarly, the thermal resistance between the thermal port and the interior of the chamber is assumed to be negligible. The temperature in the interior is equal to that at the thermal port.

Mass Balance

Mass can enter and exit the chamber through ports **A** and **B**. The volume of the chamber is fixed but the compressibility of the fluid means that its mass can change with pressure and temperature. The rate of mass accumulation in the chamber must exactly equal the mass flow rates in through ports **A** and **B**:

$$\left[\left(\frac{\partial \rho}{\partial p} \right)_u \frac{dp}{dt} + \left(\frac{\partial \rho}{\partial u} \right)_p \frac{du}{dt} \right] V = \dot{m}_A + \dot{m}_B + \epsilon_M,$$

where the left-hand side is the rate of mass accumulation and:

- ρ is the density.
- p is the pressure.
- u is the specific internal energy.
- V is the volume.
- \dot{m} is the mass flow rate.
- ϵ_M is a correction term introduced to account for a numerical error caused by the smoothing of the partial derivatives.

Correction Term for Partial-Derivative Smoothing

The partial derivatives in the mass balance equation are computed by applying the finite-difference method to the tabulated data in the Two-Phase Fluid Properties (2P) block and interpolating the results. The partial derivatives are then smoothed at the phase-transition boundaries by means of cubic polynomial functions. These functions apply between:

- The subcooled liquid and two-phase mixture phase domains when the vapor quality is in the 0-0.1 range.
- The two-phase mixture and superheated vapor phase domains when the vapor quality is in the 0-0.9 range.

The smoothing introduces a small numerical error that the block adjusts for by adding to the mass balance the correction term ϵ_M , defined as:

$$\epsilon_M = \frac{M - V/\nu}{\tau}.$$

where:

- M is the fluid mass in the chamber.
- ν is the specific volume.
- τ is the characteristic duration of a phase-change event.

The fluid mass in the chamber is obtained from the equation:

$$\frac{dM}{dt} = \dot{m}_A + \dot{m}_B.$$

Energy Balance

Energy can enter and exit the chamber in two ways: with fluid flow through ports **A** and **B** and with heat flow through port **H**. No work is done on or by the fluid inside the chamber. The rate of energy accumulation in the internal fluid volume must then equal the sum of the energy flow rates in through ports **A**, **B**, and **H**:

$$\dot{E} = \phi_A + \phi_B + Q_H,$$

where:

- ϕ is energy flow rate.
- Q is heat flow rate.
- E is total energy.

Neglecting the kinetic energy of the fluid, the total energy in the chamber is:

$$E = Mu.$$

Momentum Balance

The pressure drop due to viscous friction between the individual ports and the interior of the chamber is assumed to be negligible. Gravity is ignored as are other body forces. The pressure in the internal fluid volume must then equal that at port **A** and that at port **B**:

$$p = p_A = p_B.$$

Assumptions

- The chamber has a fixed volume of fluid.
- The flow resistance between the inlet and the interior of the chamber is negligible.
- The thermal resistance between the thermal port and the interior of the chamber is negligible.
- The kinetic energy of the fluid in the chamber is negligible.

Ports

Conserving

A — Fluid inlet

two-phase fluid

Opening through which fluid can enter and exit the chamber.

B — Fluid inlet

two-phase fluid

Opening through which fluid can enter and exit the chamber.

H — Thermal port

thermal

Interface through which the fluid in the chamber exchanges heat with a thermal network.

Parameters

Parameters Tab

Chamber volume — Volume of fluid inside the chamber

293.15 K (default) | scalar with units of area

Volume of fluid in the chamber. This volume is constant during simulation.

Cross-sectional area at port A — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of area

Inlet area normal to the direction of flow.

Cross-sectional area at port B — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of area

Inlet area normal to the direction of flow.

Effects and Initial Conditions Tab

Initial fluid energy specification — Thermodynamic variable whose initial value to set

Temperature (default) | Vapor quality | Vapor void fraction | Specific enthalpy | Specific internal energy

Thermodynamic variable in terms of which to define the initial conditions of the component.

Initial pressure — Absolute pressure at the start of simulation

0.101325 MPa (default) | scalar with units of pressure

Pressure in the chamber at the start of simulation, specified against absolute zero.

Initial temperature — Absolute temperature at the start of simulation

293.15 K (default) | scalar with units of temperature

Temperature in the chamber at the start of simulation, specified against absolute zero.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Temperature.

Initial vapor quality — Mass fraction of vapor at the start of simulation

0.5 (default) | unitless scalar between 0 and 1

Mass fraction of vapor in the chamber at the start of simulation.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Vapor quality.

Initial vapor void fraction — Volume fraction of vapor at the start of simulation

0.5 (default) | unitless scalar between 0 and 1

Volume fraction of vapor in the chamber at the start of simulation.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Vapor void fraction.

Initial specific enthalpy — Specific enthalpy at the start of simulation

1500 kJ/kg (default) | scalar with units of energy/mass

Specific enthalpy of the fluid in the chamber at the start of simulation.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Specific enthalpy.

Initial specific internal energy — Specific internal energy at the start of simulation

1500 kJ/kg (default) | scalar with units of energy/mass

Specific internal energy of the fluid in the chamber at the start of simulation.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Specific internal energy.

Phase change time constant — Characteristic duration of a phase-change event

0.1 s (default) | scalar with units of time

Characteristic time to equilibrium of a phase-change event taking place in the chamber. Increase this parameter to slow the rate of phase change or decrease it to speed the rate.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

3-Port Constant Volume Chamber (2P) | Constant Volume Chamber (2P) | Reservoir (2P)

Introduced in R2017b

3-Port Constant Volume Chamber (2P)

Chamber with three ports and fixed volume of two-phase fluid

Library: Simscape / Foundation Library / Two-Phase Fluid / Elements



Description

The 3-Port Constant Volume Chamber (2P) block models the accumulation of mass and energy in a chamber containing a fixed volume of two-phase fluid. The chamber has three inlets, labeled **A**, **B**, and **C**, through which fluid can flow. The fluid volume can exchange heat with a thermal network, for example one representing the chamber surroundings, through a thermal port labeled **H**.

The mass of the fluid in the chamber varies with density, a property that in a two-phase fluid is generally a function of pressure and temperature. Fluid enters when the pressure upstream of an inlet rises above that in the chamber and exits when the pressure gradient is reversed. The effect in a model is often to smooth out sudden changes in pressure, much like an electrical capacitor does with voltage.

The flow resistance between each inlet and the interior of the chamber is assumed to be negligible. The pressure in the interior is therefore equal to that at the inlets. Similarly, the thermal resistance between the thermal port and the interior of the chamber is assumed to be negligible. The temperature in the interior is equal to that at the thermal port.

Mass Balance

Mass can enter and exit the chamber through ports **A**, **B**, and **C**. The volume of the chamber is fixed but the compressibility of the fluid means that its mass can change with pressure and temperature. The rate of mass accumulation in the chamber must exactly equal the mass flow rates in through ports **A**, **B**, and **C**:

$$\left[\left(\frac{\partial \rho}{\partial p} \right)_u \frac{dp}{dt} + \left(\frac{\partial \rho}{\partial u} \right)_p \frac{du}{dt} \right] V = \dot{m}_A + \dot{m}_B + \dot{m}_C + \epsilon_M,$$

where the left-hand side is the rate of mass accumulation and:

- ρ is the density.
- p is the pressure.
- u is the specific internal energy.
- V is the volume.
- \dot{m} is the mass flow rate.
- ϵ_M is a correction term introduced to account for a numerical error caused by the smoothing of the partial derivatives.

Correction Term for Partial-Derivative Smoothing

The partial derivatives in the mass balance equation are computed by applying the finite-difference method to the tabulated data in the Two-Phase Fluid Properties (2P) block and interpolating the results. The partial derivatives are then smoothed at the phase-transition boundaries by means of cubic polynomial functions. These functions apply between:

- The subcooled liquid and two-phase mixture phase domains when the vapor quality is in the 0-0.1 range.
- The two-phase mixture and superheated vapor phase domains when the vapor quality is in the 0-0.9 range.

The smoothing introduces a small numerical error that the block adjusts for by adding to the mass balance the correction term ϵ_M , defined as:

$$\epsilon_M = \frac{M - V/\nu}{\tau}.$$

where:

- M is the fluid mass in the chamber.
- ν is the specific volume.
- τ is the characteristic duration of a phase-change event.

The fluid mass in the chamber is obtained from the equation:

$$\frac{dM}{dt} = \dot{m}_A + \dot{m}_B + \dot{m}_C.$$

Energy Balance

Energy can enter and exit the chamber in two ways: with fluid flow through ports **A**, **B**, and **C**, and with heat flow through port **H**. No work is done on or by the fluid inside the chamber. The rate of energy accumulation in the internal fluid volume must then equal the sum of the energy flow rates in through ports **A**, **B**, **C**, and **H**:

$$\dot{E} = \phi_A + \phi_B + \phi_C + Q_H.$$

where:

- ϕ is the energy flow rate.
- Q is the heat flow rate.
- E is the total energy.

Neglecting the kinetic energy of the fluid, the total energy in the chamber is:

$$E = Mu.$$

Momentum Balance

The pressure drop due to viscous friction between the individual ports and the interior of the chamber is assumed to be negligible. Gravity is ignored as are other body forces. The pressure in the internal fluid volume must then equal that at port **A**, port **B**, and port **C**:

$$p = p_A = p_B = p_C.$$

Assumptions

- The chamber has a fixed volume of fluid.
- The flow resistance between the inlet and the interior of the chamber is negligible.
- The thermal resistance between the thermal port and the interior of the chamber is negligible.
- The kinetic energy of the fluid in the chamber is negligible.

Ports

Conserving

A – Fluid inlet

two-phase fluid

Opening through which fluid can enter and exit the chamber.

B – Fluid inlet

two-phase fluid

Opening through which fluid can enter and exit the chamber.

C – Fluid inlet

two-phase fluid

Opening through which fluid can enter and exit the chamber.

H – Thermal port

thermal

Interface through which the fluid in the chamber exchanges heat with a thermal network.

Parameters

Parameters Tab

Chamber volume – Volume of fluid inside the chamber

293.15 K (default) | scalar with units of area

Volume of fluid in the chamber. This volume is constant during simulation.

Cross-sectional area at port A – Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of area

Inlet area normal to the direction of flow.

Cross-sectional area at port B – Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of area

Inlet area normal to the direction of flow.

Cross-sectional area at port C – Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of area

Inlet area normal to the direction of flow.

Effects and Initial Conditions Tab**Initial fluid energy specification — Thermodynamic variable whose initial value to set**

Temperature (default) | Vapor quality | Vapor void fraction | Specific enthalpy | Specific internal energy

Thermodynamic variable in terms of which to define the initial conditions of the component.

Initial pressure — Absolute pressure at the start of simulation

0.101325 MPa (default) | scalar with units of pressure

Pressure in the chamber at the start of simulation, specified against absolute zero.

Initial temperature — Absolute temperature at the start of simulation

293.15 K (default) | scalar with units of temperature

Temperature in the chamber at the start of simulation, specified against absolute zero.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Temperature.

Initial vapor quality — Mass fraction of vapor at the start of simulation

0.5 (default) | unitless scalar between 0 and 1

Mass fraction of vapor in the chamber at the start of simulation.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Vapor quality.

Initial vapor void fraction — Volume fraction of vapor at the start of simulation

0.5 (default) | unitless scalar between 0 and 1

Volume fraction of vapor in the chamber at the start of simulation.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Vapor void fraction.

Initial specific enthalpy — Specific enthalpy at the start of simulation

1500 kJ/kg (default) | scalar with units of energy/mass

Specific enthalpy of the fluid in the chamber at the start of simulation.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Specific enthalpy.

Initial specific internal energy — Specific internal energy at the start of simulation

1500 kJ/kg (default) | scalar with units of energy/mass

Specific internal energy of the fluid in the chamber at the start of simulation.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to **Specific internal energy**.

Phase change time constant — Characteristic duration of a phase-change event

0.1 s (default) | scalar with units of time

Characteristic time to equilibrium of a phase-change event taking place in the chamber. Increase this parameter to slow the rate of phase change or decrease it to speed the rate.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

2-Port Constant Volume Chamber (2P) | Constant Volume Chamber (2P) | Reservoir (2P)

Introduced in R2017b

Absolute Reference (2P)

Reference point at zero absolute pressure and specific internal energy



Library

Two-Phase Fluid/Elements

Description

The Absolute Reference (2P) block represents a reference point at zero absolute pressure and specific internal energy. Use with the Pressure & Internal Energy Sensor (2P) block to measure the absolute pressure and specific internal energy at a two-phase fluid node.

Ports

The block has a two-phase fluid conserving port. This port identifies the two-phase fluid node set to zero pressure and specific internal energy.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Pressure & Internal Energy Sensor (2P) | Reservoir (2P)

Introduced in R2015b

Absolute Reference (G)

Reference point at zero absolute temperature and pressure

Library: Simscape / Foundation Library / Gas / Elements



Description

The Absolute Reference (G) block represents an absolute reference for the pressure and temperature in a gas network. At port **A**, the pressure and temperature are both equal to zero.

Unlike other domains, where each topologically distinct circuit within a domain must contain at least one reference block, gas networks do not have this requirement. For more information, see “Reference Node and Grounding Rules”.

The purpose of the Absolute Reference (G) block is to provide a reference for the Pressure & Temperature Sensor (G). To measure the absolute pressure and absolute temperature of a node, connect that node to the **A** port of a Pressure & Temperature Sensor (G) block. Connect the **B** port of the Pressure & Temperature Sensor (G) block to an Absolute Reference (G) block.

If you use the Absolute Reference (G) block elsewhere in a gas network, it triggers a simulation assertion because gas pressure and temperature cannot be at absolute zero.

Ports

Conserving

A — Pressure and temperature are zero

gas

Gas conserving port where the pressure and temperature are both equal to zero.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Pressure & Temperature Sensor (G)

Topics

“Modeling Gas Systems”

Introduced in R2016b

Absolute Reference (MA)

Reference point at zero absolute temperature and pressure

Library: Simscape / Foundation Library / Moist Air / Elements



Description

The Absolute Reference (MA) block represents an absolute reference for the pressure and temperature in a moist air network. At port **A**, the pressure and temperature are equal to absolute zero. Specific humidity and trace gas mass fraction are also equal to zero.

Unlike other domains, where each topologically distinct circuit within a domain must contain at least one reference block, moist air networks do not have this requirement. For more information, see “Reference Node and Grounding Rules”.

The purpose of the Absolute Reference (MA) block is to provide a reference for the Pressure & Temperature Sensor (MA). For example, to measure the absolute pressure and absolute temperature of a node, connect that node to the **A** port of a Pressure & Temperature Sensor (MA) block. Connect the **B** port of the Pressure & Temperature Sensor (MA) block to an Absolute Reference (MA) block.

If you use the Absolute Reference (MA) block elsewhere in a moist air network, it triggers a simulation assertion because air mixture pressure and temperature cannot be at absolute zero.

Ports

Conserving

A — Pressure, temperature, specific humidity, and trace gas mass fraction are zero
moist air

Moist air conserving port where the pressure and temperature are equal to absolute zero, and specific humidity and trace gas mass fraction are equal to zero.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Pressure & Temperature Sensor (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Absolute Reference (TL)

Reference point at zero absolute temperature and pressure

Library: Simscape / Foundation Library / Thermal Liquid / Elements



Description

The Absolute Reference (TL) block represents an absolute reference for the pressure and temperature in thermal liquid systems. At port **A**, the pressure and temperature are both equal to zero.

Ports

Conserving

A — Pressure and temperature are zero

thermal liquid

Thermal liquid conserving port where the pressure and temperature are both equal to zero.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Reservoir (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2013b

AC Current Source

Ideal sinusoidal current source



Library

Electrical Sources

Description

The AC Current Source block represents an ideal current source that maintains sinusoidal current through it, independent of the voltage across its terminals.

The output current is defined by the following equation:

$$I = I_0 \cdot \sin(2\pi \cdot f \cdot t + \varphi)$$

where

I	Current
I_0	Peak amplitude
f	Frequency
φ	Phase shift
t	Time

The positive direction of the current flow is indicated by the arrow.

Note For Release 2012b and earlier, the unit definition for Hz was rev/s, whereas in R2013a it was changed to be 1/s, in compliance with the SI unit system. For this block it means that you must specify frequency in units of Hz or directly convertible to Hz, such as 1/s, kHz, MHz and GHz. In 2012b and earlier you could also specify frequency in angular units (such as rad/s or rpm), but this is no longer possible because the internal equation of the block now uses the 2π conversion factor to account for the 1/s unit definition. If you use this block in a model created prior to R2013a, update it by using the Upgrade Advisor. For more information, see the R2013a Release Notes.

Parameters

Peak amplitude

Peak current amplitude. The default value is 1 A.

Phase shift

Phase shift in angular units. The default value is 0.

Frequency

Current frequency, specified in Hz or units directly convertible to Hz (where Hz is defined as 1/s). For example, kHz and MHz are valid units, but rad/s is not. The default value is 60 Hz.

Ports

The block has two electrical conserving ports associated with its terminals.

See Also

AC Voltage Source

Introduced in R2007a

AC Voltage Source

Ideal constant voltage source



Library

Electrical Sources

Description

The AC Voltage Source block represents an ideal voltage source that maintains sinusoidal voltage across its output terminals, independent of the current flowing through the source.

The output voltage is defined by the following equation:

$$V = V_0 \cdot \sin(2\pi \cdot f \cdot t + \varphi)$$

where

V	Voltage
V_0	Peak amplitude
f	Frequency
φ	Phase shift
t	Time

Connections + and - are conserving electrical ports corresponding to the positive and negative terminals of the voltage source, respectively. The current is positive if it flows from positive to negative, and the voltage across the source is equal to the difference between the voltage at the positive and the negative terminal, $V(+)$ - $V(-)$.

Note For Release 2012b and earlier, the unit definition for Hz was rev/s, whereas in R2013a it was changed to be 1/s, in compliance with the SI unit system. For this block it means that you must specify frequency in units of Hz or directly convertible to Hz, such as 1/s, kHz, MHz and GHz. In 2012b and earlier you could also specify frequency in angular units (such as rad/s or rpm), but this is no longer possible because the internal equation of the block now uses the 2π conversion factor to account for the 1/s unit definition. If you use this block in a model created prior to R2013a, update it by using the Upgrade Advisor. For more information, see the R2013a Release Notes.

Parameters

Peak amplitude

Peak voltage amplitude. The default value is 1 V.

Phase shift

Phase shift in angular units. The default value is 0.

Frequency

Voltage frequency, specified in Hz or units directly convertible to Hz (where Hz is defined as 1/s). For example, kHz and MHz are valid units, but rad/s is not. The default value is 60 Hz.

Ports

The block has the following ports:

+

Electrical conserving port associated with the source positive terminal.

–

Electrical conserving port associated with the source negative terminal.

See Also

AC Current Source

Introduced in R2007a

Adiabatic Cup

Thermal element with no thermal mass and perfect insulation



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Adiabatic Cup block 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.

Ports

The block has one thermal conserving port.

See Also

Cap (TL) | Hydraulic Cap | Open Circuit | Perfect Insulator | Rotational Free End | Translational Free End

Introduced in R2009b

Cap (2P)

Perfectly insulated stop to fluid flow

Library: Simscape / Foundation Library / Two-Phase Fluid / Elements



Description

The Cap (2P) block represents a perfectly insulated terminus for a two-phase fluid branch. There is no fluid flow or heat transfer through the cap.

You can use this block to terminate two-phase fluid ports on other blocks that you want to cap. This is not necessary because if you leave a conserving port unconnected, the physical network sets all the Through variables at that port to 0. However, terminator blocks improve diagram readability.

You can also use this block to set the initial pressure and specific internal energy at a node.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Gas Volume”.

Ports

Conserving

A — Mass flow rate and energy flow rate are zero

two-phase fluid

Two-phase fluid conserving port where the mass flow rate and energy flow rate are both equal to zero.

Compatibility Considerations

Capping unconnected ports is no longer required

Behavior changed in R2019b

This block is no longer required. Starting in R2019b, the restriction that disallowed unconnected conserving ports in Simscape models has been lifted. Now, if you leave a conserving port unconnected, the physical network sets all the Through variables at this port to 0. However, you can still use terminator blocks to improve diagram readability.

There are no plans to remove the terminator blocks. All existing models that use these blocks work the same as in previous releases.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Cap (MA) | Cap (G) | Cap (TL) | Hydraulic Cap | Open Circuit | Perfect Insulator | Rotational Free End
| Translational Free End

Introduced in R2015b

Cap (G)

Gas port terminator with zero flow

Library: Simscape / Foundation Library / Gas / Elements



Description

The Cap (G) block represents a perfectly insulated terminus for a gas network branch. There is no mass or energy flow through the cap.

You can use this block to terminate gas ports on other blocks that you want to cap. This is not necessary because if you leave a conserving port unconnected, the physical network sets all the Through variables at that port to 0. However, terminator blocks improve diagram readability.

You can also use this block to set the initial pressure and temperature at a node.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Gas Volume”.

Ports

Conserving

A — Mass flow rate and energy flow rate are zero

gas

Gas conserving port where the mass flow rate and energy flow rate are both equal to zero.

Compatibility Considerations

Capping unconnected ports is no longer required

Behavior changed in R2019b

This block is no longer required. Starting in R2019b, the restriction that disallowed unconnected conserving ports in Simscape models has been lifted. Now, if you leave a conserving port unconnected, the physical network sets all the Through variables at this port to 0. However, you can still use terminator blocks to improve diagram readability.

There are no plans to remove the terminator blocks. All existing models that use these blocks work the same as in previous releases.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Cap (MA) | Cap (2P) | Cap (TL) | Hydraulic Cap | Open Circuit | Perfect Insulator | Rotational Free End | Translational Free End

Topics

“Modeling Gas Systems”

Introduced in R2016b

Cap (MA)

Moist air port terminator with zero flow

Library: Simscape / Foundation Library / Moist Air / Elements



Description

The Cap (MA) block represents a perfectly insulated terminus for a moist air network branch. There is no mass or energy flow through the cap.

You can use this block to terminate moist air ports on other blocks that you want to cap. This is not necessary because if you leave a conserving port unconnected, the physical network sets all the Through variables at that port to 0. However, terminator blocks improve diagram readability.

You can also use this block to set the initial variables, such as pressure or temperature, at a node prior to simulation.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Moist Air Volume”.

Ports

Conserving

A — Mass flow rate and energy flow rate are zero

moist air

Moist air conserving port where the mass flow rate and energy flow rate are both equal to zero.

Compatibility Considerations

Capping unconnected ports is no longer required

Behavior changed in R2019b

This block is no longer required. Starting in R2019b, the restriction that disallowed unconnected conserving ports in Simscape models has been lifted. Now, if you leave a conserving port unconnected, the physical network sets all the Through variables at this port to 0. However, you can still use terminator blocks to improve diagram readability.

There are no plans to remove the terminator blocks. All existing models that use these blocks work the same as in previous releases.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Cap (2P) | Cap (G) | Cap (TL) | Hydraulic Cap | Open Circuit | Perfect Insulator | Rotational Free End | Translational Free End

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Cap (TL)

Perfectly insulated stop to fluid flow

Library: Simscape / Foundation Library / Thermal Liquid / Elements



Description

The Cap (TL) block represents a terminus to a thermal liquid branch. The stop is perfectly insulated, preventing heat transfer with its surroundings.

You can use this block to terminate thermal liquid ports on other blocks that you want to cap. This is not necessary because if you leave a conserving port unconnected, the physical network sets all the Through variables at that port to 0. However, terminator blocks improve diagram readability.

You can also use this block to set the initial pressure and temperature at a node.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Gas Volume”.

Ports

Conserving

A — Mass flow rate and energy flow rate are zero

thermal liquid

Thermal liquid conserving port where the mass flow rate and energy flow rate are both equal to zero.

Compatibility Considerations

Capping unconnected ports is no longer required

Behavior changed in R2019b

This block is no longer required. Starting in R2019b, the restriction that disallowed unconnected conserving ports in Simscape models has been lifted. Now, if you leave a conserving port unconnected, the physical network sets all the Through variables at this port to 0. However, you can still use terminator blocks to improve diagram readability.

There are no plans to remove the terminator blocks. All existing models that use these blocks work the same as in previous releases.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Cap (MA) | Cap (2P) | Cap (G) | Hydraulic Cap | Open Circuit | Perfect Insulator | Rotational Free End
| Translational Free End

Introduced in R2013b

Capacitor

Linear capacitor in electrical systems



Library

Electrical Elements

Description

The Capacitor block models a linear capacitor, described with the following equation:

$$I = C \frac{dV}{dt}$$

where

I	Current
V	Voltage
C	Capacitance
t	Time

The **Series resistance** and **Parallel conductance** parameters represent small parasitic effects. The parallel conductance directly across the capacitor can be used to model dielectric losses, or equivalently leakage current per volt. The series resistance can be used to represent component effective series resistance (ESR) or connection resistance. Simulation of some circuits may require the presence of the small series resistance. For more information, see “Modeling Best Practices”.

Connections + and - are conserving electrical ports corresponding to the positive and negative terminals of the capacitor, respectively. The current is positive if it flows from positive to negative, and the voltage across the capacitor is equal to the difference between the voltage at the positive and the negative terminal, $V(+)$ - $V(-)$.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Capacitance

Capacitance, in farads. The default value is 1 μ F.

Series resistance

Represents small parasitic effects. The series resistance can be used to represent component internal resistance. Simulation of some circuits may require the presence of the small series resistance. The default value is 1 $\mu\Omega$.

Parallel conductance

Represents small parasitic effects. The parallel conductance directly across the capacitor can be used to model leakage current per volt. The default value is 0.

Ports

The block has the following ports:

+

Electrical conserving port associated with the capacitor positive terminal.

-

Electrical conserving port associated with the capacitor negative terminal.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

Introduced in R2007a

Conductive Heat Transfer

Heat transfer by conduction



Library

Thermal Elements

Description

The Conductive Heat Transfer block represents a heat transfer by conduction between two layers of the same material. The transfer is governed by the Fourier law and is described with the following equation:

$$Q = k \cdot \frac{A}{D}(T_A - T_B)$$

where

Q	Heat flow
k	Material thermal conductivity
A	Area normal to the heat flow direction
D	Distance between layers (thickness of material)
T_A, T_B	Temperatures of the layers

Connections A and B are thermal conserving ports associated with material layers. The block positive direction is from port A to port B. This means that the heat flow is positive if it flows from A to B.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Area

Area of heat transfer, normal to the heat flow direction. The default value is 0.0001 m^2 .

Thickness

Thickness of material, that is, distance between layers. The default value is 0.1 m .

Thermal conductivity

Thermal conductivity of the material. The default value is 401 W/m/K .

Ports

The block has the following ports:

A

Thermal conserving port associated with layer A.

B

Thermal conserving port associated with layer B.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Convective Heat Transfer | Radiative Heat Transfer

Introduced in R2007b

Connection Label

Virtual connection between conserving ports

Library: Simscape / Utilities



Description

The Connection Label block lets you specify virtual connections between Simscape conserving ports, similar to the Goto and From blocks in Simulink® diagrams. These connection lines are nondirectional, which means that all sides of the virtual connection use the same type of block.

You can also use the Connection Label block to specify virtual connections between other types of nondirectional ports, such as Simscape Bus ports or Simscape Multibody™ frame, geometry, and belt-cable ports.

Virtual connectivity is established by the label name. If two or more Connection Label blocks in a model subsystem have the same label, then the conserving ports of other blocks connected to them behave as though they were physically connected. Virtual connections cannot cross subsystem boundaries.

The label name is displayed next to the block icon on the model canvas. You specify the label name by using the **Label** parameter.

Use the Connection Label block to reduce diagram clutter by breaking off tangled connection lines.

Ports

Conserving

Port_1 — Connection port

untyped conserving port

Conserving connection port. By default, this port is untyped.

You define the type of the port by connecting it to a conserving port of another block in the subsystem, or to a Simscape Bus port. Once you establish a connection, other Connection Label blocks that are in the same subsystem and have the same label can be connected only to the same type of port.

Parameters

Label — Name of connection label

Label1 (default) | character vector | string

Label name, which establishes the virtual connection. If two or more Connection Label blocks in the same subsystem of a model have the same label, then the conserving ports of the blocks connected to them behave as though they were physically connected.

The value of this parameter appears as a label next to the block icon on the model canvas.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Connection Port | From | Goto

Introduced in R2019b

Connection Port

Physical Modeling connector port for subsystem



Library

Utilities

Description

The Connection Port block transfers a physical connection or signal across subsystem boundaries. Physical connections include Simscape conserving and physical signal connections, Simscape Multibody connections, and Vehicle Dynamics Blockset™ two-way connection ports, among others. This block is similar in function to the Inport and Outport blocks in the Simulink library.

A subsystem needs a Connection Port block for each physical connection line that crosses its boundary. You can manually place a Connection Port block inside a subsystem, or Simulink can automatically insert a Connection Port block when you create a subsystem within an existing network.

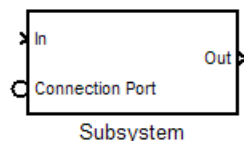
Port Appearance on Subsystem Block

The Connection Port block adds a port to the parent Subsystem block. The port type depends on the connection or signal it transfers. The port appearance on the Subsystem block matches the port to which the Connection Port block connects inside the subsystem. For example, if the port transfers a Simscape conserving connection, then it appears on the Subsystem block as a Simscape Conserving port.

Port Location and Orientation on Subsystem Block

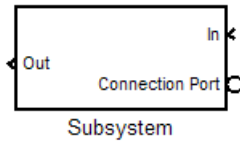
The orientation of the parent subsystem block and your choice of port location determine the Connection Port block port location on the parent subsystem block.

- A subsystem is in its fundamental orientation when its Simulink signal inports appear on its left side and its Simulink signal outports appear on its right side.



When a subsystem is oriented in this way, the actual port location on the subsystem block respects your choice of port location (left or right) for the connector port.

- A subsystem orientation is reversed, with left and right interchanged, when its Simulink signal inports occur on its right side and its Simulink signal outports occur on its left side.



When a subsystem is oriented in this way, the actual port location on the subsystem block reverses your choice of port location. If you choose left, the port appears on the right side. If you choose right, the port appears on the left side.

Parameters

Port number

Labels the subsystem connector port that this block creates. Each connector port requires a unique number as a label. The default value for the first port is 1.

Port location on parent subsystem

Choose here which side of the parent subsystem block the port is located. The choices are **Left** or **Right**. The default choice is **Left**.

See "Port Location and Orientation on Subsystem Block" on page 1-36.

See Also

In the Simulink documentation, see "Create Subsystems".

Introduced in R2007a

Constant Area Hydraulic Orifice

Hydraulic orifice with constant cross-sectional area



Library

Hydraulic Elements

Description

The Constant Area Hydraulic Orifice block models a sharp-edged constant-area orifice. The flow rate through the orifice is proportional to the pressure differential across the orifice, and is determined according to the following equations:

$$q = C_D \cdot A \sqrt{\frac{2}{\rho}} \cdot \frac{p}{(p^2 + p_{cr}^2)^{1/4}}$$

$$p = p_A - p_B$$

where

q	Flow rate
p	Pressure differential
p_A, p_B	Gauge pressures at the block terminals
C_D	Flow discharge coefficient
A	Orifice passage area
ρ	Fluid density
p_{cr}	Minimum pressure for turbulent flow, when the block transitions from laminar to turbulent regime

The minimum pressure for turbulent flow, p_{cr} , is calculated according to the laminar transition specification method:

- By pressure ratio — The transition from laminar to turbulent regime is defined by the following equations:

$$p_{cr} = (p_{avg} + p_{atm})(1 - B_{lam})$$

$$p_{avg} = (p_A + p_B)/2$$

where

p_{avg}	Average pressure between the block terminals
-----------	--

p_{atm}	Atmospheric pressure, 101325 Pa
B_{lam}	Pressure ratio at the transition between laminar and turbulent regimes (Laminar flow pressure ratio parameter value)

- By Reynolds number — The transition from laminar to turbulent regime is defined by the following equations:

$$p_{cr} = \frac{\rho}{2} \left(\frac{Re_{cr} \cdot \nu}{C_D \cdot D_H} \right)^2$$

$$D_H = \sqrt{\frac{4A}{\pi}}$$

where

D_H	Orifice hydraulic diameter
ν	Fluid kinematic viscosity
Re_{cr}	Critical Reynolds number (Critical Reynolds number parameter value)

The block positive direction is from port A to port B. This means that the flow rate is positive if it flows from A to B, and the pressure differential is determined as $p = p_A - p_B$.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Basic Assumptions and Limitations

- Fluid inertia is not taken into account.

Parameters

Orifice area

Orifice passage area. The default value is $1e-4 \text{ m}^2$.

Flow discharge coefficient

Semi-empirical parameter for orifice capacity characterization. Its value depends on the geometrical properties of the orifice, and usually is provided in textbooks or manufacturer data sheets. The default value is 0.7 .

Laminar transition specification

Select how the block transitions between the laminar and turbulent regimes:

- **Pressure ratio** — The transition from laminar to turbulent regime is smooth and depends on the value of the **Laminar flow pressure ratio** parameter. This method provides better simulation robustness.
- **Reynolds number** — The transition from laminar to turbulent regime is assumed to take place when the Reynolds number reaches the value specified by the **Critical Reynolds number** parameter.

Laminar flow pressure ratio

Pressure ratio at which the flow transitions between laminar and turbulent regimes. The default value is 0.999. This parameter is visible only if the **Laminar transition specification** parameter is set to Pressure ratio.

Critical Reynolds number

The maximum Reynolds number for laminar flow. The value of the parameter depends on the orifice geometrical profile. You can find recommendations on the parameter value in hydraulics textbooks. The default value is 12, which corresponds to a round orifice in thin material with sharp edges. This parameter is visible only if the **Laminar transition specification** parameter is set to Reynolds number.

Global Parameters

Parameters determined by the type of working fluid:

- **Fluid density**
- **Fluid kinematic viscosity**

Use the Hydraulic Fluid block or the Custom Hydraulic Fluid block to specify the fluid properties.

Ports

The block has the following ports:

A

Hydraulic conserving port associated with the orifice inlet.

B

Hydraulic conserving port associated with the orifice outlet.

References

[1] Meritt, H.E. *Hydraulic Control Systems*. New York: John Wiley & Sons, 1967.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Variable Area Hydraulic Orifice

Introduced in R2009b

Constant Area Pneumatic Orifice

Sharp-edged orifice in pneumatic systems



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Constant Area Pneumatic Orifice block models the flow rate of an ideal gas through a sharp-edged orifice.

The flow rate through the orifice is proportional to the orifice area and the pressure differential across the orifice.

$$G = C_d \cdot A \cdot p_i \sqrt{\frac{2\gamma}{\gamma-1} \cdot \frac{1}{RT_i} \left[\left(\frac{p_o}{p_i} \right)^{\frac{2}{\gamma}} - \left(\frac{p_o}{p_i} \right)^{\frac{\gamma+1}{\gamma}} \right]}$$

where

G	Mass flow rate
C_d	Discharge coefficient, to account for effective loss of area due to orifice shape
A	Orifice cross-sectional area
p_i, p_o	Absolute pressures at the orifice inlet and outlet, respectively. The inlet and outlet change depending on flow direction. For positive flow ($G > 0$), $p_i = p_A$, otherwise $p_i = p_B$.
γ	The ratio of specific heats at constant pressure and constant volume, c_p / c_v
R	Specific gas constant
T	Absolute gas temperature

The choked flow occurs at the critical pressure ratio defined by

$$\beta_{cr} = \frac{p_o}{p_i} = \left(\frac{2}{\gamma+1} \right)^{\frac{\gamma}{\gamma-1}}$$

after which the flow rate depends on the inlet pressure only and is computed with the expression

$$G = C_d \cdot A \cdot p_i \sqrt{\frac{\gamma}{RT_i} \cdot \beta_{cr} \frac{\gamma+1}{\gamma}}$$

The square root relationship has infinite gradient at zero flow, which can present numerical solver difficulties. Therefore, for very small pressure differences, defined by $p_o / p_i > 0.999$, the flow equation is replaced by a linear flow-pressure relationship

$$G = k C_d \cdot A \cdot T_i^{-0.5} (p_i - p_o)$$

where k is a constant such that the flow predicted for p_o / p_i is the same as that predicted by the original flow equation for $p_o / p_i = 0.999$.

The heat flow out of the orifice is assumed equal to the heat flow into the orifice, based on the following considerations:

- The orifice is square-edged or sharp-edged, and as such is characterized by an abrupt change of the downstream area. This means that practically all the dynamic pressure is lost in the expansion.
- The lost energy appears in the form of internal energy that rises the output temperature and makes it very close to the inlet temperature.

Therefore, $q_i = q_o$, where q_i and q_o are the input and output heat flows, respectively.

The block positive direction is from port A to port B. This means that the flow rate is positive if it flows from A to B.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Basic Assumptions and Limitations

- The gas is ideal.
- Specific heats at constant pressure and constant volume, c_p and c_v , are constant.
- The process is adiabatic, that is, there is no heat transfer with the environment.
- Gravitational effects can be neglected.
- The orifice adds no net heat to the flow.

Parameters

Discharge coefficient, C_d

Semi-empirical parameter for orifice capacity characterization. Its value depends on the geometrical properties of the orifice, and usually is provided in textbooks or manufacturer data sheets. The default value is 0.82.

Orifice area

Specify the orifice cross-sectional area. The default value is $1e-5 \text{ m}^2$.

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the orifice inlet for positive flow.

B

Pneumatic conserving port associated with the orifice outlet for positive flow.

See Also

Constant Area Pneumatic Orifice (ISO 6358) | Variable Area Pneumatic Orifice

Introduced in R2009b

Constant Area Pneumatic Orifice (ISO 6358)

Fixed-area pneumatic orifice complying with ISO 6358 standard



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Constant Area Pneumatic Orifice (ISO 6358) block models the flow rate of an ideal gas through a fixed-area sharp-edged orifice. The model conforms to the ISO 6358 standard and is based on the following flow equations, originally proposed by Sanville [1 on page 1-47]:

$$G = \begin{cases} k_1 \cdot p_i \left(1 - \frac{p_o}{p_i}\right) \sqrt{\frac{T_{ref}}{T_i}} \cdot \text{sign}(p_i - p_o) & \text{if } \frac{p_o}{p_i} > \beta_{lam} \text{ (laminar)} \\ p_i \cdot C \cdot \rho_{ref} \sqrt{\frac{T_{ref}}{T_i}} \cdot \sqrt{1 - \left(\frac{p_o}{p_i} - b\right)^2} & \text{if } \beta_{lam} > \frac{p_o}{p_i} > b \text{ (subsonic)} \\ p_i \cdot C \cdot \rho_{ref} \sqrt{\frac{T_{ref}}{T_i}} & \text{if } \frac{p_o}{p_i} \leq b \text{ (choked)} \end{cases}$$

$$k_1 = \frac{1}{1 - \beta_{lam}} \cdot C \cdot \rho_{ref} \sqrt{1 - \left(\frac{\beta_{lam} - b}{1 - b}\right)^2}$$

where

G	Mass flow rate
β_{lam}	Pressure ratio at laminar flow, a value between 0.999 and 0.995
b	Critical pressure ratio, that is, the ratio between the outlet pressure p_o and inlet pressure p_i at which the gas velocity achieves sonic speed

C	Sonic conductance of the component, that is, the ratio between the mass flow rate and the product of inlet pressure p_i and the mass density at standard conditions when the flow is choked
ρ_{ref}	Gas density at which the sonic conductance was measured (1.185 kg/m ³ for air)
p_i, p_o	Absolute pressures at the orifice inlet and outlet, respectively. The inlet and outlet change depending on flow direction. For positive flow ($G > 0$), $p_i = p_A$, otherwise $p_i = p_B$.
T_i, T_o	Absolute gas temperatures at the orifice inlet and outlet, respectively
T_{ref}	Gas temperature at which the sonic conductance was measured ($T_{ref} = 293.15$ K)

The equation itself, parameters b and C , and the heuristic on how to measure these parameters experimentally form the basis for the standard ISO 6358 (1989). The values of the critical pressure ratio b and the sonic conductance C depend on a particular design of a component. Typically, they are determined experimentally and are sometimes given on a manufacturer data sheet.

The block can also be parameterized in terms of orifice effective area or flow coefficient, instead of sonic conductance. When doing so, block parameters are converted into an equivalent value for sonic conductance. When specifying effective area, the following formula proposed by Gidlund and detailed in [2 on page 1-47] is used:

$$C = 0.128 d^2$$

where

C	Sonic conductance in dm ³ /(s*bar)
d	Inner diameter of restriction in mm

The effective area (whether specified directly, or calculated when the orifice is parameterized in terms of C_v or K_v , as described below) is used to determine the inner diameter d in the Gidlund formula, assuming a circular cross section.

Gidlund also gives an approximate formula for the critical pressure ratio in terms of the pneumatic line diameter D ,

$$b = 0.41 + 0.272 d / D$$

This equation is not used by the block and you must specify the critical pressure ratio directly.

If the orifice is parameterized in terms of the C_v [2 on page 1-47] coefficient, then the C_v coefficient is turned into an equivalent effective orifice area for use in the Gidlund formula:

$$A = 1.6986e - 5 C_v$$

By definition, an opening or restriction has a C_v coefficient of 1 if it passes 1 gpm (gallon per minute) of water at pressure drop of 1 psi.

If the orifice is parameterized in terms of the K_v [2 on page 1-47] coefficient, then the K_v coefficient is turned into an equivalent effective orifice area for use in the Gidlund formula:

$$A = 1.1785e - 6 C_v$$

K_v is the SI counterpart of C_v . An opening or restriction has a K_v coefficient of 1 if it passes 1 lpm (liter per minute) of water at pressure drop of 1 bar.

The heat flow out of the orifice is assumed equal to the heat flow into the orifice, based on the following considerations:

- The orifice is square-edged or sharp-edged, and as such is characterized by an abrupt change of the downstream area. This means that practically all the dynamic pressure is lost in the expansion.
- The lost energy appears in the form of internal energy that rises the output temperature and makes it very close to the inlet temperature.

Therefore, $q_i = q_o$, where q_i and q_o are the input and output heat flows, respectively.

The block positive direction is from port A to port B. This means that the flow rate is positive if it flows from A to B.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Basic Assumptions and Limitations

- The gas is ideal.
- Specific heats at constant pressure and constant volume, c_p and c_v , are constant.
- The process is adiabatic, that is, there is no heat transfer with the environment.
- Gravitational effects can be neglected.
- The orifice adds no net heat to the flow.

Parameters

Orifice is specified with

Select one of the following model parameterization methods:

- **Sonic conductance** — Provide value for the sonic conductance of the orifice. The values of the sonic conductance and the critical pressure ratio form the basis for the ISO 6358 compliant flow equations for the orifice. This is the default method.
- **Effective area** — Provide value for the orifice effective area. This value is internally converted by the block into an equivalent value for sonic conductance.
- **Cv coefficient (USCU)** — Provide value for the flow coefficient specified in US units. This value is internally converted by the block into an equivalent value for the orifice effective area.
- **Kv coefficient (SI)** — Provide value for the flow coefficient specified in SI units. This value is internally converted by the block into an equivalent value for the orifice effective area.

Sonic conductance

Specify the sonic conductance of the orifice, that is, the ratio between the mass flow rate and the product of upstream pressure and the mass density at standard conditions when the flow is choked. This value depends on the geometrical properties of the orifice, and usually is provided in textbooks or manufacturer data sheets. The default value is 1.6 l/s/bar. This parameter appears in the dialog box if **Orifice is specified with** parameter is set to **Sonic conductance**.

Effective area

Specify the orifice cross-sectional area. The default value is $1e-5 \text{ m}^2$. This parameter appears in the dialog box if **Orifice is specified with** parameter is set to Effective area.

Cv coefficient

Specify the value for the flow coefficient in US units. The default value is 0.6. This parameter appears in the dialog box if **Orifice is specified with** parameter is set to Cv coefficient (USCU).

Kv coefficient

Specify the value for the flow coefficient in SI units. The default value is 8.5. This parameter appears in the dialog box if **Orifice is specified with** parameter is set to Kv coefficient (SI).

Critical pressure ratio

Specify the critical pressure ratio, that is, the ratio between the downstream pressure and the upstream pressure at which the gas velocity achieves sonic speed. The default value is 0.528.

Pressure ratio at laminar flow

Specify the ratio between the downstream pressure and the upstream pressure at laminar flow. This value can be in the range between 0.995 and 0.999. The default value is 0.999.

Reference temperature

Specify the gas temperature at which the sonic conductance was measured. The default value is 293.15 K.

Density at reference conditions

Specify the gas density at which the sonic conductance was measured. The default value is 1.185 kg/m^3 .

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the orifice inlet for positive flow.

B

Pneumatic conserving port associated with the orifice outlet for positive flow.

References

[1] Sanville, F. E. "A New Method of Specifying the Flow Capacity of Pneumatic Fluid Power Valves." Paper D3, p.37-47. BHRA. Second International Fluid Power Symposium, Guildford, England, 1971.

[2] Beater, P. *Pneumatic Drives. System Design, Modeling, and Control*. New York: Springer, 2007.

See Also

Constant Area Pneumatic Orifice | Variable Area Pneumatic Orifice

Introduced in R2009b

Constant Volume Chamber (2P)

Chamber with one port and fixed volume of two-phase fluid

Library: Simscape / Foundation Library / Two-Phase Fluid / Elements



Description

The Constant Volume Chamber (2P) block models the accumulation of mass and energy in a chamber containing a fixed volume of two-phase fluid. The chamber has one inlet, labeled **A**, through which fluid can flow. The fluid volume can exchange heat with a thermal network, for example one representing the chamber surroundings, through a thermal port labeled **H**.

The mass of the fluid in the chamber varies with density, a property that in a two-phase fluid is generally a function of pressure and temperature. Fluid enters when the pressure upstream of the inlet rises above that in the chamber and exits when the pressure gradient is reversed. The effect in a model is often to smooth out sudden changes in pressure, much like an electrical capacitor does with voltage.

The flow resistance between the inlet and interior of the chamber is assumed to be negligible. The pressure in the interior is therefore equal to that at the inlet. Similarly, the thermal resistance between the thermal port and interior of the chamber is assumed to be negligible. The temperature in the interior is equal to that at the thermal port.

Mass Balance

Mass can enter and exit the chamber through port **A**. The volume of the chamber is fixed but the compressibility of the fluid means that its mass can change with pressure and temperature. The rate of mass accumulation in the chamber must exactly equal the mass flow rate in through port **A**:

$$\left[\left(\frac{\partial \rho}{\partial p} \right)_u \frac{dp}{dt} + \left(\frac{\partial \rho}{\partial u} \right)_p \frac{du}{dt} \right] V = \dot{m}_A + \epsilon_M,$$

where the left-hand side is the rate of mass accumulation and:

- ρ is the density.
- p is the pressure.
- u is the specific internal energy.
- V is the volume.
- \dot{m} is the mass flow rate.
- ϵ_M is a correction term introduced to account for a numerical error caused by the smoothing of the partial derivatives.

Correction Term for Partial-Derivative Smoothing

The partial derivatives in the mass balance equation are computed by applying the finite-difference method to the tabulated data in the Two-Phase Fluid Properties (2P) block and interpolating the

results. The partial derivatives are then smoothed at the phase-transition boundaries by means of cubic polynomial functions. These functions apply between:

- The subcooled liquid and two-phase mixture phase domains when the vapor quality is in the 0-0.1 range.
- The two-phase mixture and superheated vapor phase domains when the vapor quality is in the 0-0.9 range.

The smoothing introduces a small numerical error that the block adjusts for by adding to the mass balance the correction term ϵ_M , defined as:

$$\epsilon_M = \frac{M - V/\nu}{\tau}.$$

where:

- M is the fluid mass in the chamber.
- ν is the specific volume.
- τ is the characteristic duration of a phase-change event.

The fluid mass in the chamber is obtained from the equation:

$$\frac{dM}{dt} = \dot{m}_A.$$

Energy Balance

Energy can enter and exit the chamber in two ways: with fluid flow through port **A** and with heat flow through port **H**. No work is done on or by the fluid inside the chamber. The rate of energy accumulation in the internal fluid volume must then equal the sum of the energy flow rates in through ports **A** and **H**:

$$\dot{E} = \phi_A + Q_H,$$

where:

- ϕ is energy flow rate.
- Q is heat flow rate.
- E is total energy.

Neglecting the kinetic energy of the fluid, the total energy in the chamber is:

$$E = Mu.$$

Momentum Balance

The pressure drop due to viscous friction between port **A** and the interior of the chamber is assumed to be negligible. Gravity is ignored as are other body forces. The pressure in the internal fluid volume must then equal that at port **A**:

$$p = p_A.$$

Assumptions

- The chamber has a fixed volume of fluid.
- The flow resistance between the inlet and the interior of the chamber is negligible.
- The thermal resistance between the thermal port and the interior of the chamber is negligible.
- The kinetic energy of the fluid in the chamber is negligible.

Ports

Conserving

A — Fluid inlet

two-phase fluid

Opening through which fluid enters or exits the chamber.

H — Thermal port

thermal

Interface through which the fluid in the chamber exchanges heat with a thermal network.

Parameters

Parameters Tab

Chamber volume — Volume of fluid inside the chamber

293.15 K (default) | scalar with units of area

Volume of fluid in the chamber. This volume is constant during simulation.

Cross-sectional area at port A — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of area

Inlet area normal to the direction of flow.

Effects and Initial Conditions Tab

Initial fluid energy specification — Thermodynamic variable whose initial value to set

Temperature (default) | Vapor quality | Vapor void fraction | Specific enthalpy | Specific internal energy

Thermodynamic variable in terms of which to define the initial conditions of the component.

Initial pressure — Absolute pressure at the start of simulation

0.101325 MPa (default) | scalar with units of pressure

Pressure in the chamber at the start of simulation, specified against absolute zero.

Initial temperature — Absolute temperature at the start of simulation

293.15 K (default) | scalar with units of temperature

Temperature in the chamber at the start of simulation, specified against absolute zero.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Temperature.

Initial vapor quality — Mass fraction of vapor at the start of simulation

0.5 (default) | unitless scalar between 0 and 1

Mass fraction of vapor in the chamber at the start of simulation.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Vapor quality.

Initial vapor void fraction — Volume fraction of vapor at the start of simulation

0.5 (default) | unitless scalar between 0 and 1

Volume fraction of vapor in the chamber at the start of simulation.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Vapor void fraction.

Initial specific enthalpy — Specific enthalpy at the start of simulation

1500 kJ/kg (default) | scalar with units of energy/mass

Specific enthalpy of the fluid in the chamber at the start of simulation.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Specific enthalpy.

Initial specific internal energy — Specific internal energy at the start of simulation

1500 kJ/kg (default) | scalar with units of energy/mass

Specific internal energy of the fluid in the chamber at the start of simulation.

Dependencies

This parameter is active when the **Initial fluid energy specification** option is set to Specific internal energy.

Phase change time constant — Characteristic duration of a phase-change event

0.1 s (default) | scalar with units of time

Characteristic time to equilibrium of a phase-change event taking place in the chamber. Increase this parameter to slow the rate of phase change or decrease it to speed the rate.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

2-Port Constant Volume Chamber (2P) | 3-Port Constant Volume Chamber (2P) | Reservoir (2P)

Introduced in R2015b

Constant Volume Chamber (G)

Chamber with fixed volume of gas and variable number of ports

Library: Simscape / Foundation Library / Gas / Elements



Description

The Constant Volume Chamber (G) block models mass and energy storage in a gas network. The chamber contains a constant volume of gas. It can have between one and four inlets. The enclosure can exchange mass and energy with the connected gas network and exchange heat with the environment, allowing its internal pressure and temperature to evolve over time. The pressure and temperature evolve based on the compressibility and thermal capacity of the gas volume.

Mass Balance

Mass conservation relates the mass flow rates to the dynamics of the pressure and temperature of the internal node representing the gas volume:

$$\frac{\partial M}{\partial p} \cdot \frac{dp_I}{dt} + \frac{\partial M}{\partial T} \cdot \frac{dT_I}{dt} = \dot{m}_A + \dot{m}_B + \dot{m}_C + \dot{m}_D,$$

where:

- $\frac{\partial M}{\partial p}$ is the partial derivative of the mass of the gas volume with respect to pressure at constant temperature and volume.
- $\frac{\partial M}{\partial T}$ is the partial derivative of the mass of the gas volume with respect to temperature at constant pressure and volume.
- p_I is the pressure of the gas volume. The pressure at ports **A**, **B**, **C**, and **D** is assumed to be equal to this pressure, $p_A = p_B = p_C = p_D = p_I$.
- T_I is the temperature of the gas volume. The temperature at port **H** is assumed to be equal to this temperature, $T_H = T_I$.
- t is time.
- \dot{m}_A is the mass flow rate at port **A**. The flow rate associated with a port is positive when it flows into the block.
- \dot{m}_B is the mass flow rate at port **B**. The flow rate associated with a port is positive when it flows into the block.
- \dot{m}_C is the mass flow rate at port **C**. The flow rate associated with a port is positive when it flows into the block.
- \dot{m}_D is the mass flow rate at port **D**. The flow rate associated with a port is positive when it flows into the block.

Energy Balance

Energy conservation relates the energy and heat flow rates to the dynamics of the pressure and temperature of the internal node representing the gas volume:

$$\frac{\partial U}{\partial p} \cdot \frac{dp_I}{dt} + \frac{\partial U}{\partial T} \cdot \frac{dT_I}{dt} = \Phi_A + \Phi_B + \Phi_C + \Phi_D + Q_H,$$

where:

- $\frac{\partial U}{\partial p}$ is the partial derivative of the internal energy of the gas volume with respect to pressure at constant temperature and volume.
- $\frac{\partial U}{\partial T}$ is the partial derivative of the internal energy of the gas volume with respect to temperature at constant pressure and volume.
- Φ_A is the energy flow rate at port **A**.
- Φ_B is the energy flow rate at port **B**.
- Φ_C is the energy flow rate at port **C**.
- Φ_D is the energy flow rate at port **D**.
- Q_H is the heat flow rate at port **H**.

Partial Derivatives for Perfect and Semiperfect Gas Models

The partial derivatives of the mass M and the internal energy U of the gas volume, with respect to pressure and temperature at constant volume, depend on the gas property model. For perfect and semiperfect gas models, the equations are:

$$\frac{\partial M}{\partial p} = V \frac{\rho_I}{p_I}$$

$$\frac{\partial M}{\partial T} = -V \frac{\rho_I}{T_I}$$

$$\frac{\partial U}{\partial p} = V \left(\frac{h_I}{ZRT_I} - 1 \right)$$

$$\frac{\partial U}{\partial T} = V \rho_I \left(c_{pI} - \frac{h_I}{T_I} \right)$$

where:

- ρ_I is the density of the gas volume.
- V is the volume of gas.
- h_I is the specific enthalpy of the gas volume.
- Z is the compressibility factor.
- R is the specific gas constant.
- c_{pI} is the specific heat at constant pressure of the gas volume.

Partial Derivatives for Real Gas Model

For real gas model, the partial derivatives of the mass M and the internal energy U of the gas volume, with respect to pressure and temperature at constant volume, are:

$$\frac{\partial M}{\partial p} = V \frac{\rho_I}{\beta_I}$$

$$\frac{\partial M}{\partial T} = -V \rho_I \alpha_I$$

$$\frac{\partial U}{\partial p} = V \left(\frac{\rho_I h_I}{\beta_I} - T_I \alpha_I \right)$$

$$\frac{\partial U}{\partial T} = V \rho_I (c_{pI} - h_I \alpha_I)$$

where:

- β is the isothermal bulk modulus of the gas volume.
- α is the isobaric thermal expansion coefficient of the gas volume.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Gas Volume”.

Assumptions and Limitations

- The chamber walls are perfectly rigid.
- There is no flow resistance between ports **A**, **B**, **C**, and **D** and the chamber interior.
- There is no thermal resistance between port **H** and the chamber interior.

Ports

Conserving

A – Chamber inlet

gas

Gas conserving port associated with the chamber inlet.

B – Chamber inlet

gas

Gas conserving port associated with the second chamber inlet.

Dependencies

This port is visible if you set the **Number of ports** parameter to 2, 3, or 4.

C – Chamber inlet

gas

Gas conserving port associated with the third chamber inlet.

Dependencies

This port is visible if you set the **Number of ports** parameter to 3 or 4.

D – Chamber inlet

gas

Gas conserving port associated with the fourth chamber inlet. If a chamber has four inlet ports, you can use it as a junction in a cross connection.

Dependencies

This port is visible only if you set the **Number of ports** parameter to 4.

H – Temperature inside chamber

thermal

Thermal conserving port associated with the temperature of the gas inside the chamber.

Parameters**Chamber volume – Volume of gas in the chamber**

0.001 m³ (default)

Volume of gas in the chamber. The chamber is rigid and therefore its volume is constant during simulation. The chamber is assumed to be completely filled with gas at all times.

Number of ports – Number of inlet ports in the chamber

1 (default) | 2 | 3 | 4

Number of inlet ports in the chamber. The chamber can have between one and four ports, labeled from **A** to **D**. When you modify the parameter value, the corresponding ports are exposed or hidden in the block icon.

Cross-sectional area at port A – Area normal to flow path at the chamber inlet

0.01 m² (default)

Cross-sectional area of the chamber inlet at port **A**, in the direction normal to gas flow path.

Cross-sectional area at port B – Area normal to flow path at the chamber inlet

0.01 m² (default)

Cross-sectional area of the chamber inlet at port **B**, in the direction normal to gas flow path.

Dependencies

Enabled when port **B** is visible, that is, when the **Number of ports** parameter is set to 2, 3, or 4.

Cross-sectional area at port C – Area normal to flow path at the chamber inlet

0.01 m² (default)

Cross-sectional area of the chamber inlet at port **C**, in the direction normal to gas flow path.

Dependencies

Enabled when port **C** is visible, that is, when the **Number of ports** parameter is set to 3 or 4.

Cross-sectional area at port D – Area normal to flow path at the chamber inlet

0.01 m² (default)

Cross-sectional area of the chamber inlet at port **D**, in the direction normal to gas flow path.

Dependencies

Enabled when port **D** is visible, that is, when the **Number of ports** parameter is set to 4.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Reservoir (G)

Topics

“Modeling Gas Systems”

Introduced in R2016b

Constant Volume Chamber (IL)

Chamber with one port and fixed volume of isothermal liquid

Library: Simscape / Foundation Library / Isothermal Liquid / Elements



Description

The Constant Volume Chamber (IL) block models accumulation of mass in a chamber containing a fixed volume of isothermal liquid. The chamber has one inlet, labeled **A**.

The chamber can exchange mass with the connected isothermal liquid network. The pressure inside the chamber evolves based on the compressibility of the fluid volume.

Mass conservation relates the mass flow rate to the pressure of the internal node representing the fluid volume:

$$\frac{d\rho_{\text{mix}}}{dp_I} \dot{p}_I V = \dot{m}_A,$$

where:

- \dot{m}_A is the mass flow rate at port **A**. The flow rate associated with a port is positive when it flows into the block.
- p_I is the pressure inside the chamber. There is no flow resistance between port **A** and the chamber interior, therefore, pressure at port **A** is assumed to be equal to this pressure, $p_A = p_I$.
- V is the volume of fluid inside the chamber.
- ρ_{mix} is fluid mixture density.

The fluid can be a mixture of pure liquid and a small amount of entrained air, as specified by the Isothermal Liquid Properties (IL) block connected to the circuit. Equations used to compute ρ_{mix} depend on the selected isothermal liquid model. For detailed information, see “Isothermal Liquid Modeling Options”.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Assumptions and Limitations

- The chamber walls are perfectly rigid.
- There is no flow resistance between port **A** and the chamber interior.

Ports

Conserving

A – Chamber inlet

isothermal liquid

Isothermal liquid conserving port associated with the chamber inlet.

Parameters

Chamber volume – Volume of fluid inside the chamber

0.001 m³ (default) | positive scalar

Volume of fluid in the chamber. This volume is constant during simulation.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Reservoir (IL)

Topics

“Modeling Isothermal Liquid Systems”

“Isothermal Liquid Modeling Options”

Introduced in R2020a

Constant Volume Chamber (MA)

Chamber with fixed volume of moist air and variable number of ports

Library: Simscape / Foundation Library / Moist Air / Elements



Description

The Constant Volume Chamber (MA) block models mass and energy storage in a moist air network. The chamber contains a constant volume of moist air. It can have between one and four inlets. The enclosure can exchange mass and energy with the connected moist air network and exchange heat with the environment, allowing its internal pressure and temperature to evolve over time. The pressure and temperature evolve based on the compressibility and thermal capacity of the moist air volume. Liquid water condenses out of the moist air volume when it reaches saturation.

The block equations use these symbols. Subscripts a, w, and g indicate the properties of dry air, water vapor, and trace gas, respectively. Subscript ws indicates water vapor at saturation. Subscripts A, B, C, D, H, and S indicate the appropriate port. Subscript I indicates the properties of the internal moist air volume.

\dot{m}	Mass flow rate
Φ	Energy flow rate
Q	Heat flow rate
p	Pressure
ρ	Density
R	Specific gas constant
V	Volume of moist air inside the chamber
c_v	Specific heat at constant volume
h	Specific enthalpy
u	Specific internal energy
x	Mass fraction (x_w is specific humidity, which is another term for water vapor mass fraction)
y	Mole fraction
φ	Relative humidity
r	Humidity ratio
T	Temperature
t	Time

The net flow rates into the moist air volume inside the chamber are

$$\begin{aligned}\dot{m}_{net} &= \dot{m}_A + \dot{m}_B + \dot{m}_C + \dot{m}_D - \dot{m}_{condense} + \dot{m}_{wS} + \dot{m}_{gS} \\ \Phi_{net} &= \Phi_A + \Phi_B + \Phi_C + \Phi_D + Q_H - \Phi_{condense} + \Phi_S \\ \dot{m}_{w,net} &= \dot{m}_{wA} + \dot{m}_{wB} + \dot{m}_{wC} + \dot{m}_{wD} - \dot{m}_{condense} + \dot{m}_{wS} \\ \dot{m}_{g,net} &= \dot{m}_{gA} + \dot{m}_{gB} + \dot{m}_{gC} + \dot{m}_{gD} + \dot{m}_{gS}\end{aligned}$$

where:

- $\dot{m}_{condense}$ is the rate of condensation.
- $\Phi_{condense}$ is the rate of energy loss from the condensed water.
- Φ_S is the rate of energy added by the sources of moisture and trace gas. \dot{m}_{wS} and \dot{m}_{gS} are mass flow rates of water and gas, respectively, through port **S**. The values of \dot{m}_{wS} , \dot{m}_{gS} , and Φ_S are determined by the moisture and trace gas sources connected to port **S** of the chamber, or by the corresponding parameter values on the **Moisture and Trace Gas** tab.

If a port is not visible, then the terms with the subscript corresponding to the port name are 0.

Water vapor mass conservation relates the water vapor mass flow rate to the dynamics of the moisture level in the internal moist air volume:

$$\frac{dx_{wI}}{dt} \rho_I V + x_{wI} \dot{m}_{net} = \dot{m}_{w,net}$$

Similarly, trace gas mass conservation relates the trace gas mass flow rate to the dynamics of the trace gas level in the internal moist air volume:

$$\frac{dx_{gI}}{dt} \rho_I V + x_{gI} \dot{m}_{net} = \dot{m}_{g,net}$$

Mixture mass conservation relates the mixture mass flow rate to the dynamics of the pressure, temperature, and mass fractions of the internal moist air volume:

$$\left(\frac{1}{\rho_I} \frac{d\rho_I}{dt} - \frac{1}{T_I} \frac{dT_I}{dt} \right) \rho_I V + \frac{R_a - R_w}{R_I} (\dot{m}_{w,net} - x_w \dot{m}_{net}) + \frac{R_a - R_g}{R_I} (\dot{m}_{g,net} - x_g \dot{m}_{net}) = \dot{m}_{net}$$

Finally, energy conservation relates the energy flow rate to the dynamics of the pressure, temperature, and mass fractions of the internal moist air volume:

$$\rho_I c_{vI} V \frac{dT_I}{dt} + (u_{wI} - u_{aI}) (\dot{m}_{w,net} - x_w \dot{m}_{net}) + (u_{gI} - u_{aI}) (\dot{m}_{g,net} - x_g \dot{m}_{net}) + u_I \dot{m}_{net} = \Phi_{net}$$

The equation of state relates the mixture density to the pressure and temperature:

$$\rho_I = \rho_I R_I T_I$$

The mixture specific gas constant is

$$R_I = x_{aI} R_a + x_{wI} R_w + x_{gI} R_g$$

Flow resistance and thermal resistance are not modeled in the chamber:

$$\begin{aligned}p_A &= p_B = p_C = p_D = p_I \\ T_H &= T_I\end{aligned}$$

When the moist air volume reaches saturation, condensation may occur. The specific humidity at saturation is

$$x_{wsI} = \varphi_{ws} \frac{R_I p_{wsI}}{R_w p_I}$$

where:

- φ_{ws} is the relative humidity at saturation (typically 1).
- p_{wsI} is the water vapor saturation pressure evaluated at T_I .

The rate of condensation is

$$\dot{m}_{condense} = \begin{cases} 0, & \text{if } x_{wI} \leq x_{wsI} \\ \frac{x_{wI} - x_{wsI}}{\tau_{condense}} \rho_I V, & \text{if } x_{wI} > x_{wsI} \end{cases}$$

where $\tau_{condense}$ is the value of the **Condensation time constant** parameter.

The condensed water is subtracted from the moist air volume, as shown in the conservation equations. The energy associated with the condensed water is

$$\Phi_{condense} = \dot{m}_{condense} (h_{wI} - \Delta h_{vapI})$$

where Δh_{vapI} is the specific enthalpy of vaporization evaluated at T_I .

Other moisture and trace gas quantities are related to each other as follows:

$$\varphi_{wI} = \frac{y_{wI} p_I}{p_{wsI}}$$

$$y_{wI} = \frac{x_{wI} R_w}{R_I}$$

$$r_{wI} = \frac{x_{wI}}{1 - x_{wI}}$$

$$y_{gI} = \frac{x_{gI} R_g}{R_I}$$

$$x_{aI} + x_{wI} + x_{gI} = 1$$

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Moist Air Volume”.

Assumptions and Limitations

- The chamber walls are perfectly rigid.
- Flow resistance between the chamber inlet and the moist air volume is not modeled. Connect a Local Restriction (MA) block or a Flow Resistance (MA) block to port **A** to model the pressure losses associated with the inlet.

- Thermal resistance between port **H** and the moist air volume is not modeled. Use Thermal library blocks to model thermal resistances between the moist air mixture and the environment, including any thermal effects of a chamber wall.

Ports

Output

W – Rate of condensation measurement, kg/s

physical signal

Physical signal output port that measures the rate of condensation in the chamber.

F – Vector physical signal containing pressure, temperature, humidity, and trace gas levels data

physical signal vector

Physical signal output port that outputs a vector signal. The vector contains the pressure (in Pa), temperature (in K), moisture level, and trace gas level measurements inside the component. Use the Measurement Selector (MA) block to unpack this vector signal.

Conserving

A – Chamber inlet

moist air

Moist air conserving port associated with the chamber inlet.

B – Chamber inlet

moist air

Moist air conserving port associated with the second chamber inlet.

Dependencies

This port is visible if you set the **Number of ports** parameter to 2, 3, or 4.

C – Chamber inlet

moist air

Moist air conserving port associated with the third chamber inlet.

Dependencies

This port is visible if you set the **Number of ports** parameter to 3 or 4.

D – Chamber inlet

moist air

Moist air conserving port associated with the fourth chamber inlet. If a chamber has four inlet ports, you can use it as a junction in a cross connection.

Dependencies

This port is visible only if you set the **Number of ports** parameter to 4.

H — Temperature inside chamber

thermal

Thermal conserving port associated with the temperature of the air mixture inside the chamber.

S — Inject or extract moisture and trace gas

moist air source

Connect this port to port **S** of a block from the Moisture & Trace Gas Sources library to add or remove moisture and trace gas. For more information, see “Using Moisture and Trace Gas Sources”.

Dependencies

This port is visible only if you set the **Moisture and trace gas source** parameter to **Controlled**.

Parameters**Main****Chamber volume — Volume of moist air in the chamber**0.001 m³ (default)

Volume of moist air in the chamber. The chamber is rigid and therefore its volume is constant during simulation. The chamber is assumed to be completely filled with moist air at all times.

Number of ports — Number of inlet ports in the chamber

1 (default) | 2 | 3 | 4

Number of inlet ports in the chamber. The chamber can have between one and four ports, labeled from **A** to **D**. When you modify the parameter value, the corresponding ports are exposed or hidden in the block icon.

Cross-sectional area at port A — Area normal to flow path at the chamber inlet0.01 m² (default)

Cross-sectional area of the chamber inlet at port **A**, in the direction normal to air flow path.

Cross-sectional area at port B — Area normal to flow path at the chamber inlet0.01 m² (default)

Cross-sectional area of the chamber inlet at port **B**, in the direction normal to air flow path.

Dependencies

Enabled when port **B** is visible, that is, when the **Number of ports** parameter is set to 2, 3, or 4.

Cross-sectional area at port C — Area normal to flow path at the chamber inlet0.01 m² (default)

Cross-sectional area of the chamber inlet at port **C**, in the direction normal to air flow path.

Dependencies

Enabled when port **C** is visible, that is, when the **Number of ports** parameter is set to 3 or 4.

Cross-sectional area at port D — Area normal to flow path at the chamber inlet0.01 m² (default)

Cross-sectional area of the chamber inlet at port **D**, in the direction normal to air flow path.

Dependencies

Enabled when port **D** is visible, that is, when the **Number of ports** parameter is set to 4.

Moisture and Trace Gas

Relative humidity at saturation — Relative humidity above which condensation occurs 1 (default)

Relative humidity above which condensation occurs.

Condensation time constant — Time scale for condensation 1e-3 s (default)

Characteristic time scale at which an oversaturated moist air volume returns to saturation by condensing out excess moisture.

Moisture and trace gas source — Model moisture and trace gas levels None (default) | Constant | Controlled

This parameter controls visibility of port **S** and provides these options for modeling moisture and trace gas levels inside the component:

- **None** — No moisture or trace gas is injected into or extracted from the block. Port **S** is hidden. This is the default.
- **Constant** — Moisture and trace gas are injected into or extracted from the block at a constant rate. The same parameters as in the Moisture Source (MA) and Trace Gas Source (MA) blocks become available in the **Moisture and Trace Gas** section of the block interface. Port **S** is hidden.
- **Controlled** — Moisture and trace gas are injected into or extracted from the block at a time-varying rate. Port **S** is exposed. Connect the Controlled Moisture Source (MA) and Controlled Trace Gas Source (MA) blocks to this port.

Moisture added or removed — Select whether the block adds or removes moisture as water vapor or liquid water Vapor (default) | Liquid

Select whether the block adds or removes moisture as water vapor or liquid water:

- **Vapor** — The enthalpy of the added or removed moisture corresponds to the enthalpy of water vapor, which is greater than that of liquid water.
- **Liquid** — The enthalpy of the added or removed moisture corresponds to the enthalpy of liquid water, which is less than that of water vapor.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Rate of added moisture — Constant mass flow rate through the block 0 kg/s (default)

Water vapor mass flow rate through the block. A positive value adds moisture to the connected moist air volume. A negative value extracts moisture from that volume.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Added moisture temperature specification — Select specification method for the temperature of added moisture

Atmospheric temperature (default) | Specified temperature

Select a specification method for the moisture temperature:

- **Atmospheric temperature** — Use the atmospheric temperature, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified temperature** — Specify a value by using the **Temperature of added moisture** parameter.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Temperature of added moisture — Moisture temperature

293.15 K (default)

Enter the desired temperature of added moisture. This temperature remains constant during simulation. The block uses this value to evaluate the specific enthalpy of the added moisture only. The specific enthalpy of removed moisture is based on the temperature of the connected moist air volume.

Dependencies

Enabled when the **Added moisture temperature specification** parameter is set to Specified temperature.

Rate of added trace gas — Constant mass flow rate through the block

0 kg/s (default)

Trace gas mass flow rate through the block. A positive value adds trace gas to the connected moist air volume. A negative value extracts trace gas from that volume.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Added trace gas temperature specification — Select specification method for the temperature of added trace gas

Atmospheric temperature (default) | Specified temperature

Select a specification method for the trace gas temperature:

- **Atmospheric temperature** — Use the atmospheric temperature, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified temperature** — Specify a value by using the **Temperature of added trace gas** parameter.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Temperature of added trace gas — Trace gas temperature

293.15 K (default)

Enter the desired temperature of added trace gas. This temperature remains constant during simulation. The block uses this value to evaluate the specific enthalpy of the added trace gas only. The specific enthalpy of removed trace gas is based on the temperature of the connected moist air volume.

Dependencies

Enabled when the **Added trace gas temperature specification** parameter is set to Specified temperature.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Moisture Source (MA) | Trace Gas Source (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Constant Volume Chamber (TL)

Chamber with fixed volume of thermal liquid and variable number of ports

Library: Simscape / Foundation Library / Thermal Liquid / Elements



Description

The Constant Volume Chamber (TL) block models the accumulation of mass and energy in a chamber containing a fixed volume of thermal liquid. The chamber can have between one and four inlets, labeled from **A** to **D**, through which fluid can flow. The fluid volume can exchange heat with a thermal network, such as a network representing the chamber surroundings, through the thermal port **H**.

The mass of the fluid in the chamber varies with density, a property that in a thermal liquid is generally a function of pressure and temperature. Fluid enters when the pressure upstream of the inlet rises above that in the chamber and exits when the pressure gradient is reversed. The effect in a model is often to smooth out sudden changes in pressure, much like an electrical capacitor does with voltage.

The flow resistance between the inlet and the interior of the chamber is assumed to be negligible. The pressure in the interior is therefore equal to the pressure at the inlet. Similarly, the thermal resistance between the thermal port and the interior of the chamber is assumed to be negligible. The temperature in the interior is equal to the temperature at the thermal port.

Mass Balance

Mass can enter and exit the chamber through ports **A**, **B**, **C**, and **D**. The volume of the chamber is fixed, but the compressibility of the fluid means that its mass can change with pressure and temperature. The rate of mass accumulation in the chamber must exactly equal the mass flow rates in through ports **A**, **B**, **C**, and **D**:

$$\left(\frac{1}{\beta} \frac{dp}{dt} - \alpha \frac{dT}{dt} \right) \rho V = \dot{m}_A + \dot{m}_B + \dot{m}_C + \dot{m}_D,$$

where the left-hand side is the rate of mass accumulation and:

- p is the pressure.
- T is the temperature.
- β is the isothermal bulk modulus.
- α is the isobaric thermal expansion coefficient.
- \dot{m} is the mass flow rate.

Energy Balance

Energy can enter and exit the chamber in two ways: with fluid flow through ports **A**, **B**, **C**, and **D**, and with heat flow through port **H**. No work is done on or by the fluid inside the chamber. The rate of energy accumulation in the internal fluid volume must therefore equal the sum of the energy flow rates in through ports **A**, **B**, **C**, **D**, and **H**:

$$\left[\left(\frac{h}{\beta} - \frac{T\alpha}{\rho} \right) \frac{dp}{dt} + (c_p - h\alpha) \frac{dT}{dt} \right] \rho V = \phi_A + \phi_B + \phi_C + \phi_D + Q_H,$$

where the left-hand side is the rate of energy accumulation and:

- h is the enthalpy.
- ρ is the density.
- c_p is the specific heat.
- V is the chamber volume.
- ϕ is the energy flow rate.
- Q is the heat flow rate.

Momentum Balance

The pressure drop due to viscous friction between the individual ports and the interior of the chamber is assumed to be negligible. Gravity is ignored, as are other body forces. The pressure in the internal fluid volume must therefore equal the pressure at ports **A**, **B**, **C**, and **D**:

$$p = p_A = p_B = p_C = p_D.$$

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Assumptions and Limitations

- The chamber has a fixed volume of fluid.
- The flow resistance between the inlet and the interior of the chamber is negligible.
- The thermal resistance between the thermal port and the interior of the chamber is negligible.
- The kinetic energy of the fluid in the chamber is negligible.

Ports

Conserving

A – Chamber inlet

thermal liquid

Thermal liquid conserving port associated with the chamber inlet.

B – Chamber inlet

thermal liquid

Thermal liquid conserving port associated with the second chamber inlet.

Dependencies

This port is visible if you set the **Number of ports** parameter to 2, 3, or 4.

C – Chamber inlet

thermal liquid

Thermal liquid conserving port associated with the third chamber inlet.

Dependencies

This port is visible if you set the **Number of ports** parameter to 3 or 4.

D – Chamber inlet

thermal liquid

Thermal liquid conserving port associated with the fourth chamber inlet. If a chamber has four inlet ports, you can use it as a junction in a cross connection.

Dependencies

This port is visible only if you set the **Number of ports** parameter to 4.

H – Thermal port

thermal

Thermal conserving port through which the fluid in the chamber exchanges heat with a thermal network.

Parameters**Chamber volume – Volume of fluid inside the chamber**

0.001 m³ (default)

Volume of fluid in the chamber. This volume is constant during simulation.

Number of ports – Number of inlet ports in the chamber

1 (default) | 2 | 3 | 4

Number of inlet ports in the chamber. The chamber can have between one and four ports, labeled from **A** to **D**. When you modify the parameter value, the corresponding ports are exposed or hidden in the block icon.

Cross-sectional area at port A – Inlet area normal to the direction of flow

0.01 m² (default)

Inlet area normal to the direction of flow.

Cross-sectional area at port B – Inlet area normal to the direction of flow

0.01 m² (default)

Inlet area normal to the direction of flow.

Dependencies

Enabled when port **B** is visible, that is, when the **Number of ports** parameter is set to 2, 3, or 4.

Cross-sectional area at port C – Inlet area normal to the direction of flow

0.01 m² (default)

Inlet area normal to the direction of flow.

Dependencies

Enabled when port **C** is visible, that is, when the **Number of ports** parameter is set to 3 or 4.

Cross-sectional area at port D — Inlet area normal to the direction of flow

0.01 m² (default)

Inlet area normal to the direction of flow.

Dependencies

Enabled when port **D** is visible, that is, when the **Number of ports** parameter is set to 4.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Reservoir (TL)

Introduced in R2013b

Constant Volume Hydraulic Chamber

Hydraulic capacity of constant volume



Library

Hydraulic Elements

Description

The Constant Volume Hydraulic Chamber block models a fixed-volume chamber with rigid or flexible walls, to be used in hydraulic valves, pumps, manifolds, pipes, hoses, and so on. Use this block in models where you have to account for some form of fluid compressibility. You can select the appropriate representation of fluid compressibility using the block parameters.

Fluid compressibility in its simplest form is simulated according to the following equations:

$$V_f = V_c + \frac{V_c}{E} p$$

$$q = \frac{V_c}{E} \cdot \frac{dp}{dt}$$

where

q	Flow rate into the chamber
V_f	Volume of fluid in the chamber
V_c	Geometrical chamber volume
E	Fluid bulk modulus
p	Gauge pressure of fluid in the chamber

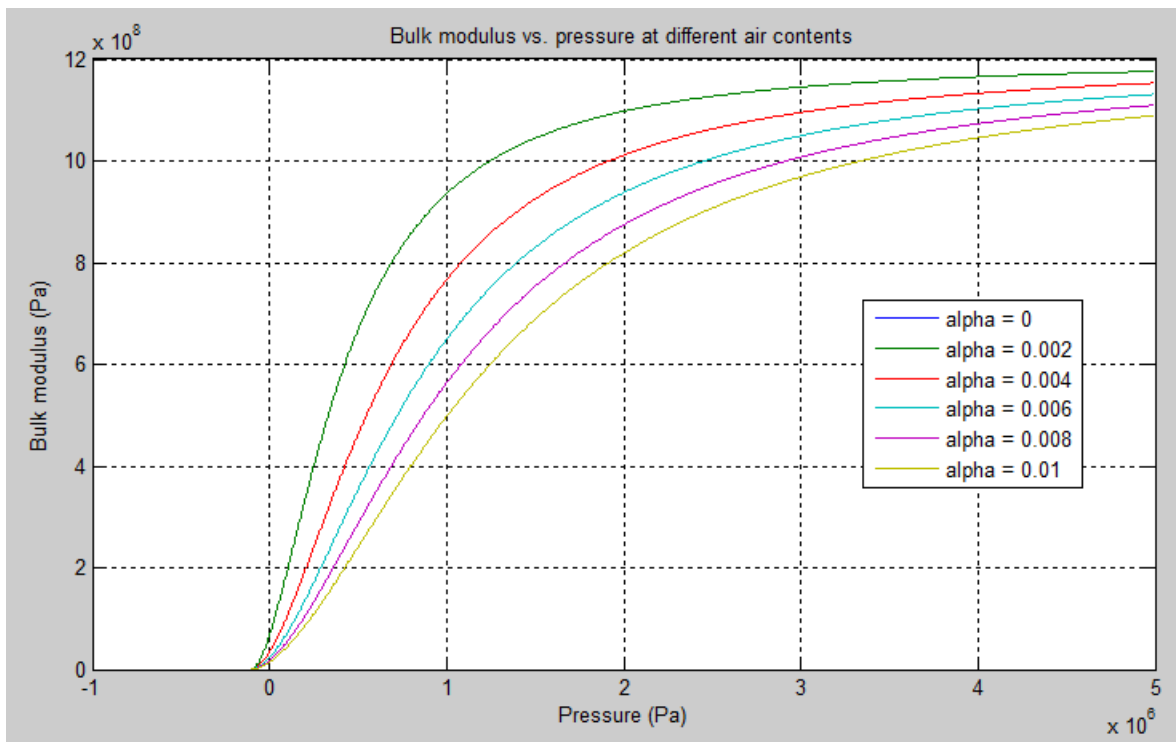
If pressure in the chamber is likely to fall to negative values and approach cavitation limit, the above equations must be enhanced. In this block, it is done by representing the fluid in the chamber as a mixture of liquid and a small amount of entrained, nondissolved gas (see [1 on page 1-77, 2 on page 1-77]). The mixture bulk modulus is determined as:

$$E = E_l \frac{1 + \alpha \left(\frac{p_a}{p_a + p} \right)^{1/n}}{1 + \alpha \frac{p_a^{1/n}}{n \cdot (p_a + p)^{\frac{n+1}{n}}} E_l}$$

where

E_l	Pure liquid bulk modulus
p_α	Atmospheric pressure
α	Relative gas content at atmospheric pressure, $\alpha = V_G/V_L$
V_G	Gas volume at atmospheric pressure
V_L	Volume of liquid
n	Gas-specific heat ratio

The main objective of representing fluid as a mixture of liquid and gas is to introduce an approximate model of cavitation, which takes place in a chamber if pressure drops below fluid vapor saturation level. As it is seen in the graph below, the bulk modulus of a mixture decreases as the gauge pressure approaches zero, thus considerably slowing down further pressure change. At gauge pressures far above zero, a small amount of undissolved gas has practically no effect on the system behavior.



To reproduce this graph, copy and paste the following script in your MATLAB® Command Window:

```
% Parameters
p_atm = 1.01325e5; % Atmospheric pressure [Pa]
pressure = -1.01325e5:1e3:5e6; % Pressure (gauge) [Pa]
alpha = 0:2e-3:0.01; % Relative amount of trapped air
k_sh = 1.4; % Specific heat ratio
bulk = 1.24285e+09; % Bulk modulus at atmospheric pressure and no gas [Pa]

% Absolute pressure
p_abs = p_atm + pressure;
% Relative absolute pressure
p_nom = (p_atm./p_abs).^(1/k_sh);
p_den = p_nom .* bulk ./ (k_sh .* p_abs);
% Instantaneous bulk modulus
```

```

bulk_inst = bulk * (1+ bsxfun(@times, alpha', p_nom)) ./ (1 + bsxfun(@times, alpha', p_den));

% Reuse figure if it exists, else create new figure
if ~exist('h1_bulk_modulus', 'var') || ~isgraphics(h1_bulk_modulus, 'figure')
    h1_bulk_modulus = figure('Name', 'h1_bulk_modulus');
end
figure(h1_bulk_modulus)
clf(h1_bulk_modulus)
legend_label = cell(length(alpha),1);

for i=1:length(alpha)
    plot(pressure, bulk_inst(i,:))
    hold on
    legend_label{i,1} = ['alpha = ', num2str(alpha(i))];
end

grid on
xlabel('Pressure (Pa)')
ylabel('Bulk modulus (Pa)')
title('Bulk modulus vs. pressure at different air contents')
legend(legend_label, 'Location', 'Best')
hold off

```

Cavitation is an inherently thermodynamic process, requiring consideration of multiple-phase fluids, heat transfers, etc., and as such cannot be accurately simulated with Simscape software. But the simplified version implemented in the block is good enough to signal if pressure falls below dangerous level, and to prevent computation failure that normally occurs at negative pressures.

If pressure falls below absolute vacuum (-101325 Pa), the simulation stops and an error message is displayed.

If chamber walls have noticeable compliance, the above equations must be further enhanced by representing geometrical chamber volume as a function of pressure:

$$V_c = \pi d^2 / 4 \cdot L$$

$$d(s) = \frac{K_p}{1 + \tau s} p(s)$$

where

d	Internal diameter of the cylindrical chamber
L	Length of the cylindrical chamber
K_p	Proportionality coefficient (m/Pa)
τ	Time constant
s	Laplace operator

Coefficient K_p establishes relationship between pressure and the internal diameter at steady-state conditions. For metal tubes, the coefficient can be computed as (see [2 on page 1-77]):

$$K_p = \frac{d}{E_M} \left(\frac{D^2 + d^2}{D^2 - d^2} + \nu \right)$$

where

D	Pipe external diameter
E_M	Modulus of elasticity (Young's modulus) for the pipe material
ν	Poisson's ratio for the pipe material

For hoses, the coefficient can be provided by the manufacturer.

The process of expansion and contraction in pipes and especially in hoses is a complex combination of nonlinear elastic and viscoelastic deformations. This process is approximated in the block with the first-order lag, whose time constant is determined empirically (for example, see [3 on page 1-77]).

As a result, by selecting appropriate values, you can implement four different models of fluid compressibility with this block:

- Chamber with rigid walls, no entrained gas in the fluid
- Cylindrical chamber with compliant walls, no entrained gas in the fluid
- Chamber with rigid walls, fluid with entrained gas
- Cylindrical chamber with compliant walls, fluid with entrained gas

The block allows two methods of specifying the chamber size:

- By volume — Use this option for cylindrical or non-cylindrical chambers with rigid walls. You only need to know the volume of the chamber. This chamber type does not account for wall compliance.
- By length and diameter — Use this option for cylindrical chambers with rigid or compliant walls, such as circular pipes or hoses.

The block has one hydraulic conserving port associated with the chamber inlet. The block positive direction is from its port to the reference point. This means that the flow rate is positive if it flows into the chamber.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Basic Assumptions and Limitations

- No inertia associated with pipe walls is taken into account.
- Chamber with compliant walls is assumed to have a cylindrical shape. Chamber with rigid wall can have any shape.

Parameters

Chamber specification

The parameter can have one of two values: **By volume** or **By length and diameter**. The value **By length and diameter** is recommended if a chamber is formed by a circular pipe. If the parameter is set to **By volume**, wall compliance is not taken into account. The default value of the parameter is **By volume**.

Chamber wall type

The parameter can have one of two values: **Rigid** or **Compliant**. If the parameter is set to **Rigid**, wall compliance is not taken into account, which can improve computational efficiency.

The value **Compliant** is recommended for hoses and metal pipes, where compliance can affect the system behavior. The default value of the parameter is **Rigid**. The parameter is used if the **Chamber specification** parameter is set to **By length and diameter**.

Chamber volume

Volume of fluid in the chamber. The default value is $1e-4 \text{ m}^3$. The parameter is used if the **Chamber specification** parameter is set to **By volume**.

Chamber internal diameter

Internal diameter of the cylindrical chamber. The default value is 0.01 m . The parameter is used if the **Chamber specification** parameter is set to **By length and diameter**.

Cylindrical chamber length

Length of the cylindrical chamber. The default value is 1 m . The parameter is used if the **Chamber specification** parameter is set to **By length and diameter**.

Static pressure-diameter coefficient

Coefficient K_p that establishes relationship between pressure and the internal diameter at steady-state conditions. The parameter can be determined analytically or experimentally. The default value is $1.2e-12 \text{ m/Pa}$. The parameter is used if **Chamber wall type** is set to **Compliant**.

Viscoelastic process time constant

Time constant in the transfer function relating pipe internal diameter to pressure variations. With this parameter, the simulated elastic or viscoelastic process is approximated with the first-order lag. The parameter is determined experimentally or provided by the manufacturer. The default value is 0.01 s . The parameter is used if **Chamber wall type** is set to **Compliant**.

Specific heat ratio

Gas-specific heat ratio. The default value is 1.4 .

Restricted Parameters

When your model is in Restricted editing mode, you cannot modify the following parameters:

- **Chamber specification**
- **Chamber wall type**

All other block parameters are available for modification. The actual set of modifiable block parameters depends on the values of the **Tube cross section type** and **Chamber wall type** parameters at the time the model entered Restricted mode.

Global Parameters

Parameters determined by the type of working fluid:

- **Fluid bulk modulus**
- **Nondissolved gas ratio** — Nondissolved gas relative content determined as a ratio of gas volume to the liquid volume.

Use the Hydraulic Fluid block or the Custom Hydraulic Fluid block to specify the fluid properties.

Ports

The block has one hydraulic conserving port associated with the chamber inlet.

References

- [1] Manring, N.D., *Hydraulic Control Systems*, John Wiley & Sons, New York, 2005
- [2] Meritt, H.E., *Hydraulic Control Systems*, John Wiley & Sons, New York, 1967
- [3] Holcke, Jan, *Frequency Response of Hydraulic Hoses*, RIT, FTH, Stockholm, 2002

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Variable Hydraulic Chamber

Introduced in R2009b

Constant Volume Pneumatic Chamber

Constant volume pneumatic chamber based on ideal gas law



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Constant Volume Pneumatic Chamber block models a constant volume pneumatic chamber based on the ideal gas law and assuming constant specific heats.

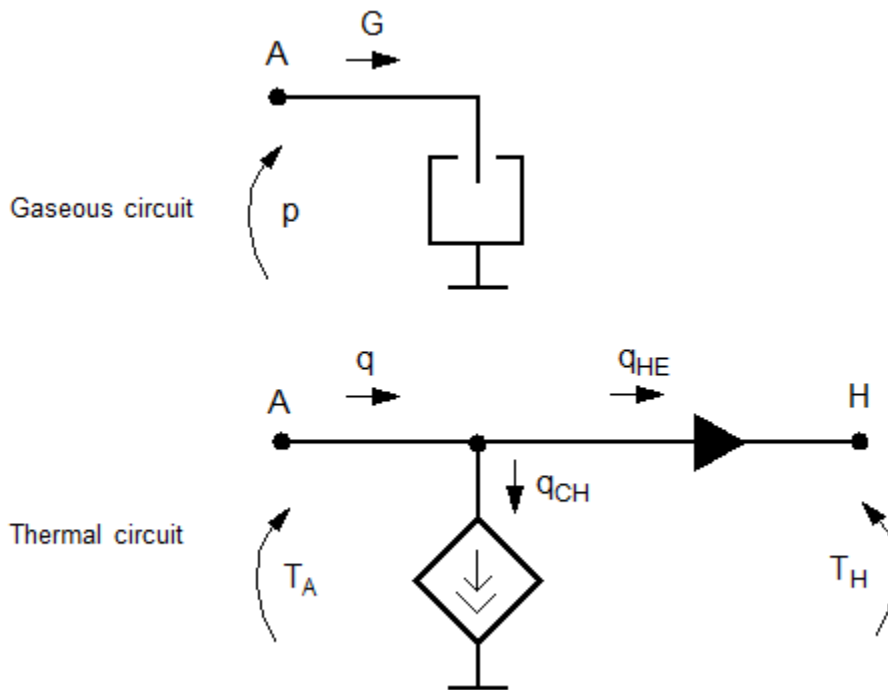
The continuity equation for the network representation of the constant chamber is

$$G = \frac{V}{RT} \left(\frac{dp}{dt} - \frac{p}{T} \frac{dT}{dt} \right)$$

where

G	Mass flow rate at input port
V	Chamber volume
p	Absolute pressure in the chamber
R	Specific gas constant
T	Absolute gas temperature
t	Time

The equivalent circuit of the Constant Volume Pneumatic Chamber block model is shown in the following illustration. Port A is the pneumatic conserving port associated with the chamber inlet. Port A connects both to the gaseous and the thermal circuit. Port H is a thermal conserving port through which heat exchange with the environment takes place. Port H connects only to the thermal circuit.



The diagram shows that the heat flow q to the chamber consists of two components:

- Heat flow q_{CH} , associated with the gaseous process
- Heat flow q_{HE} , associated with the heat exchange with the environment

The heat flow due to gas inflow is

$$q_{CH} = \frac{c_v V}{R} \cdot \frac{dp}{dt}$$

where c_v is specific heat at constant volume.

The heat exchange with the environment happens through port H, connected to thermal components. To determine the value of the heat exchange flow, the model contains a short-circuit element, resulting in the equation

$$T_A = T_H$$

where both T_A and T_H represent the gas temperature.

The gas flow and the heat flow are considered positive if they flow into the chamber.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see "Set Priority and Initial Target for Block Variables".

Basic Assumptions and Limitations

- The gas is ideal.
- Specific heats at constant pressure and constant volume, c_p and c_v , are constant.

Parameters

Chamber volume

Specify the volume of the chamber. The default value is $.001 \text{ m}^3$.

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the chamber inlet.

H

Thermal conserving port through which heat exchange with the environment takes place.

See Also

Pneumatic Piston Chamber | Rotary Pneumatic Piston Chamber

Introduced in R2009b

Controlled Current Source

Ideal current source driven by input signal



Library

Electrical Sources

Description

The Controlled Current Source block represents an ideal current source that is powerful enough to maintain the specified current through it regardless of the voltage across the source.

The output current is $I = I_s$, where I_s is the numerical value presented at the physical signal port.

The positive direction of the current flow is indicated by the arrow.

Ports

The block has one physical signal input port and two electrical conserving ports associated with its electrical terminals.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Voltage Source

Introduced in R2007a

Controlled Flux Source

Ideal flux source driven by input signal



Library

Magnetic Sources

Description

The Controlled Flux Source block represents an ideal flux source that is powerful enough to maintain the specified flux through it regardless of the mmf across the source.

The output flux is $\Phi = \Phi_{in}$, where Φ_{in} is the numerical value presented at the physical signal port.

The positive direction of the flux flow is indicated by the arrow.

Ports

The block has one physical signal input port and two magnetic conserving ports associated with its magnetic terminals.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Flux Source

Introduced in R2010a

Controlled Heat Flow Rate Source

Variable source of thermal energy, characterized by heat flow

Library: Simscape / Foundation Library / Thermal / Thermal Sources



Description

The Controlled Heat Flow Rate Source block represents an ideal source of thermal energy that is powerful enough to maintain specified heat flow at its outlet regardless of the temperature difference across the source.

Connections A and B are thermal conserving ports corresponding to the source inlet and outlet, respectively. Port S is a physical signal port, through which the control signal that drives the source is applied. You can use the entire variety of Simulink signal sources to generate the desired heat flow variation profile. The heat flow through the source is directly proportional to the signal at the control port S.

The block positive direction is from port A to port B. This means that positive signal at port S generates heat flow in the direction from A to B.

Ports

Input

S — Heat flow control signal, W

physical signal

Input physical signal that specifies the heat flow through the source.

Conserving

A — Source inlet

thermal

Thermal conserving port associated with the source inlet.

B — Source outlet

thermal

Thermal conserving port associated with the source outlet.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Heat Flow Rate Source

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007b

Controlled Mass Flow Rate Source (2P)

Generate time-varying mass flow rate



Library

Two-Phase Fluid/Sources

Description

The Controlled Mass Flow Rate Source (2P) block generates a variable mass flow rate in a two-phase fluid network branch. The source has two inlets, labeled **A** and **B**, with independently specified cross-sectional areas. By default, the source does isentropic work on the fluid, though the block provides the option to ignore this work.

The source is ideal. In other words, it maintains the specified flow rate regardless of the pressure differential produced between its ports. In addition, because the source is isentropic, there is no viscous friction between the ports and no heat exchange with the environment. Use this block to model an idealized pump or compressor or to set a boundary condition in a model.

Use physical signal port **M** to specify the desired mass flow rate. Use positive values for flows directed from port **A** to port **B** and negative values for flows directed from port **B** to port **A**.

Mass Balance

The volume of fluid in the source is considered negligible and is ignored in a model. There is no fluid accumulation between the ports and the sum of all mass flow rates into the source must therefore equal zero:

$$\dot{m}_A + \dot{m}_B = 0,$$

where \dot{m} denotes the mass flow rate into the source through a port. The block accepts as input the mass flow rate at port **A**. The flow is directed from port **A** to port **B** when the specified value is positive.

Energy Balance

By default, the source maintains the specified flow rate by performing isentropic work on the incoming fluid, though the block provides the option to ignore this term. The rate at which the source does work, if considered in the model, must equal the sum of the energy flow rates through the ports:

$$\phi_A + \phi_B + \phi_{\text{Work}} = 0,$$

where ϕ denotes the energy flow rate into the source through a port or by means of work. The energy flow rate due to work is equal to the power generated by the source. Its value is calculated from the specific total enthalpies at the ports:

$$\phi_{\text{Work}} = \dot{m}_A(h_A - h_B).$$

The specific total enthalpy h is defined as:

$$h_* = u_* + p_*v_* + \frac{1}{2}\left(\frac{\dot{m}_*v_*}{S}\right)^2,$$

where the asterisk denotes a port (**A** or **B**) and:

- u is specific internal energy.
- p is pressure.
- S is flow area.

The specific internal energy in the equation is obtained from the tabulated data of the Two-Phase Fluid Properties (2P) block. Its value is uniquely determined from the constraint that the work done by the source is isentropic. The specific entropy, a function of specific internal energy, must then have the same value at ports **A** and **B**:

$$s_A(p_A, u_A) = s_B(p_B, u_B),$$

where s is specific entropy. If the **Power added** parameter is set to **None**, the specific total enthalpies at the ports have the same value ($h_A = h_B$) and the work done by the source reduces to zero ($\phi_{\text{Work}} = 0$).

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

A — Inlet

two-phase fluid

Opening through fluid can enter and exit the source.

B — Inlet

two-phase fluid

Opening through fluid can enter and exit the source.

Input

M — Mass flow rate

physical signal

Value of the mass flow rate from port **A** to port **B**.

Parameters

Power added — Parameterization for the calculation of power

Isentropic power (default) | None

Parameterization for the calculation of power. Work is isentropic and its calculation is based on the assumptions of zero friction losses and zero heat exchange with the environment. Change to None to prevent the source from impacting the temperature of the fluid—for example, when using this block as a boundary condition in a model.

Cross-sectional area at port A — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of volume/time

Area of the fluid opening normal to the direction of flow.

Cross-sectional area at port B — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of volume/time

Area of the fluid opening normal to the direction of flow.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Volumetric Flow Rate Source (2P) | Mass Flow Rate Source (2P) | Volumetric Flow Rate Source (2P)

Introduced in R2015b

Controlled Mass Flow Rate Source (G)

Generate time-varying mass flow rate

Library: Simscape / Foundation Library / Gas / Sources



Description

The Controlled Mass Flow Rate Source (G) block represents an ideal mechanical energy source in a gas network. The mass flow rate is controlled by the input physical signal at port **M**. The source can maintain the specified mass flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. A positive mass flow rate causes gas to flow from port **A** to port **B**.

You can choose whether the source performs work on the gas flow:

- If the source is isentropic (**Power added** parameter is set to `Isentropic power`), then the isentropic relation depends on the gas property model.

Gas Model	Equations
Perfect gas	$\frac{(p_A)^{Z \cdot R/c_p}}{T_A} = \frac{(p_B)^{Z \cdot R/c_p}}{T_B}$
Semiperfect gas	$\int_0^{T_A} \frac{c_p(T)}{T} dT - Z \cdot R \cdot \ln(p_A) = \int_0^{T_B} \frac{c_p(T)}{T} dT - Z \cdot R \cdot \ln(p_B)$
Real gas	$s(T_A, p_A) = s(T_B, p_B)$

The power delivered to the gas flow is based on the specific total enthalpy associated with the isentropic process.

$$\Phi_{work} = -\dot{m}_A \left(h_A + \frac{w_A^2}{2} \right) - \dot{m}_B \left(h_B + \frac{w_B^2}{2} \right)$$

- If the source performs no work (**Power added** parameter is set to `None`), then the defining equation states that the specific total enthalpy is equal on both sides of the source. It is the same for all three gas property models.

$$h_A + \frac{w_A^2}{2} = h_B + \frac{w_B^2}{2}$$

The power delivered to the gas flow $\Phi_{work} = 0$.

The equations use these symbols:

c_p	Specific heat at constant pressure
h	Specific enthalpy
\dot{m}	Mass flow rate (flow rate associated with a port is positive when it flows into the block)
p	Pressure
R	Specific gas constant
s	Specific entropy
T	Temperature
w	Flow velocity
Z	Compressibility factor
Φ_{work}	Power delivered to the gas flow through the source

Subscripts A and B indicate the appropriate port.

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Input

M – Mass flow rate control signal, kg/s

physical signal

Input physical signal that specifies the mass flow rate of the gas through the source.

Conserving

A – Source inlet

gas

Gas conserving port. A positive mass flow rate causes gas to flow from port **A** to port **B**.

B – Source outlet

gas

Gas conserving port. A positive mass flow rate causes gas to flow from port **A** to port **B**.

Parameters

Power added – Select whether the source performs work

Isentropic power (default) | None

Select whether the source performs work on the gas flow:

- **Isentropic power** – The source performs isentropic work on the gas to maintain the specified mass flow rate, regardless of the pressure differential. Use this option to represent an idealized pump or compressor and properly account for the energy input and output, especially in closed-loop systems.

- **None** — The source performs no work on the flow, neither adding nor removing power, regardless of the mass flow rate produced by the source. Use this option to set up the desired flow condition upstream of the system, without affecting the temperature of the flow.

Cross-sectional area at port A — Area normal to flow path at port A

0.01 m² (default)

Area normal to flow path at port **A**.

Cross-sectional area at port B — Area normal to flow path at port B

0.01 m² (default)

Area normal to flow path at port **B**.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Mass Flow Rate Source (G)

Topics

“Modeling Gas Systems”

Introduced in R2016b

Controlled Mass Flow Rate Source (MA)

Generate time-varying mass flow rate

Library: Simscape / Foundation Library / Moist Air / Sources



Description

The Controlled Mass Flow Rate Source (MA) block represents an ideal mechanical energy source in a moist air network. The mass flow rate is controlled by the input physical signal at port **M**. The source can maintain the specified mass flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. A positive mass flow rate causes moist air to flow from port **A** to port **B**.

The equations describing the source use these symbols.

c_p	Specific heat at constant pressure
h	Specific enthalpy
h_t	Specific total enthalpy
\dot{m}	Mass flow rate (flow rate associated with a port is positive when it flows into the block)
p	Pressure
ρ	Density
R	Specific gas constant
s	Specific entropy
T	Temperature
Φ_{work}	Power delivered to the moist air flow through the source

Subscripts A and B indicate the appropriate port.

Mass balance:

$$\dot{m}_A + \dot{m}_B = 0$$

$$\dot{m}_{wA} + \dot{m}_{wB} = 0$$

$$\dot{m}_{gA} + \dot{m}_{gB} = 0$$

Energy balance:

$$\Phi_A + \Phi_B + \Phi_{work} = 0$$

If the source performs no work (**Power added** parameter is set to None), then $\Phi_{work} = 0$.

If the source is isentropic (**Power added** parameter is set to Isentropic power), then

$$\Phi_{work} = \dot{m}_A(h_{tB} - h_{tA})$$

where

$$h_{tA} = h_A + \frac{1}{2} \left(\frac{\dot{m}_A}{\rho_A S_A} \right)^2$$

$$h_{tB} = h_B + \frac{1}{2} \left(\frac{\dot{m}_B}{\rho_B S_B} \right)^2$$

The mixture-specific enthalpies, $h_A = h(T_A)$ and $h_B = h(T_B)$, are constrained by the isentropic relation, that is, there is no change in entropy:

$$\int_{T_A}^{T_B} \frac{1}{T} dh(T) = R \ln \left(\frac{p_B}{p_A} \right)$$

The quantity specified by the input signal of the source is

$$\dot{m}_A = \dot{m}_{\text{specified}}$$

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Input

M – Mass flow rate control signal, kg/s

physical signal

Input physical signal that specifies the mass flow rate of the moist air through the source.

Conserving

A – Source inlet

moist air

Moist air conserving port. A positive mass flow rate causes moist air to flow from port **A** to port **B**.

B – Source outlet

moist air

Moist air conserving port. A positive mass flow rate causes moist air to flow from port **A** to port **B**.

Parameters

Power added – Select whether the source performs work

Isentropic power (default) | None

Select whether the source performs work on the moist air flow:

- **Isentropic power** – The source performs isentropic work on the moist air to maintain the specified mass flow rate, regardless of the pressure differential. Use this option to represent an

idealized pump or compressor and properly account for the energy input and output, especially in closed-loop systems.

- **None** — The source performs no work on the flow, neither adding nor removing power, regardless of the mass flow rate produced by the source. Use this option to set up the desired flow condition upstream of the system, without affecting the temperature of the flow.

Cross-sectional area at port A — Area normal to flow path at port A

0.01 m² (default)

Area normal to flow path at port **A**.

Cross-sectional area at port B — Area normal to flow path at port B

0.01 m² (default)

Area normal to flow path at port **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mass Flow Rate Source (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Controlled Mass Flow Rate Source (TL)

Generate time-varying mass flow rate

Library: Simscape / Foundation Library / Thermal Liquid / Sources



Description

The Controlled Mass Flow Rate Source (TL) block represents an ideal mechanical energy source in a thermal liquid network. The mass flow rate is controlled by the input physical signal at port **M**. The source can maintain the specified mass flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. A positive mass flow rate causes the fluid to flow from port **A** to port **B**.

The energy balance at the source is a function of the energy flow rates through ports **A** and **B** and the work done on the fluid:

$$\phi_A + \phi_B + \phi_{work} = 0,$$

where:

- ϕ_A is the energy flow rate into the source through port **A**.
- ϕ_B is the energy flow rate into the source through port **B**.
- ϕ_{work} is the isentropic work done on the fluid.

The isentropic work term is

$$\phi_{work} = \frac{\dot{m}(p_B - p_A)}{\rho_{avg}},$$

where:

- ϕ_{work} is the isentropic work done on the thermal liquid.
- p_A is the pressure at port **A**.
- p_B is the pressure at port **B**.
- ρ_{avg} is the average liquid density,

$$\rho_{avg} = \frac{\rho_A + \rho_B}{2}.$$

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Input

M — Mass flow rate control signal, kg/s

physical signal

Input physical signal that specifies the mass flow rate through the source.

Conserving

A — Source inlet

thermal liquid

Thermal liquid conserving port. A positive mass flow rate causes the fluid to flow from port **A** to port **B**.

B — Source outlet

thermal liquid

Thermal liquid conserving port. A positive mass flow rate causes the fluid to flow from port **A** to port **B**.

Parameters

Cross-sectional area at ports A and B — Area normal to flow path at inlet and outlet

0.01 m² (default)

Area normal to the direction of flow at the source inlet and outlet. The two cross-sectional areas are assumed identical.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Pressure Source (TL) | Controlled Volumetric Flow Rate Source (TL) | Mass Flow Rate Source (TL) | Pressure Source (TL) | Volumetric Flow Rate Source (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2013b

Controlled MMF Source

Ideal magnetomotive force source driven by input signal



Library

Magnetic Sources

Description

The Controlled MMF Source block represents an ideal magnetomotive force (mmf) source that is powerful enough to maintain the specified mmf at its output regardless of the flux passing through it.

The output mmf is $MMF = MMFI$, where $MMFI$ is the numerical value presented at the physical signal port.

Ports

The block has one physical signal input port and two magnetic conserving ports associated with its magnetic terminals.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

MMF Source

Introduced in R2010a

Controlled Moisture Source (MA)

Inject or extract moisture at a time-varying rate

Library: Simscape / Foundation Library / Moist Air / Sources /
Moisture & Trace Gas Sources



Description

The Controlled Moisture Source (MA) block represents a time-varying source or sink of moisture for the connected moist air volume. Two physical signal input ports, **M** and **T**, supply the mass flow rate and temperature values, respectively. A positive or negative moisture mass flow rate results in moisture being added or removed, respectively.

You can add moisture as water vapor or liquid water. For water vapor, the energy associated with the added or removed moisture is

$$\Phi_s = \begin{cases} \dot{m}_{\text{specified}} \cdot h_w(T_{\text{specified}}), & \text{if } \dot{m}_{\text{specified}} \geq 0 \\ \dot{m}_{\text{specified}} \cdot h_w(T_s), & \text{if } \dot{m}_{\text{specified}} < 0 \end{cases}$$

where:

- $\dot{m}_{\text{specified}}$ is the water vapor mass flow rate specified by the input physical signal at port **M**.
- h_w is the water vapor specific enthalpy.
- $T_{\text{specified}}$ is the temperature of added moisture specified by the input physical signal at port **T**. The block uses this value to evaluate the specific enthalpy of the added moisture only. The specific enthalpy of removed moisture is based on the temperature of the connected moist air volume.
- T_s is the temperature at port **S**, which is the same as the temperature of the connected moist air volume.

For liquid water, the energy associated with the added or removed moisture is

$$\Phi_s = \begin{cases} \dot{m}_{\text{specified}} \cdot (h_w(T_{\text{specified}}) - \Delta h_{\text{vap}}(T_{\text{specified}})), & \text{if } \dot{m}_{\text{specified}} \geq 0 \\ \dot{m}_{\text{specified}} \cdot (h_w(T_s) - \Delta h_{\text{vap}}(T_s)), & \text{if } \dot{m}_{\text{specified}} < 0 \end{cases}$$

where Δh_{vap} is the water specific enthalpy of vaporization.

Port **S** is a moist air source conserving port. Connect this port to port **S** of a block with finite moist air volume to add or remove moisture through that block. For more information, see “Using Moisture and Trace Gas Sources”.

Ports

Input

M — Mass flow rate control signal, kg/s

physical signal

Input physical signal that specifies the water vapor mass flow rate through the source.

T — Temperature of added moisture, K

physical signal

Input physical signal that specifies the temperature of added moisture. The block uses this value to evaluate the specific enthalpy of the added moisture only. The specific enthalpy of removed moisture is based on the temperature of the connected moist air volume.

Conserving

S — Inject or extract moisture

moist air source

Connect this port to port **S** of a block with finite moist air volume to add or remove moisture through that block.

Parameters

Moisture added or removed — Select whether the source adds or removes moisture as water vapor or liquid water

Vapor (default) | Liquid

Select whether the source adds or removes moisture as water vapor or liquid water:

- **Vapor** — The enthalpy of the added or removed moisture corresponds to the enthalpy of water vapor, which is greater than that of liquid water.
- **Liquid** — The enthalpy of the added or removed moisture corresponds to the enthalpy of liquid water, which is less than that of water vapor.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Moisture Source (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Controlled Pneumatic Flow Rate Source

Ideal compressor with signal-controlled mass flow rate



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Controlled Pneumatic Flow Rate Source block represents an ideal compressor that maintains a mass flow rate equal to the numerical value presented at physical signal port F. The compressor adds no heat. Block connections A and B correspond to the pneumatic inlet and outlet ports, respectively, and connection F represents a control signal port.

The block positive direction is from port A to port B. This means that the flow rate is positive if it flows from A to B. The pressure differential is determined as $p = p_A - p_B$ and is negative if pressure at the source outlet is greater than pressure at its inlet. The power generated by the source is negative if the source adds energy to the flow.

Warning Be careful when driving an orifice directly from a flow rate source. The choked flow condition limits the flow that is possible through an orifice as a function of upstream pressure and temperature. Hence the flow rate value produced by the flow rate source must be compatible with upstream pressure and temperature. Specifying a flow rate that is too high will result in an unsolvable set of equations.

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the source inlet.

B

Pneumatic conserving port associated with the source outlet.

F

Control signal port.

See Also

Pneumatic Flow Rate Source | Pneumatic Mass & Heat Flow Sensor

Introduced in R2009b

Controlled Pneumatic Pressure Source

Ideal compressor with signal-controlled pressure difference



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Controlled Pneumatic Pressure Source block represents an ideal compressor that maintains a pressure difference equal to the numerical value presented at physical signal port F. The compressor adds no heat. Block connections A and B correspond to the pneumatic inlet and outlet ports, respectively, and connection F represents a control signal port.

A positive pressure difference results in the pressure at port B being higher than the pressure at port A.

Ports

The block has the following ports:

- A
Pneumatic conserving port associated with the source inlet.
- B
Pneumatic conserving port associated with the source outlet.
- F
Control signal port.

See Also

Pneumatic Pressure & Temperature Sensor | Pneumatic Pressure Source

Introduced in R2009b

Controlled Pressure Source (2P)

Generate time-varying pressure differential



Library

Two-Phase Fluid/Sources

Description

The Controlled Pressure Source (2P) block generates a variable pressure differential in a two-phase fluid network branch. The source has two inlets, labeled **A** and **B**, with independently specified cross-sectional areas. By default, the source does isentropic work on the fluid, though the block provides the option to ignore this work.

The source is ideal. In other words, it maintains the specified pressure differential regardless of the mass flow rate produced through its ports. In addition, because the source is isentropic, there is no viscous friction between the ports and no heat exchange with the environment. Use this block to model an idealized pump or compressor or to set a boundary condition in a model.

Use physical signal port **P** to specify the desired pressure differential. Use positive values for pressures that increase from port **A** to port **B** and negative value for pressures that increase from port **B** to port **A**.

Mass Balance

The volume of fluid in the source is considered negligible and is ignored in a model. There is no fluid accumulation between the ports and the sum of all mass flow rates into the source must therefore equal zero:

$$\dot{m}_A + \dot{m}_B = 0,$$

where \dot{m} denotes the mass flow rate into the source through a port. The block accepts as input the mass flow rate at port **A**. The flow is directed from port **A** to port **B** when the specified value is positive.

Energy Balance

By default, the source maintains the specified flow rate by performing isentropic work on the incoming fluid, though the block provides the option to ignore this term. The rate at which the source does work, if considered in the model, must equal the sum of the energy flow rates through the ports:

$$\phi_A + \phi_B + \phi_{\text{Work}} = 0,$$

where ϕ denotes the energy flow rate into the source through a port or by means of work. The energy flow rate due to work is equal to the power generated by the source. Its value is calculated from the specific total enthalpies at the ports:

$$\phi_{\text{Work}} = \dot{m}_A(h_A - h_B).$$

The specific total enthalpy h is defined as:

$$h_* = u_* + p_*v_* + \frac{1}{2} \left(\frac{\dot{m}_* v_*}{S} \right)^2,$$

where the asterisk denotes a port (**A** or **B**) and:

- u is specific internal energy.
- p is pressure.
- S is flow area.

The specific internal energy in the equation is obtained from the tabulated data of the Two-Phase Fluid Properties (2P) block. Its value is uniquely determined from the constraint that the work done by the source is isentropic. The specific entropy, a function of specific internal energy, must then have the same value at ports **A** and **B**:

$$s_A(p_A, u_A) = s_B(p_B, u_B),$$

where s is specific entropy. If the **Power added** parameter is set to **None**, the specific total enthalpies at the ports have the same value ($h_A = h_B$) and the work done by the source reduces to zero ($\phi_{\text{Work}} = 0$).

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

A — Inlet

two-phase fluid

Opening through fluid can enter and exit the source.

B — Inlet

two-phase fluid

Opening through fluid can enter and exit the source.

Input

P — Pressure differential

physical signal

Value of the pressure gain from port **A** to port **B**.

Parameters

Power added — Parameterization for the calculation of power

Isentropic power (default) | None

Parameterization for the calculation of power. Work is isentropic and its calculation is based on the assumptions of zero friction losses and zero heat exchange with the environment. Change to None to prevent the source from impacting the temperature of the fluid—for example, when using this block as a boundary condition in a model.

Cross-sectional area at port A — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of volume/time

Area of the fluid opening normal to the direction of flow.

Cross-sectional area at port B — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of volume/time

Area of the fluid opening normal to the direction of flow.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Pressure Source (2P)

Introduced in R2015b

Controlled Pressure Source (G)

Generate time-varying pressure differential

Library: Simscape / Foundation Library / Gas / Sources



Description

The Controlled Pressure Source (G) block represents an ideal mechanical energy source in a gas network. The pressure differential is controlled by the input physical signal at port **P**. The source can maintain the specified pressure differential across its ports regardless of the mass flow rate through the source. There is no flow resistance and no heat exchange with the environment. A positive signal at port **P** causes the pressure at port **B** to be greater than the pressure at port **A**.

You can choose whether the source performs work on the gas flow:

- If the source is isentropic (**Power added** parameter is set to `Isentropic power`), then the isentropic relation depends on the gas property model.

Gas Model	Equations
Perfect gas	$\frac{(p_A)^{Z \cdot R/c_p}}{T_A} = \frac{(p_B)^{Z \cdot R/c_p}}{T_B}$
Semiperfect gas	$\int_0^{T_A} \frac{c_p(T)}{T} dT - Z \cdot R \cdot \ln(p_A) = \int_0^{T_B} \frac{c_p(T)}{T} dT - Z \cdot R \cdot \ln(p_B)$
Real gas	$s(T_A, p_A) = s(T_B, p_B)$

The power delivered to the gas flow is based on the specific total enthalpy associated with the isentropic process.

$$\Phi_{work} = -\dot{m}_A \left(h_A + \frac{w_A^2}{2} \right) - \dot{m}_B \left(h_B + \frac{w_B^2}{2} \right)$$

- If the source performs no work (**Power added** parameter is set to `None`), then the defining equation states that the specific total enthalpy is equal on both sides of the source. It is the same for all three gas property models.

$$h_A + \frac{w_A^2}{2} = h_B + \frac{w_B^2}{2}$$

The power delivered to the gas flow $\Phi_{work} = 0$.

The equations use these symbols:

c_p	Specific heat at constant pressure
h	Specific enthalpy
\dot{m}	Mass flow rate (flow rate associated with a port is positive when it flows into the block)
p	Pressure
R	Specific gas constant
s	Specific entropy
T	Temperature
w	Flow velocity
Z	Compressibility factor
Φ_{work}	Power delivered to the gas flow through the source

Subscripts A and B indicate the appropriate port.

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Input

P — Pressure differential control signal, Pa

physical signal

Input physical signal that specifies the pressure differential of the gas across the source. A positive signal at port **P** causes the pressure at port **B** to be greater than the pressure at port **A**.

Conserving

A — Source inlet

gas

Gas conserving port. A positive signal at port **P** causes the pressure at port **B** to be greater than the pressure at port **A**.

B — Source outlet

gas

Gas conserving port. A positive signal at port **P** causes the pressure at port **B** to be greater than the pressure at port **A**.

Parameters

Power added — Select whether the source performs work

Isentropic power (default) | None

Select whether the source performs work on the gas flow:

- **Isentropic power** — The source performs isentropic work on the gas to maintain the specified pressure differential, regardless of the mass flow rate. Use this option to represent an idealized pump or compressor and properly account for the energy input and output, especially in closed-loop systems.
- **None** — The source performs no work on the flow, neither adding nor removing power, regardless of the pressure differential produced by the source. Use this option to set up the desired flow condition upstream of the system, without affecting the temperature of the flow.

Cross-sectional area at port A — Area normal to flow path at port A

0.01 m² (default)

Area normal to flow path at port **A**.

Cross-sectional area at port B — Area normal to flow path at port B

0.01 m² (default)

Area normal to flow path at port **B**.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Pressure Source (G)

Topics

“Modeling Gas Systems”

Introduced in R2016b

Controlled Pressure Source (MA)

Generate time-varying pressure differential

Library: Simscape / Foundation Library / Moist Air / Sources



Description

The Controlled Pressure Source (MA) block represents an ideal mechanical energy source in a moist air network. The pressure differential is controlled by the input physical signal at port **P**. The source can maintain the specified pressure differential across its ports regardless of the mass flow rate through the source. There is no flow resistance and no heat exchange with the environment. A positive signal at port **P** causes the pressure at port **B** to be greater than the pressure at port **A**.

The equations describing the source use these symbols.

c_p	Specific heat at constant pressure
h	Specific enthalpy
h_t	Specific total enthalpy
\dot{m}	Mass flow rate (flow rate associated with a port is positive when it flows into the block)
p	Pressure
ρ	Density
R	Specific gas constant
s	Specific entropy
T	Temperature
Φ_{work}	Power delivered to the moist air flow through the source

Subscripts A and B indicate the appropriate port.

Mass balance:

$$\dot{m}_A + \dot{m}_B = 0$$

$$\dot{m}_{wA} + \dot{m}_{wB} = 0$$

$$\dot{m}_{gA} + \dot{m}_{gB} = 0$$

Energy balance:

$$\Phi_A + \Phi_B + \Phi_{work} = 0$$

If the source performs no work (**Power added** parameter is set to None), then $\Phi_{work} = 0$.

If the source is isentropic (**Power added** parameter is set to Isentropic power), then

$$\Phi_{work} = \dot{m}_A(h_{tB} - h_{tA})$$

where

$$h_{tA} = h_A + \frac{1}{2} \left(\frac{\dot{m}_A}{\rho_A S_A} \right)^2$$

$$h_{tB} = h_B + \frac{1}{2} \left(\frac{\dot{m}_B}{\rho_B S_B} \right)^2$$

The mixture-specific enthalpies, $h_A = h(T_A)$ and $h_B = h(T_B)$, are constrained by the isentropic relation, that is, there is no change in entropy:

$$\int_A^{T_B} \frac{1}{T} dh(T) = R \ln \left(\frac{p_B}{p_A} \right)$$

The quantity specified by the input signal of the source is

$$p_B - p_A = \Delta p_{\text{specified}}$$

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Input

P – Pressure differential control signal, Pa

physical signal

Input physical signal that specifies the pressure differential of the moist air mixture across the source. A positive signal at port **P** causes the pressure at port **B** to be greater than the pressure at port **A**.

Conserving

A – Source inlet

moist air

Moist air conserving port. A positive signal at port **P** causes the pressure at port **B** to be greater than the pressure at port **A**.

B – Source outlet

moist air

Moist air conserving port. A positive signal at port **P** causes the pressure at port **B** to be greater than the pressure at port **A**.

Parameters

Power added – Select whether the source performs work

Isentropic power (default) | None

Select whether the source performs work on the moist air flow:

- **Isentropic power** — The source performs isentropic work on the moist air to maintain the specified mass flow rate, regardless of the pressure differential. Use this option to represent an idealized pump or compressor and properly account for the energy input and output, especially in closed-loop systems.
- **None** — The source performs no work on the flow, neither adding nor removing power, regardless of the mass flow rate produced by the source. Use this option to set up the desired flow condition upstream of the system, without affecting the temperature of the flow.

Cross-sectional area at port A — Area normal to flow path at port A

0.01 m² (default)

Area normal to flow path at port A.

Cross-sectional area at port B — Area normal to flow path at port B

0.01 m² (default)

Area normal to flow path at port B.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Pressure Source (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Controlled Pressure Source (TL)

Generate time-varying pressure differential

Library: Simscape / Foundation Library / Thermal Liquid / Sources



Description

The Controlled Pressure Source (TL) block represents an ideal mechanical energy source in a thermal liquid network. The pressure differential is controlled by the input physical signal at port **P**. The source can maintain the specified pressure differential across its ports regardless of the mass flow rate through the source. There is no flow resistance and no heat exchange with the environment. A positive signal at port **P** causes the pressure at port **B** to be greater than the pressure at port **A**.

The energy balance at the source is a function of the energy flow rates through ports **A** and **B** and the work done on the fluid:

$$\phi_A + \phi_B + \phi_{work} = 0,$$

where:

- ϕ_A is the energy flow rate into the source through port **A**.
- ϕ_B is the energy flow rate into the source through port **B**.
- ϕ_{work} is the isentropic work done on the fluid.

The isentropic work term is

$$\phi_{work} = \frac{\dot{m}(p_B - p_A)}{\rho_{avg}},$$

where:

- ϕ_{work} is the isentropic work done on the thermal liquid.
- p_A is the pressure at port **A**.
- p_B is the pressure at port **B**.
- ρ_{avg} is the average liquid density,

$$\rho_{avg} = \frac{\rho_A + \rho_B}{2}.$$

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Input

P — Pressure differential control signal, Pa

physical signal

Input physical signal that specifies the pressure differential across the source.

Conserving

A — Source inlet

thermal liquid

Thermal liquid conserving port. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

B — Source outlet

thermal liquid

Thermal liquid conserving port. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

Parameters

Cross-sectional area at ports A and B — Area normal to flow path at inlet and outlet

0.01 m² (default)

Area normal to the direction of flow at the source inlet and outlet. The two cross-sectional areas are assumed identical.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Mass Flow Rate Source (TL) | Controlled Volumetric Flow Rate Source (TL) | Mass Flow Rate Source (TL) | Pressure Source (TL) | Volumetric Flow Rate Source (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2013b

Controlled Reservoir (2P)

Two-phase fluid reservoir at variable pressure and temperature



Library

Two-Phase Fluid/Elements

Description

The Reservoir (2P) block sets boundary conditions in a two-phase fluid network. Those conditions are specified in terms of pressure and—depending on the **Reservoir energy specification** setting—either specific enthalpy or specific internal energy. The reservoir is assumed infinite in size, causing its pressure and specific internal energy to remain constant.

Port **A** represents the reservoir inlet. The flow resistance between port A and the reservoir interior is assumed negligible. The pressure at port A is therefore equal to that inside the reservoir.

Port **P** sets the absolute pressure in the reservoir (in units of MPa). Ports **H** and **U** set the specific enthalpy and specific internal energy, respectively (both in units of kJ/kg). Which of these two ports is active depends on the energy specification setting of the block.

The specific enthalpy and specific internal energy at the reservoir inlet depend on the direction of flow. If fluid leaves the reservoir, these variables are the same as inside the reservoir. The reservoir then acts as an energy source. If fluid enters the reservoir, these variables are the same as upstream of the reservoir. The reservoir acts as an energy sink.

This block also serves as a reference connection for the Pressure & Internal Energy Sensor (2P) block. In this case, the measured pressure and specific internal energy are relative to the reservoir pressure and specific internal energy.

Assumptions and Limitations

- The flow resistance between port A and the reservoir interior is negligible. Pressure is the same at port A and in the reservoir interior.

Parameters

Reservoir energy specification

Measure of energy to specify during simulation. The default setting is **Specific enthalpy**. Changing this setting changes port **H** to port **U**. Use the active port to specify the chosen measure of energy.

Cross-sectional area at port A

Flow area of the reservoir inlet, represented by port **A**. The flow area must be greater than zero. The default value is 0.01 m^2 .

Ports

The block has four ports:

- **A** — Two-phase fluid port representing the inlet of the reservoir.
- **P** — Physical signal input port for the (absolute) pressure of the fluid in the reservoir.
- **H** — Physical signal input port for the specific enthalpy of the fluid in the reservoir. Whether this port is active depends on the **Reservoir energy specification** setting of the block.
- **U** — Physical signal input port for the specific internal energy of the fluid in the reservoir. Whether this port is active depends on the **Reservoir energy specification** setting of the block.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Constant Volume Chamber (2P) | Reservoir (2P)

Introduced in R2015b

Controlled Reservoir (G)

Boundary conditions for gas network at time-varying pressure and temperature

Library: Simscape / Foundation Library / Gas / Elements



Description

The Controlled Reservoir (G) block represents an infinite reservoir at variable pressure and temperature. Port **A**, a gas conserving port, represents the reservoir inlet. Port **P**, a physical signal port, provides the reservoir pressure control signal. Port **T**, a physical signal port, provides the reservoir temperature control signal.

The volume of gas inside the reservoir is assumed infinite. Therefore, the flow is assumed quasi-steady.

Gas enters and leaves the reservoir at the reservoir pressure, but its temperature is determined by the direction of gas flow. If gas flows into the reservoir, its temperature is determined by the gas network upstream. The reservoir acts as a heat sink. If gas flows out of the reservoir, its temperature equals that of the reservoir. The reservoir acts as a heat source.

Assumptions and Limitations

- Gas in the reservoir is quasi-steady.

Ports

Input

P — Pressure control signal, Pa

physical signal

Physical signal port that provides the reservoir pressure control signal.

T — Temperature control signal, K

physical signal

Physical signal port that provides the reservoir temperature control signal.

Conserving

A — Reservoir inlet

gas

Gas conserving port associated with the reservoir inlet.

Parameters

Cross-sectional area at port A — Area normal to flow path at the reservoir inlet

0.01 m² (default)

The cross-sectional area of the reservoir inlet, in the direction normal to gas flow path.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Reservoir (G)

Topics

“Modeling Gas Systems”

Introduced in R2016b

Controlled Reservoir (MA)

Boundary conditions for moist air network at time-varying pressure, temperature, moisture, and trace gas levels

Library: Simscape / Foundation Library / Moist Air / Elements



Description

The Controlled Reservoir (MA) block sets controlled boundary conditions in a moist air network. The volume of moist air inside the reservoir is assumed infinite. Therefore, the flow is assumed quasi-steady. Moist air leaves the reservoir at the reservoir pressure, temperature, specific humidity, and trace gas mass fraction. Moist air enters the reservoir at the reservoir pressure, but the temperature, specific humidity, and trace gas mass fraction are determined by the moist air network upstream.

You specify the reservoir pressure, temperature, amount of moisture, and amount of trace gas by control physical signals at ports **P**, **T**, **W**, and **G**, respectively. The inputs are limited by their valid ranges. For pressure and temperature, the valid range is between the minimum and maximum values specified in the Moist Air Properties (MA) block connected to the circuit. For the amount of moisture, the valid range is between zero and either saturation or 100 percent water vapor. For the amount of trace gas, the valid range is between zero and either the fraction left over after water vapor or 100 percent trace gas. The input **G** is ignored if **Trace gas model** in the Moist Air Properties (MA) block is set to None.

You can specify moisture as one of:

- Relative humidity, ϕ_w
- Specific humidity, x_w
- Water vapor mole fraction, y_w
- Humidity ratio, r_w

You can specify trace gas as one of:

- Trace gas mass fraction, x_g
- Trace gas mole fraction, y_g

These moisture and trace gas quantities are related to each other as follows:

$$\phi_w = \frac{y_w p}{p_{ws}}$$

$$y_w = \frac{x_w R_w}{R}$$

$$r_w = \frac{x_w}{1 - x_w}$$

$$y_g = \frac{x_g R_g}{R}$$

$$x_a + x_w + x_g = 1$$

$$R = x_a R_a + x_w R_w + x_g R_g$$

where:

- p is pressure.
- R is specific gas constant.

Subscripts a, w, and g indicate the properties of dry air, water vapor, and trace gas, respectively. Subscript ws indicates water vapor at saturation.

Ports

Input

P – Pressure control signal, Pa

physical signal

Physical signal port that provides the reservoir pressure control signal.

T – Temperature control signal, K

physical signal

Physical signal port that provides the reservoir temperature control signal.

W – Moisture control signal, unitless

physical signal

Physical signal port that controls the reservoir moisture level. To select the quantity that the control signal represents, use the **Reservoir moisture specification** parameter.

G – Trace gas control signal, unitless

physical signal

Physical signal port that controls the reservoir trace gas level. To select the quantity that the control signal represents, use the **Reservoir trace gas specification** parameter.

Conserving

A – Reservoir inlet

moist air

Moist air conserving port associated with the reservoir inlet.

Parameters

Reservoir moisture specification — Select the moisture property that the control signal represents

Relative humidity (default) | Specific humidity | Mole fraction | Humidity ratio

Select a moisture property:

- Relative humidity — Physical signal at port **W** specifies the relative humidity.
- Specific humidity — Physical signal at port **W** specifies the specific humidity.
- Mole fraction — Physical signal at port **W** specifies the water vapor mole fraction.
- Humidity ratio — Physical signal at port **W** specifies the humidity ratio.

Reservoir trace gas specification — Select the trace gas property that the control signal represents

Mass fraction (default) | Mole fraction

Select a trace gas property:

- Mass fraction — Physical signal at port **G** specifies the trace gas mass fraction.
- Mole fraction — Physical signal at port **G** specifies the trace gas mole fraction.

Relative humidity at saturation — Relative humidity above which condensation occurs

1 (default)

Relative humidity above which condensation occurs. Amount of moisture in the reservoir must be less than saturation.

Cross-sectional area at port A — Area normal to flow path at the reservoir inlet

0.01 m² (default)

The cross-sectional area of the reservoir inlet.

Inputs outside valid range — Select what happens when the input signal values are outside of valid range

Warn and limit to valid values (default) | Limit to valid values | Error

Select what happens when the input signal values are outside of valid range:

- Limit to valid values — The block uses the minimum or maximum valid values, but does not issue a warning.
- Warn and limit to valid values — The block issues a warning and uses the corresponding minimum or maximum valid values.
- Error — Simulation stops with an error.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Reservoir (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Controlled Reservoir (TL)

Thermal liquid reservoir at time-varying temperature

Library: Simscape / Foundation Library / Thermal Liquid / Elements



Description

The Controlled Reservoir (TL) block represents an infinite reservoir at fixed pressure and variable temperature. The reservoir and its inlet can be at atmospheric pressure or at a specified pressure. Port **A**, a thermal liquid conserving port, represents the reservoir inlet. Port **T**, a physical signal port, provides the reservoir temperature control signal.

The inlet temperature depends on the direction of liquid flow. If the liquid flows into the reservoir, the inlet temperature equals that of the upstream liquid and the reservoir acts as a heat sink. If liquid flows out of the reservoir, the inlet temperature equals that of the reservoir and the reservoir acts as a heat source.

To ensure a smooth temperature change at the reservoir inlet during liquid flow reversal, the block includes heat conduction along a length equal to the effective diameter of the inlet pipe. This diameter is a function of the specified cross-sectional area of the inlet pipe.

Ports

Input

T — Temperature control signal, K

physical signal

Physical signal port that provides the reservoir temperature control signal.

Conserving

A — Reservoir inlet

thermal liquid

Thermal liquid conserving port associated with the reservoir inlet.

Parameters

Reservoir pressure specification — Specification method for reservoir pressure

Atmospheric pressure (default) | Specified pressure

Specification method for the reservoir pressure:

- **Atmospheric pressure** — Use the atmospheric pressure specified by a Thermal Liquid Settings (TL) or Thermal Liquid Properties (TL) block connected to the circuit.

- **Specified pressure** — Specify a value by using the **Reservoir pressure** parameter.

Reservoir pressure — Pressure in reservoir

0.101325 MPa (default)

Desired pressure in the reservoir. This pressure remains constant during simulation.

Dependencies

Enabled when the **Reservoir pressure specification** parameter is set to **Specified pressure**.

Cross-sectional area at port A — Area normal to flow path at reservoir inlet

0.01 m² (default)

Cross-sectional area of the reservoir inlet pipe. The block uses this area to determine the characteristic length of the pipe along which heat conduction occurs.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Constant Volume Chamber (TL) | Reservoir (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2015a

Controlled Temperature Source

Variable source of thermal energy, characterized by temperature

Library: Simscape / Foundation Library / Thermal / Thermal Sources



Description

The Controlled Temperature Source block represents an ideal source of thermal energy that is powerful enough to maintain specified temperature difference across the source regardless of the heat flow consumed by the system.

Connections A and B are thermal conserving ports corresponding to the source inlet and outlet, respectively. Port S is a physical signal port, through which the control signal that drives the source is applied. You can use the entire variety of Simulink signal sources to generate the desired heat flow variation profile. The temperature differential across the source is directly proportional to the signal at the control port S.

The block positive direction is from port A to port B. This means that the temperature differential is determined as $T_B - T_A$, where T_B and T_A are the temperatures at source ports.

Ports

Input

S — Temperature control signal, K

physical signal

Input physical signal that specifies the temperature difference across the source.

Conserving

A — Source inlet

thermal

Thermal conserving port associated with the source inlet.

B — Source outlet

thermal

Thermal conserving port associated with the source outlet.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Temperature Source

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007b

Controlled Trace Gas Source (MA)

Inject or extract trace gas at a time-varying rate

Library: Simscape / Foundation Library / Moist Air / Sources /
Moisture & Trace Gas Sources



Description

The Controlled Trace Gas Source (MA) block represents a time-varying source or sink of trace gas for the connected moist air volume. Two physical signal input ports, **M** and **T**, supply the mass flow rate and temperature values, respectively. A positive or negative trace gas mass flow rate causes trace gas levels to increase or decrease, respectively.

The energy associated with the added or removed trace gas is

$$\Phi_s = \begin{cases} \dot{m}_{\text{specified}} \cdot h_g(T_{\text{specified}}), & \text{if } \dot{m}_{\text{specified}} \geq 0 \\ \dot{m}_{\text{specified}} \cdot h_g(T_s), & \text{if } \dot{m}_{\text{specified}} < 0 \end{cases}$$

where:

- $\dot{m}_{\text{specified}}$ is the trace gas mass flow rate specified by the input physical signal at port **M**.
- h_g is the trace gas specific enthalpy.
- $T_{\text{specified}}$ is the temperature of added trace gas specified by the input physical signal at port **T**. The block uses this value to evaluate the specific enthalpy of the added trace gas only. The specific enthalpy of removed trace gas is based on the temperature of the connected moist air volume.
- T_s is the temperature at port **S**, which is the same as the temperature of the connected moist air volume.

Port **S** is a moist air source conserving port. Connect this port to port **S** of a block with finite moist air volume to add or remove trace gas through that block. For more information, see “Using Moisture and Trace Gas Sources”.

Ports

Input

M — Mass flow rate control signal, kg/s
physical signal

Input physical signal that specifies the trace gas mass flow rate through the source.

T — Temperature of added trace gas, K
physical signal

Input physical signal that specifies the temperature of added trace gas. The block uses this value to evaluate the specific enthalpy of the added trace gas only. The specific enthalpy of removed trace gas is based on the temperature of the connected moist air volume.

Conserving

S — Inject or extract trace gas

moist air source

Connect this port to port **S** of a block with finite moist air volume to add or remove trace gas through that block.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Trace Gas Source (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Controlled Voltage Source

Ideal voltage source driven by input signal



Library

Electrical Sources

Description

The Controlled Voltage Source block represents an ideal voltage source that is powerful enough to maintain the specified voltage at its output regardless of the current flowing through the source.

The output voltage is $V = V_s$, where V_s is the numerical value presented at the physical signal port.

Ports

The block has one physical signal input port and two electrical conserving ports associated with its electrical terminals.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Current Source

Introduced in R2007a

Controlled Volumetric Flow Rate Source (2P)

Generate time-varying volumetric flow rate



Library

Thermal Liquid/Sources

Description

The Controlled Volumetric Flow Rate Source (2P) block generates a variable volumetric flow rate in a two-phase fluid network branch. The source has two inlets, labeled **A** and **B**, with independently specified cross-sectional areas. By default, the source does isentropic work on the fluid, though the block provides the option to ignore this work.

The source is ideal. In other words, it maintains the specified flow rate regardless of the pressure differential produced between its ports. In addition, because the source is isentropic, there is no viscous friction between the ports and no heat exchange with the environment. Use this block to model an idealized pump or compressor or to set a boundary condition in a model.

Use physical signal port **V** to specify the desired volumetric flow rate. Use positive values for flows directed from port **A** to port **B** and negative values for flows directed from port **B** to port **A**.

Mass Balance

The volume of fluid in the source is considered negligible and is ignored in a model. There is no fluid accumulation between the ports and the sum of all mass flow rates into the source must therefore equal zero:

$$\dot{m}_A + \dot{m}_B = 0,$$

where \dot{m} denotes the mass flow rate into the source through a port. Its value at port **A** is calculated from the specified volumetric flow rate:

$$\dot{m}_A = \begin{cases} \frac{\dot{V}}{v_B}, & \text{if } \dot{V} \geq 0 \\ \frac{\dot{V}}{v_A}, & \text{otherwise} \end{cases},$$

where \dot{V} is volumetric flow rate and v is specific volume.

Energy Balance

By default, the source maintains the specified flow rate by performing isentropic work on the incoming fluid, though the block provides the option to ignore this term. The rate at which the source does work, if considered in the model, must equal the sum of the energy flow rates through the ports:

$$\phi_A + \phi_B + \phi_{\text{Work}} = 0,$$

where ϕ denotes the energy flow rate into the source through a port or by means of work. The energy flow rate due to work is equal to the power generated by the source. Its value is calculated from the specific total enthalpies at the ports:

$$\phi_{\text{Work}} = \dot{m}_A(h_A - h_B).$$

The specific total enthalpy h is defined as:

$$h_* = u_* + p_*v_* + \frac{1}{2} \left(\frac{\dot{m}_*v_*}{S} \right)^2,$$

where the asterisk denotes a port (**A** or **B**) and:

- u is specific internal energy.
- p is pressure.
- S is flow area.

The specific internal energy in the equation is obtained from the tabulated data of the Two-Phase Fluid Properties (2P) block. Its value is uniquely determined from the constraint that the work done by the source is isentropic. The specific entropy, a function of specific internal energy, must then have the same value at ports **A** and **B**:

$$s_A(p_A, u_A) = s_B(p_B, u_B),$$

where s is specific entropy. If the **Power added** parameter is set to **None**, the specific total enthalpies at the ports have the same value ($h_A = h_B$) and the work done by the source reduces to zero ($\phi_{\text{Work}} = 0$).

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

A – Fluid opening

two-phase fluid

Opening through fluid can enter and exit the source.

B – Fluid opening

two-phase fluid

Opening through fluid can enter and exit the source.

Input

V – Volumetric flow rate

physical signal

Value of the volumetric flow rate from port **A** to port **B**.

Parameters

Power added — Parameterization for the calculation of power

Isentropic power (default) | None

Parameterization for the calculation of power. Work is isentropic and its calculation is based on the assumptions of zero friction losses and zero heat exchange with the environment. Change to **None** to prevent the source from impacting the temperature of the fluid—for example, when using this block as a boundary condition in a model.

Cross-sectional area at port A — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of volume/time

Area of the fluid opening normal to the direction of flow.

Cross-sectional area at port B — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of volume/time

Area of the fluid opening normal to the direction of flow.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Mass Flow Rate Source (2P) | Mass Flow Rate Source (2P) | Volumetric Flow Rate Source (2P)

Introduced in R2015b

Controlled Volumetric Flow Rate Source (G)

Generate time-varying volumetric flow rate

Library: Simscape / Foundation Library / Gas / Sources



Description

The Controlled Volumetric Flow Rate Source (G) block represents an ideal mechanical energy source in a gas network. The volumetric flow rate is controlled by the input physical signal at port **V**. The source can maintain the specified volumetric flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. A positive volumetric flow rate causes gas to flow from port **A** to port **B**.

The volumetric flow rate and mass flow rate are related through the expression

$$\dot{m} = \begin{cases} \rho_B \dot{V} & \text{for } \dot{V} \geq 0 \\ \rho_A \dot{V} & \text{for } \dot{V} < 0 \end{cases}$$

where:

- \dot{m} is the mass flow rate from port **A** to port **B**.
- ρ_A and ρ_B are densities at ports **A** and **B**, respectively.
- \dot{V} is the volumetric flow rate.

You can choose whether the source performs work on the gas flow:

- If the source is isentropic (**Power added** parameter is set to **Isentropic power**), then the isentropic relation depends on the gas property model.

Gas Model	Equations
Perfect gas	$\frac{(p_A)^{Z \cdot R/c_p}}{T_A} = \frac{(p_B)^{Z \cdot R/c_p}}{T_B}$
Semiperfect gas	$\int_{T_A}^{T_B} \frac{c_p(T)}{T} dT - Z \cdot R \cdot \ln(p_A) = \int_{T_B}^{T_C} \frac{c_p(T)}{T} dT - Z \cdot R \cdot \ln(p_B)$
Real gas	$s(T_A, p_A) = s(T_B, p_B)$

The power delivered to the gas flow is based on the specific total enthalpy associated with the isentropic process.

$$\Phi_{work} = -\dot{m}_A \left(h_A + \frac{w_A^2}{2} \right) - \dot{m}_B \left(h_B + \frac{w_B^2}{2} \right)$$

- If the source performs no work (**Power added** parameter is set to **None**), then the defining equation states that the specific total enthalpy is equal on both sides of the source. It is the same for all three gas property models.

$$h_A + \frac{w_A^2}{2} = h_B + \frac{w_B^2}{2}$$

The power delivered to the gas flow $\Phi_{\text{work}} = 0$.

The equations use these symbols:

c_p	Specific heat at constant pressure
h	Specific enthalpy
\dot{m}	Mass flow rate (flow rate associated with a port is positive when it flows into the block)
p	Pressure
R	Specific gas constant
s	Specific entropy
T	Temperature
w	Flow velocity
Z	Compressibility factor
Φ_{work}	Power delivered to the gas flow through the source

Subscripts A and B indicate the appropriate port.

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Input

V – Volumetric flow rate control signal, m³/s

physical signal

Input physical signal that specifies the volumetric flow rate of gas through the source.

Conserving

A – Source inlet

gas

Gas conserving port. A positive mass flow rate causes gas to flow from port **A** to port **B**.

B – Source outlet

gas

Gas conserving port. A positive mass flow rate causes gas to flow from port **A** to port **B**.

Parameters

Power added — Select whether the source performs work

Isentropic power (default) | None

Select whether the source performs work on the gas flow:

- **Isentropic power** — The source performs isentropic work on the gas to maintain the specified mass flow rate, regardless of the pressure differential. Use this option to represent an idealized pump or compressor and properly account for the energy input and output, especially in closed-loop systems.
- **None** — The source performs no work on the flow, neither adding nor removing power, regardless of the mass flow rate produced by the source. Use this option to set up the desired flow condition upstream of the system, without affecting the temperature of the flow.

Cross-sectional area at port A — Area normal to flow path at port A

0.01 m² (default)

Area normal to flow path at port **A**.

Cross-sectional area at port B — Area normal to flow path at port B

0.01 m² (default)

Area normal to flow path at port **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Volumetric Flow Rate Source (G)

Topics

“Modeling Gas Systems”

Introduced in R2019a

Controlled Volumetric Flow Rate Source (MA)

Generate time-varying volumetric flow rate

Library: Simscape / Foundation Library / Moist Air / Sources



Description

The Controlled Volumetric Flow Rate Source (MA) block represents an ideal mechanical energy source in a moist air network. The volumetric flow rate is controlled by the input physical signal at port **V**. The source can maintain the specified volumetric flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. A positive volumetric flow rate causes moist air to flow from port **A** to port **B**.

The volumetric flow rate and mass flow rate are related through the expression

$$\dot{m} = \begin{cases} \rho_B \dot{V} & \text{for } \dot{V} \geq 0 \\ \rho_A \dot{V} & \text{for } \dot{V} < 0 \end{cases}$$

where:

- \dot{m} is the mass flow rate from port **A** to port **B**.
- ρ_A and ρ_B are densities at ports **A** and **B**, respectively.
- \dot{V} is the volumetric flow rate.

The equations describing the source use these symbols.

c_p	Specific heat at constant pressure
h	Specific enthalpy
h_t	Specific total enthalpy
\dot{m}	Mass flow rate (flow rate associated with a port is positive when it flows into the block)
p	Pressure
ρ	Density
R	Specific gas constant
s	Specific entropy
T	Temperature
Φ_{work}	Power delivered to the moist air flow through the source

Subscripts A and B indicate the appropriate port.

Mass balance:

$$\dot{m}_A + \dot{m}_B = 0$$

$$\dot{m}_{wA} + \dot{m}_{wB} = 0$$

$$\dot{m}_{gA} + \dot{m}_{gB} = 0$$

Energy balance:

$$\Phi_A + \Phi_B + \Phi_{work} = 0$$

If the source performs no work (**Power added** parameter is set to None), then $\Phi_{work} = 0$.

If the source is isentropic (**Power added** parameter is set to Isentropic power), then

$$\Phi_{work} = \dot{m}_A(h_{tB} - h_{tA})$$

where

$$h_{tA} = h_A + \frac{1}{2} \left(\frac{\dot{m}_A}{\rho_A S_A} \right)^2$$

$$h_{tB} = h_B + \frac{1}{2} \left(\frac{\dot{m}_B}{\rho_B S_B} \right)^2$$

The mixture-specific enthalpies, $h_A = h(T_A)$ and $h_B = h(T_B)$, are constrained by the isentropic relation, that is, there is no change in entropy:

$$\int_{T_A}^{T_B} \frac{1}{T} dh(T) = R \ln \left(\frac{p_B}{p_A} \right)$$

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Input

V – Volumetric flow rate control signal, m³/s

physical signal

Input physical signal that specifies the volumetric flow rate of the moist air through the source.

Conserving

A – Source inlet

moist air

Moist air conserving port. A positive volumetric flow rate causes moist air to flow from port **A** to port **B**.

B – Source outlet

moist air

Moist air conserving port. A positive volumetric flow rate causes moist air to flow from port **A** to port **B**.

Parameters

Power added — Select whether the source performs work

Isentropic power (default) | None

Select whether the source performs work on the moist air flow:

- **Isentropic power** — The source performs isentropic work on the moist air to maintain the specified mass flow rate, regardless of the pressure differential. Use this option to represent an idealized pump or compressor and properly account for the energy input and output, especially in closed-loop systems.
- **None** — The source performs no work on the flow, neither adding nor removing power, regardless of the mass flow rate produced by the source. Use this option to set up the desired flow condition upstream of the system, without affecting the temperature of the flow.

Cross-sectional area at port A — Area normal to flow path at port A

0.01 m² (default)

Area normal to flow path at port **A**.

Cross-sectional area at port B — Area normal to flow path at port B

0.01 m² (default)

Area normal to flow path at port **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Volumetric Flow Rate Source (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Controlled Volumetric Flow Rate Source (TL)

Generate time-varying volumetric flow rate

Library: Simscape / Foundation Library / Thermal Liquid / Sources



Description

The Controlled Volumetric Flow Rate Source (TL) block represents an ideal mechanical energy source in a thermal liquid network. The volumetric flow rate is controlled by the input physical signal at port **V**. The source can maintain the specified volumetric flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. A positive volumetric flow rate causes the fluid to flow from port **A** to port **B**.

The volumetric flow rate and mass flow rate are related through the expression

$$\dot{m} = \begin{cases} \rho_B \dot{V} & \text{for } \dot{V} \geq 0 \\ \rho_A \dot{V} & \text{for } \dot{V} < 0 \end{cases}$$

where:

- \dot{m} is the mass flow rate from port **A** to port **B**.
- ρ_A and ρ_B are densities at ports **A** and **B**, respectively.
- \dot{V} is the volumetric flow rate.

The energy balance at the source is a function of the energy flow rates through ports **A** and **B** and the work done on the fluid:

$$\phi_A + \phi_B + \phi_{work} = 0,$$

where:

- ϕ_A is the energy flow rate into the source through port **A**.
- ϕ_B is the energy flow rate into the source through port **B**.
- ϕ_{work} is the isentropic work done on the fluid.

The isentropic work term is

$$\phi_{work} = \frac{\dot{m}(p_B - p_A)}{\rho_{avg}},$$

where:

- ϕ_{work} is the isentropic work done on the thermal liquid.
- p_A is the pressure at port **A**.
- p_B is the pressure at port **B**.

- ρ_{avg} is the average liquid density,

$$\rho_{avg} = \frac{\rho_A + \rho_B}{2}.$$

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Input

V — Volumetric flow rate control signal, m³/s

physical signal

Input physical signal that specifies the volumetric flow rate through the source.

Conserving

A — Source inlet

thermal liquid

Thermal liquid conserving port. A positive flow rate causes the fluid to flow from port **A** to port **B**.

B — Source outlet

thermal liquid

Thermal liquid conserving port. A positive flow rate causes the fluid to flow from port **A** to port **B**.

Parameters

Cross-sectional area at ports A and B — Area normal to flow path at inlet and outlet

0.01 m² (default)

Area normal to the direction of flow at the source inlet and outlet. The two cross-sectional areas are assumed identical.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Mass Flow Rate Source (TL) | Controlled Pressure Source (TL) | Mass Flow Rate Source (TL) | Pressure Source (TL) | Volumetric Flow Rate Source (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2016a

Convective Heat Transfer

Heat transfer by convection



Library

Thermal Elements

Description

The Convective Heat Transfer block represents a heat transfer by convection between two bodies by means of fluid motion. The transfer is governed by the Newton law of cooling and is described with the following equation:

$$Q = k \cdot A \cdot (T_A - T_B)$$

where

Q	Heat flow
k	Convection heat transfer coefficient
A	Surface area
T_A, T_B	Temperatures of the bodies

Connections A and B are thermal conserving ports associated with the points between which the heat transfer by convection takes place. The block positive direction is from port A to port B. This means that the heat flow is positive if it flows from A to B.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Area

Surface area of heat transfer. The default value is 0.0001 m².

Heat transfer coefficient

Convection heat transfer coefficient. The default value is 20 W/m²/K.

Ports

The block has the following ports:

A

Thermal conserving port associated with body A.

B

Thermal conserving port associated with body B.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

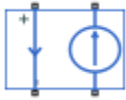
See Also

Conductive Heat Transfer | Radiative Heat Transfer

Introduced in R2007b

Current-Controlled Current Source

Linear current-controlled current source



Library

Electrical Sources

Description

The Current-Controlled Current Source block models a linear current-controlled current source, described with the following equation:

$$I2 = K \cdot I1$$

where

$I2$	Output current
K	Current gain
$I1$	Current flowing from the + to the - control port

To use the block, connect the + and - ports on the left side of the block (the control ports) to the control current source. The arrow between these ports indicates the positive direction of the control current flow. The two ports on the right side of the block (the output ports) generate the output current, with the arrow between them indicating the positive direction of the output current flow.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Current gain K

Ratio of the current between the two output terminals to the current passing between the two control terminals. The default value is 1.

Ports

The block has four electrical conserving ports. Connections + and - on the left side of the block are the control ports. The other two ports are the electrical terminals that provide the output current. The arrows between each pair of ports indicate the positive direction of the current flow.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

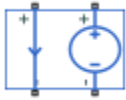
See Also

Current-Controlled Voltage Source | Voltage-Controlled Current Source | Voltage-Controlled Voltage Source

Introduced in R2007a

Current-Controlled Voltage Source

Linear current-controlled voltage source



Library

Electrical Sources

Description

The Current-Controlled Voltage Source block models a linear current-controlled voltage source, described with the following equation:

$$V = K \cdot I1$$

where

V	Voltage
K	Transresistance
$I1$	Current flowing from the + to the - control port

To use the block, connect the + and - ports on the left side of the block (the control ports) to the control current source. The arrow indicates the positive direction of the current flow. The two ports on the right side of the block (the output ports) generate the output voltage. Polarity is indicated by the + and - signs.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Transresistance K

Ratio of the voltage between the two output terminals to the current passing between the two control terminals. The default value is 1Ω .

Ports

The block has four electrical conserving ports. Connections + and - on the left side of the block are the control ports. The arrow indicates the positive direction of the current flow. The other two ports are the electrical terminals that provide the output voltage. Polarity is indicated by the + and - signs.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Current-Controlled Current Source | Voltage-Controlled Current Source | Voltage-Controlled Voltage Source

Introduced in R2007a

Current Sensor

Current sensor in electrical systems



Library

Electrical Sensors

Description

The Current Sensor block represents an ideal current sensor, that is, a device that converts current measured in any electrical branch into a physical signal proportional to the current.

Connections + and - are electrical conserving ports through which the sensor is inserted into the circuit. Connection I is a physical signal port that outputs the measurement result.

Ports

The block has the following ports:

+

Electrical conserving port associated with the sensor positive terminal.

-

Electrical conserving port associated with the sensor negative terminal.

I

Physical signal output port for current.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS-Simulink Converter | Voltage Sensor

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007a

Custom Hydraulic Fluid

Working fluid properties, set by specifying parameter values



Library

Hydraulic Utilities

Description

The Custom Hydraulic Fluid block lets you specify the type of hydraulic fluid used in a loop of hydraulic blocks. It provides the hydraulic fluid properties, such as kinematic viscosity, density, and bulk modulus, for all the hydraulic blocks in the loop. These fluid properties are assumed to be constant during simulation time.

The Custom Hydraulic Fluid block lets you specify the fluid properties, such as kinematic viscosity, density, bulk modulus, and relative amount of entrapped air, as block parameters.

The Custom Hydraulic Fluid block has one port. You can connect it to a hydraulic diagram by branching a connection line off the main line and connecting it to the port. When you connect the Custom Hydraulic Fluid block to a hydraulic line, the software automatically identifies the hydraulic blocks connected to the particular loop and propagates the hydraulic fluid properties to all the hydraulic blocks in the loop.

Each topologically distinct hydraulic loop in a diagram requires the properties of the working fluid to be specified. You can specify these properties by using either a Custom Hydraulic Fluid block or a Hydraulic Fluid block, which is available with Simscape Fluids™ libraries. If no Hydraulic Fluid block or Custom Hydraulic Fluid block is attached to a loop, the hydraulic blocks in this loop use the default fluid, which is equivalent to fluid defined by a Custom Hydraulic Fluid block with the default parameter values.

Parameters

Fluid density

Density of the working fluid. The default value is 850 kg/m^3 .

Kinematic viscosity

Kinematic viscosity of the working fluid. The default value is $1.8\text{e-}5 \text{ m}^2/\text{s}$.

Bulk modulus at atm. pressure and no gas

Bulk modulus of the working fluid, at atmospheric pressure and with no entrapped air. The default value is $8\text{e}8 \text{ Pa}$.

Relative amount of trapped air

Amount of entrained, nondissolved gas in the fluid. The amount is specified as the ratio of gas volume at normal conditions to the fluid volume in the chamber. Therefore, the parameter value

must be less than 1. In practice, the relative amount of trapped air is always greater than 0. If set to 0, ideal fluid is assumed. The default value is 0.005.

Absolute pressure below absolute zero in blocks with fluid compressibility

During simulation, the software checks that the absolute pressure does not fall below absolute zero. This check is performed only for those blocks in the hydraulic circuit where fluid compressibility is important. The value of this parameter determines how the block handles the out-of-range assertion during simulation:

- **Error** — If the pressure falls below absolute zero, the simulation stops and you get an error message. This is the default.
- **Warning** — If the pressure falls below absolute zero, you get a warning but the simulation continues. Use this option when modeling systems where cavitation can occur in extreme cases.

Ports

The block has one hydraulic conserving port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Hydraulic Fluid

Introduced in R2007a

DC Current Source

Ideal constant current source



Library

Electrical Sources

Description

The DC Current Source block represents an ideal current source that is powerful enough to maintain specified current through it regardless of the voltage across the source.

You specify the output current by using the **Constant current** parameter, which can be positive or negative.

The positive direction of the current flow is indicated by the arrow.

Parameters

Constant current

Output current. You can specify positive or negative values. The default value is 1 A.

Ports

The block has two electrical conserving ports associated with its terminals.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

DC Voltage Source

Introduced in R2007a

DC Voltage Source

Ideal constant voltage source



Library

Electrical Sources

Description

The DC Voltage Source block represents an ideal voltage source that is powerful enough to maintain specified voltage at its output regardless of the current flowing through the source.

You specify the output voltage by using the **Constant voltage** parameter, which can be positive or negative.

Connections + and - are conserving electrical ports corresponding to the positive and negative terminals of the voltage source, respectively. The current is positive if it flows from positive to negative, and the voltage across the source is equal to the difference between the voltage at the positive and the negative terminal, $V(+)$ - $V(-)$.

Parameters

Constant voltage

Output voltage. You can specify positive or negative values. The default value is 1 V.

Ports

The block has the following ports:

+

Electrical conserving port associated with the source positive terminal.

-

Electrical conserving port associated with the source negative terminal.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

DC Current Source

Introduced in R2007a

Diode

Piecewise linear diode in electrical systems



Library

Electrical Elements

Description

The Diode block models a piecewise linear diode. If the voltage across the diode is bigger than the **Forward voltage** parameter value, then the diode behaves like a linear resistor with low resistance, given by the **On resistance** parameter value, plus a series voltage source. If the voltage across the diode is less than the forward voltage, then the diode behaves like a linear resistor with low conductance given by the **Off conductance** parameter value.

When forward biased, the series voltage source is described with the following equation:

$$V = Vf(1 - R_{on}G_{off}),$$

where

V	Voltage
Vf	Forward voltage
R_{on}	On resistance
G_{off}	Off conductance

The $R_{on}G_{off}$ term ensures that the diode current is exactly zero when the voltage across it is zero.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Forward voltage

Minimum voltage that needs to be applied for the diode to become forward-biased. The default value is 0.6 V.

On resistance

The resistance of a forward-biased diode. The default value is 0.3 Ω .

Off conductance

The conductance of a reverse-biased diode. The default value is 1e-8 1/ Ω .

Ports

The block has the following ports:

+

Electrical conserving port associated with the diode positive terminal.

-

Electrical conserving port associated with the diode negative terminal.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

Introduced in R2007a

Electrical Reference

Connection to electrical ground



Library

Electrical Elements

Description

The Electrical Reference block represents an electrical ground. Electrical conserving ports of all the blocks that are directly connected to ground must be connected to an Electrical Reference block. A model with electrical elements must contain at least one Electrical Reference block.

Ports

The block has one electrical conserving port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

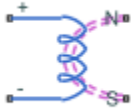
Topics

“Grounding Rules”

Introduced in R2007a

Electromagnetic Converter

Lossless electromagnetic energy conversion device



Library

Magnetic Elements

Description

The Electromagnetic Converter block provides a generic interface between the electrical and magnetic domains.

The block is based on the following equations:

$$MMF = N \cdot I$$

$$V = -N \cdot \frac{d\Phi}{dt}$$

where

MMF	Magnetomotive force (mmf) across the magnetic ports
Φ	Flux through the magnetic ports
I	Current through the electrical ports
V	Voltage across the electrical ports
N	Number of electrical winding turns
t	Simulation time

Connections N and S are magnetic conserving ports, and connections + and - are electrical conserving ports. If the current from the electrical + to - ports is positive, then the resulting mmf is positive acting across the magnetic N to S ports. A positive rate of change of flux flowing from N to S results in a negative induced voltage across the + and - ports.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Basic Assumptions and Limitations

Electromagnetic energy conversion is lossless.

Parameters

Number of winding turns

Number of electrical winding turns. The default value is 1.

Ports

The block has the following ports:

N

Magnetic conserving port associated with the block North terminal.

S

Magnetic conserving port associated with the block South terminal.

+

Positive electrical conserving port.

-

Negative electrical conserving port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Reluctance Force Actuator

Introduced in R2010a

Flow Rate Sensor (IL)

Measure mass flow rate and volumetric flow rate

Library: Simscape / Foundation Library / Isothermal Liquid / Sensors



Description



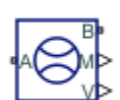
The Flow Rate Sensor (IL) block represents an ideal sensor that measures mass flow rate and volumetric flow rate in an isothermal liquid network. There is no change in pressure or temperature across the sensor.

Because the flow rate is a Through variable, the block must connect in series with the component being measured.

Ports **A** and **B** represent the sensor inlet and outlet. The relative orientation of ports **A** and **B** establishes the measurement sign. The sign is positive if flow occurs from port **A** to port **B**. Switching port connections reverses the measurement sign.

Physical signal ports **M** and **V** output the mass flow rate and the volumetric flow rate through the sensor, respectively. The **Sensor Measurements** parameter controls the measurement type by exposing or hiding the respective ports. Connect an exposed physical signal port to a PS-Simulink Converter block to transform the output physical signal into a Simulink signal, for example, for plotting or additional data processing.

The block icon changes depending on the value of the **Sensor Measurements** parameter.

Sensor Measurements	Block Icon
Mass flow rate	
Volumetric flow rate	
Mass and volumetric flow rates	

If the sensor measurements include volumetric flow rate, the block calculates it from the measured mass flow rate:

$$q = \frac{\dot{m}}{\rho}$$

where:

- q is the volumetric flow rate.
- \dot{m} is the measured mass flow rate.
- ρ is fluid mixture density at port **A** or **B**. There is no pressure difference across the sensor, therefore the densities at ports **A** and **B** are identical. Equations used to compute the fluid mixture density depend on the selected isothermal liquid model. For detailed information, see “Isothermal Liquid Modeling Options”.

Ports

Output

M — Mass flow rate, kg/s

physical signal

Physical signal output port for the mass flow rate measurement.

Dependencies

This port is visible if you set the **Sensor Measurements** parameter to `Mass flow rate` or `Mass and volumetric flow rate`.

V — Volumetric flow rate, m³/s

physical signal

Physical signal output port for the volumetric flow rate measurement.

Dependencies

This port is visible if you set the **Sensor Measurements** parameter to `Volumetric flow rate` or `Mass and volumetric flow rate`.

Conserving

A — Sensor inlet

isothermal liquid

Isothermal liquid conserving port. Flow rate is positive if fluid flows from port **A** to port **B**.

B — Sensor outlet

isothermal liquid

Isothermal liquid conserving port. Flow rate is positive if fluid flows from port **A** to port **B**.

Parameters

Sensor Measurements — Select whether the sensor measures mass flow rate, volumetric flow rate, or both

`Mass flow rate (default)` | `Volumetric flow rate` | `Mass and volumetric flow rate`

Select the type of measurements:

- `Mass flow rate` — The sensor measures the mass flow rate. Selecting this option exposes the output port **M**.

- **Volumetric flow rate** — The sensor measures the volumetric flow rate. Selecting this option exposes the output port **V**.
- **Mass and volumetric flow rate** — The sensor measures both the mass flow rate and the volumetric flow rate. Selecting this option exposes the output ports **M** and **V**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Liquid Properties Sensor (IL) | Pressure Sensor (IL)

Topics

“Modeling Isothermal Liquid Systems”

“Isothermal Liquid Modeling Options”

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2020a

Flow Rate Source (IL)

Generate specified mass flow rate or volumetric flow rate

Library: Simscape / Foundation Library / Isothermal Liquid / Sources



Description

The Flow Rate Source (IL) block represents an ideal mechanical energy source in an isothermal liquid network. The source can maintain the specified mass flow rate or volumetric flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. You specify the flow rate type by using the **Flow rate type** parameter.

Ports **A** and **B** represent the source inlet and outlet. The input physical signal at port **M** or **V**, depending on the flow rate type, specifies the flow rate. Alternatively, you can specify a fixed flow rate as a block parameter. A positive flow rate causes the fluid to flow from port **A** to port **B**.

The block icon changes depending on the values of the **Source type** and **Flow rate type** parameters.

Source Type	Flow Rate Type	Block Icon
Controlled	Mass flow rate	
	Volumetric flow rate	
Constant	Mass flow rate	
	Volumetric flow rate	

The block calculates the work performed on the fluid and includes the results in the simulation data log for information purposes:

$$W_{mech} = \dot{m} \frac{p_B - p_A}{\bar{\rho}}$$

$$\bar{\rho} = \frac{\rho_A + \rho_B}{2}$$

where:

- W_{mech} is the mechanical work performed by the source.
- \dot{m} is the mass flow rate generated by the source.
- p_A and p_B are pressures at ports **A** and **B**, respectively.
- $\bar{\rho}$ is the average fluid mixture density.
- ρ_A and ρ_B are fluid mixture density values at ports **A** and **B**, respectively. Equations used to compute the fluid mixture density depend on the selected isothermal liquid model. For detailed information, see “Isothermal Liquid Modeling Options”.

If the **Flow rate type** parameter is set to **Volumetric flow rate**, the block calculates the mass flow rate from the specified volumetric flow rate:

$$\dot{m} = \rho_{\text{out}}q$$

$$\rho_{\text{out}} = \begin{cases} \rho_B, & q > 0 \\ \rho_A, & q \leq 0 \end{cases}$$

where:

- ρ_{out} is the outflow density.
- q is the volumetric flow rate.

For information on viewing logged simulation data, see “Data Logging”.

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Input

M – Mass flow rate control signal, kg/s

physical signal

Input physical signal that specifies the mass flow rate through the source.

Dependencies

This port is visible if you set the **Source type** parameter to **Controlled** and **Flow rate type** parameter to **Mass flow rate**.

V – Volumetric flow rate control signal, m³/s

physical signal

Input physical signal that specifies the volumetric flow rate through the source.

Dependencies

This port is visible if you set the **Source type** parameter to **Controlled** and **Flow rate type** parameter to **Volumetric flow rate**.

Conserving

A — Source inlet

isothermal liquid

Isothermal liquid conserving port. A positive flow rate causes the fluid to flow from port **A** to port **B**.

B — Source outlet

isothermal liquid

Isothermal liquid conserving port. A positive flow rate causes the fluid to flow from port **A** to port **B**.

Parameters

Source type — Select whether flow rate can change during simulation

Controlled (default) | Constant

Select whether the flow rate generated by the source can change during simulation:

- **Controlled** — The flow rate is variable, controlled by an input physical signal. Selecting this option exposes the input port **M** or **V**, depending on the value of the **Flow rate type** parameter.
- **Constant** — The flow rate is constant during simulation, specified by a block parameter. Selecting this option enables the **Mass flow rate** or **Volumetric flow rate** parameter, depending on the value of the **Flow rate type** parameter.

Mass flow rate — Constant mass flow rate through the source

0 kg/s (default) | scalar

Desired mass flow rate of fluid through the source.

Dependencies

Enabled when you set the **Source type** parameter to Constant and **Flow rate type** parameter to Mass flow rate.

Volumetric flow rate — Constant volumetric flow rate through the source

0 m³/s (default) | scalar

Desired volumetric flow rate of fluid through the source.

Dependencies

Enabled when you set the **Source type** parameter to Constant and **Flow rate type** parameter to Volumetric flow rate.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Pressure Source (IL)

Topics

“Modeling Isothermal Liquid Systems”

“Isothermal Liquid Modeling Options”

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2020a

Flow Resistance (2P)

General resistance in a two-phase fluid branch

Library: Simscape / Foundation Library / Two-Phase Fluid / Elements



Description

The Flow Resistance (2P) block models a general pressure drop in a two-phase fluid network branch. The pressure drop is proportional to the square of the mass flow rate and to the density of the two-phase fluid. The constant of proportionality is determined from a nominal operating condition specified in the block dialog box.

Use this block when the only data available for a component is its pressure drop as a function of its mass flow rate. Combine the block with others to create a custom component that more accurately captures the pressure drop that it induces—for example, a heat exchanger based on a chamber block.

Mass Balance

The volume of fluid inside the flow resistance is assumed to be negligible. The mass flow rate in through one port must then exactly equal the mass flow rate out through the other port:

$$\dot{m}_A + \dot{m}_B = 0,$$

where \dot{m}_A and \dot{m}_B are defined as the mass flow rates into the component through ports **A** and **B**, respectively.

Energy Balance

Energy can enter and leave the flow resistance through the two-phase fluid ports only. No heat exchange occurs between the wall and the environment. In addition, no work is done on or by the fluid. The energy flow rate in through one port must then exactly equal the energy flow rate out through the other port:

$$\phi_A + \phi_B = 0,$$

where ϕ_A and ϕ_B are the energy flow rates into the flow resistance through ports **A** and **B**.

Momentum Balance

The relevant external forces on the fluid include those due to pressure at the ports and those due to viscous friction at the component walls. Gravity is ignored as are other body forces. Expressing the frictional forces in terms of a loss factor ξ yields the semi-empirical expression:

$$\Delta p = \xi \frac{\nu \dot{m}^2}{2S^2},$$

where:

- Δp is the pressure drop from port **A** to port **B**—that is, $p_A - p_B$.
- ξ is the loss factor.
- ν is the specific volume, the inverse of the mass density ρ —that is, $1/\rho$.
- S is the flow area.

The pressure drop equation is implemented with two modifications. First, to allow for a change in sign upon reversal of flow direction, it is rewritten:

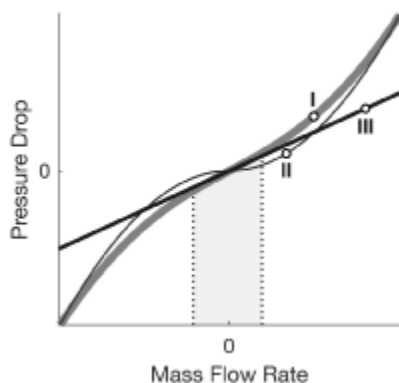
$$\Delta p = \xi \frac{\nu \dot{m} |\dot{m}|}{2S^2},$$

where the pressure drop is positive only if the mass flow rate is too. Second, to eliminate singularities due to flow reversal—singularities that can pose a challenge for numerical solvers during simulation—it is linearized in a small region of near-zero flow:

$$\Delta p = \xi \frac{\nu \dot{m} \sqrt{\dot{m}^2 + \dot{m}_{Th}^2}}{2S^2},$$

where \dot{m}_{Th} is a threshold mass flow rate below which the pressure drop is linearized. The figure shows the modified pressure drop against the local mass flow rate (curve **I**):

- Above \dot{m}_{Th} , the pressure drop approximates that expressed in the original equation (curve **II**) and it varies with \dot{m}^2 . This dependence is commensurate with that observed in turbulent flows.
- Below \dot{m}_{Th} , the pressure drop approximates a straight line with slope partly dependent on \dot{m}_{Th} (curve **III**) and it varies with \dot{m} . This dependence is commensurate with that observed in laminar flows.



For ease of modeling, the loss factor ξ is not required as a block parameter. Instead, it is automatically computed from the nominal condition specified in the block dialog box:

$$\frac{\xi}{2S^2} = \frac{\Delta p_*}{\nu_* \dot{m}_*^2},$$

where the asterisk (*) denotes a value at the nominal operating condition. Underlying all of these calculations is the assumption that the threshold mass flow rate \dot{m}_{Th} is much smaller than the nominal value \dot{m}_* . Replacing the fraction $\xi/(2S^2)$ in the expression for the pressure drop yields:

$$\Delta p = \frac{\nu \Delta p_*}{\nu_* \dot{m}_*^2} (\dot{m} \sqrt{\dot{m}^2 + \dot{m}_{Th}^2}).$$

or, equivalently:

$$\Delta p = C \nu \dot{m} \sqrt{\dot{m}^2 + \dot{m}_{Th}^2},$$

where C is a constant of proportionality between the pressure drop across the flow resistance and the local mass flow rate. It is defined as:

$$C = \frac{\Delta p_*}{\nu_* \dot{m}_*^2}.$$

If the specific volume—and therefore the mass density—is assumed to be invariant, then its nominal and actual values must always be equal. This is the case whenever the nominal value is specified in the block dialog box as Θ —a special value used to signal to the block that the specific volume is a constant. The ratio of the two is then 1 and the product $C\nu$ reduces to:

$$C\nu = \frac{\Delta p_*}{\dot{m}_*^2}.$$

Ports

Conserving

A — Flow resistance port

two-phase fluid

Port through which the two-phase fluid enters or exits the flow resistance.

B — Flow resistance port

two-phase fluid

Port through which the two-phase fluid enters or exits the flow resistance.

Parameters

Nominal pressure drop — Pressure drop at a known operating condition

0.001 MPa (default) | scalar with units of pressure

Pressure drop from inlet to outlet at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate.

Nominal mass flow rate — Mass flow rate at a known operating condition

0.1 kg/s (default) | scalar with units of mass/time

Mass flow rate through the component at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate.

Nominal specific volume — Specific volume at a known operating condition

0 m³/kg (default) | scalar with units of volume/mass

Specific volume inside the flow resistance at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate. Set this parameter to zero to ignore the dependence of the pressure drop on the specific volume.

Cross-sectional area at ports A and B — Flow area at the ports of the flow resistance

0.01 m² (default) | scalar with units of area

Flow area at the ports of the flow resistance. The ports are assumed to be identical in size.

Fraction of nominal mass flow rate for laminar flow — Ratio of threshold to nominal mass flow rates

1e-3 (default) | unitless scalar

Ratio of the threshold mass flow rate to the nominal mass flow rate. The block uses this parameter to calculate the threshold mass flow rate—and ultimately to set the limits of linearization for the pressure drop.

Variables**Mass flow rate into port A — Mass flow rate into the resistance through port A**

293.15 K (default) | positive scalar with units of temperature

Mass flow rate into the resistance through port **A** at the start of simulation.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (2P) | Local Restriction (2P)

Introduced in R2017b

Flow Resistance (G)

General resistance in a gas branch

Library: Simscape / Foundation Library / Gas / Elements



Description

The Flow Resistance (G) block models a general pressure drop in a gas network branch. The pressure drop is proportional to the square of the mass flow rate and to the density of the gas. The constant of proportionality is determined from a nominal operating condition specified in the block dialog box.

Use this block when the only data available for a component is its pressure drop as a function of its mass flow rate. Combine the block with others to create a custom component that more accurately captures the pressure drop that it induces—for example, a heat exchanger based on a chamber block.

Mass Balance

The volume of gas inside the flow resistance is assumed to be negligible. The mass flow rate in through one port must then exactly equal the mass flow rate out through the other port:

$$\dot{m}_A + \dot{m}_B = 0,$$

where \dot{m}_A and \dot{m}_B are defined as the mass flow rates into the component through ports **A** and **B**, respectively.

Energy Balance

Energy can enter and leave the flow resistance through the gas conserving ports only. No heat exchange occurs between the wall and the environment. In addition, no work is done on or by the fluid. The energy flow rate in through one port must then exactly equal the energy flow rate out through the other port:

$$\phi_A + \phi_B = 0,$$

where ϕ_A and ϕ_B are the energy flow rates into the flow resistance through ports **A** and **B**.

Momentum Balance

The relevant external forces on the fluid include those due to pressure at the ports and those due to viscous friction at the component walls. Gravity is ignored as are other body forces. Expressing the frictional forces in terms of a loss factor ξ yields the semi-empirical expression:

$$\Delta p = \xi \frac{\dot{m}^2}{2\rho S^2},$$

where:

- Δp is the pressure drop from port **A** to port **B**—that is, $p_A - p_B$.

- ξ is the loss factor.
- ρ is the fluid density.
- S is the flow area.

The pressure drop equation is implemented with two modifications. First, to allow for a change in sign upon reversal of flow direction, it is rewritten:

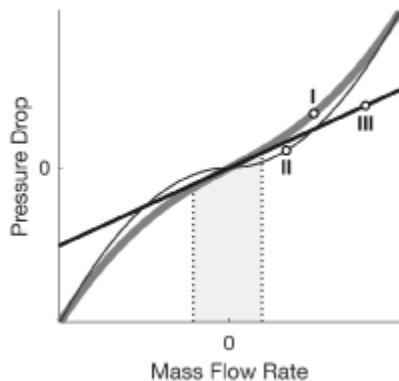
$$\Delta p = \xi \frac{\dot{m}|\dot{m}|}{2\rho S^2},$$

where the pressure drop is positive only if the mass flow rate is too. Second, to eliminate singularities due to flow reversal—which can pose a challenge for numerical solvers during simulation—it is linearized in a small region of near-zero flow:

$$\Delta p = \xi \frac{\dot{m}\sqrt{\dot{m}^2 + \dot{m}_{Th}^2}}{2\rho S^2},$$

where \dot{m}_{Th} is a threshold mass flow rate below which the pressure drop is linearized. The figure shows the modified pressure drop against the local mass flow rate (curve **I**):

- Above \dot{m}_{Th} , the pressure drop approximates that expressed in the original equation (curve **II**) and it varies with \dot{m}^2 . This dependence is commensurate with that observed in turbulent flows.
- Below \dot{m}_{Th} , the pressure drop approximates a straight line with slope partly dependent on \dot{m}_{Th} (curve **III**) and it varies with \dot{m} . This dependence is commensurate with that observed in laminar flows.



For ease of modeling, the loss factor ξ is not required as a block parameter. Instead, it is automatically computed from the nominal condition specified in the block dialog box:

$$\frac{\xi}{2S^2} = \frac{\rho_* \Delta p_*}{\dot{m}_*^2},$$

where the asterisk (*) denotes a value at the nominal operating condition. Underlying all of these calculations is the assumption that the threshold mass flow rate \dot{m}_{Th} is much smaller than the nominal value \dot{m}_* . Replacing the fraction $\xi/(2S^2)$ in the expression for the pressure drop yields:

$$\Delta p = \frac{\rho_* \Delta p_*}{\rho \dot{m}_*^2} (\dot{m} \sqrt{\dot{m}^2 + \dot{m}_{Th}^2}).$$

or, equivalently:

$$\Delta p = \frac{C \dot{m}}{\rho} \sqrt{\dot{m}^2 + \dot{m}_{Th}^2},$$

where C is a constant of proportionality between the pressure drop across the flow resistance and the local mass flow rate. It is defined as:

$$C = \frac{\rho_* \Delta p_*}{\dot{m}_*^2}.$$

If the fluid density is assumed to be invariant, then its nominal and actual values must always be equal. This is the case whenever the nominal value is specified in the block dialog box as \emptyset —a special value used to signal to the block that the fluid density is a constant. The ratio of the two is then 1 and the fraction C/ρ reduces to:

$$\frac{C}{\rho} = \frac{\Delta p_*}{\dot{m}_*^2}.$$

Ports

Conserving

A — Inlet or outlet

gas

Gas conserving port associated with the inlet or outlet of the flow resistance. This block has no intrinsic directionality.

B — Inlet or outlet

gas

Gas conserving port associated with the inlet or outlet of the flow resistance. This block has no intrinsic directionality.

Parameters

Nominal pressure drop — Pressure drop at a known operating condition

0.001 MPa (default)

Pressure drop from inlet to outlet at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate.

Nominal mass flow rate — Mass flow rate at a known operating condition

0.1 kg/s (default)

Mass flow rate through the component at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate.

Nominal density — Density at a known operating condition

0 m³/kg (default)

Mass density inside the flow resistance at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate. Set this parameter to zero to ignore the dependence of the pressure drop on the fluid density.

Cross-sectional area at ports A and B — Flow area at the ports of the flow resistance

0.01 m² (default)

Flow area at the ports of the flow resistance. The ports are assumed to be identical in size.

Fraction of nominal mass flow rate for laminar flow — Ratio of threshold to nominal mass flow rates

1e-3 (default)

Ratio of the threshold mass flow rate to the nominal mass flow rate. The block uses this parameter to calculate the threshold mass flow rate—and ultimately to set the limits of linearization for the pressure drop.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (G)

Introduced in R2017b

Flow Resistance (IL)

General resistance in an isothermal liquid branch

Library: Simscape / Foundation Library / Isothermal Liquid / Elements



Description

The Flow Resistance (IL) block models a general pressure drop in an isothermal-liquid network branch. The pressure drop is proportional to the square of the mass flow rate. The constant of proportionality is determined from a nominal operating condition specified in the block dialog box.

Use this block when the only data available for a component is the typical pressure drop and flow rate. This block is useful for representing complex components, where it is difficult to determine theoretical pressure loss from the geometry.

The volume of fluid inside the flow resistance is assumed to be negligible. The mass flow rate in through one port must then exactly equal the mass flow rate out through the other port:

$$\dot{m}_A + \dot{m}_B = 0,$$

where \dot{m}_A and \dot{m}_B are defined as the mass flow rates into the component through ports **A** and **B**, respectively.

The pressure drop is assumed proportional to the square of the mass flow rate. The square of the mass flow rate is linearized in a small laminar flow region near zero flow, resulting in

$$p_A - p_B = K\dot{m}_A\sqrt{\dot{m}_A^2 + \dot{m}_{lam}^2},$$

where:

- p_A and p_B are pressures at ports **A** and **B**, respectively.
- \dot{m}_{lam} is the mass flow rate threshold for laminar transition.

K is the proportionality constant,

$$K = \frac{\Delta p_{nom}}{\dot{m}_{nom}^2},$$

where Δp_{nom} is the **Nominal pressure drop** parameter value and \dot{m}_{nom} is the **Nominal mass flow rate** parameter value.

Ports

Conserving

A — Inlet or outlet

isothermal liquid

Isothermal liquid conserving port associated with the inlet or outlet of the flow resistance. This block has no intrinsic directionality.

B — Inlet or outlet

isothermal liquid

Isothermal liquid conserving port associated with the inlet or outlet of the flow resistance. This block has no intrinsic directionality.

Parameters

Nominal pressure drop — Pressure drop at a known operating condition

0.001 MPa (default) | positive scalar

Pressure drop from inlet to outlet at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate.

Nominal mass flow rate — Mass flow rate at a known operating condition

0.1 kg/s (default) | positive scalar

Mass flow rate through the component at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate.

Fraction of nominal mass flow rate for laminar flow — Ratio of threshold to nominal mass flow rates

1e-3 (default) | scalar in the range [0,1]

Ratio of the threshold mass flow rate to the nominal mass flow rate. The block uses this parameter to calculate the threshold mass flow rate—and ultimately to set the limits of linearization for the pressure drop.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (IL) | Laminar Leakage (IL) | Local Restriction (IL)

Topics

“Modeling Isothermal Liquid Systems”

“Isothermal Liquid Modeling Options”

Introduced in R2020a

Flow Resistance (MA)

General resistance in a moist air branch

Library: Simscape / Foundation Library / Moist Air / Elements



Description

The Flow Resistance (MA) block models a general pressure drop in a moist air network branch. The drop in pressure is proportional to the square of the mixture mass flow rate and inversely proportional to the mixture density. The constant of proportionality is determined from a nominal operating condition specified in the block dialog box. Set the **Nominal mixture density** parameter to zero to omit the density dependence.

Use this block when empirical data on the pressure losses and flow rates through a component is available, but detailed geometry information is unavailable.

The block equations use these symbols.

\dot{m}	Mass flow rate
Φ	Energy flow rate
p	Pressure
ρ	Density
R	Specific gas constant
S	Cross-sectional area
K	Proportionality constant
h	Specific enthalpy
T	Temperature

Subscripts a, w, and g indicate the properties of dry air, water vapor, and trace gas, respectively. Subscripts A and B indicate the appropriate port.

Mass balance:

$$\dot{m}_A + \dot{m}_B = 0$$

$$\dot{m}_{wA} + \dot{m}_{wB} = 0$$

$$\dot{m}_{gA} + \dot{m}_{gB} = 0$$

Energy balance:

$$\Phi_A + \Phi_B = 0$$

If the **Nominal mixture density** parameter value, ρ_{nom} , is greater than zero, then the block computes the pressure drop as

$$p_A - p_B = K_1 \dot{m}_A \sqrt{\dot{m}_A^2 + (f_{lam} \dot{m}_{nom})^2} \frac{RT_{in}}{p_{in}}$$

where:

- f_{lam} is the fraction of nominal mixture mass flow rate for laminar flow transition.
- p_{in} is the inlet pressure (p_A or p_B , depending on flow direction).
- T_{in} is the inlet temperature (T_A or T_B , depending on flow direction).

The proportionality constant is computed from the nominal flow conditions as

$$K_1 = \frac{\Delta p_{nom} \rho_{nom}}{\dot{m}_{nom}^2}$$

If the **Nominal mixture density** parameter is set to zero, then the block computes the pressure drop as

$$p_A - p_B = K_2 \dot{m}_A \sqrt{\dot{m}_A^2 + (f_{lam} \dot{m}_{nom})^2}$$

where the proportionality constant is computed from the nominal flow conditions as

$$K_2 = \frac{\Delta p_{nom}}{\dot{m}_{nom}^2}$$

The flow resistance is assumed adiabatic, so the mixture specific total enthalpies are equal

$$h_A + \frac{1}{2} \left(\frac{\dot{m}_A R T_A}{S p_A} \right) = h_B + \frac{1}{2} \left(\frac{\dot{m}_B R T_B}{S p_B} \right)$$

Assumptions and Limitations

- The resistance is adiabatic. It does not exchange heat with the environment.
- The pressure drop is assumed to be proportional to the square of the mixture mass flow rate and inversely proportional to the mixture density.

Ports

Conserving

A – Inlet or outlet

moist air

Moist air conserving port associated with the inlet or outlet of the flow resistance. This block has no intrinsic directionality.

B – Inlet or outlet

moist air

Moist air conserving port associated with the inlet or outlet of the flow resistance. This block has no intrinsic directionality.

Parameters

Nominal pressure drop — Pressure drop at a known operating condition

0.001 MPa (default)

Pressure drop from inlet to outlet at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate.

Nominal mixture mass flow rate — Mass flow rate at a known operating condition

0.1 kg/s (default)

Mass flow rate of the air mixture through the block at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate.

Nominal mixture density — Density at a known operating condition

0 kg/m³ (default)

Mass density of the moist air mixture inside the flow resistance at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate. Set this parameter to zero to ignore the dependence of the pressure drop on the air mixture density.

Cross-sectional area at ports A and B — Flow area at the ports of the flow resistance

0.01 m² (default)

Flow area at the ports of the flow resistance. The ports are assumed to be identical in size.

Fraction of nominal mixture mass flow rate for laminar flow transition — Ratio of threshold to nominal mass flow rates

1e-3 (default)

Ratio of the threshold mass flow rate to the nominal mass flow rate of the air mixture. The block uses this parameter to set the threshold for the linearization of the pressure drop.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Flow Resistance (TL)

General resistance in a thermal liquid branch

Library: Simscape / Foundation Library / Thermal Liquid / Elements



Description

The Flow Resistance (TL) block models a general pressure drop in a thermal-liquid network branch. The pressure drop is proportional to the square of the mass flow rate. The constant of proportionality is determined from a nominal operating condition specified in the block dialog box.

Use this block when the only data available for a component is the typical pressure drop and flow rate. This block is useful for representing complex components, where it is difficult to determine theoretical pressure loss from the geometry.

Mass Balance

The volume of fluid inside the flow resistance is assumed to be negligible. The mass flow rate in through one port must then exactly equal the mass flow rate out through the other port:

$$\dot{m}_A + \dot{m}_B = 0,$$

where \dot{m}_A and \dot{m}_B are defined as the mass flow rates into the component through ports **A** and **B**, respectively.

Energy Balance

Energy can enter and leave the flow resistance through the thermal liquid ports only. No heat exchange occurs between the wall and the environment. In addition, no work is done on or by the fluid. The energy flow rate in through one port must then exactly equal the energy flow rate out through the other port:

$$\phi_A + \phi_B = 0,$$

where ϕ_A and ϕ_B are the energy flow rates into the flow resistance through ports **A** and **B**.

Momentum Balance

The pressure drop is assumed proportional to the square of the mass flow rate. The square of the mass flow rate is linearized in a small laminar flow region near zero flow, resulting in

$$p_A - p_B = K\dot{m}_A\sqrt{\dot{m}_A^2 + \dot{m}_{lam}^2},$$

where:

- p_A and p_B are pressures at ports **A** and **B**, respectively.
- \dot{m}_{lam} is the mass flow rate threshold for laminar transition.

K is the proportionality constant,

$$K = \frac{\Delta p_{nom}}{\dot{m}_{nom}^2},$$

where Δp_{nom} is the **Nominal pressure drop** parameter value and \dot{m}_{nom} is the **Nominal mass flow rate** parameter value.

Ports

Conserving

A — Flow resistance port

thermal liquid

Port through which the thermal liquid enters or exits the flow resistance.

B — Flow resistance port

thermal liquid

Port through which the thermal liquid enters or exits the flow resistance.

Parameters

Nominal pressure drop — Pressure drop at a known operating condition

0.001 MPa (default)

Pressure drop from inlet to outlet at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate.

Nominal mass flow rate — Mass flow rate at a known operating condition

0.1 kg/s (default)

Mass flow rate through the component at a known operating condition. The block uses the nominal parameters to calculate the constant of proportionality between the pressure drop and the mass flow rate.

Cross-sectional area at ports A and B — Flow area at the ports of the flow resistance

0.01 m² (default)

Flow area at the ports of the flow resistance. The ports are assumed to be identical in size.

Fraction of nominal mass flow rate for laminar flow — Ratio of threshold to nominal mass flow rates

1e-3 (default)

Ratio of the threshold mass flow rate to the nominal mass flow rate. The block uses this parameter to calculate the threshold mass flow rate—and ultimately to set the limits of linearization for the pressure drop.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (TL) | Local Restriction (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2017b

Fluid Inertia

Pressure differential across tube or channel due to change in fluid velocity



Library

Hydraulic Elements

Description

The Fluid Inertia block models pressure differential, due to change in fluid velocity, across a fluid passage of constant cross-sectional area. The pressure differential is determined according to the following equation:

$$p = \rho \frac{L}{A} \frac{dq}{dt}$$

where

p	Pressure differential
ρ	Fluid density
L	Passage length
A	Passage area
q	Flow rate
t	Time

Use this block in various pipe or channel models that require fluid inertia to be accounted for.

The block positive direction is from port A to port B. This means that the flow rate is positive if it flows from A to B.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Assumptions and Limitations

Fluid density remains constant.

Parameters

Passage area

Fluid passage cross-sectional area. The default value is $8e-5 \text{ m}^2$.

Passage length

Length of the fluid passage. The default value is 1 m.

Initial flow rate

Initial flow rate through the passage. This parameter specifies the initial condition for use in computing the block's initial state at the beginning of a simulation run. For more information, see "Initial Conditions Computation". The default value is 0.

Global Parameters

Parameter determined by the type of working fluid:

- **Fluid density**

Use the Hydraulic Fluid block or the Custom Hydraulic Fluid block to specify the fluid properties.

Ports

The block has the following ports:

A

Hydraulic conserving port associated with the passage inlet.

B

Hydraulic conserving port associated with the passage outlet.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

Introduced in R2007a

Flux Sensor

Ideal flux sensor



Library

Magnetic Sensors

Description

The Flux Sensor block represents an ideal flux sensor, that is, a device that converts flux measured in any magnetic branch into a physical signal proportional to the flux.

Connections N and S are conserving magnetic ports through which the sensor is inserted into the circuit. The physical signal port outputs the value of the flux, which is positive when the flux flows from the N to the S port.

Ports

The block has the following ports:

N

Magnetic conserving port associated with the sensor North terminal.

S

Magnetic conserving port associated with the sensor South terminal.

The block also has a physical signal output port, which outputs the value of the flux.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

[Controlled Flux Source](#) | [Flux Source](#) | [PS-Simulink Converter](#)

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2010a

Flux Source

Ideal flux source



Library

Magnetic Sources

Description

The Flux Source block represents an ideal flux source that is powerful enough to maintain specified constant flux through it, regardless of the mmf across its terminals.

You specify the output flux by using the **Constant flux** parameter, which can be positive, negative, or zero.

The positive direction of the flux flow is indicated by the arrow.

Parameters

Constant flux

Output flux. You can specify any real value. The default value is 0.001 Wb.

Ports

The block has two magnetic conserving ports associated with its terminals.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Flux Source

Introduced in R2010a

Fundamental Reluctance

Simplified implementation of magnetic reluctance



Library

Magnetic Elements

Description

The Fundamental Reluctance block 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.

The block is based on the following equation:

$$MMF = \Phi \cdot \mathfrak{R}$$

where

MMF	Magnetomotive force (mmf) across the component
Φ	Flux through the component
\mathfrak{R}	Reluctance

Connections N and S are magnetic conserving ports. The mmf across the reluctance is given by $MMF(N) - MMF(S)$, and the sign of the flux is positive when flowing through the device from N to S.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Reluctance

The ratio of the magnetomotive force (mmf) across the component to the resulting flux that flows through the component. The default value is $8e4$ 1/H.

Ports

The block has the following ports:

N

Magnetic conserving port associated with the block North terminal.

S

Magnetic conserving port associated with the block South terminal.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Reluctance | Variable Reluctance

Introduced in R2014a

Gas Properties

Pneumatic domain properties for attached circuit



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Gas Properties block defines pneumatic domain properties for a circuit, that is, the gas properties that act as global parameters for all the blocks connected to the pneumatic circuit. These gas properties are assumed to be constant during simulation time.

The Gas Properties block lets you specify the gas properties, such as specific heat at constant pressure and constant volume, as well as viscosity, as block parameters. It also lets you specify ambient pressure and ambient temperature.

The Gas Properties block has one port. You can connect it to a pneumatic diagram by branching a connection line off the main line and connecting it to the port. When you connect the Gas Properties block to a pneumatic line, the software automatically identifies the pneumatic blocks connected to the particular circuit and propagates the gas properties to all the pneumatic blocks in the circuit.

Each topologically distinct pneumatic circuit in a diagram requires exactly one Gas Properties block to be connected to it. Therefore, there must be as many Gas Properties blocks as there are pneumatic circuits in the system. If no Gas Properties block is attached to a circuit, the pneumatic blocks in this circuit use the gas properties corresponding to the default Gas Properties block parameter values.

Parameters

Specific heat at constant pressure

Specify the gas specific heat at constant pressure. The default value is $1.005e3$ J/kg/K.

Specific heat at constant volume

Specify the gas specific heat at constant volume. The default value is 717.95 J/kg/K.

Dynamic viscosity

Specify the gas dynamic viscosity. The default value is $1.821e-5$ s*Pa.

Ambient pressure

Specify the gas ambient pressure. The default value is 101325 Pa.

Ambient temperature

Specify the gas ambient temperature. The default value is 293.15 K.

Pressure or temperature below absolute zero

Determines how the block handles the out-of-range assertion during simulation:

- **Error** — If the pressure or temperature falls below absolute zero, the simulation stops and you get an error message. This is the default.
- **Warning** — If the pressure or temperature falls below absolute zero, you get a warning but the simulation continues.

Ports

The block has one pneumatic conserving port.

Introduced in R2009b

Gas Properties (G)

Global gas properties for attached circuit

Library: Simscape / Foundation Library / Gas / Utilities



Description

The Gas Properties (G) block defines the gas properties that act as global parameters for all the blocks connected to a circuit. The default gas is dry air.

Each topologically distinct gas circuit in a diagram can have a Gas Properties (G) block connected to it. If no Gas Properties (G) block is attached to a circuit, the blocks in this circuit use the gas properties corresponding to the default Gas Properties (G) block parameter values.

The Gas Properties (G) block lets you select between three gas property models: perfect gas, semiperfect gas, and real gas. The three gas property models provide trade-offs between simulation speed and accuracy. They also enable the incremental workflow: you start with a simple model, which requires minimal information about the working gas, and then build upon the model when more detailed gas property data becomes available. The following table lists the information that must be specified for each of the three gas property models.

Gas Property Model	Physical Properties	
Perfect	R	Specific gas constant
	Z	Compressibility factor, $Z = p/(\rho RT)$
	T_{ref}	Reference temperature
	h_{ref}	Specific enthalpy at reference temperature
	c_p	Specific heat at constant pressure, $c_p = \left(\frac{\partial h}{\partial T}\right)_p$
	μ	Dynamic viscosity
	k	Thermal conductivity
Semiperfect	R	Specific gas constant
	Z	Compressibility factor
	h	Specific enthalpy vector
	c_p	Specific heat at constant pressure vector
	μ	Dynamic viscosity vector
	k	Thermal conductivity vector
	T	Temperature vector
Real	ρ	Density table
	s	Specific entropy table

Gas Property Model	Physical Properties	
	h	Specific enthalpy table
	c_p	Specific heat at constant pressure table
	μ	Dynamic viscosity table
	k	Thermal conductivity table
	β	Isothermal bulk modulus table, $\beta = \rho \left(\frac{\partial p}{\partial \rho} \right)_T$
	α	Isobaric thermal expansion coefficient table, $\alpha = - \frac{1}{\rho} \left(\frac{\partial \rho}{\partial T} \right)_p$
	T	Temperature vector
	p	Pressure vector

For semiperfect gas, caloric and transport properties are functions of temperature. You specify them as one-dimensional arrays corresponding to the **Temperature vector**. For real gas, all properties are functions of temperature and pressure. You specify them as two-dimensional arrays in which the rows correspond to the **Temperature vector** and the columns correspond to the **Pressure vector**.

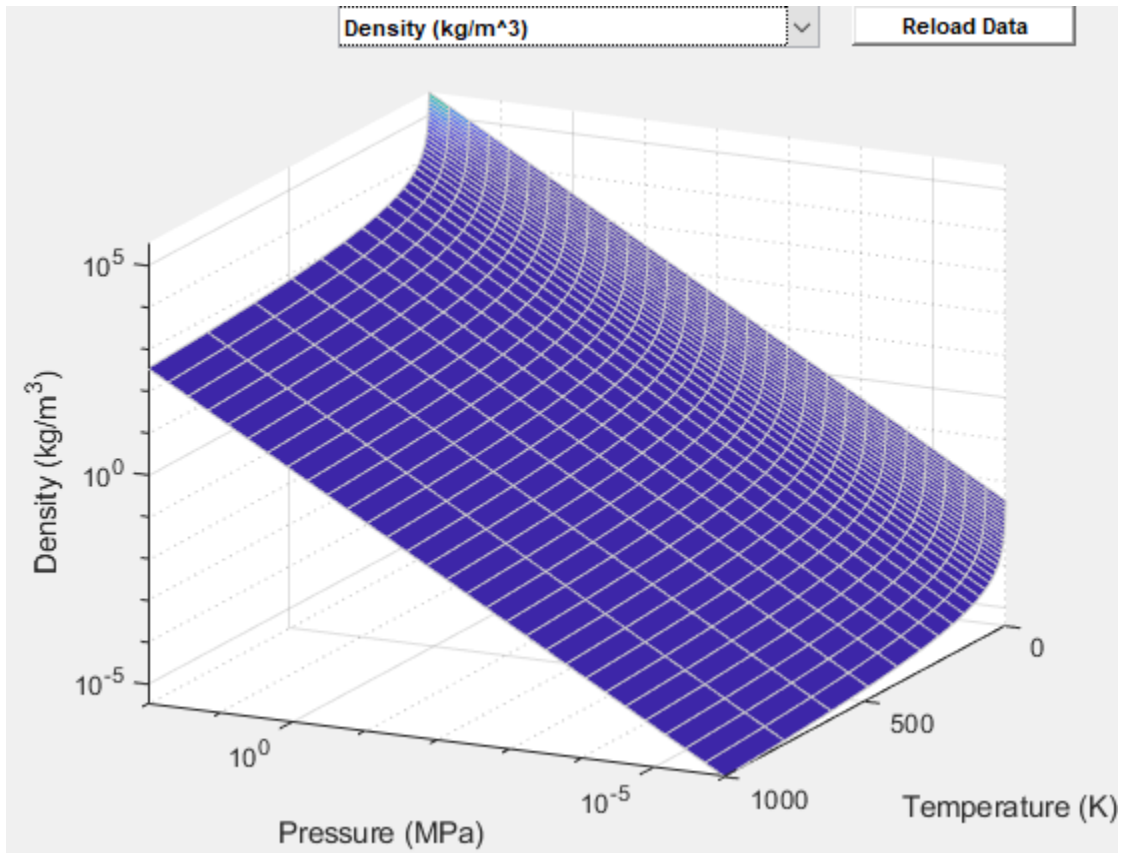
Simulation issues an error when temperature or pressure is out of range:

- For perfect gas, you specify the minimum and maximum permissible values for pressure and temperature in the **Parameters** section.
- For semiperfect gas, the **Valid temperature range parameterization** parameter gives you an option to define the permissible range of temperatures by using the lowest and highest values of the temperature vector. You have to specify the minimum and maximum permissible values for pressure.
- For real gas, the **Valid pressure-temperature region parameterization** parameter gives you an option to define the permissible ranges of pressure and temperature by using the lowest and highest values of the pressure and temperature vectors, respectively. You can also specify a validity matrix for the pressure-temperature value combinations.

Data Visualization

The block provides the option to plot the specified gas properties as a function of temperature and pressure. Plotting the properties lets you visualize the data before simulating the model.

To plot the data, right-click the Gas Properties (G) block in your model and, from the context menu, select **Foundation Library > Plot Gas Properties**. Use the drop-down list located at the top of the plot to select the gas property to visualize. Click the **Reload** button to regenerate a plot following a block parameter update.



Gas Properties Plot

Note To facilitate plot readability, density data is shown in log-log scale, while other properties are plotted in semi-log scale. That is, for a density plot as a function of temperature and pressure, both density and pressure are in log scales, while the temperature values are not in log scale. For other properties, only the pressure is in log scale.

Ports

Conserving

A — Connection port

gas

Gas conserving port that connects the block to the gas network. You can connect it to any point on a gas connection line in a block diagram. When you connect the Gas Properties (G) block to a gas line, the software automatically identifies the gas blocks connected to the particular circuit and propagates the gas properties to all the blocks in the circuit.

Settings

Physical Properties

Gas specification — Select gas property model

Perfect (default) | Semiperfect | Real

Select the gas property model, which defines the level of idealization: perfect gas, semiperfect gas, or real gas. The gas property model determines the set of physical properties and parameters that you can specify for the working gas. For more information, see “Gas Property Models”.

Specific gas constant — Perfect or semiperfect gas

0.287 kJ/kg/K (default)

Universal gas constant divided by the molar mass of the gas.

Compressibility factor — Perfect or semiperfect gas

1 (default)

Compressibility factor that accounts for deviation from the ideal gas law. It is assumed constant during simulation.

Reference temperature for gas properties — Perfect gas

293.15 K (default) | scalar

Temperature at which the perfect gas properties are specified.

Specific enthalpy at reference temperature — Perfect gas

420 kJ/kg (default) | scalar

Specific enthalpy of the perfect gas at the reference temperature.

Specific heat at constant pressure — Perfect gas

1 kJ/kg/K (default) | scalar

Specific heat capacity of the perfect gas at constant pressure.

Dynamic viscosity — Perfect gas

18 s* μ Pa (default) | scalar

Dynamic viscosity of the perfect gas.

Thermal conductivity — Perfect gas

26 mW/m/K (default) | scalar

Thermal conductivity of the perfect gas.

Temperature vector — Semiperfect or real gas

vector

Vector of gas temperature values, to be used for table lookup of other gas properties.

The block allows easy switching between the gas property models, without having to change the values of the parameters. Although the parameter name is the same for semiperfect and real gas, the values are independent from each other. The block stores them separately. This way, you can have different lookup tables stored within the block for semiperfect and for real gas.

The default value, both for semiperfect and real gas, is [150 : 10 : 200, 250 : 50 : 1000, 1500, 2000] K.

Specific enthalpy vector – Semiperfect gas

vector

The vector of specific enthalpy values of the semiperfect gas, for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size.

The default value is [275.264783730547; 285.377054177734; 295.474578903607; 305.560871069627; 315.638490783961; 325.709373787179; 376.008033649461; 426.297784741196; 476.678788323875; 527.253890684429; 578.12722735187; 629.395040865183; 681.137690131207; 733.415442873049; 786.267894376777; 839.715851152006; 893.764459999509; 948.406676876931; 1003.62653219711; 1059.40193232821; 1115.70691941005; 1172.51341226958; 1762.18512007361; 2377.14064127409] kJ/kg.

Specific heat at constant pressure vector – Semiperfect gas

vector

The vector of specific heat at constant pressure values, for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size.

The default value is [1.01211492398124; 1.01042105529234; 1.00914159006373; 1.00815898273273; 1.00739688341918; 1.00680484569075; 1.00554191216852; 1.00637389921747; 1.0092105862442; 1.01414404435034; 1.02111299587909; 1.02986878923959; 1.04003820504128; 1.05120320041448; 1.06296312852444; 1.07497111709439; 1.08694865668905; 1.0986858495625; 1.11003396604319; 1.12089493621399; 1.13121051694521; 1.14095251451866; 1.21001953144; 1.24628356718428] kJ/kg/K.

Dynamic viscosity vector – Semiperfect gas

vector

The vector of gas dynamic viscosity values, for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size.

The default value is [10.3766056544352; 10.9908682444892; 11.5932000841352; 12.1841060353213; 12.764067709023; 13.3335437107148; 16.038149065057; 18.5373404836509; 20.8671495114253; 23.0554226800681; 25.1239717991959; 27.0901379793485; 28.96790749834; 30.7687111230031; 32.5020036481173; 34.17569027566; 35.7964450415303; 37.3699520751368; 38.9010908824946; 40.3940804368247; 41.8525925389218; 43.279841956487; 56.3254722619962; 68.068290080945] s* μ Pa.

Thermal conductivity vector – Semiperfect gas

vector

The vector of gas thermal conductivity values, for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size.

The default value is [14.1517155766309; 15.0474512994325; 15.9300520513026; 16.7998749295306; 17.6573089963228; 18.5027588910197; 22.5644026699512;

26.384465676638; 30.0032801168886; 33.4532006033488; 36.7600619429332;
 39.9446251241844; 43.0237448740114; 46.0112537033799; 48.9186234516389;
 51.7554617696538; 54.5298878921773; 57.2488200454925; 59.9181976996146;
 62.5431553064749; 65.128159527924; 67.6771186900213; 91.7815514375552;
 114.486299090693] mW/m/K.

Pressure vector — Real gas

vector

Vector of gas pressure values, to be used together with the vector of temperature values for two-dimensional table lookup of other gas properties.

The default value is [0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0] Mpa.

Density table — Real gas

matrix

The matrix of gas density values, for two-dimensional table lookup based on pressure and temperature. The matrix size must correspond to the sizes of the pressure and temperature vectors. The table rows correspond to the **Temperature vector** values, and the columns correspond to the **Pressure vector** values.

The default is [0.232389928446798, 0.465063671432694, 1.16479567270147,
 2.33676613945539, 4.70263894299404, 11.983516436453, 24.7878864888476,
 53.4753944012435, 188.202476404452, 488.088746553617; 0.217841319668304,
 0.43590005028115, 1.09138524660849, 2.18825141646267, 4.39865940277095,
 11.1676860192343, 22.9408518273177, 48.6188318442922, 151.41140977555,
 397.086323615566; 0.205009108095605, 0.410186693889784, 1.02673290215943,
 2.05770371150129, 4.13249094693848, 10.4620818492884, 21.3797637857971,
 44.7465382224065, 130.788260466556, 323.533424670971; 0.193606079729805,
 0.38734394286521, 0.969349656759853, 1.94200819142115, 3.89732426996839,
 9.84460606064557, 20.0372743829505, 41.551685035359, 116.744404785154,
 273.404057887341; 0.183405807765996, 0.36691546510227, 0.918068262208759,
 1.83874006288175, 3.68793116346732, 9.2989642809239, 18.8669316601087,
 38.8510263281099, 106.248040134752, 239.063674783047; 0.174227353976232,
 0.348537012803259, 0.871960127107058, 1.74598097193374, 3.50022045445381,
 8.81280884432948, 17.8353318467781, 36.5262009931833, 97.9572133071041,
 214.144269454883; 0.139362218300841, 0.278750918818379, 0.697075681624698,
 1.39481110010495, 2.79224958681861, 7.00009934436889, 14.0635945083875,
 28.3693859898697, 72.4958995959279, 148.061190553083; 0.116127751690325,
 0.232262711724454, 0.580710652868886, 1.16159962682988, 2.32390314733333,
 5.81485119489922, 11.6454651017326, 23.345179818482, 58.6036404676478,
 116.933272564178; 0.0995348730218713, 0.199069527735142, 0.497672062326111,
 0.995337472158662, 1.99064239855983, 4.97624478331013, 9.95052507838832,
 19.8877675359538, 49.5222664696343, 97.8281817865855; 0.0870915231079571,
 0.174179875730153, 0.435425837514054, 0.870771682736456, 1.74121977223468,
 4.35055112014231, 8.6923131825558, 17.3462098741118, 43.0204270425471,
 84.5862815012905; 0.0774139632775453, 0.154823661826077, 0.387027124125729,
 0.773947181340524, 1.54746386958083, 3.86538722275345, 7.71958432856407,
 15.3924298760864, 38.0966874262009, 74.7409240194489; 0.0696722132563204,
 0.139339881699215, 0.348315590097498, 0.696517280056477, 1.3925775726502,
 3.47798929739441, 6.94428820700405, 13.840595299045, 34.2211656015534,
 67.0763619510542; 0.0633382100675146, 0.126671957763271, 0.316646408860559,

0.633181081307499, 1.26591433962438, 3.16140997822029, 6.31145690049322, 12.5766989218007, 31.0826018729834, 60.9113572079736; 0.0580599590057729, 0.116115694052286, 0.2902575444054221, 0.580409375636267, 1.1603953396719, 2.89780177016166, 5.78491146603688, 11.5265728971708, 28.4843351052556, 55.829341571381; 0.0535937940886494, 0.107183660899874, 0.267929690316126, 0.535761125401431, 1.07112886747126, 2.67486471725533, 5.33982631291416, 10.63973457213, 26.2952097273985, 51.5589206387574; 0.0497656797149008, 0.0995277403610491, 0.248792203124892, 0.497493882078744, 0.99462543314985, 2.48384153794703, 4.95858078750581, 9.88055926765067, 24.4240170579045, 47.9145569639003; 0.0464479959361466, 0.0928926705710533, 0.232206763597689, 0.464330464081941, 0.928328524958611, 2.31832536465517, 4.62831259866885, 9.22315228584034, 22.8051678288226, 44.7645110499732; 0.0435450312171066, 0.0870870188301226, 0.217694718092182, 0.435313326650428, 0.870322119623417, 2.17351945401494, 4.33940803061583, 8.64821978531377, 21.3901818076149, 42.0123332867245; 0.0409835963547538, 0.0819644033075145, 0.204890086547796, 0.409710426055867, 0.819141805232369, 2.04576053908084, 4.08453435907903, 8.14108047721294, 20.1423910541739, 39.5856012611331; 0.0387067675798369, 0.077410976131965, 0.193508246917561, 0.386952511151441, 0.773649057561575, 1.93220226835442, 3.85799954128176, 7.6903585147706, 19.0335137737723, 37.4287645642868; 0.0366696057813522, 0.073336860231159, 0.183324515224783, 0.366590243538238, 0.732945322059316, 1.8305992514921, 3.65531662685461, 7.28709889581498, 18.0413651664581, 35.4984330382504; 0.0348361601073285, 0.0696701556937661, 0.174159155188816, 0.348264195999875, 0.696311928298658, 1.73915624407237, 3.47290023125895, 6.92415359223964, 17.1482850565279, 33.7601783509398; 0.0232242994681783, 0.0464475414045395, 0.11611092226157, 0.232195408621678, 0.464285085702862, 1.15991997307222, 2.31719916466188, 4.62384864786652, 11.4808038114339, 22.7013239671071; 0.0174183167348545, 0.0348360245853652, 0.0870854949659176, 0.174155769175675, 0.348250662116806, 0.870170220523828, 1.73881992076876, 3.47156495737229, 8.63350555136021, 17.1168158717553] kg/m³.

Specific entropy table – Real gas

matrix

The matrix of specific entropy values, for two-dimensional table lookup based on pressure and temperature. The matrix size must correspond to the sizes of the pressure and temperature vectors. The table rows correspond to the **Temperature vector** values, and the columns correspond to the **Pressure vector** values.

The default is [3.85666832168988, 3.65733557342939, 3.39321600973057, 3.19240764614444, 2.98972303348322, 2.71525667312277, 2.4961408855038, 2.25200576011077, 1.78471429674036, 1.29090458888978; 3.92140869957028, 3.72213261304626, 3.45818397201293, 3.25766364401107, 3.0555672742907, 2.78296953737326, 2.56736420779637, 2.33223141053168, 1.92951303779799, 1.47834112059833; 3.98221529565273, 3.7829836555054, 3.51916895627942, 3.31887389296688, 3.11723580728132, 2.84607806940339, 2.63311263769974, 2.40436393408814, 2.03542502560031, 1.64260706565665; 4.03954033696417, 3.84034421897669, 3.57663648301021, 3.37652102412054, 3.17524721731924, 2.90522439822934, 2.69430151549255, 2.47029333509785, 2.12268019468985, 1.77344207366055; 4.09376228675006, 3.89459501864723, 3.63097409868183, 3.43100422585914, 3.23002497599441, 2.96091365856363, 2.75160785927873, 2.5312406415135, 2.19841869007162, 1.87901583141216; 4.14520117995298, 3.94605767166668, 3.68250821276855, 3.48265804957869, 3.28192051690123,

3.0135531037355, 2.80555209686157, 2.58805540086205, 2.26610502263839,
 1.96775904763196; 4.36901028390122, 4.16994018348893, 3.90661129562974,
 3.70712991087092, 3.5071342997304, 3.24102784511277, 3.03691403727937,
 2.82764560970477, 2.53381584725277, 2.28577338343632; 4.55205623466113,
 4.35302221894054, 4.08980164185938, 3.89050096377458, 3.69086747132005,
 3.42585296452877, 3.22357730852106, 3.01804958620108, 2.73582153398541,
 2.50683391368241; 4.70718999888292, 4.5081763447144, 4.24501685439271,
 4.04581799448331, 3.84638816350602, 3.58198482518679, 3.38072826636793,
 3.17723930571407, 2.90109390657126, 2.68174878534546; 4.84213514624873,
 4.64313406945348, 4.38001230330443, 4.18087629053765, 3.98157205463241,
 3.71754470888327, 3.51691216187067, 3.31466120289276, 3.04214220284213,
 2.82845136827457; 4.96189181371153, 4.76289902096378, 4.49980209857165,
 4.30070746415684, 4.10148588060503, 3.83770566300567, 3.63748226836764,
 3.43603958793625, 3.1658665252885, 2.9558053312017; 5.06986353974934,
 4.87087647196321, 4.60779671754215, 4.40873067340924, 4.20956618447661,
 3.9459565693246, 3.74601527912977, 3.54512868450336, 3.27656221849516,
 3.06897900823943; 5.1684503865325, 4.96946742614353, 4.70639998856773,
 4.50735445472532, 4.30823091994561, 4.04474364100682, 3.84500452317677,
 3.64451604005125, 3.37709737816777, 3.17128346671381; 5.2593908173202,
 5.06041089369949, 4.7973525623652, 4.59832219207578, 4.39922893392441,
 4.13583208512233, 3.93624237769884, 3.73604799461671, 3.46947669719846,
 3.26497004191754; 5.34397260579747, 5.14499498347528, 4.88194355295007,
 4.68292467374206, 4.48385435941153, 4.22052603846292, 4.02104955215385,
 3.82107803046202, 3.55514894085648, 3.35163463875693; 5.42316925832833,
 5.22419341632272, 4.96114732438797, 4.7621373349829, 4.5630847709502,
 4.29980946881718, 4.10042058896199, 3.90062154398139, 3.63518977950306,
 3.43244552095257; 5.49773109148187, 5.29875665110057, 5.03571476221911,
 4.83671177184987, 4.63767318328301, 4.3744396285261, 4.17511974266743,
 3.97545657469879, 3.71041694885177, 3.5082812277818; 5.56824735227027,
 5.3692740321233, 5.10623550252735, 4.90723810625184, 4.70821068826088,
 4.44501050574435, 4.24574578441128, 4.04619129997113, 3.78146565369216,
 3.57981830676003; 5.63518925543468, 5.43621684242134, 5.17318103311078,
 4.97418816693336, 4.77516979527807, 4.51199664197531, 4.31277661041644,
 4.11331020953596, 3.84883930692141, 3.64758910924957; 5.69894018571711,
 5.4999685157543, 5.2369349347088, 5.03794577935713, 4.83893481833662,
 4.57578380965866, 4.37660040041591, 4.17720621324442, 3.91294438708545,
 3.71202087418194; 5.75981714571907, 5.56084609060782, 5.2978143534057,
 5.09882826876, 4.89982344031189, 4.63669075931028, 4.43753766711061,
 4.23820328614963, 3.97411480122343, 3.77346272019934; 5.81808616796,
 5.61911562623141, 5.35608542860148, 5.15710190799382, 4.95810220046551,
 4.69498482551242, 4.49585705785849, 4.29657265520884, 4.03262917849163,
 3.83220462904951; 6.29530146610539, 6.09633331903482, 5.8333103029986,
 5.63433874379363, 5.43536292947713, 5.17231699952891, 4.97330753860033,
 4.77425696756918, 4.51099451008297, 4.31164629242141; 6.64886963403215,
 6.44990215110977, 6.18688112698696, 5.98791288587009, 5.78894370112331,
 5.52591760746728, 5.32694103515787, 5.12795561761856, 4.86488401452708,
 4.66584072487367] kJ/kg/K.

Specific enthalpy table – Real gas

matrix

The matrix of specific enthalpy values, for two-dimensional table lookup based on pressure and temperature. The matrix size must correspond to the sizes of the pressure and temperature vectors. The table rows correspond to the **Temperature vector** values, and the columns correspond to the **Pressure vector** values.

The default is [276.007989595737, 275.926922934925, 275.68326231383, 275.275612961165, 274.45440990197, 271.941510053176, 267.573991878308, 258.028969756929, 218.234362269026, 158.930370089034; 286.039258534729, 285.966962517021, 285.749758776967, 285.386693516142, 284.656540566853, 282.432853052032, 278.608576829258, 270.456137821523, 240.62825338508, 187.975341616082; 296.069269816893, 296.004298839236, 295.809169446516, 295.483228714877, 294.828601531559, 292.84221621424, 289.452931620771, 282.352183856267, 258.083267319111, 215.051052511419; 306.098418400666, 306.03965742714, 305.863227349031, 305.568684845356, 304.977739633649, 303.189749788045, 300.157496456351, 293.885241012016, 273.341211471863, 237.921819553097; 316.127034494775, 316.073605930689, 315.913222268783, 315.64558865716, 315.109086307672, 313.489564435311, 310.756170955578, 305.156737704007, 287.344997830989, 257.435968836394; 326.155419682752, 326.106620570346, 325.960160622186, 325.715851541006, 325.226445621544, 323.751863790821, 321.272681065641, 316.232458831075, 300.538097964225, 274.729457520261; 376.304967312284, 376.272444481832, 376.174889100496, 376.012340447409, 375.687406489001, 374.713906995184, 373.095751316588, 369.876236784684, 360.387900683659, 345.656435942873; 426.50485367445, 426.482160685975, 426.414109983542, 426.30078622341, 426.074490520224, 425.398399641627, 424.280795394447, 422.079563123224, 415.746855514936, 406.194464895021; 476.825819267465, 476.80969998052, 476.761370979607, 476.680918679615, 476.520372988485, 476.041583484634, 475.252944483118, 473.709732112807, 469.339610018801, 462.897504479846; 527.358078674724, 527.346652677109, 527.312400552407, 527.255399745011, 527.141719692163, 526.803228763503, 526.247426409769, 525.16611881262, 522.148839900958, 517.815837109305; 578.199407091628, 578.191488376844, 578.16775454136, 578.128272369734, 578.049585297172, 577.815721490642, 577.433139940652, 576.694019389874, 574.669839072011, 571.873288825611; 629.442482115764, 629.437274460453, 629.421670526216, 629.395727286419, 629.344077339647, 629.191001924645, 628.942009298704, 628.466216590227, 627.202800994792, 625.579623790386; 681.165504740472, 681.162448079495, 681.153294327517, 681.138092074412, 681.107889301156, 681.018879721989, 680.875761746077, 680.608453039536, 679.94611657735, 679.249483320862; 733.427358544504, 733.426044380332, 733.422115787073, 733.415614375573, 733.40278431897, 733.365663494888, 733.308277054267, 733.209722450195, 733.03360169447, 733.087076022656; 786.266709844873, 786.266831443852, 786.267208214465, 786.267876004164, 786.269360429697, 786.274993640674, 786.2882450665, 786.328732242753, 786.553281957475, 787.225888563409; 839.703718528092, 839.70504000887, 839.70901483498, 839.715674095171, 839.729121711022, 839.770488095606, 839.842783833376, 839.999515710826, 840.559250908246, 841.750855761636; 893.743067589143, 893.745403899365, 893.75242189651, 893.764148726438, 893.787715116484, 893.859308222756, 893.981558350577, 894.236670780038, 895.080332077458, 896.712551381498; 948.377371826816, 948.380575306181, 948.390193713629, 948.406250911057, 948.438464405059, 948.535890862389, 948.700843735304, 949.040088044586, 950.12680220025, 952.137078607614; 1003.59040526852, 1003.59435635928, 1003.60621668171, 1003.62600734625, 1003.66567635309, 1003.78537887654, 1003.98716277073,

1004.39900119214, 1005.69566363563, 1008.03325638583; 1059.35987800545,
 1059.36447866476, 1059.37828691794, 1059.40132155581, 1059.44746887997,
 1059.58653006743, 1059.82032895022, 1060.29529625187, 1061.77473328752,
 1064.39791677957; 1115.6596794705, 1115.66484839484, 1115.68036078582,
 1115.70623346662, 1115.758048708, 1115.91404892212, 1116.17586781833,
 1116.70611010185, 1118.34575590913, 1121.21980977189; 1172.46160798047,
 1172.46727707642, 1172.48428942135, 1172.51266015931, 1172.56946454382,
 1172.74037694276, 1173.02686874382, 1173.60580340131, 1175.38672698925,
 1178.48246092596; 1762.10727721046, 1762.11579940935, 1762.1413682507,
 1762.18399045824, 1762.26926281627, 1762.52530190486, 1762.95276369405,
 1763.81035090328, 1766.40303160672, 1770.78232137303; 2377.0524638672,
 2377.06211829984, 2377.09108293987, 2377.13936180819, 2377.23593626125,
 2377.52579256081, 2378.00932430138, 2378.97799140869, 2381.89608390415,
 2386.79535914098] kJ/kg.

Specific heat at constant pressure table – Real gas

matrix

The matrix of specific heat at constant pressure values, for two-dimensional table lookup based on pressure and temperature. The matrix size must correspond to the sizes of the pressure and temperature vectors. The table rows correspond to the **Temperature vector** values, and the columns correspond to the **Pressure vector** values.

The default is [1.00320557010184, 1.0041691525775, 1.00707743852905,
 1.01198388197822, 1.02202562270308, 1.05412155485561, 1.1152747482059,
 1.27776397304517, 2.72714600163742, 2.84169679152775; 1.00305665529268,
 1.00385495020235, 1.00626171270995, 1.01031300874797, 1.01856901047877,
 1.04463433942978, 1.09292211458076, 1.21254825029424, 1.91568051250176,
 2.88130807863504; 1.00295209523415, 1.00362423915973, 1.00564889473126,
 1.00905096181284, 1.01596048941702, 1.03756832264122, 1.07677086800609,
 1.16935473257714, 1.61274080548658, 2.49705970753673; 1.00288306639866,
 1.00345683185285, 1.00518392449607, 1.0080818551325, 1.013951550188,
 1.03217016931395, 1.0646970984316, 1.13890069327182, 1.45314157485364,
 2.09724000634052; 1.00284513880189, 1.00334073461589, 1.00483166592398,
 1.00733042924969, 1.01238048621931, 1.02796073691597, 1.05542748615461,
 1.11647221332449, 1.35460688567128, 1.82497183987797; 1.00283682944629,
 1.00326928939605, 1.00456966478083, 1.00674697608445, 1.01113942325315,
 1.02462437914299, 1.04815719159727, 1.09940879194009, 1.2879608304898,
 1.64561032045572; 1.00329934910967, 1.00354447996773, 1.00428050378407,
 1.00550931299369, 1.00797480662471, 1.01543403864644, 1.02807301255901,
 1.05410032215668, 1.1370595668667, 1.27667353640799; 1.00493370280677,
 1.00509133864519, 1.00556434123231, 1.00635299455648, 1.0079314755531,
 1.01267611184663, 1.02061296964177, 1.03658189787277, 1.08483165641663,
 1.16220451192724; 1.0082099496314, 1.00831954600127, 1.0086482943854,
 1.00919607243928, 1.01029111748123, 1.01357213131286, 1.01902645924599,
 1.02987923459813, 1.06187268740862, 1.11203831447519; 1.01341075301128,
 1.01349109533105, 1.01373205197484, 1.01413341227812, 1.01493525704896,
 1.0173338224635, 1.02130841135621, 1.02917221070611, 1.05207076302405,
 1.08759133887527; 1.02055448721779, 1.02061569051992, 1.02079923080097,
 1.02110489958894, 1.02171537134855, 1.02353991759304, 1.02655828571218,
 1.03251272842721, 1.04974356502866, 1.07634460433342; 1.02943075893538,
 1.02947876408642, 1.02962271860113, 1.02986244006153, 1.03034112568935,
 1.03177118158355, 1.03413496644912, 1.03879129407206, 1.05222519740105,

1.07293262035703; 1.03968665510974, 1.03972518411936, 1.03984071999137,
 1.04003310960439, 1.04041725327376, 1.041564650546, 1.04346052313106,
 1.04719276076415, 1.05794765935823, 1.07453027993477; 1.05091575080444,
 1.05094725491763, 1.05104172492289, 1.05119903411067, 1.05151312659059,
 1.05245123964959, 1.05400115405601, 1.0570518469246, 1.06584135589456,
 1.0794113758917; 1.06272444629524, 1.0627506054135, 1.06282904784038,
 1.06295966903439, 1.06322047753513, 1.06399946734372, 1.06528655888261,
 1.06782023265042, 1.07512332807677, 1.08641966249765; 1.07477034892988,
 1.07479235240011, 1.07485833393168, 1.074968207086, 1.07518759463145,
 1.07584291626895, 1.0769258365524, 1.07905818118525, 1.08520928708625,
 1.09474437534767; 1.08677790841906, 1.08679662142856, 1.08685273646968,
 1.08694618174342, 1.08713277427591, 1.08769019167808, 1.08861150786242,
 1.09042630393675, 1.0956664697493, 1.10380810039455; 1.09853924561766,
 1.09855531217341, 1.09860349180281, 1.09868372452655, 1.0988439410057,
 1.09932261842318, 1.10011396834262, 1.10167339765528, 1.10618098976507,
 1.11320064188411; 1.10990704597111, 1.1099209549981, 1.10996266523763,
 1.1100321262765, 1.11017083908205, 1.11058531966622, 1.11127070044324,
 1.11262188650428, 1.11653182839406, 1.12263462387947; 1.12078425520571,
 1.12079638431193, 1.12083275738759, 1.12089333179633, 1.12101430361411,
 1.12137581669106, 1.12197375228244, 1.12315305305123, 1.12656934044836,
 1.13191335135503; 1.13111337419617, 1.13112401943828, 1.13115594304562,
 1.13120910873864, 1.13131528950622, 1.1316326382829, 1.1321576486921,
 1.133193557109, 1.13619765700072, 1.14090677336816; 1.14086676413044,
 1.14087616073784, 1.14090434018739, 1.14095127142654, 1.14104500497982,
 1.14132518387657, 1.14178880550594, 1.14270395912918, 1.14536059930382,
 1.14953335297966; 1.20998897700762, 1.20999232462239, 1.21000236461065,
 1.21001908842037, 1.21005250052026, 1.21015245505421, 1.21031812238981,
 1.21064610721112, 1.21160534237447, 1.2131333248498; 1.24627002647896,
 1.24627150992239, 1.24627595914621, 1.24628337083698, 1.24629818045446,
 1.24634250007779, 1.24641600783175, 1.24656172222959, 1.24698925111466,
 1.24767439351222] kJ/kg/K.

Dynamic viscosity table – Real gas

matrix

The matrix of gas dynamic viscosity values, for two-dimensional table lookup based on pressure and temperature. The matrix size must correspond to the sizes of the pressure and temperature vectors. The table rows correspond to the **Temperature vector** values, and the columns correspond to the **Pressure vector** values.

The default is [10.3604759816291, 10.3621937105615, 10.367418080323,
 10.3763644808926, 10.3951680979376, 10.4591651061333, 10.5933716881858,
 10.987254253246, 14.2772531553238, 32.0640438449792; 10.9746289690763,
 10.9763664398934, 10.9816387279312, 10.9906266336238, 11.0093642467284,
 11.0718587716065, 11.1983236137388, 11.5473941927686, 13.80253919041,
 24.9939602435999; 11.5769808861739, 11.5787223615312, 11.583997583134,
 11.5929596854945, 11.6115273211419, 11.6724951531801, 11.7924961900945,
 12.1087051696631, 13.877110798453, 21.116594552727; 12.1679978404446,
 12.1697321846652, 12.1749786474567, 12.1838679911623, 12.2021948975387,
 12.2616345418299, 12.3760758859915, 12.6668675990252, 14.1428657176503,
 19.3071407602894; 12.7481340945151, 12.7498534307175, 12.7550488382933,
 12.7638328050275, 12.7818715111764, 12.8397983982715, 12.9493527652261,
 13.219710064689, 14.4977302956073, 18.4912497838941; 13.3178287127886,

13.3195275009732, 13.3246562824217, 13.3333124788606, 13.3510320108467,
 13.4074736485989, 13.5126622664861, 13.7661027806947, 14.899955910071,
 18.1598898734484; 16.0238079360955, 16.0253666205878, 16.0300598781818,
 16.0379392868608, 16.053912766856, 16.1035452030589, 16.191935922972,
 16.3896878927643, 17.1441537726836, 18.884784635824; 18.52440681911,
 18.5258161888637, 18.5300544205539, 18.5371518302972, 18.5514726042656,
 18.5954348735271, 18.6719871639656, 18.8370142902459, 19.4199345833489,
 20.6372440527557; 20.8554697382748, 20.8567443227183, 20.8605744863513,
 20.8669794187721, 20.8798689481311, 20.9191686895803, 20.986731360858,
 21.1292934645023, 21.6108097810363, 22.5597942408407; 23.0448221434545,
 23.045979989021, 23.0494578145236, 23.0552684559991, 23.066943007854,
 23.1023882980934, 23.1628395000098, 23.2886837392316, 23.7017174154222,
 24.4855712668615; 25.1142940198393, 25.1153517007868, 25.1185277377966,
 25.1238310913551, 25.134474978366, 25.1667008098841, 25.2213696262148,
 25.3341476729655, 25.6971247787102, 26.3680691177854; 27.0812512958046,
 27.0822229155161, 27.0851399370654, 27.090008831616, 27.0997734703662,
 27.1292797980586, 27.1791493931782, 27.2813713204132, 27.6058089463127,
 28.1940892928547; 28.9597031311334, 28.9606004148378, 28.9632938713994,
 28.9677883052456, 28.9767971072563, 29.003981220333, 29.049802171496,
 29.1432877879835, 29.4369487246637, 29.9617496853132; 30.7610994221431,
 30.7619320706261, 30.7644312356394, 30.7686005669006, 30.7769543732151,
 30.8021356113189, 30.8444950072474, 30.930615426825, 31.1990251289407,
 31.6733383603249; 32.4949104479269, 32.495686507264, 32.4980156297584,
 32.5019006419902, 32.5096823953741, 32.5331204679996, 32.572486613314,
 32.6523054692036, 32.8995616662979, 33.3326296905506; 34.1690538279474,
 34.1697800092973, 34.1719592969339, 34.1755939162799, 34.1828723898823,
 34.2047808981699, 34.2415335080216, 34.3158950975646, 34.5451352599228,
 34.9437909088635; 35.7902135588225, 35.7908954982821, 35.7929419104473,
 35.796354572472, 35.8031872716339, 35.8237437459906, 35.8581950302175,
 35.9277821776786, 36.141470539637, 36.5109227626193; 37.3640817965628,
 37.3647242614964, 37.366652136349, 37.3698668579186, 37.3763022640149,
 37.3956556877903, 37.4280654747596, 37.4934390588987, 37.6935517485496,
 38.037872360799; 38.8955445574042, 38.8961516091219, 38.8979731565972,
 38.9010103740854, 38.9070896826512, 38.9253661946702, 38.9559530260828,
 39.0175798676091, 39.2057292225718, 39.5281681110305; 40.3888261334681,
 40.3894012555341, 40.3911269454317, 40.3940041720465, 40.3997626466744,
 40.4170699132082, 40.4460193409127, 40.5042924094875, 40.6818132979733,
 40.9850102421168; 41.8476027037491, 41.8481489035896, 41.8497877724624,
 41.8525201165967, 41.8579881512737, 41.8744187556912, 41.901889650535,
 41.9571430391398, 42.1251545145845, 42.4112860620607; 43.275092666491,
 43.2756125563266, 43.27717245166, 43.2797730284532, 43.2849769878333,
 43.3006110869677, 43.3267405070731, 43.3792607438865, 43.5387111552397,
 43.8095952349356; 56.3222978677009, 56.3226454260967, 56.3236881526017,
 56.3254262008214, 56.3289029348368, 56.3393381859437, 56.3567467614485,
 56.3916232579666, 56.4966803757243, 56.6729203832773; 68.0659308723044,
 68.0661891931937, 68.0669641709277, 68.0682558505842, 68.0708393968717,
 68.0785915136088, 68.0915165160473, 68.1173836790054, 68.1951058070175,
 68.324944028235] s* μ Pa.

Thermal conductivity table – Real gas
 matrix

The matrix of thermal conductivity values, for two-dimensional table lookup based on pressure and temperature. The matrix size must correspond to the sizes of the pressure and temperature vectors. The table rows correspond to the **Temperature vector** values, and the columns correspond to the **Pressure vector** values.

The default is [14.0896194596466, 14.0962928994967, 14.1164979631452, 14.1507959891554, 14.2217750346856, 14.4546939871274, 14.9161812046845, 16.179147415959, 25.804405174674, 52.7180018225101; 14.9897852748967, 14.9959964117649, 15.0147814269664, 15.0465994004966, 15.1121738136527, 15.3249205125914, 15.7364707474682, 16.804366372496, 22.9562135535487, 43.3833329114209; 15.8762358183473, 15.8820425495945, 15.8995893400156, 15.9292585639261, 15.9902051816438, 16.1862002209989, 16.5586124847781, 17.4902098051164, 22.1528330719619, 36.7631825210105; 16.7494370491423, 16.7548870807582, 16.7713444308513, 16.7991324227521, 16.8560641436321, 17.0378636225901, 17.3785258158075, 18.208203678567, 22.0215304626411, 32.8930646908785; 17.6098605465768, 17.6149936632712, 17.630484995726, 17.6566114067637, 17.7100224746424, 17.8796016859028, 18.1938407516341, 18.9436656412164, 22.1982657206096, 30.777075291909; 18.4579748810943, 18.4628246117595, 18.4774535090791, 18.5021011959309, 18.5523972214712, 18.711323043571, 19.0031367814908, 19.6882500463792, 22.5419671243229, 29.6342111066617; 22.5295267397569, 22.5333162187732, 22.5447280689821, 22.5638923617408, 22.6027614529468, 22.7236585032217, 22.9392024892054, 23.4208794883241, 25.2225807004481, 29.1026020666553; 26.3559981789686, 26.3590969468369, 26.3684204301145, 26.3840499604917, 26.4156464271078, 26.513103457303, 26.6842088495029, 27.0572951588736, 28.3894828113755, 31.116172718458; 29.9792852821715, 29.9818997749358, 29.9897623413869, 30.00293009458, 30.0295024972172, 30.1110920118903, 30.2531619878245, 30.5589676635475, 31.6266997546166, 33.7713242983997; 33.4325033482028, 33.4347601010226, 33.4415445093619, 33.452898913298, 33.4757833616242, 33.5458253676961, 33.6670810329056, 33.9257157838018, 34.8145797429682, 36.5779479385118; 36.7418926226882, 36.7438747571878, 36.7498320768626, 36.7597972498181, 36.7798631433013, 36.8411331524036, 36.9467440951269, 37.1704725669481, 37.9301765555925, 39.4231380752276; 39.9284529105416, 39.9302178746677, 39.9355214503588, 39.9443896273473, 39.9622338457127, 40.0166204072958, 40.1100512227945, 40.3069184840544, 40.9690202447697, 42.2600441586569; 43.0091886076603, 43.0107777144628, 43.0155521112874, 43.0235329814107, 43.0395826599714, 43.0884283290468, 43.1721143078645, 43.3476855159646, 43.9335022352695, 45.0680736353814; 45.9980307084909, 45.9994746322735, 46.0038122846688, 46.0110612730578, 46.0256323609001, 46.0699251693948, 46.1456429179092, 46.3039277327615, 46.8285393394153, 47.8384752869958; 48.9065184578233, 48.9078405799009, 48.9118119135018, 48.9184473324447, 48.9317799444647, 48.9722677483055, 49.0413526520279, 49.1853349000785, 49.6597989829678, 50.5682381944555; 51.7443071656044, 51.745525704373, 51.7491855737184, 51.7552995100503, 51.767580290433, 51.8048425014379, 51.8683232820388, 52.0002826738489, 52.4329446456495, 53.257237057186; 54.5195506971908, 54.5206801157423, 54.5240720551906, 54.5297375484385, 54.5411143746451, 54.5756088111794, 54.634294449072, 54.7560113342558, 55.1533210316785, 55.9068028282042; 57.2391930354411, 57.240245001546, 57.2434041225393, 57.2486800512407, 57.2592720210842, 57.2913665575319, 57.3459044849862, 57.4587952617007, 57.8258378690484, 58.5189735848795; 59.909193195724, 59.9101772544579, 59.9131322744835, 59.9180667746078, 59.927971180289, 59.9579657490833, 60.0088820020188,

60.1140915894389, 60.4549450900886, 61.0960905721369; 62.5347007241231,
 62.5356247803198, 62.5383994775854, 62.5430323913023, 62.5523297075283,
 62.5804719765559, 62.6281996229267, 62.7266663614742, 63.0446493817009,
 63.6405775919021; 65.1201941202401, 65.1210647917066, 65.1236790697485,
 65.1280437362695, 65.1368012634004, 65.1632980493667, 65.208197772669,
 65.3007000594371, 65.5985480944572, 66.1548211331343; 67.6695911218085,
 67.6704140026442, 67.6728846837353, 67.6770092731228, 67.6852838475087,
 67.7103095833674, 67.7526848810435, 67.8398753764544, 68.1198688564118,
 68.6411071959795; 91.7767359068442, 91.777262593984, 91.7788435541879,
 91.7814814816141, 91.7867685465899, 91.8027190941739, 91.8295990025123,
 91.8844513412603, 92.0574313529676, 92.3717892854824; 114.482791123862,
 114.483174899984, 114.484326734451, 114.48624814467, 114.490097281331,
 114.501695087767, 114.521191867175, 114.560805453965, 114.684476478976,
 114.905858092359] mW/m/K.

Isothermal bulk modulus table – Real gas

matrix

The matrix of isothermal bulk modulus values, for two-dimensional table lookup based on pressure and temperature. The matrix size must correspond to the sizes of the pressure and temperature vectors. The table rows correspond to the **Temperature vector** values, and the columns correspond to the **Pressure vector** values.

The default is [0.00999390291790365, 0.019975596477506, 0.0498471914323395,
 0.0993868331912267, 0.197531539031374, 0.484259268054388, 0.934728751330763,
 1.71724988255064, 2.82175958388315, 17.9919018393343; 0.00999501582294484,
 0.019980055533591, 0.049875200783815, 0.0994998167287049, 0.197991211869307,
 0.487286155539529, 0.947989796271856, 1.78170846489774, 3.51015719198841,
 10.7214594343083; 0.00999589475132194, 0.0199835757487714, 0.049897286918626,
 0.0995887316618911, 0.198351512022939, 0.489628938251755, 0.958019328446712,
 1.82784028886759, 3.90652386037032, 8.82854192149657; 0.0099965989898174,
 0.019986395438532, 0.0499149614555173, 0.0996597753736587, 0.198638486959409,
 0.491476766853061, 0.965793651429376, 1.8622384486163, 4.16822627329493,
 8.63159094055481; 0.00999717031901561, 0.0199886824058013,
 0.0499292860323451, 0.0997172816775755, 0.198870192541928, 0.492957196986466,
 0.971938886453558, 1.88865947552405, 4.35409769664694, 8.80496894225828;
 0.00999763885669209, 0.0199905575293509, 0.0499410238357645,
 0.09976435529279, 0.199059473751491, 0.494159033891154, 0.976874655018731,
 1.90941822598397, 4.49230805929173, 9.03075785062252; 0.00999904969083958,
 0.0199962017056947, 0.0499763157291795, 0.099905628722424, 0.199625422103725,
 0.497712504275578, 0.991197318806328, 1.96746753165082, 4.84750864309542,
 9.81988814731239; 0.00999968937543632, 0.0199987597506444,
 0.0499922905319115, 0.0999694417192566, 0.199879988705673, 0.499290875622076,
 0.99742780597502, 1.9917193153671, 4.98305568535804, 10.1656019796519;
 0.0100000107657759, 0.0200000446683482, 0.0500003092207186,
 0.100001436447427, 0.200007331564404, 0.500075036239648, 1.00048844086038,
 2.0033761441506, 5.04492274361954, 10.3235244831897; 0.0100001818834321,
 0.0200007286818531, 0.0500045757502084, 0.100018445745085, 0.200074917321734,
 0.500489132153313, 1.00209119595608, 2.00937998338643, 5.07536569274062,
 10.3970361107304; 0.0100002753602103, 0.0200011022772309, 0.0500069048874091,
 0.100027723548317, 0.200111720745558, 0.500713487337442, 1.00295213774867,
 2.01254849426626, 5.09054105189364, 10.4290658432466; 0.0100003261002374,
 0.0200013050236998, 0.0500081680546832, 0.10003274966352, 0.200131614263656,

0.500833938761101, 1.00340895811156, 2.01418774348902, 5.09767889148856, 10.4394281011518; 0.0100003522309734, 0.020001409397665, 0.0500088176038191, 0.100035329341532, 0.200141785852635, 0.500894802476762, 1.00363498281765, 2.01496079090984, 5.10036467376937, 10.4380608503546; 0.0100003637388679, 0.0200014553232053, 0.0500091026538814, 0.100036456359626, 0.200146189184624, 0.500920394361745, 1.00372493336464, 2.01522723999957, 5.10050925168426, 10.4302658232288; 0.0100003663834363, 0.0200014658244477, 0.0500091668449028, 0.100036703546432, 0.200147101823945, 0.500924696260925, 1.00373311072209, 2.01519203184266, 5.0991858431757, 10.4190022258625; 0.010000363608138, 0.0200014546662049, 0.0500090960380804, 0.100036413225327, 0.200145884159056, 0.50091604682661, 1.00369181722325, 2.01497646633471, 5.09702101425378, 10.4059756798434; 0.0100003575307486, 0.0200014303136357, 0.0500089430294319, 0.100035795841818, 0.200143372106845, 0.500899562667401, 1.00362082635598, 2.01465443675056, 5.09439104481377, 10.3921899924774; 0.0100003494824879, 0.0200013980876295, 0.0500087409998702, 0.100034983623855, 0.200140090642286, 0.500878452458178, 1.00353250756031, 2.01427193112161, 5.09152635088116, 10.3782418599722; 0.0100003403137851, 0.0200013613871755, 0.0500085111420191, 0.100034061002855, 0.200136374797818, 0.500854760597534, 1.00343471996805, 2.01385799749085, 5.08856963592464, 10.3644851864513; 0.0100003305738062, 0.020001322407037, 0.0500082671375832, 0.100033082469423, 0.200132440659468, 0.500829803088868, 1.00333250556547, 2.01343113818249, 5.08560952488836, 10.3511260030461; 0.010000320619021, 0.0200012825717463, 0.0500080178646851, 0.100032083368648, 0.200128428277648, 0.50080443076327, 1.00322911019209, 2.01300315461294, 5.08270063956026, 10.3382789225521; 0.0100003106805437, 0.0200012428047909, 0.0500077690769735, 0.100031086594575, 0.200124428270713, 0.500779192346422, 1.0031266152441, 2.01258152001199, 5.07987590059229, 10.3260015341067; 0.0100002276847675, 0.0200009107668482, 0.0500056928131132, 0.100022774714135, 0.200091126444364, 0.500570052304359, 1.00228355378324, 2.0091599660732, 5.05768572125532, 10.2331043051344; 0.0100001747871207, 0.0200006991625897, 0.0500043700304798, 0.100017481881046, 0.200069941554726, 0.500437395676774, 1.00175129454241, 2.00701846253067, 5.04409493230387, 10.177673467778] MPa.

Isobaric thermal expansion coefficient table – Real gas

matrix

The matrix of isobaric thermal expansion coefficient values, for two-dimensional table lookup based on pressure and temperature. The matrix size must correspond to the sizes of the pressure and temperature vectors. The table rows correspond to the **Temperature vector** values, and the columns correspond to the **Pressure vector** values.

The default is [0.00667922012518017, 0.00669181264110188, 0.00672982638662978, 0.00679398124264439, 0.00692537286982025, 0.00734607851757868, 0.00815022150220267, 0.0102947700145685, 0.0286130234990625, 0.0187953609548895; 0.00625983959082828, 0.00626970330044927, 0.00629944001931492, 0.0063494918362647, 0.00645147598349833, 0.00677332847886827, 0.00736905474929384, 0.00884048760073163, 0.0171339202490674, 0.0215315077643186; 0.00589018585424886, 0.00589803389264854, 0.00592166917961251, 0.00596136776571166, 0.00604193334161856, 0.00629337413340494, 0.00674772192415558, 0.00781179743638987, 0.0126760057430892, 0.0188265238969795; 0.00556187450638498, 0.00556820302846878, 0.00558724622228694, 0.00561917817400221, 0.00568377600485627, 0.00588362426414759,

0.00623812701166661, 0.00703668964338148, 0.0102469955676518,
0.0149536825294414; 0.00526831453344121, 0.00527347724162598,
0.00528900187610229, 0.00531499852393791, 0.00536745444819242,
0.00552860555277807, 0.00581033085505093, 0.00642649496887873,
0.00869646281395399, 0.012066880795797; 0.00500425061604882,
0.00500850505700949, 0.0050212913702516, 0.00504267877028035,
0.00508574365995129, 0.00521729340118635, 0.00544459697246268,
0.00593041440285951, 0.0076105627631654, 0.0100682280023522;
0.00400181304245358, 0.00400362618568443, 0.00400906621781262,
0.00401813493211499, 0.00403627972376539, 0.00409077021034855,
0.00418174921054286, 0.00436400960219121, 0.00490100541701872,
0.00562155604159635; 0.00333421185313783, 0.00333509006258143,
0.00333772282956964, 0.00334210457480867, 0.00335084484885457,
0.00337688044325411, 0.00341965705023207, 0.00350288036408,
0.00373210009503289, 0.0040224471305652; 0.00285760133449809,
0.00285805953708679, 0.002859432498067, 0.00286171528786412,
0.0028662604025581, 0.00287973354731443, 0.00290165734133489,
0.00294356774142556, 0.00305431746464114, 0.00318740088788344;
0.00250024882385804, 0.00250049745078393, 0.00250124215161053,
0.00250247939627841, 0.00250493924559049, 0.0025122030852077,
0.00252393267225762, 0.00254603777605415, 0.00260236336428197,
0.0026655593424356; 0.00222235848069695, 0.00222249460364612,
0.00222290216059238, 0.00222357872330766, 0.00222492178356112,
0.00222887152831193, 0.00223519655385453, 0.00224692658003741,
0.00227548830007393, 0.0023039296686364; 0.00200007258789285,
0.00200014508258874, 0.00200036200840209, 0.00200072169593176,
0.00200143415322425, 0.0020035169729466, 0.00200681155082699,
0.00201277258031385, 0.0020261922956013, 0.00203612832096266;
0.00181821703439595, 0.00181825218577952, 0.00181835725159714,
0.00181853107073621, 0.00181887389835542, 0.00181986446354205,
0.00182139265566498, 0.00182401402332969, 0.00182881438125047,
0.00182849858742558; 0.0016666793946039, 0.00166669207675119,
0.00166672984892135, 0.00166679189109629, 0.00166691257836838,
0.00166724787274696, 0.00166772010434241, 0.00166835822480996,
0.00166812873286004, 0.00166203844048902; 0.00153846054408761,
0.00153845951685567, 0.00153845623835195, 0.00153845012018689,
0.0015384354465394, 0.00153837222503799, 0.00153820477492471,
0.00153765053910512, 0.00153446053814666, 0.00152514199244349;
0.0014285620398155, 0.00142855262711849, 0.00142852424563152,
0.00142847646666545, 0.0014283791330723, 0.00142807314680965,
0.00142751797086101, 0.00142624811075033, 0.00142133264209363,
0.00141028344512266; 0.00133331886909244, 0.00133330438715713,
0.00133326083537194, 0.0013331878969186, 0.00133304070781509,
0.00133258880731091, 0.00133180225897506, 0.00133011148807634,
0.00132422665389017, 0.00131234790632645; 0.00124998256666653,
0.00124996512008288, 0.00124991270097266, 0.00124982507210603,
0.00124964883188495, 0.00124911237594876, 0.00124819330564196,
0.0012462672139215, 0.00123988413227689, 0.00122772507871996;
0.00117645154049269, 0.00117643248270835, 0.00117637524921376,
0.00117627966023511, 0.0011760877378041, 0.00117550611047333,
0.00117451781904419, 0.00117247471993822, 0.00116588996983256,
0.00115378631688242; 0.0011110913270673, 0.00111107153532981,

```
0.00111101211403981, 0.00111091292547076, 0.00111071397803639,
0.00111011264786176, 0.00110909595498133, 0.00110701172424232,
0.00110041248132117, 0.00108856745884026; 0.00105261162892232,
0.00105259167294376, 0.00105253176935347, 0.00105243181158577,
0.00105223145483156, 0.00105162691347276, 0.00105060815541382,
0.00104853139690437, 0.00104203499016234, 0.00103056828403983;
0.000999980252515709, 0.000999960500382609, 0.000999901216143385,
0.000999802316594989, 0.000999604173050738, 0.000999007033534597,
0.00099800307517343, 0.000995964605933906, 0.000989643139947452,
0.00097862116221275; 0.00066665347214985, 0.000666640277074793,
0.000666600688508808, 0.000666534696481079, 0.000666402671244935,
0.000666006273741644, 0.000665344588714039, 0.00066401774733083,
0.00066001610662481, 0.000653313601854791; 0.000499991591899168,
0.000499983183702468, 0.000499957958539613, 0.000499915914707016,
0.00049983182004726, 0.000499579482293665, 0.000499158754482985,
0.00049831678545942, 0.000495788736070635, 0.000491579671000215] 1/K.
```

Parameters

Minimum valid temperature — Perfect gas, semiperfect gas, or real gas

1 K (default)

Lowest temperature allowed in the gas network. The simulation issues an error when temperature is out of range. For semiperfect or real gas, you have an option to use the lowest and highest values of the temperature vector as the permissible range of temperatures, instead of specifying this parameter.

Maximum valid temperature — Perfect gas, semiperfect gas, or real gas

inf K (default)

Highest temperature allowed in the gas network. The simulation issues an error when temperature is out of range. For semiperfect or real gas, you have an option to use the lowest and highest values of the temperature vector as the permissible range of temperatures, instead of specifying this parameter.

Minimum valid pressure — Perfect gas, semiperfect gas, or real gas

1e-6 MPa (default)

Lowest pressure allowed in the gas network. The simulation issues an error when pressure is out of range. For real gas, you have an option to use the lowest and highest values of the pressure vector as the permissible range of pressures, instead of specifying this parameter.

Maximum valid pressure — Perfect gas, semiperfect gas, or real gas

inf MPa (default)

Highest pressure allowed in the gas network. The simulation issues an error when pressure is out of range. For real gas, you have an option to use the lowest and highest values of the pressure vector as the permissible range of pressures, instead of specifying this parameter.

Valid temperature range parameterization — Semiperfect gas

Range of gas property vectors (default) | Specified minimum and maximum temperatures

Select how the block specifies the permissible range of temperatures:

- Range of gas property vectors — Use the lowest and highest values of the **Temperature vector** in the **Physical Properties** section.
- Specified minimum and maximum temperatures — Use the **Minimum valid temperature** and **Maximum valid temperature** parameter values.

Valid pressure-temperature region parameterization — Real gas

Range of gas property tables (default) | Specified minimum and maximum values | Validity matrix

Select how the block specifies the permissible pressure-temperature region:

- Range of gas property tables — Use the lowest and highest values of the **Temperature vector** and the **Pressure vector** in the **Physical Properties** section.
- Specified minimum and maximum values — Use the **Minimum valid temperature**, **Maximum valid temperature**, **Minimum valid pressure**, and **Maximum valid pressure** parameter values.
- Validity matrix — Use the **Pressure-temperature validity matrix** parameter to specify valid pressure-temperature pairs.

Pressure-temperature validity matrix — Real gas

ones(24, 10) (default) | matrix

A matrix of validity indicators, where the rows correspond to the **Temperature vector** and the columns correspond to the **Pressure vector** in the **Physical Properties** section. Validity indicator is 1 for valid pressure-temperature pairs and 0 for invalid pressure-temperature pairs.

Dependencies

Enabled when the **Valid pressure-temperature region parameterization** parameter is set to Validity matrix.

Atmospheric pressure — Perfect gas, semiperfect gas, or real gas

0.101325 MPa (default)

Absolute pressure of the environment.

Mach number threshold for flow reversal — Perfect gas, semiperfect gas, or real gas

0.001 (default)

Mach number below which flow reversal occurs. During flow reversal, the energy flow rate at the ports smoothly transitions between the upstream value and the downstream value.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Topics

“Modeling Gas Systems”

Introduced in R2016b

Gear Box

Gear box in mechanical systems

Library: Simscape / Foundation Library / Mechanical / Mechanisms



Description

The Gear Box block represents an ideal, nonplanetary, fixed gear ratio gear box. The gear ratio is determined as the ratio of the input shaft angular velocity to that of the output shaft.

The gear box is described with the following equations:

$$\omega_S = N \cdot \omega_O$$

$$T_O = N \cdot T_S$$

$$P_S = \omega_S \cdot T_S$$

$$P_O = -\omega_O \cdot T_O$$

where

- ω_S is input shaft angular velocity.
- ω_O is output shaft angular velocity.
- N is gear ratio.
- T_S is torque on the input shaft.
- T_O is torque on the output shaft.
- P_S is power on the input shaft.
- P_O is power on the output shaft. Notice the minus sign in computing P_O . One of the network rules is that the power flowing through a conserving port is positive if it is removed (dissipated) from the circuit, and is negative if the component generates power into the system.

Connections S and O are mechanical rotational conserving ports associated with the box input and output shaft, respectively. The block positive directions are from S to the reference point and from the reference point to O.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

S – Input shaft

mechanical rotational

Mechanical rotational conserving port associated with the input shaft.

0 – Output shaft

mechanical rotational

Mechanical rotational conserving port associated with the output shaft.

Parameters

Gear ratio – Ratio of gear velocities

5 (default)

The ratio of the input shaft angular velocity to that of the output shaft. You can specify both positive and negative values.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

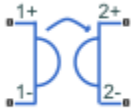
See Also

Lever | Slider-Crank | Wheel and Axle

Introduced in R2007a

Gyrator

Ideal gyrator in electrical systems



Library

Electrical Elements

Description

Gyrators can be used to implement an inductor with a capacitor. The main benefit is that an equivalent inductance can be created with a much smaller physically sized capacitance. In practice, a gyrator is implemented with an op-amp plus additional passive components.

The Gyrator block models an ideal gyrator with no losses, described with the following equations:

$$I1 = G \cdot V2$$

$$I2 = G \cdot V1$$

where

$V1$	Input voltage
$V2$	Output voltage
$I1$	Current flowing into the input + terminal
$I2$	Current flowing out of the output + terminal
G	Gyration conductance

The two electrical networks connected to the primary and secondary windings must each have their own Electrical Reference block.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Gyration conductance

The gyration conductance constant G . The default value is 1.

Ports

The block has four electrical conserving ports. Polarity is indicated by the + and - signs. Ports labeled 1+ and 1- are connected to the primary winding. Ports labeled 2+ and 2- are connected to the secondary winding.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

Introduced in R2008a

Heat Flow Rate Sensor

Ideal heat flow meter



Library

Thermal Sensors

Description

The Heat Flow Rate Sensor block represents an ideal heat flow meter, that is, a device that converts a heat flow passing through the meter into a control signal proportional to this flow. The meter must be connected in series with the component whose heat flow is being monitored.

Connections A and B are thermal conserving ports. Port H is a physical signal port that outputs the heat flow value.

The block positive direction is from port A to port B.

Ports

The block has the following ports:

A

Thermal conserving port associated with the sensor positive probe.

B

Thermal conserving port associated with the sensor negative probe.

H

Physical signal output port for heat flow.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS-Simulink Converter

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007b

Heat Flow Rate Source

Constant source of thermal energy, characterized by heat flow

Library: Simscape / Foundation Library / Thermal / Thermal Sources



Description

The Heat Flow Rate Source block represents an ideal source of thermal energy that is powerful enough to maintain specified heat flow at its outlet regardless of the temperature difference across the source.

The source generates constant heat flow rate, defined by the **Heat flow rate** parameter value.

Ports

Conserving

A — Source inlet

thermal

Thermal conserving port associated with the source inlet.

B — Source outlet

thermal

Thermal conserving port associated with the source outlet.

Parameters

Heat flow rate — Heat flow rate through the source

0 W (default)

Constant Heat flow rate through the source. Positive heat flow rate flows from port A to port B.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Heat Flow Rate Source

Introduced in R2017b

Humidity & Trace Gas Sensor (MA)

Measure humidity and trace gas levels in moist air network

Library: Simscape / Foundation Library / Moist Air / Sensors



Description

The Humidity & Trace Gas Sensor (MA) block measures the amount of moisture and trace gas in a moist air network. There is no mass or energy flow through the sensor.

The physical signal ports **W** and **G** report the moisture level and trace gas level, respectively, measured at port **A**. Connect these ports to PS-Simulink Converter blocks to transform the output physical signals into Simulink signals, for example, for plotting or additional data processing.

The sensor measurements correspond to the moisture and trace gas level upstream of the measured node. For example, if port **A** of the Humidity & Trace Gas Sensor (MA) is connected to a node between pipe 1 and pipe 2 and the moist air flows from pipe 1 to pipe 2, then the moisture and trace gas level measured at port **A** are those of the moist air volume in pipe 1. If the air flow then switches direction and flows from pipe 2 to pipe 1, then the moisture and trace gas level measured at port **A** are those of the moist air volume in pipe 2. If the moisture and trace gas levels of the two moist air volumes are different, then the measured values change when the flow direction switches. If there are two or more upstream flow paths merging at the node, then the moisture and trace gas level measurement at the node represents the weighted average based on the ideal mixing of the merging flow.

Ports

Output

W — Moisture level measurement, unitless

physical signal

Physical signal output port for moisture level measurement. To select the quantity you want to measure, use the **Measured moisture specification** parameter.

G — Trace gas level measurement, unitless

physical signal

Physical signal output port for trace gas level measurement. To select the quantity you want to measure, use the **Measured trace gas specification** parameter.

Conserving

A — Sensor inlet

moist air

Moist air conserving port. Connect this port to the node in the moist air network where you want to measure the moisture and trace gas level.

Parameters

Measured moisture specification — Select the moisture property to measure

Relative humidity (default) | Specific humidity | Mole fraction | Humidity ratio

Select the property to measure:

- Relative humidity — Physical signal port **W** reports the relative humidity.
- Specific humidity — Physical signal port **W** reports the specific humidity.
- Mole fraction — Physical signal port **W** reports the water vapor mole fraction.
- Humidity ratio — Physical signal port **W** reports the humidity ratio.

Measured trace gas specification — Select the trace gas property to measure

Mass fraction (default) | Mole fraction

Select the property to measure:

- Mass fraction — Physical signal port **G** reports the trace gas mass fraction.
- Mole fraction — Physical signal port **G** reports the trace gas mole fraction.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Measurement Selector (MA) | Pressure & Temperature Sensor (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Hydraulic Cap

Hydraulic port terminator with zero flow

Library: Simscape / Foundation Library / Hydraulic / Hydraulic Elements



Description

The Hydraulic Cap block represents a hydraulic plug, that is, a hydraulic port with zero flow through it.

You can use this block to terminate hydraulic ports on other blocks that you want to cap. This is not necessary because if you leave a conserving port unconnected, the physical network sets all the Through variables at that port to 0. However, terminator blocks improve diagram readability.

You can also use this block to set the initial gauge pressure at a node.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

P — Zero flow rate

hydraulic

Hydraulic conserving port with zero flow rate.

Compatibility Considerations

Capping unconnected ports is no longer required

Behavior changed in R2019b

This block is no longer required. Starting in R2019b, the restriction that disallowed unconnected conserving ports in Simscape models has been lifted. Now, if you leave a conserving port unconnected, the physical network sets all the Through variables at this port to 0. However, you can still use terminator blocks to improve diagram readability.

There are no plans to remove the terminator blocks. All existing models that use these blocks work the same as in previous releases.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Cap (2P) | Cap (G) | Cap (MA) | Cap (TL) | Open Circuit | Perfect Insulator | Rotational Free End | Translational Free End

Introduced in R2012b

Hydraulic Constant Flow Rate Source

Ideal source of hydraulic energy, characterized by constant flow rate



Library

Hydraulic Sources

Description

The 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.

Block connections T and P correspond to the hydraulic inlet and outlet ports, respectively. The block positive direction is from port T to port P.

Parameters

Source flow rate

Specifies the flow rate through the source. The default value is $0.001 \text{ m}^3/\text{s}$.

Ports

The block has the following ports:

T

Hydraulic conserving port associated with the source inlet.

P

Hydraulic conserving port associated with the source outlet.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Hydraulic Flow Rate Sensor | Hydraulic Flow Rate Source

Introduced in R2011a

Hydraulic Constant Mass Flow Rate Source

Ideal source of mechanical energy in hydraulic network, characterized by constant mass flow rate



Library

Hydraulic Sources

Description

The Hydraulic Constant Mass Flow Rate Source block represents an ideal mechanical energy source in a hydraulic network that can maintain a constant mass flow rate regardless of the pressure differential across the source. The source does not generate any losses due to friction. The **Mass flow rate** parameter specifies the mass flow rate through the source.

Block connections T and P correspond to the hydraulic inlet and outlet ports, respectively. A positive mass flow rate causes liquid to flow from port T to port P.

Parameters

Mass flow rate

Specifies the mass flow rate through the source. The default value is 0.001 kg/s.

Ports

The block has the following ports:

T

Hydraulic conserving port associated with the source inlet.

P

Hydraulic conserving port associated with the source outlet.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Hydraulic Mass Flow Rate Source

Introduced in R2016b

Hydraulic Constant Pressure Source

Ideal source of hydraulic energy, characterized by constant pressure



Library

Hydraulic Sources

Description

The 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

$$p = p_P - p_T$$

where p_P , p_T are the gauge pressures at the source ports.

Block connections T and P correspond to the hydraulic inlet and outlet ports, respectively. The block positive direction is from port P to port T.

Parameters

Pressure

Specifies the pressure difference between the source inlet and outlet. The default value is 1e6 Pa.

Ports

The block has the following ports:

T

Hydraulic conserving port associated with the source inlet.

P

Hydraulic conserving port associated with the source outlet.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Hydraulic Pressure Sensor | Hydraulic Pressure Source

Introduced in R2011a

Hydraulic Flow Rate Sensor

Ideal flow meter



Library

Hydraulic Sensors

Description

The Hydraulic Flow Rate Sensor block represents an ideal flow meter, that is, a device that converts volumetric flow rate through a hydraulic line into a control signal proportional to this flow rate. Connection Q is a physical signal port that outputs the volumetric flow rate value. Connection M is a physical signal port that outputs the mass flow rate value. The sensor is ideal because it does not account for inertia, friction, delays, pressure loss, and so on.

Connections A and B are conserving hydraulic ports connecting the sensor to the hydraulic line. The sensor positive direction is from A to B. This means that the flow rate is positive if it flows from A to B.

Ports

The block has the following ports:

A

Hydraulic conserving port associated with the sensor positive probe.

B

Hydraulic conserving port associated with the sensor negative (reference) probe.

Q

Physical signal port that outputs the volumetric flow rate value.

M

Physical signal port that outputs the mass flow rate value.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Hydraulic Constant Flow Rate Source | Hydraulic Flow Rate Source | PS-Simulink Converter

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2009b

Hydraulic Flow Rate Source

Ideal source of hydraulic energy, characterized by flow rate



Library

Hydraulic Sources

Description

The Hydraulic 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. Block connections T and P correspond to the hydraulic inlet and outlet ports, respectively, and connection S represents a control signal port. The flow rate through the source is directly proportional to the signal at the control port S. The entire variety of Simulink signal sources can be used to generate desired flow rate variation profile.

The block positive direction is from port T to port P. This means that the flow rate is positive if it flows from T to P. The pressure differential is determined as $p = p_P - p_T$ and is negative if pressure at the source outlet is greater than pressure at its inlet. The power generated by the source is negative if the source delivers energy to port P.

Ports

The block has the following ports:

T

Hydraulic conserving port associated with the source inlet.

P

Hydraulic conserving port associated with the source outlet.

S

Control signal port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Hydraulic Constant Flow Rate Source | Hydraulic Flow Rate Sensor

Introduced in R2009b

Hydraulic Mass Flow Rate Source

Ideal source of mechanical energy in hydraulic network, characterized by mass flow rate



Library

Hydraulic Sources

Description

The Hydraulic Mass Flow Rate Source block represents an ideal mechanical energy source in a hydraulic network that can maintain a specified mass flow rate regardless of the pressure differential across the source. The source does not generate any losses due to friction. Block connections T and P correspond to the hydraulic inlet and outlet ports, respectively. The mass flow rate through the source is directly proportional to the signal at the control port M.

A positive mass flow rate causes liquid to flow from port T to port P.

Ports

The block has the following ports:

T

Hydraulic conserving port associated with the source inlet.

P

Hydraulic conserving port associated with the source outlet.

M

Control signal port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Hydraulic Constant Mass Flow Rate Source

Introduced in R2016b

Hydraulic Piston Chamber

Variable volume hydraulic capacity in cylinders



Library

Hydraulic Elements

Description

Note Starting in Release R2014a, you can specify fluid compressibility directly in the hydro-mechanical converter blocks. It is recommended that, instead of using the Hydraulic Piston Chamber block connected to a converter, you use the **Compressibility** parameter in the converter block dialog box, because the new method provides more accurate results and also because the Hydraulic Piston Chamber block may be removed in a future release. For more information, see the R2014a Release Notes.

The Hydraulic Piston Chamber block models fluid compressibility in a chamber created by a piston of a cylinder. The fluid is considered to be a mixture of liquid and a small amount of entrained, nondissolved gas. Use this block together with the Translational Hydro-Mechanical Converter block. The Hydraulic Piston Chamber block takes into account only the flow rate caused by fluid compressibility. The fluid volume consumed to create piston velocity is accounted for in the Translational Hydro-Mechanical Converter block.

The chamber is simulated according to the following equations (see [1 on page 1-232, 2 on page 1-232]):

$$q = \frac{V_0 + A(x_0 + x \cdot or)}{E} \cdot \frac{dp}{dt}$$

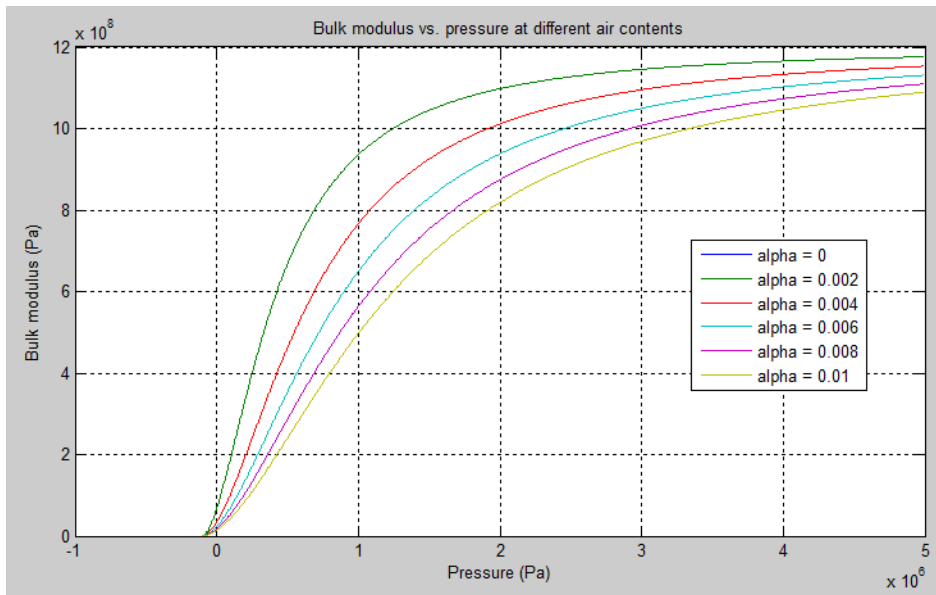
$$E = E_l \frac{1 + \alpha \left(\frac{p_a}{p_a + p} \right)^{1/n}}{1 + \alpha \frac{p_a^{1/n}}{n \cdot (p_a + p)^{\frac{n+1}{n}}} E_l}$$

where

q	Flow rate due to fluid compressibility
V_0	Dead volume
A	Effective piston area

x_0	Piston initial position
x	Piston displacement from initial position
or	Chamber orientation with respect to the globally assigned positive direction. If displacement in positive direction increases the volume of the chamber, or equals 1. If displacement in positive direction decreases the volume of the chamber, or equals -1.
E	Fluid bulk modulus
E_l	Pure liquid bulk modulus
p	Gauge pressure of fluid in the chamber
p_α	Atmospheric pressure
α	Relative gas content at atmospheric pressure, $\alpha = V_G/V_L$
V_G	Gas volume at atmospheric pressure
V_L	Volume of liquid
n	Gas-specific heat ratio

The main objective of representing fluid as a mixture of liquid and gas is to introduce an approximate model of cavitation, which takes place in a chamber if pressure drops below fluid vapor saturation level. As it is seen in the graph below, the bulk modulus of a mixture decreases as the gauge pressure approaches zero, thus considerably slowing down further pressure change. At gauge pressures far above zero, a small amount of undissolved gas has practically no effect on the system behavior.



Cavitation is an inherently thermodynamic process, requiring consideration of multiple-phase fluids, heat transfers, etc., and as such cannot be accurately simulated with Simscape software. But the simplified version implemented in the block is good enough to signal if pressure falls below dangerous level, and to prevent computation failure that normally occurs at negative pressures.

If pressure falls below absolute vacuum (-101325 Pa), the simulation stops and an error message is displayed.

Port A is a hydraulic conserving port associated with the chamber inlet. Port P is a physical signal port that controls piston displacement.

The block positive direction is from port A to the reference point. This means that the flow rate is positive if it flows into the chamber.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Basic Assumptions and Limitations

- Fluid density remains constant.
- Chamber volume cannot be less than the dead volume.
- Fluid fills the entire chamber volume.

Parameters

Piston area

Effective piston area. The default value is $5e-4 \text{ m}^2$.

Piston initial position

Initial offset of the piston from the cylinder cap. The default value is 0.

Chamber orientation

Specifies chamber orientation with respect to the globally assigned positive direction. The chamber can be installed in two different ways, depending upon whether the piston motion in the positive direction increases or decreases the volume of the chamber. If piston motion in the positive direction decreases the chamber volume, set the parameter to `Positive displacement decreases volume`. The default value is `Positive displacement increases volume`.

Chamber dead volume

Volume of fluid in the chamber at zero piston position. The default value is $1e-4 \text{ m}^3$.

Specific heat ratio

Gas-specific heat ratio. The default value is 1.4.

Restricted Parameters

When your model is in Restricted editing mode, you cannot modify the following parameter:

- **Chamber orientation**

All other block parameters are available for modification.

Global Parameters

Parameters determined by the type of working fluid:

- **Fluid density**
- **Fluid kinematic viscosity**

Use the Hydraulic Fluid block or the Custom Hydraulic Fluid block to specify the fluid properties.

Ports

The block has the following ports:

A

Hydraulic conserving port associated with the chamber inlet.

P

Physical signal port that controls piston displacement.

References

[1] Manring, N.D., *Hydraulic Control Systems*, John Wiley & Sons, New York, 2005

[2] Meritt, H.E., *Hydraulic Control Systems*, John Wiley & Sons, New York, 1967

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Constant Volume Hydraulic Chamber | Rotational Hydro-Mechanical Converter | Translational Hydro-Mechanical Converter | Variable Hydraulic Chamber

Introduced in R2009b

Hydraulic Pressure Sensor

Ideal pressure sensing device



Library

Hydraulic Sensors

Description

The Hydraulic Pressure Sensor block represents an ideal hydraulic pressure sensor, that is, a device that converts hydraulic pressure differential measured between two points into a control signal proportional to this pressure. The sensor is ideal because it does not account for inertia, friction, delays, pressure loss, and so on.

Connections A and B are conserving hydraulic ports connecting the sensor to the hydraulic line. Connection P is a physical signal port that outputs the pressure value. The sensor positive direction is from A to B. This means that the pressure differential is determined as $p = p_A - p_B$.

Ports

The block has the following ports:

A

Hydraulic conserving port associated with the sensor positive probe.

B

Hydraulic conserving port associated with the sensor negative (reference) probe.

P

Physical signal port that outputs the pressure value.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Hydraulic Constant Pressure Source | Hydraulic Pressure Source | PS-Simulink Converter

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2009b

Hydraulic Pressure Source

Ideal source of hydraulic energy, characterized by pressure



Library

Hydraulic Sources

Description

The Hydraulic Pressure Source block represents an ideal source of hydraulic energy that is powerful enough to maintain specified pressure at its outlet regardless of the flow rate consumed by the system. Block connections T and P correspond to the hydraulic inlet and outlet ports, respectively, and connection S represents a control signal port. The pressure differential across the source

$$p = p_P - p_T$$

where p_P , p_T are the gauge pressures at the source ports, is directly proportional to the signal at the control port S. The entire variety of Simulink signal sources can be used to generate desired pressure variation profile.

The block positive direction is from port P to port T. This means that the flow rate is positive if it flows from P to T. The power generated by the source is negative if the source delivers energy to port P.

Ports

The block has the following ports:

T

Hydraulic conserving port associated with the source inlet.

P

Hydraulic conserving port associated with the source outlet.

S

Control signal port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Hydraulic Constant Pressure Source | Hydraulic Pressure Sensor

Introduced in R2009b

Hydraulic Reference

Connection to atmospheric pressure



Library

Hydraulic Elements

Description

The Hydraulic Reference block represents a connection to atmospheric pressure. 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.

Ports

The block has one hydraulic conserving port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Topics

“Grounding Rules”

Introduced in R2007a

Hydraulic Resistive Tube

Hydraulic pipeline which accounts for friction losses only



Library

Hydraulic Elements

Description

The Hydraulic Resistive Tube block models hydraulic pipelines with circular and noncircular cross sections and accounts for resistive property only. In other words, the block is developed with the basic assumption of the steady state fluid momentum conditions. Neither fluid compressibility nor fluid inertia is considered in the model, meaning that features such as water hammer cannot be investigated. If necessary, you can add fluid compressibility, fluid inertia, and other effects to your model using other blocks, thus producing a more comprehensive model.

The end effects are also not considered, assuming that the flow is fully developed along the entire pipe length. To account for local resistances, such as bends, fittings, inlet and outlet losses, and so on, all the resistances are converted into their equivalent lengths, and then the total length of all the resistances is added to the pipe geometrical length.

Pressure loss due to friction is computed with the Darcy equation, in which losses are proportional to the flow regime-dependable friction factor and the square of the flow rate. The friction factor in turbulent regime is determined with the Haaland approximation (see [1] on page 1-241). The friction factor during transition from laminar to turbulent regimes is determined with the linear interpolation between extreme points of the regimes. As a result of these assumptions, the tube is simulated according to the following equations:

$$p = f \frac{(L + L_{eq})}{D_H} \frac{\rho}{2A^2} q \cdot |q|$$

$$f = \begin{cases} K_S/Re & \text{for } Re < = Re_L \\ f_L + \frac{f_T - f_L}{Re_T - Re_L} (Re - Re_L) & \text{for } Re_L < Re < Re_T \\ \frac{1}{\left(-1.8 \log_{10} \left(\frac{6.9}{Re} + \left(\frac{r/D_H}{3.7} \right)^{1.11} \right) \right)^2} & \text{for } Re > = Re_T \end{cases}$$

$$Re = \frac{q \cdot D_H}{A \cdot \nu}$$

where

p	Pressure loss along the pipe due to friction
q	Flow rate through the pipe
Re	Reynolds number
Re_L	Maximum Reynolds number at laminar flow
Re_T	Minimum Reynolds number at turbulent flow
K_s	Shape factor that characterizes the pipe cross section
f_L	Friction factor at laminar border
f_T	Friction factor at turbulent border
A	Pipe cross-sectional area
D_H	Pipe hydraulic diameter
L	Pipe geometrical length
L_{eq}	Aggregate equivalent length of local resistances
r	Height of the roughness on the pipe internal surface
ν	Fluid kinematic viscosity

The block positive direction is from port A to port B. This means that the flow rate is positive if it flows from A to B, and the pressure loss is determined as $p = p_A - p_B$.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Basic Assumptions and Limitations

- Flow is assumed to be fully developed along the pipe length.
- Fluid inertia, fluid compressibility, and wall compliance are not taken into account.

Parameters

Tube cross section type

The type of tube cross section: **Circular** or **Noncircular**. For a circular tube, you specify its internal diameter. For a noncircular tube, you specify its hydraulic diameter and tube cross-sectional area. The default value of the parameter is **Circular**.

Tube internal diameter

Tube internal diameter. The parameter is used if **Tube cross section type** is set to **Circular**. The default value is 0.01 m.

Noncircular tube cross-sectional area

Tube cross-sectional area. The parameter is used if **Tube cross section type** is set to **Noncircular**. The default value is $1e-4 \text{ m}^2$.

Noncircular tube hydraulic diameter

Hydraulic diameter of the tube cross section. The parameter is used if **Tube cross section type** is set to **Noncircular**. The default value is 0.0112 m.

Geometrical shape factor

Used for computing friction factor at laminar flow. The shape of the tube cross section determines the value. For a tube with a noncircular cross section, set the factor to an appropriate value, for example, 56 for a square, 96 for concentric annulus, 62 for rectangle (2:1), and so on [1 on page 1-241]. The default value is 64, which corresponds to a tube with a circular cross section.

Tube length

Tube geometrical length. The default value is 5 m.

Aggregate equivalent length of local resistances

This parameter represents total equivalent length of all local resistances associated with the tube. You can account for the pressure loss caused by local resistances, such as bends, fittings, armature, inlet/outlet losses, and so on, by adding to the pipe geometrical length an aggregate equivalent length of all the local resistances. The default value is 1 m.

Internal surface roughness height

Roughness height on the tube internal surface. The parameter is typically provided in data sheets or manufacturer's catalogs. The default value is $1.5e-5$ m, which corresponds to drawn tubing.

Laminar flow upper margin

Specifies the Reynolds number at which the laminar flow regime is assumed to start converting into turbulent. Mathematically, this is the maximum Reynolds number at fully developed laminar flow. The default value is 2000.

Turbulent flow lower margin

Specifies the Reynolds number at which the turbulent flow regime is assumed to be fully developed. Mathematically, this is the minimum Reynolds number at turbulent flow. The default value is 4000.

Restricted Parameters

When your model is in Restricted editing mode, you cannot modify the following parameter:

- **Tube cross section type**

All other block parameters are available for modification. The actual set of modifiable block parameters depends on the value of the **Tube cross section type** parameter at the time the model entered Restricted mode.

Global Parameters

Parameters determined by the type of working fluid:

- **Fluid density**
- **Fluid kinematic viscosity**

Use the Hydraulic Fluid block or the Custom Hydraulic Fluid block to specify the fluid properties.

Ports

The block has the following ports:

A

Hydraulic conserving port associated with the tube inlet.

B

Hydraulic conserving port associated with the tube outlet.

References

[1] White, F.M., *Viscous Fluid Flow*, McGraw-Hill, 1991

See Also

Linear Hydraulic Resistance

Introduced in R2009b

Ideal Angular Velocity Source

Ideal angular velocity source in mechanical rotational systems

Library: Simscape / Foundation Library / Mechanical / Mechanical Sources



Description

The Ideal Angular Velocity Source block represents an ideal source of angular velocity that generates velocity differential at its terminals proportional to the input physical signal. The source is ideal in a sense that it is assumed to be powerful enough to maintain specified velocity regardless of the torque exerted on the system.

Connections R and C are mechanical rotational conserving ports. Port S is a physical signal port, through which the control signal that drives the source is applied. The relative velocity (velocity differential) across the source is directly proportional to the signal at the control port S. The entire variety of Simulink signal sources can be used to generate the desired velocity variation profile.

The block positive direction is from port R to port C. This means that the velocity is measured as $\omega = \omega_R - \omega_C$, where ω_R , ω_C are the absolute angular velocities at ports R and C, respectively, and torque through the source is positive if it is directed from R to C. The power generated by the source is negative if the source delivers energy to port R.

Ports

Input

S — Control signal, rad/s

physical signal

Physical signal input port, through which the control signal that drives the source is applied.

Conserving

R — Rod

mechanical rotational

Mechanical rotational conserving port associated with the source moving part (rod).

C — Case

mechanical rotational

Mechanical rotational conserving port associated with the source reference point (case).

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Ideal Rotational Motion Sensor

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007a

Ideal Force Sensor

Force sensor in mechanical translational systems

Library: Simscape / Foundation Library / Mechanical / Mechanical Sensors



Description

The Ideal Force Sensor block represents a device that converts a variable passing through the sensor into a control signal proportional to the force. The sensor is ideal since it does not account for inertia, friction, delays, energy consumption, and so on.

Connections R and C are mechanical translational conserving ports that connect the block to the line where force is being monitored. Connection F is a physical signal port that outputs the measurement result.

The block positive direction is from port R to port C. This means that positive force applied to port R (the sensor positive probe) generates a positive output signal.

Ports

Output

F — Force, N

physical signal

Physical signal output port for force.

Conserving

R — Rod (positive probe)

mechanical translational

Mechanical translational conserving port associated with the sensor positive probe.

C — Case (reference probe)

mechanical translational

Mechanical translational conserving port associated with the sensor negative (reference) probe.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Ideal Force Source | PS-Simulink Converter

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007a

Ideal Force Source

Ideal source of mechanical energy that generates force proportional to the input signal

Library: Simscape / Foundation Library / Mechanical / Mechanical Sources



Description

The Ideal Force Source block represents an ideal source of mechanical energy that generates force proportional to the input physical signal. The source is ideal in a sense that it is assumed to be powerful enough to maintain specified force at its output regardless of the velocity at source terminals.

Connections R and C are mechanical translational conserving ports. Port S is a physical signal port, through which the control signal that drives the source is applied. You can use the entire variety of Simulink signal sources to generate the desired force variation profile. Positive signal at port S generates force acting from C to R. The force generated by the source is directly proportional to the signal at the control port S.

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. 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 negative if the source delivers energy to port R.

Ports

Input

S — Control signal, N

physical signal

Physical signal input port, through which the control signal that drives the source is applied.

Conserving

R — Rod

mechanical translational

Mechanical translational conserving port associated with the source moving part (rod).

C — Case

mechanical translational

Mechanical translational conserving port associated with the source reference point (case).

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Ideal Force Sensor | Simulink-PS Converter

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007a

Ideal Rotational Motion Sensor

Motion sensor in mechanical rotational systems

Library: Simscape / Foundation Library / Mechanical / Mechanical Sensors



Description

The Ideal Rotational Motion Sensor block represents an ideal mechanical rotational motion sensor, that is, a device that converts an across variable measured between two mechanical rotational nodes into a control signal proportional to angular velocity or angle. You can specify the initial angular position (offset) as a block parameter.

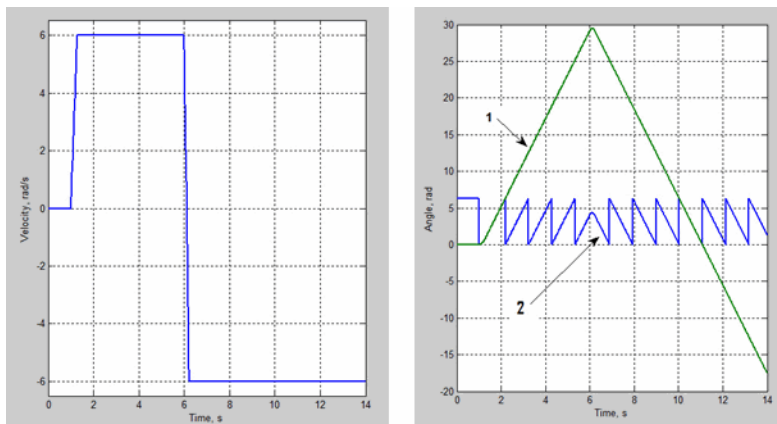
The sensor is ideal since it does not account for inertia, friction, delays, energy consumption, and so on.

Connections R and C are mechanical rotational conserving ports that connect the block to the nodes whose motion is being monitored. Connections W and A are physical signal output ports for velocity and angular displacement, respectively.

The block positive direction is from port R to port C. This means that the velocity is measured as $\omega = \omega_R - \omega_C$, where ω_R , ω_C are the absolute angular velocities at ports R and C, respectively.

The **Wrap angle to [0, 2*pi]** parameter lets you control the angular displacement output range. When set to **On**, it keeps angular displacement within the range from 0 to 2π radians (360 degrees), regardless of the number of revolutions performed by the object and the direction of rotation. When set to **Off**, the output range is unrestricted.

The figure demonstrates the difference between the two options.



In this example, the object is moving at 6 rad/s in the positive direction for the first 5 seconds, and then switches to the negative direction at the same speed. The default angular displacement output (line 1) shows that the object turned forward by 30 rad and then turned back in the negative

direction, continuing until -2π rad. If you set **Wrap angle to $[0, 2\pi]$** to **On**, the output (line 2) stays in the range from 0 to 2π rad.

Setting the **Wrap angle to $[0, 2\pi]$** parameter to **On** simplifies development of models with complex relationship between model parameters and rotation angle, such as pumps and motors.

Ports

Output

W – Angular velocity, rad/s

physical signal

Physical signal output port for angular velocity.

A – Angular displacement, rad

physical signal

Physical signal output port for angular displacement.

Conserving

R – Rod (positive probe)

mechanical rotational

Mechanical rotational conserving port associated with the sensor positive probe.

C – Case (reference probe)

mechanical rotational

Mechanical rotational conserving port associated with the sensor negative (reference) probe.

Parameters

Wrap angle to $[0, 2\pi]$ – Control output range for angular displacement

Off (default) | On

When set to **On**, keeps angular displacement output by the sensor within the range from 0 to 2π radians (360 degrees), regardless of the number of revolutions performed by the object and the direction of rotation. When set to **Off**, the output range is unrestricted.

Setting this parameter to **On** simplifies development of models with complex relationship between model parameters and rotation angle, such as pumps and motors.

Initial angle – Sensor initial offset

0 rad (default)

Sensor initial angular position (offset).

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Ideal Angular Velocity Source | PS-Simulink Converter

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007a

Ideal Torque Sensor

Torque sensor in mechanical rotational systems

Library: Simscape / Foundation Library / Mechanical / Mechanical Sensors



Description

The Ideal Torque Sensor block represents a device that converts a variable passing through the sensor into a control signal proportional to the torque. The sensor is ideal because it does not account for inertia, friction, delays, energy consumption, and so on.

Connections R and C are mechanical rotational conserving ports that connect the block to the line where torque is being monitored. Connection T is a physical signal port that outputs the measurement result.

The block positive direction is from port R to port C.

Ports

Output

T — Torque, N*m
physical signal

Physical signal output port for torque.

Conserving

R — Rod (positive probe)
mechanical rotational

Mechanical rotational conserving port associated with the sensor positive probe.

C — Case (reference probe)
mechanical rotational

Mechanical rotational conserving port associated with the sensor negative (reference) probe.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Ideal Torque Source | PS-Simulink Converter

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007a

Ideal Torque Source

Ideal source of mechanical energy that generates torque proportional to the input signal

Library: Simscape / Foundation Library / Mechanical / Mechanical Sources



Description

The Ideal Torque Source block represents an ideal source of mechanical energy that generates torque proportional to the input physical signal. The source is ideal in a sense that it is assumed to be powerful enough to maintain specified torque regardless of the angular velocity at source terminals.

Connections R and C are mechanical rotational conserving ports. Port S is a physical signal port, through which the control signal that drives the source is applied. You can use the entire variety of Simulink signal sources to generate the desired torque variation profile. Positive signal at port S generates torque acting from C to R. The torque generated by the source is directly proportional to the signal at the control port S.

The block positive direction is from port C to port R. This means that the torque is positive if it acts in the direction from C to R. The relative velocity is determined as $\omega = \omega_R - \omega_C$, where ω_R , ω_C are the absolute angular 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 negative if the source delivers energy to port R.

Ports

Input

S — Control signal, N*m

physical signal

Physical signal input port, through which the control signal that drives the source is applied.

Conserving

R — Rod

mechanical rotational

Mechanical rotational conserving port associated with the source moving part (rod).

C — Case

mechanical rotational

Mechanical rotational conserving port associated with the source reference point (case).

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Ideal Torque Sensor | Simulink-PS Converter

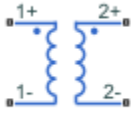
Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007a

Ideal Transformer

Ideal transformer in electrical systems



Library

Electrical Elements

Description

The Ideal Transformer block models an ideal power-conserving transformer, described with the following equations:

$$V1 = N \cdot V2$$

$$I2 = N \cdot I1$$

where

$V1$	Primary voltage
$V2$	Secondary voltage
$I1$	Current flowing into the primary + terminal
$I2$	Current flowing out of the secondary + terminal
N	Winding ratio

This block can be used to represent either an AC transformer or a solid-state DC to DC converter. To model a transformer with inductance and mutual inductance terms, use the Mutual Inductor block.

The two electrical networks connected to the primary and secondary windings must each have their own Electrical Reference block.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Winding ratio

Winding ratio of the transformer, or ratio of primary coil turns to secondary coil turns. The default value is 1.

Ports

The block has four electrical conserving ports. Polarity is indicated by the + and - signs. Ports labeled 1+ and 1- are connected to the primary winding. Ports labeled 2+ and 2- are connected to the secondary winding.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mutual Inductor

Introduced in R2007a

Ideal Translational Motion Sensor

Motion sensor in mechanical translational systems

Library: Simscape / Foundation Library / Mechanical / Mechanical Sensors



Description

The Ideal Translational Motion Sensor block represents a device that converts an across variable measured between two mechanical translational nodes into a control signal proportional to velocity or position. You can specify the initial position (offset) as a block parameter.

The sensor is ideal since it does not account for inertia, friction, delays, energy consumption, and so on.

Connections R and C are mechanical translational conserving ports that connect the block to the nodes whose motion is being monitored. Connections V and P are physical signal output ports for velocity and position, respectively.

The block positive direction is from port R to port C. This means that the velocity is measured as $v = v_R - v_C$, where v_R , v_C are the absolute velocities at ports R and C, respectively.

Ports

Output

V — Velocity, m/s

physical signal

Physical signal output port for velocity.

P — Position, m

physical signal

Physical signal output port for position.

Conserving

R — Rod (positive probe)

mechanical translational

Mechanical translational conserving port associated with the sensor positive probe.

C — Case (reference probe)

mechanical translational

Mechanical translational conserving port associated with the sensor negative (reference) probe.

Parameters

Initial position — Sensor initial offset

0 m (default)

Sensor initial position (offset).

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Ideal Translational Velocity Source | PS-Simulink Converter

Topics

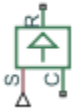
“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007a

Ideal Translational Velocity Source

Ideal velocity source in mechanical translational systems

Library: Simscape / Foundation Library / Mechanical / Mechanical Sources



Description

The Ideal Translational Velocity Source block represents an ideal source of velocity that generates velocity differential at its terminals proportional to the input physical signal. The source is ideal in a sense that it is assumed to be powerful enough to maintain specified velocity regardless of the force exerted on the system.

Connections R and C are mechanical translational conserving ports. Port S is a physical signal port, through which the control signal that drives the source is applied. The relative velocity (velocity differential) across the source is directly proportional to the signal at the control port S. The entire variety of Simulink signal sources can be used to generate the desired velocity variation profile.

The block positive direction is from port R to port C. This means that the velocity is measured as $v = v_R - v_C$, where v_R , v_C are the absolute velocities at ports R and C, respectively, and force through the source is negative if it acts from C to R. The power generated by the source is negative if the source delivers energy to port R.

Ports

Input

S — Control signal, m/s

physical signal

Physical signal input port, through which the control signal that drives the source is applied.

Conserving

R — Rod

mechanical translational

Mechanical translational conserving port associated with the source moving part (rod).

C — Case

mechanical translational

Mechanical translational conserving port associated with the source reference point (case).

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Ideal Translational Motion Sensor | Simulink-PS Converter

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007a

Inductor

Linear inductor in electrical systems



Library

Electrical Elements

Description

The Inductor block models a linear inductor, described with the following equation:

$$V = L \frac{dI}{dt}$$

where

I	Current
V	Voltage
L	Inductance
t	Time

The **Series resistance** and **Parallel conductance** parameters represent small parasitic effects. The series resistance can be used to represent the DC winding resistance or the resistance due to the skin effect. Simulation of some circuits may require the presence of a small parallel conductance. For more information, see “Modeling Best Practices”.

Connections + and - are conserving electrical ports corresponding to the positive and negative terminals of the inductor, respectively. The current is positive if it flows from positive to negative, and the voltage across the inductor is equal to the difference between the voltage at the positive and the negative terminal, $V(+)$ - $V(-)$.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Inductance

Inductance, in henries. The default value is 1 μ H.

Series resistance

Represents small parasitic effects. The series resistance can be used to represent the DC winding resistance. The default value is 0.

Parallel conductance

Represents small parasitic effects. The parallel conductance across the inductor can be used to model insulation conductance. Simulation of some circuits may require the presence of a small parallel conductance. The default value is $1e-9$ 1/Ω.

Ports

The block has the following ports:

+

Electrical conserving port associated with the inductor positive terminal.

-

Electrical conserving port associated with the inductor negative terminal.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

Introduced in R2007a

Inertia

Ideal mechanical rotational inertia

Library: Simscape / Foundation Library / Mechanical / Rotational Elements
Simscape / Multibody / Body Elements



Description

The Inertia block represents an ideal mechanical translational inertia, described with the following equation:



$$T = J \frac{d\omega}{dt}$$

where:

- T is inertia torque.
- J is inertia.
- ω is angular velocity.
- t is time.

By default, the block has one mechanical translational conserving port. The block positive direction is from its port to the reference point. This means that the inertia torque is positive if inertia is accelerated in positive direction.

In some applications, it is customary to display inertia in series with other elements in the block diagram layout. To support this use case, the **Number of graphical ports** parameter lets you display a second port on the opposite side of the block icon. The two-port variant is purely graphical: the two ports have the same angular velocity, so the block functions exactly the same whether it has one or two ports. The block icon changes depending on the value of the **Number of graphical ports** parameter.

Number of graphical ports	Block Icon
1	
2	

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

I – Port that connects the inertia to the circuit

mechanical rotational

Mechanical rotational conserving port connecting the inertia to the rest of the physical network.

J – Second graphical port

mechanical rotational

Second mechanical rotational conserving port that lets you connect the inertia in series with other elements in the block diagram. This port is rigidly connected to port **I**, therefore the difference between the one-port and two-port block representations is purely graphical.

Dependencies

This port is visible if you set the **Number of graphical ports** parameter to 2.

Parameters

Inertia – Constant inertia

0.001 kg*m² (default) | positive scalar

Inertia value. The inertia is constant during simulation.

Number of graphical ports – Select whether the block is graphically connected to the side of a circuit or in series with other elements

1 (default) | 2

Select how to connect the block to the rest of the circuit:

- 1 – The block has one conserving port, **I**, that connects it to the mechanical rotational circuit. When the block has one port, attach it to a connection line between two other blocks.
- 2 – Selecting this option exposes the second graphical port **J**, which lets you connect the block in series with other blocks in the circuit. The two ports are rigidly connected and have the same angular velocity, therefore the block functions exactly the same as if it had one port.

Block Characteristics

Data Types	
Direct Feedthrough	No
Zero-Crossing Detection	No

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mass

Topics

“Evaluating Performance of a DC Motor”

Introduced in R2007a

Infinite Flow Resistance (2P)

Perfectly insulated barrier between two-phase fluid networks



Library

Two-Phase Fluid/Elements

Description

The Infinite Flow Resistance (2P) block represents a perfectly insulated barrier between otherwise distinct two-phase fluid networks. The block joins the networks, enabling you to specify their fluid properties using a single Two-Phase Fluid Properties (2P) block. No fluid flow or heat transfer occur across the barrier.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

The block has a pair of two-phase fluid conserving ports, A and B, representing branches of adjacent two-phase fluid networks.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (IL) | Infinite Flow Resistance (MA) | Infinite Flow Resistance (G) | Infinite Flow Resistance (TL) | Infinite Hydraulic Resistance | Infinite Resistance | Infinite Thermal Resistance

Introduced in R2015b

Infinite Flow Resistance (G)

Perfectly insulated barrier between gas networks

Library: Simscape / Foundation Library / Gas / Elements



Description

The Infinite Flow Resistance (G) block represents a perfectly insulated barrier between two connected gas networks. There is no mass or energy flow through the barrier. However, gas properties are still shared between the connected gas networks.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Gas Volume”.

Ports

Conserving

A — Mass flow rate and energy flow rate are zero

gas

Gas conserving port where the mass flow rate and energy flow rate are both equal to zero.

B — Mass flow rate and energy flow rate are zero

gas

Gas conserving port where the mass flow rate and energy flow rate are both equal to zero.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (IL) | Infinite Flow Resistance (MA) | Infinite Flow Resistance (2P) | Infinite Flow Resistance (TL) | Infinite Hydraulic Resistance | Infinite Resistance | Infinite Thermal Resistance

Topics

“Modeling Gas Systems”

Introduced in R2016b

Infinite Flow Resistance (IL)

Isothermal liquid element for setting initial pressure difference between two nodes

Library: Simscape / Foundation Library / Isothermal Liquid / Elements



Description

The Infinite Flow Resistance (IL) block represents a break between two connected isothermal liquid networks. There is no liquid flow through the barrier. However, fluid properties are still shared between the connected isothermal liquid networks. Use this block to set the initial pressure difference between two isothermal liquid nodes without affecting model equations.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

A — Flow rate is zero

isothermal liquid

Isothermal liquid conserving port with no fluid flow.

B — Flow rate is zero

isothermal liquid

Isothermal liquid conserving port with no fluid flow.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (MA) | Infinite Flow Resistance (2P) | Infinite Flow Resistance (G) | Infinite Flow Resistance (TL) | Infinite Hydraulic Resistance | Infinite Resistance | Infinite Thermal Resistance

Topics

“Modeling Isothermal Liquid Systems”
“Isothermal Liquid Modeling Options”

Introduced in R2020a

Infinite Flow Resistance (MA)

Perfectly insulated barrier between moist air networks

Library: Simscape / Foundation Library / Moist Air / Elements



Description

The Infinite Flow Resistance (MA) block represents a perfectly insulated barrier between two connected moist air networks. There is no mass or energy flow through the barrier. However, moist air properties are still shared between the connected networks.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Moist Air Volume”.

Ports

Conserving

A — Mass flow rate and energy flow rate are zero

moist air

Moist air conserving port where the mass flow rate and energy flow rate are both equal to zero.

B — Mass flow rate and energy flow rate are zero

moist air

Moist air conserving port where the mass flow rate and energy flow rate are both equal to zero.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (IL) | Infinite Flow Resistance (2P) | Infinite Flow Resistance (G) | Infinite Flow Resistance (TL) | Infinite Hydraulic Resistance | Infinite Resistance | Infinite Thermal Resistance

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Infinite Flow Resistance (TL)

Perfectly insulated barrier between thermal liquid networks

Library: Simscape / Foundation Library / Thermal Liquid / Elements



Description

The Infinite Flow Resistance (TL) block represents a perfectly insulated barrier between two connected thermal liquid networks. There is no fluid or heat flow through the barrier. However, fluid properties are still shared between the connected thermal liquid networks.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

A — Flow rate is zero

thermal liquid

Thermal liquid conserving port with no fluid or heat flow.

B — Flow rate is zero

thermal liquid

Thermal liquid conserving port with no fluid or heat flow.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (IL) | Infinite Flow Resistance (MA) | Infinite Flow Resistance (2P) | Infinite Flow Resistance (G) | Infinite Hydraulic Resistance | Infinite Resistance | Infinite Thermal Resistance

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2014b

Infinite Hydraulic Resistance

Hydraulic element for setting initial pressure difference between two nodes



Library

Hydraulic Elements

Description

The Infinite Hydraulic Resistance block represents an infinite resistance, with no fluid flow through it. Use this block to set the initial pressure difference between two hydraulic nodes without affecting model equations.

The **Variables** tab lets you set the priority and initial target value for the **Pressure difference** variable prior to simulation. For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

The block has two hydraulic conserving ports.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (IL) | Infinite Flow Resistance (MA) | Infinite Flow Resistance (2P) | Infinite Flow Resistance (G) | Infinite Flow Resistance (TL) | Infinite Resistance | Infinite Thermal Resistance

Introduced in R2014b

Infinite Pneumatic Resistance

Pneumatic element for setting initial pressure and temperature difference between two nodes



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Infinite Pneumatic Resistance block represents an infinite resistance, with no gas or heat flow through it. Use this block to set the initial pressure and temperature difference between two pneumatic nodes without affecting model equations.

The **Variables** tab lets you set the priority and initial target value for the **Pressure difference** and **Temperature difference** variables prior to simulation. For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

The block has two pneumatic conserving ports.

See Also

Infinite Thermal Resistance | Infinite Flow Resistance (TL) | Infinite Hydraulic Resistance | Infinite Resistance

Introduced in R2014b

Infinite Resistance

Electrical element for setting initial voltage difference between two nodes



Library

Electrical Elements

Description

The Infinite Resistance block represents an infinite electrical resistance that draws no current. Use this block to set the initial voltage difference between two electrical nodes without affecting model equations.

The **Variables** tab lets you set the priority and initial target value for the **Voltage difference** variable prior to simulation. For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

The block has two electrical conserving ports.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (IL) | Infinite Flow Resistance (MA) | Infinite Flow Resistance (2P) | Infinite Flow Resistance (G) | Infinite Flow Resistance (TL) | Infinite Hydraulic Resistance | Infinite Thermal Resistance

Introduced in R2014b

Infinite Thermal Resistance

Thermal element for setting initial temperature difference between two nodes



Library

Thermal Elements

Description

The Infinite Thermal Resistance block represents an infinite resistance, with no heat flow through it. Use this block to set the initial temperature difference between two thermal nodes without affecting model equations.

The **Variables** tab lets you set the priority and initial target value for the **Temperature difference** variable prior to simulation. For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

The block has two thermal conserving ports.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Infinite Flow Resistance (IL) | Infinite Flow Resistance (MA) | Infinite Flow Resistance (2P) | Infinite Flow Resistance (G) | Infinite Flow Resistance (TL) | Infinite Hydraulic Resistance | Infinite Resistance

Introduced in R2014b

Interface (H-IL)

Flow connection between hydraulic and isothermal liquid networks

Library: Simscape / Foundation Library / Hydraulic / Hydraulic Utilities



Description

The Interface (H-IL) block represents a cross-domain flow connection between hydraulic and isothermal liquid networks:

- Pressure and mass flow rates are equal at the interface.
- Fluid properties are not shared across the interface. Each network has its own fluid properties.

Use this block in existing models with blocks from the Foundation > Hydraulic library, or from the Fluids > Hydraulics (Isothermal) library, to implement parts of the model using Isothermal Liquid library blocks, without converting the whole model. The Fluids > Hydraulics (Isothermal) library is available if you have a Simscape Fluids license.

The `hydraulicToIsothermalLiquid` conversion tool replaces Hydraulic blocks in your model with equivalent Isothermal Liquid blocks. If your model contains references or links to other libraries, models, or subsystems, you can use the Interface (H-IL) to restore the broken connections between the converted parts of the block diagram and the libraries, models, or subsystems that still need to be converted.

Ports

Conserving

H — Hydraulic port

hydraulic

Hydraulic conserving port representing a passage point through the interface. Connect this port to a block network built on the Simscape hydraulic domain.

IL — Isothermal liquid port

isothermal liquid

Isothermal liquid conserving port representing a passage point through the interface. Connect this port to a block network built on the Simscape isothermal liquid domain.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Custom Hydraulic Fluid | Isothermal Liquid Properties (IL)

Topics

“Upgrading Hydraulic Models To Use Isothermal Liquid Blocks”

“Upgrading Simscape Fluids Models Containing Hydraulics (Isothermal) Blocks” (Simscape Fluids)

Introduced in R2020a

Isothermal Liquid Properties (IL)

Physical properties of isothermal liquid

Library: Simscape / Foundation Library / Isothermal Liquid / Utilities



Description

The Isothermal Liquid Properties (IL) block defines the liquid properties that act as global parameters for all the blocks connected to a circuit. The default liquid is water.

Each topologically distinct isothermal liquid circuit in a diagram can have a Isothermal Liquid Properties (IL) block connected to it. If no Isothermal Liquid Properties (IL) block is attached to a circuit, the blocks in this circuit use the properties corresponding to the default Isothermal Liquid Properties (IL) block parameter values.

The Isothermal Liquid Properties (IL) block provides a choice of modeling options:

- Mixture bulk modulus: either constant or a linear function of pressure
- Entrained air: zero, constant, or a linear function of pressure

Equations used to compute various liquid properties depend on the selected isothermal liquid model. For detailed information, see “Isothermal Liquid Modeling Options”.

Ideal Fluid Models

Entrained air is the relative amount of nondissolved gas trapped in the fluid. Fluid with zero entrained air is ideal, that is, it represents pure liquid.

In its default configuration, the Isothermal Liquid Properties (IL) block models an ideal fluid with a constant bulk modulus:

- **Isothermal bulk modulus model** is Constant
- **Entrained air model** is Constant
- **Entrained air-to-liquid volumetric ratio at atmospheric pressure** is 0

In this model, the liquid bulk modulus is assumed to be constant and therefore the liquid density increases exponentially with the liquid pressure:

$$\rho_L = \rho_{L0} \cdot e^{\frac{p - p_0}{\beta_L}},$$

where:

- β_L is liquid bulk modulus.
- ρ_L is liquid density.
- ρ_{L0} is liquid density at reference pressure.

- p is liquid pressure.
- p_0 is reference pressure. By default, the block assumes reference pressure to be the atmospheric pressure, 0.101325 MPa, but you can specify a different value.

In systems where the liquid pressure can change over a wide range, and the assumption of constant bulk modulus is not valid anymore, you can use the **Isothermal bulk modulus model** parameter to define the liquid bulk modulus as a linear function of pressure:

$$\beta_L = \beta_{L0} + K_{\beta p}(p - p_0),$$

where:

- β_{L0} is liquid bulk modulus at reference pressure.
- $K_{\beta p}$ is the coefficient of proportionality between the bulk modulus and pressure increase.

If liquid pressure decreases below the reference pressure p_0 , the liquid bulk modulus value in the previous equation can become negative, which is nonphysical. To ensure that the liquid bulk modulus always stays positive, use the **Minimum valid pressure** parameter to specify the minimum valid pressure, p_{\min} :

$$p_{\min} > p_0 - \frac{\beta_{L0}}{K_{\beta p}}.$$

If liquid pressure drops below the **Minimum valid pressure** parameter value, simulation issues an error.

Fluid Models with Entrained Air

In practice, working fluid is a mixture of liquid and a small amount of entrained air. To model this type of fluid, specify a nonzero value for the **Entrained air-to-liquid volumetric ratio at atmospheric pressure** parameter, but keep the **Entrained air model** as Constant.

The mixture density at a given pressure is defined as the total mass of liquid and entrained air over the total volume of the liquid and the entrained air at that pressure. While the total mass of the mixture is conserved as the pressure changes, the mixture volume does not stay constant. The entrained air is given by a volumetric fraction:

$$\alpha_0 = \frac{V_{g0}}{V_{g0} + V_{L0}},$$

where:

- α_0 is entrained air-to-liquid volumetric ratio at reference (atmospheric) pressure.
- V_{g0} is air volume at reference pressure.
- V_{L0} is pure liquid volume at reference pressure.

The entrained air is assumed to follow the ideal gas law. The compression or expansion of air in the liquid is a polytropic process, in which the air pressure and liquid pressure are identical:

$$V_g = V_{g0} \left(\frac{p_0}{p} \right)^{1/n},$$

where:

- V_g is air volume.
- n is air polytropic index.

To model the air dissolution effects into the fluid, set the **Entrained air model** parameter to Linear function of pressure.

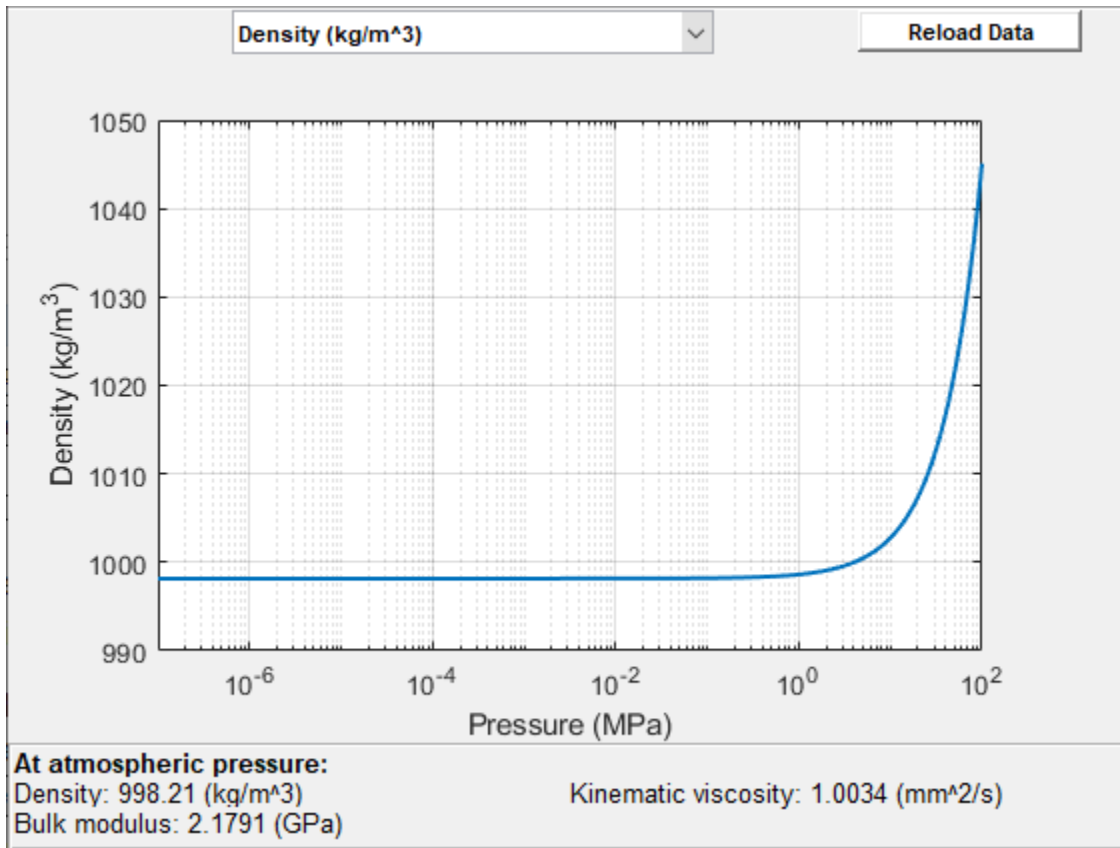
The process of dissolving air into the fluid is described by Henry's law. At pressures less than or equal to the reference pressure, p_0 (which is assumed to be equal to atmospheric pressure), all the air is assumed to be entrained. At pressures equal or higher than pressure p_c , all the entrained air has been dissolved into the liquid. At pressures between p_0 and p_c , the volumetric fraction of entrained air that is not lost to dissolution, $\theta(p)$, is a linear function of the pressure, and is approximated by a third-order polynomial function to smoothly connect the density and bulk modulus values between the three pressure regions:

$$\theta(p) = \begin{cases} 1, & p \leq p_0 \\ 0, & p \geq p_c \\ 1 - \left(3 \left(\frac{p_c - p}{p_c - p_0} \right)^2 - 2 \left(\frac{p_c - p}{p_c - p_0} \right)^3 \right), & p_0 < p < p_c \end{cases}.$$

Data Visualization

The block provides the option to plot the specified fluid properties (density and isothermal bulk modulus) as a function of pressure. Plotting the properties lets you visualize the data before simulating the model.

To plot the data, right-click the Isothermal Liquid Properties (IL) block in your model and, from the context menu, select **Foundation Library > Plot Fluid Properties**. Use the drop-down list located at the top of the plot to select the fluid property to visualize. Click the **Reload** button to regenerate a plot following a block parameter update.



Isothermal Liquid Properties Plot

Ports

Conserving

A – Connection port

isothermal liquid

Isothermal liquid conserving port that connects the block to the network. You can connect it to any point on an isothermal liquid connection line in a block diagram. When you connect the Isothermal Liquid Properties (IL) block to a connection line, the software automatically identifies the isothermal liquid blocks connected to the particular circuit and propagates the liquid properties to all the blocks in the circuit.

Settings

Liquid

Liquid density at atmospheric pressure (no entrained air) – Liquid density

998.21 kg/m³ (default) | positive scalar

Density of the isothermal liquid at atmospheric pressure, with no entrained air.

Isothermal bulk modulus model — Select the bulk modulus model

Constant (default) | Linear function of pressure

Select the bulk modulus model for the isothermal liquid:

- Constant — The bulk modulus is constant.
- Linear function of pressure — The bulk modulus is a linear function of pressure.

Isothermal bulk modulus vs. pressure increase gain — Coefficient of proportionality for linear dependency

6 (default) | positive scalar

Coefficient of proportionality between the bulk modulus and pressure increase.

DependenciesEnabled when the **Isothermal bulk modulus model** parameter is set to Linear function of pressure.**Liquid isothermal bulk modulus at atmospheric pressure (no entrained air) — Liquid isothermal bulk modulus**

2.1791e9 Pa (default) | positive scalar

Isothermal bulk modulus of the liquid at atmospheric pressure, with no entrained air.

Kinematic viscosity at atmospheric pressure — Liquid kinematic viscosity1.0034e-6 m²/s (default) | positive scalar

Kinematic viscosity of the isothermal liquid at atmospheric pressure.

Atmospheric pressure — Absolute pressure of the environment

0.101325 MPa (default) | positive scalar

Absolute pressure of the environment.

Minimum valid pressure — Lowest pressure allowed

0.1 Pa (default) | positive scalar

Lowest pressure allowed in the isothermal liquid network. The simulation issues an error when pressure is out of range.

Entrained Air**Volumetric fraction of entrained air in mixture at atmospheric pressure — Volumetric fraction of entrained air in the fluid mixture**

0 (default) | scalar in the range [0,1]

Volumetric fraction of air that is entrained in the fluid mixture at atmospheric pressure.

Air polytropic index — Air polytropic index

1.0 (default) | positive scalar

Air polytropic index. The default value of 1 represents an isothermal process, which is consistent with the assumptions of the isothermal liquid domain.

Air density at atmospheric condition — Air density at atmospheric condition

1.225 kg/m³ (default) | positive scalar

Air density at atmospheric condition.

Air dissolution model — Select the air dissolution model

Off (default) | On

Select the air dissolution model for the isothermal liquid:

- Off — The amount of entrained air is constant. Air dissolution is not modeled.
- On — The entrained air can dissolve into liquid. The amount of dissolved air is a function of pressure.

Pressure at which all entrained air is dissolved — Pressure at which all entrained air is dissolved

3 MPa (default) | positive scalar

Pressure at which all entrained air in the liquid is dissolved.

Dependencies

Enabled when the **Air dissolution model** parameter is set to On.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also**Topics**

“Modeling Isothermal Liquid Systems”

“Isothermal Liquid Modeling Options”

Introduced in R2020a

Laminar Leakage (IL)

Isothermal liquid element that models laminar leakage flow for various geometries

Library: Simscape / Foundation Library / Isothermal Liquid / Elements



Description

The Laminar Leakage (IL) block models laminar flow in isothermal liquid networks. The block includes several common flow passage geometries and a custom geometry. Select a **Cross-sectional geometry** parameter value and then specify other parameter values based on the selected option:

- **Circular** — Provide the diameter and longitudinal length.
- **Annular** — Provide the inner diameter, outer diameter, and longitudinal length.
- **Rectangular** — Provide the width, height, and longitudinal length.
- **Elliptical** — Provide the major axis, minor axis, and longitudinal length.
- **Equilateral triangular** — Provide the length of the triangle side and longitudinal length.
- **Custom** — Provide the flow resistance value, that is, the ratio of pressure loss to mass flow rate. In this configuration, the block acts as a linear hydraulic resistance.

The flow rate calculations are based on the Hagen-Poiseuille equation, which is valid for fully developed laminar flow:

$$\dot{m}_A = \begin{cases} \frac{K_{geom}}{\nu L} (p_A - p_B), & \text{common geometries} \\ \frac{\bar{\rho}}{R} (p_A - p_B), & \text{custom} \end{cases}$$

$$K_{geom} = \begin{cases} \frac{\pi d^4}{128}, & \text{circular} \\ \frac{\pi}{128} \left(d_o^4 - d_i^4 - \frac{(d_o^2 - d_i^2)^2}{\log(d_o/d_i)} \right), & \text{annular} \\ \frac{wh^3}{12} \left(1 - \frac{192h}{w\pi^5} \tanh\left(\frac{\pi w}{2h}\right) \right), & \text{rectangular} \\ \frac{\pi(ab)^3}{64(a^2 + b^2)}, & \text{elliptical} \\ l_{side}^4 \frac{\sqrt{3}}{320}, & \text{triangular} \end{cases}$$

$$\bar{\rho} = \frac{\rho_A + \rho_B}{2}$$

where:

- \dot{m}_A is the mass flow rate through port **A**.
- p_A and p_B are pressures at ports **A** and **B**, respectively.
- K_{geom} is the coefficient calculated based on the cross-sectional geometry.
- d is the diameter of the circular cross section.
- d_o is the outer diameter of the annular cross section.
- d_i is the inner diameter of the annular cross section.
- w is the width of the rectangular cross section.
- h is the height of the rectangular cross section.
- a is the major axis of the elliptical cross section.
- b is the minor axis of the elliptical cross section.
- l_{side} is the length of the side of the triangular cross section.
- L is the longitudinal length.
- R is the linear hydraulic resistance for custom geometry.
- ν is the liquid kinematic viscosity at atmospheric pressure, which is a global parameter defined by the Isothermal Liquid Properties (IL) block connected to the circuit.
- $\bar{\rho}$ is the average fluid mixture density.
- ρ_A and ρ_B are fluid mixture density values at ports **A** and **B**, respectively. Equations used to compute the fluid mixture density depend on the selected isothermal liquid model. For detailed information, see “Isothermal Liquid Modeling Options”.

Ports

Conserving

A — Inlet or outlet

isothermal liquid

Isothermal liquid conserving port associated with the inlet or outlet of the laminar leakage. This block has no intrinsic directionality.

B — Inlet or outlet

isothermal liquid

Isothermal liquid conserving port associated with the inlet or outlet of the laminar leakage. This block has no intrinsic directionality.

Parameters

Cross-sectional geometry — Select the shape of the leakage cross section

Circular (default) | Annular | Rectangular | Elliptical | Equilateral triangular | Custom

Select a cross-sectional shape and then specify other parameter values based on the selected option.

Diameter — Diameter of the circular cross section

1e-3 m (default) | positive scalar

Diameter of the circular cross section.

Dependencies

Enabled when the **Cross-sectional geometry** parameter is set to Circular.

Inner diameter — Inner diameter of the annular cross section

5e-4 m (default) | positive scalar

Inner diameter of the annular cross section.

Dependencies

Enabled when the **Cross-sectional geometry** parameter is set to Annular.

Outer diameter — Outer diameter of the annular cross section

1e-3 m (default) | positive scalar

Outer diameter of the annular cross section.

Dependencies

Enabled when the **Cross-sectional geometry** parameter is set to Annular.

Width — Width of the rectangular cross section

1e-3 m (default) | positive scalar

Width of the rectangular cross section.

Dependencies

Enabled when the **Cross-sectional geometry** parameter is set to Rectangular.

Height — Height of the rectangular cross section

0.1 m (default) | positive scalar

Height of the rectangular cross section.

Dependencies

Enabled when the **Cross-sectional geometry** parameter is set to Rectangular.

Major axis — Major axis of the elliptical cross section

1e-3 m (default) | positive scalar

Major axis of the elliptical cross section.

Dependencies

Enabled when the **Cross-sectional geometry** parameter is set to Elliptical.

Minor axis — Minor axis of the elliptical cross section

5e-4 m (default) | positive scalar

Minor axis of the elliptical cross section.

Dependencies

Enabled when the **Cross-sectional geometry** parameter is set to Elliptical.

Side length — Length of the triangle side for the equilateral triangular cross section

1e-3 m (default) | positive scalar

Length of the triangle side for the equilateral triangular cross section.

Dependencies

Enabled when the **Cross-sectional geometry** parameter is set to Equilateral triangular.

Longitudinal length — Length of the flow passage

5 m (default) | positive scalar

Length of the flow passage.

Linear pressure-flow resistance — Ratio of pressure loss to mass flow rate

1e3 s*MPa/m³ (default) | positive scalar

Ratio of pressure loss to the mass flow rate. If cross-sectional geometry is not known, the block acts as a linear hydraulic resistance.

Dependencies

Enabled when the **Cross-sectional geometry** parameter is set to Custom.

References

[1] Manning, N. D., *Hydraulic Control Systems*. p. 71. Hoboken, New Jersey: John Wiley & Sons, 2005.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Flow Resistance (IL) | Local Restriction (IL)

Topics

“Modeling Isothermal Liquid Systems”

“Isothermal Liquid Modeling Options”

Introduced in R2020a

Lever

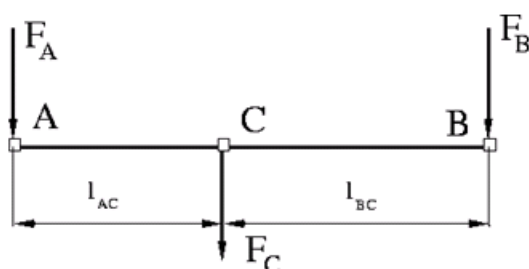
Generic mechanical lever

Library: Simscape / Foundation Library / Mechanical / Mechanisms



Description

The Lever block represents a mechanical lever in its generic form, known as a free or summing lever, shown in the following schematic.



The summing lever equations are derived with the assumption of small angle deviation from initial position:

$$v_C = K_{AC} \cdot v_A + K_{BC} \cdot v_B$$

$$F_A = K_{AC} \cdot F_C$$

$$F_B = K_{BC} \cdot F_C$$

$$K_{AC} = \frac{l_{BC}}{l_{AC} + l_{BC}}$$

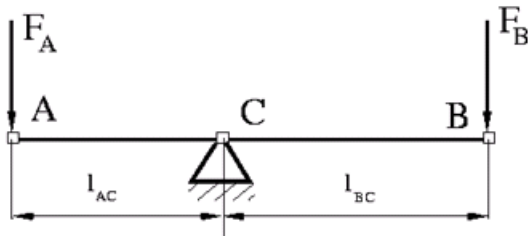
$$K_{BC} = \frac{l_{AC}}{l_{AC} + l_{BC}}$$

where

- v_A, v_B, v_C are level joint velocities.
- F_A, F_B, F_C are level joint forces.
- l_{AC}, l_{BC} are arm lengths.

The above equations are derived with the assumption that the lever sums forces and motions at node C. The assumption is arbitrary and does not impose any limitations on how the forces or motions are applied to the lever. In other words, any of the lever nodes can be “input” or “output” nodes, depending on the value of the force. Moreover, any of the block nodes can be connected to the reference point, thus converting a three-node lever into a first-class lever, with the fulcrum in the middle, or a second-class lever, with the fulcrum at the end.

The following illustration shows a schematic of a first-class lever, with the fulcrum in the middle.

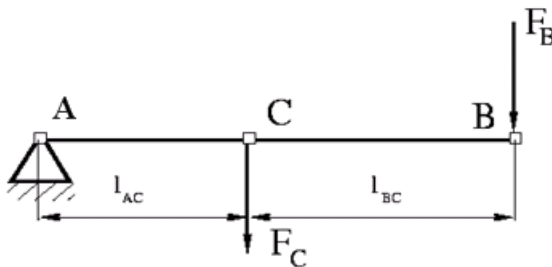


It is described with the following equations:

$$v_A = -\frac{l_{AC}}{l_{BC}} \cdot v_B$$

$$F_B = -\frac{l_{AC}}{l_{BC}} \cdot F_A$$

The next illustration shows a schematic of a second-class lever, with the fulcrum at node A.



It is described with the following equations:

$$v_C = K_{BC} \cdot v_B$$

$$F_B = K_{BC} \cdot F_C$$

The “Linkage Mechanism” example illustrates the use of the Lever block in three different modes. Linkages L_1 and L_4 simulate first-class levers with the fulcrum at the end. Linkage L_2 represents a summing lever. Linkage L_3 simulates a second-class lever with the fulcrum in the middle.

As far as the block directionality is concerned, the joints' absolute displacements are positive if they are in line with the globally assigned positive direction.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

A — Node A of the lever

mechanical translational

Mechanical translational conserving port associated with the node A of the lever.

B — Node B of the lever

mechanical translational

Mechanical translational conserving port associated with the node B of the lever.

C — Node C of the lever

mechanical translational

Mechanical translational conserving port associated with the node C of the lever.

Parameters

AC arm length — Arm length between nodes A and C

0.1 m (default)

Arm length between nodes A and C.

BC arm length — Arm length between nodes B and C

0.1 m (default)

Arm length between nodes B and C.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Gear Box | Slider-Crank | Wheel and Axle

Introduced in R2007a

Linear Hydraulic Resistance

Hydraulic pipeline with linear resistance losses



Library

Hydraulic Elements

Description

The Linear Hydraulic Resistance block represents a hydraulic resistance where pressure loss is directly proportional to flow rate. This block can be useful at preliminary stages of development, or as a powerful means to speed up the simulation, especially if the flow rate varies insignificantly with respect to the operating point.

Connections A and B are conserving hydraulic ports associated with the block inlet and outlet, respectively.

The block positive direction is from port A to port B. This means that the flow rate is positive if fluid flows from A to B, and the pressure loss is determined as $p = p_A - p_B$.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Resistance

The linear resistance coefficient. The default value is $10e9 \text{ Pa}/(\text{m}^3/\text{s})$.

Ports

The block has the following ports:

A

Hydraulic conserving port associated with the resistance inlet.

B

Hydraulic conserving port associated with the resistance outlet.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Hydraulic Resistive Tube

Introduced in R2007a

Liquid Properties Sensor (IL)

Measure density, bulk modulus, and fraction of entrained air in isothermal liquid network

Library: Simscape / Foundation Library / Isothermal Liquid / Sensors



Description

The Liquid Properties Sensor (IL) block represents an ideal sensor that lets you measure mixture density, bulk modulus, and fraction of entrained air in an isothermal liquid network. There is no mass flow through the sensor.

The block measures these properties at the node connected to port **A**. All measurements are taken with respect to absolute zero.

Ports

Output

rho — Density measurement, kg/m³

physical signal

Physical signal output port for measuring the density of isothermal liquid mixture.

Dependencies

This port is visible if you set the **Density** parameter to On.

beta — Isothermal bulk modulus measurement, Pa

physical signal

Physical signal output port for measuring the bulk modulus of isothermal liquid mixture.

Dependencies

This port is visible if you set the **Isothermal bulk modulus** parameter to On.

theta — Fraction of entrained air measurement, unitless

physical signal

Physical signal output port for measuring the fraction of entrained (undissolved) air, (*entrained air*) / (*entrained air* + *dissolved air*). Value of 1 means no air is dissolved. Value of 0 means all air is dissolved.

Dependencies

This port is visible if you set the **Fraction of air that is entrained** parameter to On.

Conserving

A — Sensor inlet

isothermal liquid

Isothermal liquid conserving port. Connect this port to the node in the isothermal liquid network where you want to measure the liquid properties. All measurements are taken with respect to absolute zero.

Parameters

Density — Control output port visibility to measure density

On (default) | Off

Setting this parameter to On exposes output port **rho**, which lets you measure fluid density.

Isothermal bulk modulus — Control output port visibility to measure bulk modulus

Off (default) | On

Setting this parameter to On exposes output port **beta**, which lets you measure bulk modulus.

Fraction of air that is entrained — Control output port visibility to measure fraction of air that is entrained

Off (default) | On

Setting this parameter to On exposes output port **theta**, which lets you measure the fraction of undissolved air.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Flow Rate Sensor (IL) | Pressure Sensor (IL)

Topics

“Modeling Isothermal Liquid Systems”

“Isothermal Liquid Modeling Options”

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2020a

Local Restriction (2P)

Fixed flow resistance



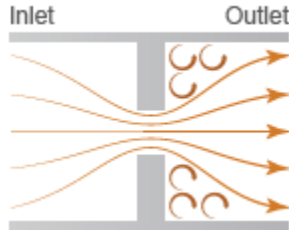
Library

Two-Phase Fluid/Elements

Description

The Local Restriction (2P) block models the pressure drop due to a fixed flow resistance such as an orifice. Ports A and B represent the restriction inlet and outlet. The restriction area, specified in the block dialog box, remains constant during simulation.

The restriction consists of a contraction followed by a sudden expansion in flow area. The contraction causes the fluid to accelerate and its pressure to drop. The expansion recovers the lost pressure though only in part, as the flow separates from the wall, losing momentum in the process.



Local Restriction Schematic

Mass Balance

The mass balance equation is

$$\dot{m}_A + \dot{m}_B = 0,$$

where:

- \dot{m}_A and \dot{m}_B are the mass flow rates into the restriction through port A and port B.

Energy Balance

The energy balance equation is

$$\phi_A + \phi_B = 0,$$

where:

- ϕ_A and ϕ_B are the energy flow rates into the restriction through port A and port B.

The local restriction is assumed to be adiabatic and the change in specific total enthalpy is therefore zero. At port A,

$$u_A + p_A \nu_A + \frac{w_A^2}{2} = u_R + p_R \nu_R + \frac{w_R^2}{2},$$

while at port B,

$$u_B + p_B \nu_B + \frac{w_B^2}{2} = u_R + p_R \nu_R + \frac{w_R^2}{2},$$

where:

- u_A , u_B , and u_R are the specific internal energies at port A, at port B, and the restriction aperture.
- p_A , p_B , and p_R are the pressures at port A, port B, and the restriction aperture.
- ν_A , ν_B , and ν_R are the specific volumes at port A, port B, and the restriction aperture.
- w_A , w_B , and w_R are the ideal flow velocities at port A, port B, and the restriction aperture.

The ideal flow velocity is computed as

$$w_A = \frac{\dot{m}_{ideal} \nu_A}{S}$$

at port A, as

$$w_B = \frac{\dot{m}_{ideal} \nu_B}{S}$$

at port B, and as

$$w_R = \frac{\dot{m}_{ideal} \nu_R}{S_R},$$

inside the restriction, where:

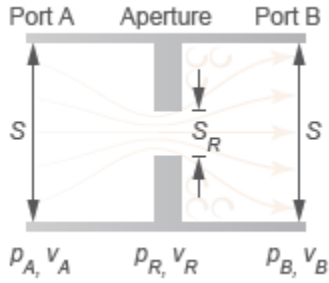
- \dot{m}_{ideal} is the ideal mass flow rate through the restriction.
- S is the flow area at port A and port B.
- S_R is the flow area of the restriction aperture.

The ideal mass flow rate through the restriction is computed as:

$$\dot{m}_{ideal} = \frac{\dot{m}_A}{C_D},$$

where:

- C_D is the flow discharge coefficient for the local restriction.



Local Restriction Variables

Momentum Balance

The change in momentum between the ports reflects in the pressure loss across the restriction. That loss depends on the mass flow rate through the restriction, though the exact dependence varies with flow regime. When the flow is turbulent:

$$\dot{m} = S_R(p_A - p_B) \sqrt{\frac{2}{|\rho_A - \rho_B| \nu_R K_T}},$$

where K_T is defined as:

$$K_T = \left(1 + \frac{S_R}{S}\right) \left(1 - \frac{\nu_{in} S_R}{\nu_{out} S}\right) - 2 \frac{S_R}{S} \left(1 - \frac{\nu_{out} S_R}{\nu_R S}\right),$$

in which the subscript *in* denotes the inlet port and the subscript *out* the outlet port. Which port serves as the inlet and which serves as the outlet depends on the pressure differential across the restriction. If pressure is greater at port **A** than at port **B**, then port **A** is the inlet; if pressure is greater at port **B**, then port **B** is the inlet.

When the flow is laminar:

$$\dot{m} = S_R(p_A - p_B) \sqrt{\frac{2}{\Delta p_{Th} \nu_R \left(1 - \frac{S_R}{S}\right)^2}},$$

where Δp_{Th} denotes the threshold pressure drop at which the flow begins to smoothly transition between laminar and turbulent:

$$\Delta p_{Th} = \left(\frac{p_A + p_B}{2}\right) (1 - B_L),$$

in which B_{Lam} is the **Laminar flow pressure ratio** block parameter. The flow is laminar if the pressure drop from port **A** to port **B** is below the threshold value; otherwise, the flow is turbulent.

The pressure at the restriction area, p_R likewise depends on the flow regime. When the flow is turbulent:

$$p_{R,L} = p_{in} - \frac{\nu_R}{2} \left(\frac{\dot{m}}{S_R}\right)^2 \left(1 + \frac{S_R}{S}\right) \left(1 - \frac{\nu_{in} S_R}{\nu_R S}\right).$$

When the flow is laminar:

$$p_{R,L} = \frac{p_A + p_B}{2}.$$

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Assumptions and Limitations

- The restriction is adiabatic. It does not exchange heat with its surroundings.

Parameters

Restriction area

Area normal to the flow path at the restriction aperture—the narrow orifice located between the ports. The default value, 0.01 m^2 , is the same as the port areas.

Cross-sectional area at ports A and B

Area normal to the flow path at the restriction ports. The ports are assumed to be identical in cross-section. The default value, 0.01 m^2 , is the same as the restriction aperture area.

Flow discharge coefficient

Ratio of the actual to the theoretical mass flow rate through the restriction. The discharge coefficient is an empirical parameter used to account for non-ideal effects such as those due to restriction geometry. The default value is 0.64 .

Laminar flow pressure ratio

Ratio of the outlet to the inlet port pressure at which the flow regime is assumed to switch from laminar to turbulent. The prevailing flow regime determines the equations used in simulation. The pressure drop across the restriction is linear with respect to the mass flow rate if the flow is laminar and quadratic (with respect to the mass flow rate) if the flow is turbulent. The default value is 0.999 .

Ports

A pair of two-phase fluid conserving ports labeled A and B represent the restriction inlet and outlet.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Variable Local Restriction (2P)

Introduced in R2015b

Local Restriction (G)

Restriction in flow area in gas network

Library: Simscape / Foundation Library / Gas / Elements



Description

The Local Restriction (G) block models the pressure drop due to a localized reduction in flow area, such as a valve or an orifice, in a gas network. Choking occurs when the restriction reaches the sonic condition.

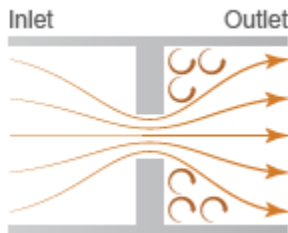
Ports **A** and **B** represent the restriction inlet and outlet. The input physical signal at port **AR** specifies the restriction area. Alternatively, you can specify a fixed restriction area as a block parameter.

The block icon changes depending on the value of the **Restriction type** parameter.

Restriction Type	Block Icon
Variable	
Fixed	

The restriction is adiabatic. It does not exchange heat with the environment.

The restriction consists of a contraction followed by a sudden expansion in flow area. The gas accelerates during the contraction, causing the pressure to drop. The gas separates from the wall during the sudden expansion, causing the pressure to recover only partially due to the loss of momentum.



Local Restriction Schematic

Caution Gas flow through this block can choke. If a Mass Flow Rate Source (G) block or a Controlled Mass Flow Rate Source (G) block connected to the Local Restriction (G) block specifies a greater mass flow rate than the possible choked mass flow rate, the simulation generates an error. For more information, see “Choked Flow”.

Mass Balance

The mass balance equation is:

$$\dot{m}_A + \dot{m}_B = 0$$

where \dot{m}_A and \dot{m}_B are the mass flow rates at ports **A** and **B**, respectively. Flow rate associated with a port is positive when it flows into the block.

Energy Balance

The energy balance equation is:

$$\Phi_A + \Phi_B = 0$$

where Φ_A and Φ_B are energy flow rates at ports **A** and **B**, respectively.

The block is assumed adiabatic. Therefore, there is no change in specific total enthalpy between port **A**, port **B**, and the restriction:

$$h_A + \frac{w_A^2}{2} = h_R + \frac{w_R^2}{2}$$

$$h_B + \frac{w_B^2}{2} = h_R + \frac{w_R^2}{2}$$

where h is the specific enthalpy at port **A**, port **B**, or restriction R, as indicated by the subscript.

The ideal flow velocities at port **A**, port **B**, and the restriction are:

$$w_A = \frac{\dot{m}_{ideal}}{\rho_A S}$$

$$w_B = \frac{\dot{m}_{ideal}}{\rho_B S}$$

$$w_R = \frac{\dot{m}_{ideal}}{\rho_R S_R}$$

where:

- S is the cross-sectional area at ports **A** and **B**.
- S_R is the cross-sectional area at the restriction.
- ρ is the density of gas volume at port **A**, port **B**, or restriction R, as indicated by the subscript.

The theoretical mass flow rate without nonideal effects is:

$$\dot{m}_{ideal} = \frac{\dot{m}_A}{C_d}$$

where C_d is the discharge coefficient.

Momentum Balance

The pressure difference between ports **A** and **B** is based on a momentum balance for flow area contraction between the inlet and the restriction, plus a momentum balance for sudden flow area expansion between the restriction and the outlet.

For flow from port **A** to port **B**:

$$\Delta p_{AB} = \rho_R \cdot w_R \cdot |w_R| \cdot \left(\frac{1+r}{2} \left(1 - r \frac{\rho_R}{\rho_A} \right) - r \left(1 - r \frac{\rho_R}{\rho_B} \right) \right)$$

where r is the area ratio, $r = S_R/S$.

For flow from port **B** to port **A**:

$$\Delta p_{BA} = \rho_R \cdot w_R \cdot |w_R| \cdot \left(\frac{1+r}{2} \left(1 - r \frac{\rho_R}{\rho_B} \right) - r \left(1 - r \frac{\rho_R}{\rho_A} \right) \right)$$

The pressure differences in the two preceding equations are proportional to the square of the flow rate. This is the typical behavior for turbulent flow. However, for laminar flow, the pressure difference becomes linear with respect to flow rate. The laminar approximation for the pressure difference is:

$$\Delta p_{lam} = \sqrt{\frac{\rho_R \Delta p_{transition}}{2}} (1 - r)$$

The threshold for transition from turbulent flow to laminar flow is defined as $\Delta p_{transition} = p_{avg}(1 - B_{lam})$, where B_{lam} is the pressure ratio at the transition between laminar and turbulent regimes (**Laminar flow pressure ratio** parameter value) and $p_{avg} = (p_A + p_B)/2$.

The pressure at the restriction is based on a momentum balance for flow area contraction between the inlet and the restriction.

For flow from port **A** to port **B**:

$$p_{R_{AB}} = p_A - \rho_R \cdot w_R \cdot |w_R| \cdot \frac{1+r}{2} \left(1 - r \frac{\rho_R}{\rho_A} \right)$$

For flow from port **B** to port **A**:

$$p_{R_{BA}} = p_B + \rho_R \cdot w_R \cdot |w_R| \cdot \frac{1+r}{2} \left(1 - r \frac{\rho_R}{\rho_B} \right)$$

For laminar flow, the pressure at the restriction is approximately

$$p_{R_{lam}} = p_{avg} - \rho_R \cdot w_R^2 \frac{1-r^2}{2}$$

The block uses a cubic polynomial in terms of $(p_A - p_B)$ to smoothly blend the pressure difference and the restriction pressure between the turbulent regime and the laminar regime:

- When $\Delta p_{transition} \leq p_A - p_B$,

then $p_A - p_B = \Delta p_{AB}$

and $p_R = p_{R_{AB}}$.

- When $0 \leq p_A - p_B < \Delta p_{\text{transition}}$,
then $p_A - p_B$ is smoothly blended between Δp_{AB} and Δp_{lam}
and p_R is smoothly blended between $p_{R_{AB}}$ and $p_{R_{\text{lam}}}$.
- When $-\Delta p_{\text{transition}} < p_A - p_B \leq 0$,
then $p_A - p_B$ is smoothly blended between Δp_{BA} and Δp_{lam}
and p_R is smoothly blended between $p_{R_{BA}}$ and $p_{R_{\text{lam}}}$.
- When $p_A - p_B \leq -\Delta p_{\text{transition}}$,
then $p_A - p_B = \Delta p_{BA}$
and $p_R = p_{R_{BA}}$.

Choked Flow

When the flow through the restriction becomes choked, further changes to the flow are dependent on the upstream conditions and are independent of the downstream conditions.

If $A.p$ is the Across variable at port **A** and $p_{B_{\text{choked}}}$ is the hypothetical pressure at port **B**, assuming choked flow from port **A** to port **B**, then

$$A.p - p_{B_{\text{choked}}} = \rho_R \cdot a_R^2 \left(\frac{1+r}{2} \left(1 - r \frac{\rho_R}{\rho_A} \right) - r \left(1 - r \frac{\rho_R}{\rho_B} \right) \right)$$

where a is speed of sound.

If $B.p$ is the Across variable at port **B** and $p_{A_{\text{choked}}}$ is the hypothetical pressure at port **A**, assuming choked flow from port **B** to port **A**, then

$$B.p - p_{A_{\text{choked}}} = \rho_R \cdot a_R^2 \left(\frac{1+r}{2} \left(1 - r \frac{\rho_R}{\rho_B} \right) - r \left(1 - r \frac{\rho_R}{\rho_A} \right) \right)$$

The actual pressures at ports **A** and **B**, p_A and p_B , respectively, depend on whether choking has occurred.

For flow from port **A** to port **B**, $p_A = A.p$ and

$$p_B = \begin{cases} B.p, & \text{if } B.p \geq p_{B_{\text{choked}}} \\ p_{B_{\text{choked}}}, & \text{if } B.p < p_{B_{\text{choked}}} \end{cases}$$

For flow from port **B** to port **A**, $p_B = B.p$ and

$$p_A = \begin{cases} A.p, & \text{if } A.p \geq p_{A_{\text{choked}}} \\ p_{A_{\text{choked}}}, & \text{if } A.p < p_{A_{\text{choked}}} \end{cases}$$

Assumptions and Limitations

- The restriction is adiabatic. It does not exchange heat with the environment.
- This block does not model supersonic flow.

Ports

Input

AR — Restriction area control signal, m²

physical signal

Input physical signal that controls the gas flow restriction area. The signal saturates when its value is outside the minimum and maximum restriction area limits, specified by the block parameters.

Dependencies

This port is visible only if you set the **Restriction type** parameter to *Variable*.

Conserving

A — Inlet or outlet

gas

Gas conserving port associated with the inlet or outlet of the local restriction. This block has no intrinsic directionality.

B — Inlet or outlet

gas

Gas conserving port associated with the inlet or outlet of the local restriction. This block has no intrinsic directionality.

Parameters

Restriction type — Specify whether restriction area can change during simulation

Variable (default) | *Fixed*

Select whether the restriction area can change during simulation:

- *Variable* — The input physical signal at port **AR** specifies the restriction area, which can vary during simulation. The **Minimum restriction area** and **Maximum restriction area** parameters specify the lower and upper bounds for the restriction area.
- *Fixed* — The restriction area, specified by the **Restriction area** block parameter value, remains constant during simulation. Port **AR** is hidden.

Minimum restriction area — Lower bound for the restriction cross-sectional area

$1e-10 \text{ m}^2$ (default)

The lower bound for the restriction cross-sectional area. You can use this parameter to represent the leakage area. The input signal AR saturates at this value to prevent the restriction area from decreasing any further.

Dependencies

Enabled when the **Restriction type** parameter is set to *Variable*.

Maximum restriction area — Upper bound for the restriction cross-sectional area

0.005 m^2 (default)

The upper bound for the restriction cross-sectional area. The input signal AR saturates at this value to prevent the restriction area from increasing any further.

Dependencies

Enabled when the **Restriction type** parameter is set to Variable.

Restriction area — Area normal to flow path at the restriction

1e-3 m² (default)

Area normal to flow path at the restriction.

Dependencies

Enabled when the **Restriction type** parameter is set to Fixed.

Cross-sectional area at ports A and B — Area normal to flow path at the ports

0.01 m² (default)

Area normal to flow path at ports **A** and **B**. This area is assumed the same for the two ports.

Discharge coefficient — Ratio of actual mass flow rate to theoretical mass flow rate through restriction

0.64 (default)

Ratio of actual mass flow rate to the theoretical mass flow rate through the restriction. The discharge coefficient is an empirical parameter that accounts for nonideal effects.

Laminar flow pressure ratio — Pressure ratio at which gas flow transitions between laminar and turbulent regimes

0.999 (default)

Pressure ratio at which the gas flow transitions between laminar and turbulent regimes. The pressure loss is linear with respect to mass flow rate in the laminar regime and quadratic with respect to mass flow rate in the turbulent regime.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Pipe (G)

Topics

“Modeling Gas Systems”

Introduced in R2016b

Local Restriction (IL)

Restriction in flow area in isothermal liquid network

Library: Simscape / Foundation Library / Isothermal Liquid / Elements

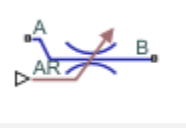



Description

The Local Restriction (IL) block models the pressure drop due to a localized reduction in flow area, such as a valve or an orifice, in an isothermal liquid network.

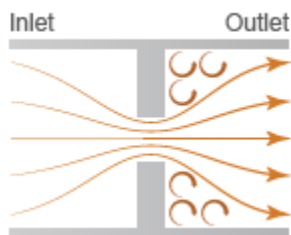
Ports **A** and **B** represent the restriction inlet and outlet. The input physical signal at port **AR** specifies the restriction area. Alternatively, you can specify a fixed restriction area as a block parameter.

The block icon changes depending on the value of the **Restriction type** parameter.

Restriction Type	Block Icon
Variable	
Fixed	

The restriction is adiabatic. It does not exchange heat with the environment.

The restriction consists of a contraction followed by a sudden expansion in flow area. The fluid accelerates during the contraction, causing the pressure to drop. In the expansion zone, if the **Pressure recovery** parameter is set to `off`, the momentum of the accelerated fluid is lost. If the **Pressure recovery** parameter is set to `on`, the sudden expansion recovers some of the momentum and allows the pressure to rise slightly after the restriction.



Local Restriction Schematic

The block equations express the mass flow rate in terms of the pressure difference between ports **A** and **B**:

$$\dot{m}_A + \dot{m}_B = 0$$

$$\dot{m}_A = C_d S_R \frac{\Delta p}{(\Delta p^2 + \Delta p_{cr}^2)^{1/4}} \sqrt{\frac{2\bar{\rho}}{PR_{loss} \left(1 - \left(\frac{S_R}{S}\right)^2\right)}}$$

$$\Delta p = p_A - p_B$$

$$\Delta p_{cr} = \frac{\pi}{4} \cdot \frac{\bar{\rho}}{2S_R} \left(\frac{Re_{cr} \nu_{atm}}{C_d}\right)^2$$

$$\bar{\rho} = \frac{\rho_A + \rho_B}{2}$$

where:

- Δp is the pressure differential.
- p_A and p_B are pressures at ports **A** and **B**, respectively.
- S_R is the cross-sectional area at the restriction.
- S is the cross-sectional area at ports **A** and **B**.
- Δp_{cr} is the critical pressure differential for the transition between the laminar and turbulent flow regimes.
- Re_{cr} is the critical Reynolds number.
- ν_{atm} is the liquid kinematic viscosity at atmospheric pressure, which is a global parameter defined by the Isothermal Liquid Properties (IL) block connected to the circuit.
- C_d is the discharge coefficient.
- $\bar{\rho}$ is the average fluid mixture density.
- ρ_A and ρ_B are fluid mixture density values at ports **A** and **B**, respectively. Equations used to compute the fluid mixture density depend on the selected isothermal liquid model. For detailed information, see "Isothermal Liquid Modeling Options".
- PR_{loss} is the pressure loss ratio.

The pressure loss ratio, PR_{loss} , depends on the **Pressure recovery** parameter value:

- If pressure recovery is off, then

$$PR_{loss} = 1.$$

- If pressure recovery is on, then

$$PR_{loss} = \frac{\sqrt{1 - \left(\frac{S_R}{S}\right)^2 (1 - C_d^2)} - C_d \frac{S_R}{S}}{\sqrt{1 - \left(\frac{S_R}{S}\right)^2 (1 - C_d^2)} + C_d \frac{S_R}{S}}.$$

The cross-sectional area at the restriction, S_R , depends on the **Restriction type** parameter value:

- For variable restrictions,

$$S_R = \begin{cases} S_{\min}, & AR \leq S_{\min} \\ S_{\max}, & AR \geq S_{\max} \\ AR, & S_{\min} < AR < S_{\max} \end{cases},$$

where AR is the value of the input physical signal, and S_{\min} and S_{\max} are the values of the **Minimum restriction area** and **Maximum restriction area** block parameters, respectively.

- For fixed restrictions, S_R is the value of the **Restriction area** parameter.

By default, the block assumes that the cross-sectional area at the restriction ports is much greater than the restriction area, setting the **Cross-sectional area at ports A and B** parameter value to ∞ and all the S_R/S terms in the equations to 0, to improve computation efficiency. Specify an actual value for the **Cross-sectional area at ports A and B** parameter if the two cross-sectional areas are comparable in size and their ratio has an impact on flow computations.

Ports

Input

AR — Restriction area control signal, m²

physical signal

Input physical signal that controls the flow restriction area. The signal saturates when its value is outside the minimum and maximum restriction area limits, specified by the block parameters.

Dependencies

This port is visible only if you set the **Restriction type** parameter to **Variable**.

Conserving

A — Inlet or outlet

isothermal liquid

Isothermal liquid conserving port associated with the inlet or outlet of the local restriction. This block has no intrinsic directionality.

B — Inlet or outlet

isothermal liquid

Isothermal liquid conserving port associated with the inlet or outlet of the local restriction. This block has no intrinsic directionality.

Parameters

Restriction type — Specify whether restriction area can change during simulation

Variable (default) | Fixed

Select whether the restriction area can change during simulation:

- **Variable** — The input physical signal at port **AR** specifies the restriction area, which can vary during simulation. The **Minimum restriction area** and **Maximum restriction area** parameters specify the lower and upper bounds for the restriction area.

- **Fixed** — The restriction area, specified by the **Restriction area** block parameter value, remains constant during simulation. Port **AR** is hidden.

Minimum restriction area — Lower bound for the restriction cross-sectional area

1e-10 m² (default) | positive scalar

The lower bound for the restriction cross-sectional area. You can use this parameter to represent the leakage area. The input signal AR saturates at this value to prevent the restriction area from decreasing any further.

Dependencies

Enabled when the **Restriction type** parameter is set to Variable.

Maximum restriction area — Upper bound for the restriction cross-sectional area

0.005 m² (default) | positive scalar

The upper bound for the restriction cross-sectional area. The input signal AR saturates at this value to prevent the restriction area from increasing any further.

Dependencies

Enabled when the **Restriction type** parameter is set to Variable.

Restriction area — Area normal to flow path at the restriction

1e-3 m² (default) | positive scalar

Area normal to flow path at the restriction.

Dependencies

Enabled when the **Restriction type** parameter is set to Fixed.

Cross-sectional area at ports A and B — Area normal to flow path at the ports

inf m² (default) | positive scalar

Area normal to flow path at ports **A** and **B**. This area is assumed to be the same for the two ports. The inf value assumes that the cross-sectional area at the ports is much greater than the restriction area, and therefore all the S_R/S terms in the block equations are negligibly small.

Discharge coefficient — Ratio of actual mass flow rate to theoretical mass flow rate through restriction

0.64 (default) | positive scalar

The discharge coefficient is a semi-empirical parameter commonly used to characterize the flow capacity of an orifice. This parameter is defined as the ratio of the actual mass flow rate through the orifice to the ideal mass flow rate.

Critical Reynolds number — Reynolds number for transition between laminar and turbulent regimes

150 (default) | positive scalar

The Reynolds number for the transition between laminar and turbulent regimes.

Pressure recovery — Specify whether to account for pressure recovery

On (default) | Off

Specify whether to account for pressure recovery at the local restriction outlet.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Flow Resistance (IL) | Laminar Leakage (IL)

Topics

“Modeling Isothermal Liquid Systems”

“Isothermal Liquid Modeling Options”

Introduced in R2020a

Local Restriction (MA)

Restriction in flow area in moist air network

Library: Simscape / Foundation Library / Moist Air / Elements

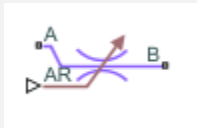



Description

The Local Restriction (MA) block models the pressure drop due to a localized reduction in flow area, such as a valve or an orifice, in a moist air network. Choking occurs when the restriction reaches the sonic condition.

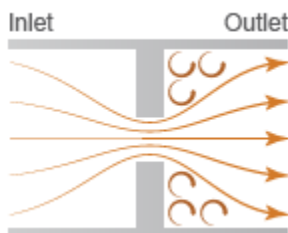
Ports **A** and **B** represent the restriction inlet and outlet. The input physical signal at port **AR** specifies the restriction area. Alternatively, you can specify a fixed restriction area as a block parameter.

The block icon changes depending on the value of the **Restriction type** parameter.

Restriction Type	Block Icon
Variable	
Fixed	

The restriction is adiabatic. It does not exchange heat with the environment.

The restriction consists of a contraction followed by a sudden expansion in flow area. The moist air accelerates during the contraction, causing the pressure to drop. The moist air separates from the wall during the sudden expansion, causing the pressure to recover only partially due to the loss of momentum.



Local Restriction Schematic

Caution Moist air flow through this block can choke. If a Mass Flow Rate Source (MA) block or a Controlled Mass Flow Rate Source (MA) block connected to the Local Restriction (MA) block specifies

a greater mass flow rate than the possible choked mass flow rate, the simulation generates an error. For more information, see “Choked Flow”.

The block equations use these symbols.

\dot{m}	Mass flow rate
Φ	Energy flow rate
p	Pressure
ρ	Density
R	Specific gas constant
S	Cross-sectional area
C_d	Discharge coefficient
h	Specific enthalpy
c_p	Specific heat at constant pressure
T	Temperature

Subscripts **a**, **w**, and **g** indicate the properties of dry air, water vapor, and trace gas, respectively. Subscripts **lam** and **tur** indicate the laminar and turbulent regime, respectively. Subscripts **A** and **B** indicate the appropriate port. Subscript **R** indicates the restriction.

Mass balance:

$$\dot{m}_A + \dot{m}_B = 0$$

$$\dot{m}_{wA} + \dot{m}_{wB} = 0$$

$$\dot{m}_{gA} + \dot{m}_{gB} = 0$$

Energy balance:

$$\Phi_A + \Phi_B = 0$$

When the flow is not choked, the mixture mass flow rate (positive from port **A** to port **B**) in the turbulent regime is

$$\dot{m}_{tur} = C_d S_R (p_A - p_B) \sqrt{\frac{2\rho_R}{|p_A - p_B|} K_{tur}}$$

$$K_{tur} = \left(1 + \frac{S_R}{S}\right) \left(1 - \frac{\rho_R S_R}{\rho_{in} S}\right) - 2 \frac{S_R}{S} \left(1 - \frac{\rho_R S_R}{\rho_{out} S}\right)$$

Subscripts **in** and **out** indicate the inlet and outlet, respectively. If $p_A \geq p_B$, the inlet is port **A** and the outlet is port **B**; otherwise, they are reversed. The cross-sectional area S is assumed to be equal at ports **A** and **B**. S_R is the area at the restriction.

The mixture mass flow rate equation is derived by combining the equations from two control volume analyses:

- Momentum balance for flow area contraction from the inlet to the restriction
- Momentum balance for sudden flow area expansion from the restriction to the outlet

In the analysis for the flow area contraction, pressure p_{in} acts on the area at the inlet, S , and pressure p_R acts on the area at the restriction, S_R . The pressure acting on the area outside the restriction, $S - S_R$, is assumed to be $(p_{in}S + p_R S_R)/(S + S_R)$.

In the analysis for the flow area expansion, the pressure acting on both the area at the restriction, S_R , and the area outside the restriction, $S - S_R$, is assumed to be p_R , because of flow separation from the restriction. The pressure acting on the area at the outlet, S , is equal to p_{out} .

The mixture mass flow rate (positive from port **A** to port **B**) in the laminar regime is linearized with respect to the pressure difference:

$$\dot{m}_{lam} = C_d S_R (p_A - p_B) \sqrt{\frac{2\rho_R}{\Delta p_{threshold} \left(1 - \frac{S_R}{S}\right)^2}}$$

where the threshold for transition between the laminar and turbulent regime is defined based on the laminar flow pressure ratio, B_{lam} , as

$$\Delta p_{threshold} = \left(\frac{p_A + p_B}{2}\right)(1 - B_{lam})$$

When $|p_A - p_B| \geq \Delta p_{threshold}$, the flow is assumed to be turbulent and therefore $\dot{m}_{unchoked} = \dot{m}_{tur}$.

When $|p_A - p_B| < \Delta p_{threshold}$, $\dot{m}_{unchoked}$ smoothly transitions to \dot{m}_{lam} .

When the flow is choked, the velocity at the restriction is equal to the speed of sound and cannot increase any further. Assuming the flow is choked, the mixture mass flow rate is

$$\dot{m}_{choked} = C_d S_R \rho_R \sqrt{\frac{\gamma_R}{RT_R}}$$

where $\gamma_R = c_{pR}/(c_{pR} - R)$. Therefore, the actual mixture mass flow rate is equal to $\dot{m}_{unchoked}$, but is limited in magnitude by \dot{m}_{choked} :

$$\dot{m}_A = \begin{cases} -\dot{m}_{choked}, & \text{if } \dot{m}_{unchoked} \leq -\dot{m}_{choked} \\ \dot{m}_{unchoked}, & \text{if } -\dot{m}_{choked} < \dot{m}_{unchoked} < \dot{m}_{choked} \\ \dot{m}_{choked}, & \text{if } \dot{m}_{unchoked} \geq \dot{m}_{choked} \end{cases}$$

The expression for the pressure at the restriction is obtained by considering the momentum balance for flow area contraction from the inlet to the restriction only. In the turbulent regime, it is

$$p_{Rtur} = p_{in} - \frac{1}{2\rho_R} \left(\frac{\dot{m}_A}{C_d S_R}\right)^2 \left(1 + \frac{S_R}{S}\right) \left(1 - \frac{\rho_R S_R}{\rho_{in} S}\right)$$

In the laminar regime, the pressure at the restriction is assumed to be

$$p_{Rlam} = \frac{p_A + p_B}{2}$$

Similar to the mixture mass flow rate, when $|p_A - p_B| \geq \Delta p_{threshold}$, the flow is assumed to be turbulent and therefore $p_R = p_{Rtur}$. When $|p_A - p_B| < \Delta p_{threshold}$, p_R smoothly transitions to p_{Rlam} .

The local restriction is assumed adiabatic, so the mixture specific total enthalpies are equal. Therefore, the changes in mixture specific enthalpies are:

$$h_A - h_R = \left(\frac{1}{\rho_R^2 S_R^2} - \frac{1}{\rho_A^2 S^2} \right) \frac{\dot{m}_A^2}{2C_D^2}$$

$$h_B - h_R = \left(\frac{1}{\rho_R^2 S_R^2} - \frac{1}{\rho_B^2 S^2} \right) \frac{\dot{m}_B^2}{2C_D^2}$$

Assumptions and Limitations

- The restriction is adiabatic. It does not exchange heat with the environment.
- This block does not model supersonic flow.

Ports

Input

AR — Restriction area control signal, m²

physical signal

Input physical signal that controls the air flow restriction area. The signal saturates when its value is outside the minimum and maximum restriction area limits, specified by the block parameters.

Dependencies

This port is visible only if you set the **Restriction type** parameter to **Variable**.

Conserving

A — Inlet or outlet

moist air

Moist air conserving port associated with the inlet or outlet of the local restriction. This block has no intrinsic directionality.

B — Inlet or outlet

moist air

Moist air conserving port associated with the inlet or outlet of the local restriction. This block has no intrinsic directionality.

Parameters

Restriction type — Specify whether restriction area can change during simulation

Variable (default) | Fixed

Select whether the restriction area can change during simulation:

- **Variable** — The input physical signal at port **AR** specifies the restriction area, which can vary during simulation. The **Minimum restriction area** and **Maximum restriction area** parameters specify the lower and upper bounds for the restriction area.

- **Fixed** — The restriction area, specified by the **Restriction area** block parameter value, remains constant during simulation. Port **AR** is hidden.

Minimum restriction area — Lower bound for the restriction cross-sectional area

1e-10 m² (default)

Lower bound for the restriction cross-sectional area. You can use this parameter to represent the leakage area. The input signal **AR** saturates at this value to prevent the restriction area from decreasing any further.

Dependencies

Enabled when the **Restriction type** parameter is set to Variable.

Maximum restriction area — Upper bound for the restriction cross-sectional area

0.005 m² (default)

Upper bound for the restriction cross-sectional area. The input signal **AR** saturates at this value to prevent the restriction area from increasing any further.

Dependencies

Enabled when the **Restriction type** parameter is set to Variable.

Restriction area — Area normal to flow path at the restriction

1e-3 m² (default)

Area normal to flow path at the restriction.

Dependencies

Enabled when the **Restriction type** parameter is set to Fixed.

Cross-sectional area at ports A and B — Area normal to flow path at the ports

0.01 m² (default)

Area normal to flow path at ports **A** and **B**. This area is assumed to be the same for the two ports.

Discharge coefficient — Ratio of actual mass flow rate to theoretical mass flow rate through the restriction

0.64 (default)

Ratio of actual mass flow rate to the theoretical mass flow rate through the restriction. The discharge coefficient is an empirical parameter that accounts for nonideal effects.

Laminar flow pressure ratio — Pressure ratio at which air flow transitions between laminar and turbulent regimes

0.999 (default)

Pressure ratio at which the moist air flow transitions between laminar and turbulent regimes. The pressure loss is linear with respect to mass flow rate in the laminar regime and quadratic with respect to mass flow rate in the turbulent regime.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Pipe (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Local Restriction (TL)

Restriction in flow area in thermal liquid network

Library: Simscape / Foundation Library / Thermal Liquid / Elements





Description

The Local Restriction (TL) block models the pressure drop due to a localized reduction in flow area, such as a valve or an orifice, in a thermal liquid network.

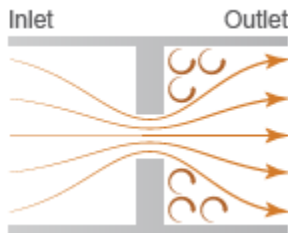
Ports **A** and **B** represent the restriction inlet and outlet. The input physical signal at port **AR** specifies the restriction area. Alternatively, you can specify a fixed restriction area as a block parameter.

The block icon changes depending on the value of the **Restriction type** parameter.

Restriction Type	Block Icon
Variable	
Fixed	

The restriction is adiabatic. It does not exchange heat with the environment.

The restriction consists of a contraction followed by a sudden expansion in flow area. The fluid accelerates during the contraction, causing the pressure to drop. In the expansion zone, if the **Pressure recovery** parameter is set to **off**, the momentum of the accelerated fluid is lost. If the **Pressure recovery** parameter is set to **on**, the sudden expansion recovers some of the momentum and allows the pressure to rise slightly after the restriction.



Local Restriction Schematic

Mass Balance

The mass balance in the restriction is

$$0 = \dot{m}_A + \dot{m}_B,$$

where:

- \dot{m}_A is the mass flow rate into the restriction through port **A**.
- \dot{m}_B is the mass flow rate into the restriction through port **B**.

Momentum Balance

The pressure difference between ports **A** and **B** follows from the momentum balance in the restriction:

$$\Delta p = \frac{1}{2\rho} \left(1 - \frac{S_R^2}{S^2} \right) v_R \sqrt{v_R^2 + v_{Rc}^2}$$

$$v_R = \frac{\dot{m}_A}{C_d \rho S_R}$$

$$v_{Rc} = \frac{Re_c \mu}{C_d \rho} \sqrt{\frac{\pi}{4S_R}}$$

where:

- Δp is the pressure differential.
- ρ is the liquid density.
- μ is the liquid dynamic viscosity.
- S is the cross-sectional area at ports **A** and **B**.
- S_R is the cross-sectional area at the restriction.
- v_R is fluid velocity at the restriction.
- v_{Rc} is the critical fluid velocity.
- Re_c is the critical Reynolds number.
- C_d is the discharge coefficient.

If pressure recovery is off, then

$$p_A - p_B = \Delta p,$$

where:

- p_A is the pressure at port **A**.
- p_B is the pressure at port **B**.

If pressure recovery is on, then

$$p_A - p_B = \Delta p \frac{\sqrt{1 - \left(\frac{S_R}{S}\right)^2 (1 - C_d^2)} - C_d \frac{S_R}{S}}{\sqrt{1 - \left(\frac{S_R}{S}\right)^2 (1 - C_d^2)} + C_d \frac{S_R}{S}}.$$

Energy Balance

The energy balance in the restriction is

$$\phi_A + \phi_B = 0,$$

where:

- ϕ_A is the energy flow rate into the restriction through port **A**.
- ϕ_B is the energy flow rate into the restriction through port **B**.

Assumptions and Limitations

- The restriction is adiabatic. It does not exchange heat with its surroundings.
- The dynamic compressibility and thermal capacity of the liquid in the restriction are negligible.

Ports

Input

AR — Restriction area control signal, m²

physical signal

Input physical signal that controls the flow restriction area. The signal saturates when its value is outside the minimum and maximum restriction area limits, specified by the block parameters.

Dependencies

This port is visible only if you set the **Restriction type** parameter to *Variable*.

Conserving

A — Inlet or outlet

thermal liquid

Thermal liquid conserving port associated with the inlet or outlet of the local restriction. This block has no intrinsic directionality.

B — Inlet or outlet

thermal liquid

Thermal liquid conserving port associated with the inlet or outlet of the local restriction. This block has no intrinsic directionality.

Parameters

Restriction type — Specify whether restriction area can change during simulation

Variable (default) | *Fixed*

Select whether the restriction area can change during simulation:

- *Variable* — The input physical signal at port **AR** specifies the restriction area, which can vary during simulation. The **Minimum restriction area** and **Maximum restriction area** parameters specify the lower and upper bounds for the restriction area.
- *Fixed* — The restriction area, specified by the **Restriction area** block parameter value, remains constant during simulation. Port **AR** is hidden.

Minimum restriction area — Lower bound for the restriction cross-sectional area

1e-10 m² (default)

The lower bound for the restriction cross-sectional area. You can use this parameter to represent the leakage area. The input signal AR saturates at this value to prevent the restriction area from decreasing any further.

Dependencies

Enabled when the **Restriction type** parameter is set to Variable.

Maximum restriction area — Upper bound for the restriction cross-sectional area

0.005 m² (default)

The upper bound for the restriction cross-sectional area. The input signal AR saturates at this value to prevent the restriction area from increasing any further.

Dependencies

Enabled when the **Restriction type** parameter is set to Variable.

Restriction area — Area normal to flow path at the restriction

1e-3 m² (default)

Area normal to flow path at the restriction.

Dependencies

Enabled when the **Restriction type** parameter is set to Fixed.

Cross-sectional area at ports A and B — Area normal to flow path at the ports

0.01 m² (default)

Area normal to flow path at ports **A** and **B**. This area is assumed to be the same for the two ports.

Discharge coefficient — Ratio of actual mass flow rate to theoretical mass flow rate through restriction

0.64 (default)

The discharge coefficient is a semi-empirical parameter commonly used to characterize the flow capacity of an orifice. This parameter is defined as the ratio of the actual mass flow rate through the orifice to the ideal mass flow rate.

Pressure recovery — Specify whether to account for pressure recovery

On (default) | Off

Specify whether to account for pressure recovery at the local restriction outlet.

Critical Reynolds number — Reynolds number for transition between laminar and turbulent regimes

12 (default)

The Reynolds number for the transition between laminar and turbulent regimes. The default value corresponds to a sharp-edged orifice.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Pipe (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2013b

Magnetic Reference

Reference connection for magnetic ports



Library

Magnetic Elements

Description

The Magnetic Reference block represents a reference point for all magnetic conserving ports. A model with magnetic elements must contain at least one Magnetic Reference block.

Ports

The block has one magnetic conserving port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Topics

“Grounding Rules”

Introduced in R2010a

Mass

Ideal mechanical translational mass

Library: Simscape / Foundation Library / Mechanical / Translational Elements



Description

The Mass block represents an ideal mechanical translational mass, described with the following equation:

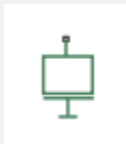
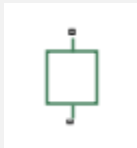
$$F = m \frac{dv}{dt}$$

where:

- F is inertia force.
- m is mass.
- v is velocity.
- t is time.

By default, the block has one mechanical translational conserving port. The block positive direction is from its port to the reference point. This means that the inertia force is positive if mass is accelerated in positive direction.

In some applications, it is customary to display mass in series with other elements in the block diagram layout. To support this use case, the **Number of graphical ports** parameter lets you display a second port on the opposite side of the block icon. The two-port variant is purely graphical: the two ports have the same velocity, so the block functions exactly the same whether it has one or two ports. The block icon changes depending on the value of the **Number of graphical ports** parameter.

Number of graphical ports	Block Icon
1	
2	

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

M – Port that connects the mass to the circuit

mechanical translational

Mechanical translational conserving port connecting the mass to the rest of the physical network.

N – Second graphical port

mechanical translational

Second mechanical translational conserving port that lets you connect the mass in series with other elements in the block diagram. This port is rigidly connected to port **M**, therefore the difference between the one-port and two-port block representations is purely graphical.

Dependencies

This port is visible if you set the **Number of graphical ports** parameter to 2.

Parameters

Mass – Constant mass

1 kg (default) | positive scalar

Mass value. The mass is constant during simulation.

Number of graphical ports – Select whether the block is graphically connected to the side of a circuit or in series with other elements

1 (default) | 2

Select how to connect the block to the rest of the circuit:

- 1 – The block has one conserving port, **M**, that connects it to the mechanical translational circuit. When the block has one port, attach it to a connection line between two other blocks.
- 2 – Selecting this option exposes the second graphical port **N**, which lets you connect the block in series with other blocks in the circuit. The two ports are rigidly connected and have the same velocity, therefore the block functions exactly the same as if it had one port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Inertia

Topics

“Essential Steps for Constructing a Physical Model”

Introduced in R2007a

Mass & Energy Flow Rate Sensor (2P)

Measure mass and energy flow rates



Library

Two-Phase Fluid/Sensors

Description

The Mass & Energy Flow Rate Sensor (2P) block measures mass and energy flow rates through the two-phase fluid branch defined by ports A and B. The energy flow rate includes contributions from internal energy advection, thermal conduction, pressure work, and kinetic energy. The flow rates are positive if mass and energy flow from port A to port B.

Ports

The block has two two-phase fluid conserving ports, A and B. Physical signal port M outputs the mass flow rate value. Physical signal port Phi outputs the energy flow rate value.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Mass Flow Rate Source (2P) | Mass Flow Rate Source (2P) | Volumetric Flow Rate Sensor (2P)

Introduced in R2015b

Mass & Energy Flow Rate Sensor (G)

Measure mass and energy flow rates

Library: Simscape / Foundation Library / Gas / Sensors



Description

The Mass & Energy Flow Rate Sensor (G) block represents an ideal sensor that measures mass flow rate and energy flow rate in a gas network. The energy flow rate is the advection of total enthalpy. There is no change in pressure or temperature across the sensor.

Because the flow rates are Through variables, the block must connect in series with the component being measured.

The relative orientation of ports **A** and **B** establishes the measurement sign. The sign is positive if flow occurs from port **A** to port **B**. Switching port connections reverses the measurement sign.

Two physical signal ports output the measurement data. Port **M** outputs the mass flow rate. Port **Phi** outputs the energy flow rate. Connect the ports to PS-Simulink Converter blocks to transform the output physical signals into Simulink signals, for example, for plotting or additional data processing.

Ports

Output

M — Mass flow rate measurement, kg/s

physical signal

Physical signal output port for mass flow rate measurement.

Phi — Energy flow rate measurement, W

physical signal

Physical signal output port for energy flow rate measurement.

Conserving

A — Sensor inlet

gas

Gas conserving port. Mass and energy flow rates are positive if gas flows from port **A** to port **B**.

B — Sensor outlet

gas

Gas conserving port. Mass and energy flow rates are positive if gas flows from port **A** to port **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Pressure & Temperature Sensor (G) | Thermodynamic Properties Sensor (G) | Volumetric Flow Rate Sensor (G)

Topics

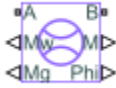
“Modeling Gas Systems”

Introduced in R2016b

Mass & Energy Flow Rate Sensor (MA)

Measure mass and energy flow rates in a moist air network

Library: Simscape / Foundation Library / Moist Air / Sensors



Description

The Mass & Energy Flow Rate Sensor (MA) block represents an ideal sensor that measures mass flow rate and energy flow rates in a moist air network. There is no change in pressure, temperature, specific humidity, or trace gas mass fraction across the sensor.

Because the flow rates are Through variables, the block must connect in series with the component being measured.

The relative orientation of ports **A** and **B** establishes the measurement sign. The sign is positive if flow occurs from port **A** to port **B**. Switching port connections reverses the measurement sign.

Four physical signal ports output the measurement data. Connect the ports to PS-Simulink Converter blocks to transform the output physical signals into Simulink signals, for example, for plotting or additional data processing.

Ports

Output

M — Mixture mass flow rate, kg/s

physical signal

Physical signal output port for the mixture mass flow rate measurement.

Phi — Mixture energy flow rate, W

physical signal

Physical signal output port for the mixture energy flow rate measurement.

Mw — Water vapor mass flow rate, kg/s

physical signal

Physical signal output port for the water vapor mass flow rate measurement.

Mg — Trace gas mass flow rate, kg/s

physical signal

Physical signal output port for the trace gas mass flow rate measurement.

Conserving

A – Sensor inlet

moist air

Moist air conserving port. Mass and energy flow rates are positive if air flows from port **A** to port **B**.

B – Sensor outlet

moist air

Moist air conserving port. Mass and energy flow rates are positive if air flows from port **A** to port **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Pressure & Temperature Sensor (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Mass & Energy Flow Rate Sensor (TL)

Measure mass and energy flow rates

Library: Simscape / Foundation Library / Thermal Liquid / Sensors



Description

The Mass & Energy Flow Rate Sensor (TL) block represents an ideal sensor that measures mass and energy flow rates through a thermal liquid node.

Because the flow rates are Through variables, the block must connect in series with the component being measured.

The relative orientation of ports **A** and **B** establishes the measurement sign. The sign is positive if flow occurs from port **A** to port **B**. Switching port connections reverses the measurement sign.

Two physical signal ports output the measurement data. Port **M** outputs the mass flow rate. Port **Phi** outputs the energy flow rate. Connect the ports to PS-Simulink Converter blocks to transform the output physical signals into Simulink signals, for example, for plotting or additional data processing.

Ports

Output

M — Mass flow rate measurement, kg/s

physical signal

Physical signal output port for mass flow rate measurement.

Phi — Energy flow rate measurement, W

physical signal

Physical signal output port for energy flow rate measurement.

Conserving

A — Sensor inlet

thermal liquid

Thermal liquid conserving port. Mass and energy flow rates are positive if fluid flows from port **A** to port **B**.

B — Sensor outlet

thermal liquid

Thermal liquid conserving port. Mass and energy flow rates are positive if fluid flows from port **A** to port **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Pressure & Temperature Sensor (TL) | Thermodynamic Properties Sensor (TL) | Volumetric Flow Rate Sensor (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2016a

Mass Flow Rate Source (2P)

Generate constant mass flow rate



Library

Two-Phase Fluid/Sources

Description

The Mass Flow Rate Source (2P) block generates a constant mass flow rate in a two-phase fluid network branch. The source has two inlets, labeled **A** and **B**, with independently specified cross-sectional areas. By default, the source does isentropic work on the fluid, though the block provides the option to ignore this work.

The source is ideal. In other words, it maintains the specified flow rate regardless of the pressure differential produced between its ports. In addition, because the source is isentropic, there is no viscous friction between the ports and no heat exchange with the environment. Use this block to model an idealized pump or compressor or to set a boundary condition in a model.

Mass Balance

The volume of fluid in the source is considered negligible and is ignored in a model. There is no fluid accumulation between the ports and the sum of all mass flow rates into the source must therefore equal zero:

$$\dot{m}_A + \dot{m}_B = 0,$$

where \dot{m} denotes the mass flow rate into the source through a port. The block accepts as input the mass flow rate at port **A**. The flow is directed from port **A** to port **B** when the specified value is positive.

Energy Balance

By default, the source maintains the specified flow rate by performing isentropic work on the incoming fluid, though the block provides the option to ignore this term. The rate at which the source does work, if considered in the model, must equal the sum of the energy flow rates through the ports:

$$\phi_A + \phi_B + \phi_{\text{Work}} = 0,$$

where ϕ denotes the energy flow rate into the source through a port or by means of work. The energy flow rate due to work is equal to the power generated by the source. Its value is calculated from the specific total enthalpies at the ports:

$$\phi_{\text{Work}} = \dot{m}_A(h_A - h_B).$$

The specific total enthalpy h is defined as:

$$h_* = u_* + p_* v_* + \frac{1}{2} \left(\frac{\dot{m}_* v_*}{S} \right)^2,$$

where the asterisk denotes a port (**A** or **B**) and:

- u is specific internal energy.
- p is pressure.
- S is flow area.

The specific internal energy in the equation is obtained from the tabulated data of the Two-Phase Fluid Properties (2P) block. Its value is uniquely determined from the constraint that the work done by the source is isentropic. The specific entropy, a function of specific internal energy, must then have the same value at ports **A** and **B**:

$$s_A(p_A, u_A) = s_B(p_B, u_B),$$

where s is specific entropy. If the **Power added** parameter is set to **None**, the specific total enthalpies at the ports have the same value ($h_A = h_B$) and the work done by the source reduces to zero ($\phi_{\text{Work}} = 0$).

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

A – Inlet

two-phase fluid

Opening through fluid can enter and exit the source.

B – Inlet

two-phase fluid

Opening through fluid can enter and exit the source.

Parameters

Power added – Parameterization for the calculation of power

Isentropic power (default) | None

Parameterization for the calculation of power. Work is isentropic and its calculation is based on the assumptions of zero friction losses and zero heat exchange with the environment. Change to **None** to prevent the source from impacting the temperature of the fluid—for example, when using this block as a boundary condition in a model.

Mass flow rate – Mass pumped per unit time from port A to port B

0 m³/s (default) | scalar with units of volume/time

Fluid mass pumped per unit time from port **A** to port **B**. A positive rate corresponds to a flow directed from port **A** to port **B**. The specified rate is maintained no matter the pressure differential produced between the ports.

Cross-sectional area at port A — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of volume/time

Area of the fluid opening normal to the direction of flow.

Cross-sectional area at port B — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of volume/time

Area of the fluid opening normal to the direction of flow.

Ports

The block has two two-phase fluid conserving ports, **A** and **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Mass Flow Rate Source (2P) | Controlled Volumetric Flow Rate Source (2P) | Volumetric Flow Rate Source (2P)

Introduced in R2015b

Mass Flow Rate Source (G)

Generate constant mass flow rate

Library: Simscape / Foundation Library / Gas / Sources



Description

The Mass Flow Rate Source (G) block represents an ideal mechanical energy source in a gas network. The source can maintain a constant mass flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. A positive mass flow rate causes gas to flow from port **A** to port **B**.

You can choose whether the source performs work on the gas flow:

- If the source is isentropic (**Power added** parameter is set to **Isentropic power**), then the isentropic relation depends on the gas property model.

Gas Model	Equations
Perfect gas	$\frac{(p_A)^{Z \cdot R/c_p}}{T_A} = \frac{(p_B)^{Z \cdot R/c_p}}{T_B}$
Semiperfect gas	$\int_{T_A}^{T_B} \frac{c_p(T)}{T} dT - Z \cdot R \cdot \ln(p_A) = \int_{T_B}^{T_C} \frac{c_p(T)}{T} dT - Z \cdot R \cdot \ln(p_B)$
Real gas	$s(T_A, p_A) = s(T_B, p_B)$

The power delivered to the gas flow is based on the specific total enthalpy associated with the isentropic process.

$$\Phi_{work} = -\dot{m}_A \left(h_A + \frac{w_A^2}{2} \right) - \dot{m}_B \left(h_B + \frac{w_B^2}{2} \right)$$

- If the source performs no work (**Power added** parameter is set to **None**), then the defining equation states that the specific total enthalpy is equal on both sides of the source. It is the same for all three gas property models.

$$h_A + \frac{w_A^2}{2} = h_B + \frac{w_B^2}{2}$$

The power delivered to the gas flow $\Phi_{work} = 0$.

The equations use these symbols:

c_p	Specific heat at constant pressure
-------	------------------------------------

h	Specific enthalpy
\dot{m}	Mass flow rate (flow rate associated with a port is positive when it flows into the block)
p	Pressure
R	Specific gas constant
s	Specific entropy
T	Temperature
w	Flow velocity
Z	Compressibility factor
Φ_{work}	Power delivered to the gas flow through the source

Subscripts A and B indicate the appropriate port.

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Conserving

A — Source inlet

gas

Gas conserving port. A positive mass flow rate causes gas to flow from port **A** to port **B**.

B — Source outlet

gas

Gas conserving port. A positive mass flow rate causes gas to flow from port **A** to port **B**.

Parameters

Power added — Select whether the source performs work

Isentropic power (default) | None

Select whether the source performs work on the gas flow:

- **Isentropic power** — The source performs isentropic work on the gas to maintain the specified mass flow rate, regardless of the pressure differential. Use this option to represent an idealized pump or compressor and properly account for the energy input and output, especially in closed-loop systems.
- **None** — The source performs no work on the flow, neither adding nor removing power, regardless of the mass flow rate produced by the source. Use this option to set up the desired flow condition upstream of the system, without affecting the temperature of the flow.

Mass flow rate — Constant mass flow rate through the source

0 kg/s (default)

Desired mass flow rate of the gas through the source.

Cross-sectional area at port A — Area normal to flow path at port A

0.01 m² (default)

Area normal to flow path at port A.

Cross-sectional area at port B — Area normal to flow path at port B

0.01 m² (default)

Area normal to flow path at port B.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Mass Flow Rate Source (G)

Topics

“Modeling Gas Systems”

Introduced in R2016b

Mass Flow Rate Source (MA)

Generate constant mass flow rate

Library: Simscape / Foundation Library / Moist Air / Sources



Description

The Mass Flow Rate Source (MA) block represents an ideal mechanical energy source in a moist air network. The source can maintain a constant mass flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. A positive mass flow rate causes moist air to flow from port **A** to port **B**.

The equations describing the source use these symbols.

c_p	Specific heat at constant pressure
h	Specific enthalpy
h_t	Specific total enthalpy
\dot{m}	Mass flow rate (flow rate associated with a port is positive when it flows into the block)
p	Pressure
ρ	Density
R	Specific gas constant
s	Specific entropy
T	Temperature
Φ_{work}	Power delivered to the moist air flow through the source

Subscripts A and B indicate the appropriate port.

Mass balance:

$$\dot{m}_A + \dot{m}_B = 0$$

$$\dot{m}_{wA} + \dot{m}_{wB} = 0$$

$$\dot{m}_{gA} + \dot{m}_{gB} = 0$$

Energy balance:

$$\Phi_A + \Phi_B + \Phi_{work} = 0$$

If the source performs no work (**Power added** parameter is set to None), then $\Phi_{work} = 0$.

If the source is isentropic (**Power added** parameter is set to Isentropic power), then

$$\Phi_{work} = \dot{m}_A(h_{tB} - h_{tA})$$

where

$$h_{tA} = h_A + \frac{1}{2} \left(\frac{\dot{m}_A}{\rho_A S_A} \right)^2$$

$$h_{tB} = h_B + \frac{1}{2} \left(\frac{\dot{m}_B}{\rho_B S_B} \right)^2$$

The mixture-specific enthalpies, $h_A = h(T_A)$ and $h_B = h(T_B)$, are constrained by the isentropic relation, that is, there is no change in entropy:

$$\int_A^{T_B} \frac{1}{T} dh(T) = R \ln \left(\frac{p_B}{p_A} \right)$$

The quantity specified by the **Mixture mass flow rate** parameter of the source is

$$\dot{m}_A = \dot{m}_{\text{specified}}$$

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Conserving

A — Source inlet

moist air

Moist air conserving port. A positive mass flow rate causes moist air to flow from port **A** to port **B**.

B — Source outlet

moist air

Moist air conserving port. A positive mass flow rate causes moist air to flow from port **A** to port **B**.

Parameters

Power added — Select whether the source performs work

Isentropic power (default) | None

Select whether the source performs work on the moist air flow:

- **Isentropic power** — The source performs isentropic work on the moist air to maintain the specified mass flow rate, regardless of the pressure differential. Use this option to represent an idealized pump or compressor and properly account for the energy input and output, especially in closed-loop systems.
- **None** — The source performs no work on the flow, neither adding nor removing power, regardless of the mass flow rate produced by the source. Use this option to set up the desired flow condition upstream of the system, without affecting the temperature of the flow.

Mixture mass flow rate — Constant mass flow rate through the source

0 kg/s (default)

Desired mass flow rate of the moist air mixture through the source.

Cross-sectional area at port A — Area normal to flow path at port A

0.01 m² (default)

Area normal to flow path at port A.

Cross-sectional area at port B — Area normal to flow path at port B

0.01 m² (default)

Area normal to flow path at port B.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Mass Flow Rate Source (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Mass Flow Rate Source (TL)

Generate constant mass flow rate

Library: Simscape / Foundation Library / Thermal Liquid / Sources



Description

The Mass Flow Rate Source (TL) block represents an ideal mechanical energy source in a thermal liquid network. The source can maintain a constant mass flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. A positive mass flow rate causes the fluid to flow from port **A** to port **B**.

The energy balance at the source is a function of the energy flow rates through ports **A** and **B** and the work done on the fluid:

$$\phi_A + \phi_B + \phi_{work} = 0,$$

where:

- ϕ_A is the energy flow rate into the source through port **A**.
- ϕ_B is the energy flow rate into the source through port **B**.
- ϕ_{work} is the isentropic work done on the fluid.

The isentropic work term is

$$\phi_{work} = \frac{\dot{m}(p_B - p_A)}{\rho_{avg}},$$

where:

- ϕ_{work} is the isentropic work done on the thermal liquid.
- p_A is the pressure at port **A**.
- p_B is the pressure at port **B**.
- ρ_{avg} is the average liquid density,

$$\rho_{avg} = \frac{\rho_A + \rho_B}{2}.$$

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Conserving

A — Source inlet

thermal liquid

Thermal liquid conserving port. A positive mass flow rate causes the fluid to flow from port **A** to port **B**.

B — Source outlet

thermal liquid

Thermal liquid conserving port. A positive mass flow rate causes the fluid to flow from port **A** to port **B**.

Parameters

Mass flow rate — Constant mass flow rate through the source

0 kg/s (default)

Desired mass flow rate of the fluid through the source.

Cross-sectional area at ports A and B — Area normal to flow path at inlet and outlet

0.01 m² (default)

Area normal to the direction of flow at the source inlet and outlet. The two cross-sectional areas are assumed identical.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Mass Flow Rate Source (TL) | Controlled Pressure Source (TL) | Controlled Volumetric Flow Rate Source (TL) | Pressure Source (TL) | Volumetric Flow Rate Source (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2013b

Measurement Selector (MA)

Measure pressure, temperature, humidity, and trace gas levels in moist air internal volumes

Library: Simscape / Foundation Library / Moist Air / Sensors



Description

The Measurement Selector (MA) block lets you access the pressure, temperature, moisture level, and trace gas level measurements inside a block with a finite moist air volume. There is no mass or energy flow through the sensor.

Blocks with a finite moist air volume contain an internal node, which represents the moist air volume inside the component. (For a complete list of these blocks, see “Blocks with Moist Air Volume”.) Regular sensors cannot connect to this internal node. Therefore, each of the blocks with a finite moist air volume has a physical signal output port **F**, which outputs a vector signal containing the pressure (in Pa), temperature (in K), moisture level, and trace gas level measurements inside the component. For more information, see “Measuring Moisture and Trace Gas Levels”.

Use the Measurement Selector (MA) block to unpack the vector signal and reassign units to pressure and temperature values. Connect the input port **F** of the Measurement Selector (MA) block to the physical signal output port **F** of a block with finite internal moist air volume to access the data.

Connect the output ports of the Measurement Selector (MA) block to PS-Simulink Converter blocks to transform the output physical signals into Simulink signals, for example, for plotting or additional data processing. Use the **Output signal unit** parameter of the PS-Simulink Converter blocks connected to ports **P** and **T** to reassign units to pressure and temperature values, respectively.

Ports

Input

F — Vector physical signal containing pressure, temperature, humidity, and trace gas levels data

physical signal vector

Connect this port to the physical signal output port **F** of a block with finite internal moist air volume to access the data. Blocks with a finite moist air volume have a physical signal output port **F**, which outputs a vector signal containing the pressure (in Pa), temperature (in K), moisture level, and trace gas level measurements inside the component. Use the Measurement Selector (MA) block to unpack this vector signal.

Output

P — Pressure measurement, Pa

physical signal

Physical signal output port for pressure measurement.

T — Temperature measurement, K

physical signal

Physical signal output port for temperature measurement.

W — Moisture level measurement, unitless

physical signal

Physical signal output port for moisture level measurement. To select the quantity you want to measure, use the **Measured moisture specification** parameter.**G — Trace gas level measurement, unitless**

physical signal

Physical signal output port for trace gas level measurement. To select the quantity you want to measure, use the **Measured trace gas specification** parameter.**Parameters****Measured moisture specification — Select the moisture property to measure**

Relative humidity (default) | Specific humidity | Mole fraction | Humidity ratio

Select the property to measure:

- Relative humidity — Physical signal port **W** reports the relative humidity.
- Specific humidity — Physical signal port **W** reports the specific humidity.
- Mole fraction — Physical signal port **W** reports the water vapor mole fraction.
- Humidity ratio — Physical signal port **W** reports the humidity ratio.

Measured trace gas specification — Select the trace gas property to measure

Mass fraction (default) | Mole fraction

Select the property to measure:

- Mass fraction — Physical signal port **G** reports the trace gas mass fraction.
- Mole fraction — Physical signal port **G** reports the trace gas mole fraction.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Humidity & Trace Gas Sensor (MA) | Pressure & Temperature Sensor (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Mechanical Rotational Reference

Reference connection for mechanical rotational ports



Library

Mechanical Rotational Elements

Description

The Mechanical Rotational Reference block represents a reference point, or frame, for all mechanical rotational ports. All rotational ports that are rigidly clamped to the frame (ground) must be connected to a Mechanical Rotational Reference block.

Ports

The block has one mechanical rotational port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mechanical Translational Reference

Topics

“Grounding Rules”

Introduced in R2007a

Mechanical Translational Reference

Reference connection for mechanical translational ports



Library

Mechanical Translational Elements

Description

The Mechanical Translational Reference block represents a reference point, or frame, for all mechanical translational ports. All translational ports that are rigidly clamped to the frame (ground) must be connected to a Mechanical Translational Reference block.

Ports

The block has one mechanical translational port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mechanical Rotational Reference

Topics

“Grounding Rules”

Introduced in R2007a

Memristor

Ideal memristor with nonlinear dopant drift approach

Library: Simscape / Foundation Library / Electrical / Electrical Elements



Description

This block allows you to model an ideal memristor with a nonlinear dopant drift approach. The behavior of memristor is similar to a resistor, except that its resistance (also called memristance) is a function of the current that has passed through the device. The memristance is defined by two states, A and B, with some fraction of the device in one of those states at a given time.

The nonlinear dopant drift model, [1], is described with the following equations:

$$V = M \cdot I$$

$$M = \xi \cdot R_A + (1 - \xi) \cdot R_B$$

$$\frac{d\xi}{dt} = \frac{I}{Q_0} F_p(\xi)$$

where

- V is the voltage across the memristor.
- M is the memristance.
- I is the current entering the + terminal.
- R_A and R_B are the resistances of the A and B states, respectively.
- ξ is the fraction of the memristor in state A. A positive current from the + terminal to the - terminal increases ξ . Similarly, a positive current from the - terminal to the + terminal decreases ξ . The value of ξ is bounded by 0 and 1.
- t is time.
- Q_0 is the total charge required to make the memristor transition from being fully in one state to being fully in the other state.
- $F_p(\xi)$ is a "window" function, which keeps ξ in the window between 1 and 0, and therefore gives zero drift at the boundaries of the device.

The window function is

$$F_p(\xi) = 1 - (2\xi - 1)^{2p}$$

where p is a positive integer. This function is modified when ξ is close to either 0 or 1, to improve numerical stability.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

+ – Positive terminal

electrical

Electrical conserving port associated with the memristor positive terminal.

- – Negative terminal

electrical

Electrical conserving port associated with the memristor negative terminal.

Parameters

Resistance of state A – Entire memristor in state A

1 Ω (default) | positive scalar

The resistance if the entire memristor is in state A, that is, if $\xi = 1$. The value should be greater than 0.

Resistance of state B – Entire memristor in state B

100 Ω (default) | positive scalar

The resistance if the entire memristor is in state B, that is, if $\xi = 0$. The value should be greater than 0.

Total charge required for full state transition – Charge required for full transition between states

10 mC (default) | positive scalar

The total charge flow that is required to transition the memristor from being fully in one state to being fully in the other state.

State A fraction at t=0 – Fraction of memristor in state A at start of simulation

0 (default) | min/max: (0,1)

The initial condition for ξ at the start of the simulation. This parameter sets a high priority variable target within the block. The value should be greater than or equal to 0 and less than or equal to 1.

Exponent of the window function – Drift control at the boundaries

2 (default) | positive integer scalar

The exponent, p , of the window function, which keeps the value of ξ between 0 and 1.

References

- [1] Joglekar, Y. N., and S. J. Wolf. "The elusive memristor: properties of basic electrical circuits." *European Journal of Physics*. 30, 2009, pp. 661-675.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Introduced in R2016b

MMF Sensor

Ideal magnetomotive force sensor



Library

Magnetic Sensors

Description

The MMF Sensor block represents an ideal magnetomotive force (mmf) sensor, that is, a device that converts the mmf measured between any magnetic connections into a physical signal proportional to the mmf.

Connections N and S are conserving magnetic ports through which the sensor is connected to the circuit. The physical signal port outputs the value of the mmf.

Ports

The block has the following ports:

N

Magnetic conserving port associated with the sensor North terminal.

S

Magnetic conserving port associated with the sensor South terminal.

The block also has a physical signal output port, which outputs the value of the mmf.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled MMF Source | MMF Source | PS-Simulink Converter

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2010a

MMF Source

Ideal magnetomotive force source



Library

Magnetic Sources

Description

The MMF Source block represents an ideal magnetomotive force (mmf) source that is powerful enough to maintain specified constant mmf across its output terminals, regardless of the flux flowing through the source.

You specify the output mmf by using the **Constant mmf** parameter, which can be positive, negative, or zero.

Parameters

Constant mmf

Output mmf. You can specify any real value. The default value is 1 A.

Ports

The block has two magnetic conserving ports associated with its terminals.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Flux Source

Introduced in R2010a

Moist Air Properties (MA)

Global moist air properties for attached circuit

Library: Simscape / Foundation Library / Moist Air / Utilities



Description

The Moist Air Properties (MA) block defines the moist air properties that act as global parameters for all the blocks connected to a circuit. The moist air mixture is composed of dry air, water vapor, and an optional trace gas. The default trace gas is carbon dioxide.

Dry air, water vapor, and trace gas are assumed to be semiperfect gas. Pressure, temperature, and density obey the ideal gas law. Other properties are functions of temperature. You specify them as one-dimensional arrays corresponding to the **Temperature vector**.

Each topologically distinct moist air circuit in a diagram can have a Moist Air Properties (MA) block connected to it. If no Moist Air Properties (MA) block is attached to a circuit, the blocks in this circuit use the properties corresponding to the default Moist Air Properties (MA) block parameter values.

Ports

Conserving

A — Connection port

moist air

Conserving port that connects the block to the moist air network. You can connect it to any point on a moist air connection line in a block diagram. When you connect the Moist Air Properties (MA) block to a connection line, the software automatically identifies the moist air blocks connected to the particular circuit and propagates the properties to all these blocks.

Parameters

Dry Air

Dry air specific gas constant — Universal gas constant divided by molar mass of dry air

287.047 J/kg/K (default)

Universal gas constant divided by molar mass of dry air.

Temperature vector — Vector of temperature values for table lookup

vector

Vector of temperature values, to be used for table lookup of other properties as a function of temperature. Dry air, water vapor, and trace gas properties share the same temperature vector.

The default value is [-56.55, -50 : 10 : -10, -5 : 1 : 5, 10 : 10 : 350]' degC.

Dry air specific enthalpy vector – Vector of specific enthalpy values for table lookup vector

The vector of specific enthalpy values of dry air, to be used for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size.

The default value is [342.416126230579; 349.005511058471; 359.063577249119; 369.119948684177; 379.175469465129; 389.230944678417; 394.258910928084; 395.264534937366; 396.27017101329; 397.275819932805; 398.281482472874; 399.287159410541; 400.292851523015; 401.298559587728; 402.304284382408; 403.310026685143; 404.315787274435; 409.34489180313; 419.404921755863; 429.468036401264; 439.535033154839; 449.606720928652; 459.683919949603; 469.767460521911; 479.858181001348; 489.956925196484; 500.064539372946; 510.181869006204; 520.309755403634; 530.449032296018; 540.60052248087; 550.765034584465; 560.943359995745; 571.136270013102; 581.344513234296; 591.568813210327; 601.809866375885; 612.068340261954; 622.344871990218; 632.640067043998; 642.95449830646; 653.288705353699; 663.643193987949; 674.018435994423; 684.414869104228; 694.832897145098; 705.272890361557; 715.735185886199; 726.220088344236; 736.727870574087; 747.258774447567; 757.813011774199] kJ/kg.

Dry air dynamic viscosity vector – Vector of dynamic viscosity values for table lookup vector

The vector of dynamic viscosity values of dry air, to be used for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size.

The default value is [14.2568883320012; 14.6140127728333; 15.1517277055779; 15.6806860159119; 16.2012351072083; 16.7137043125028; 16.9670071730589; 17.017438062815; 17.0677930842452; 17.1180725309876; 17.1682766951315; 17.2184058672268; 17.2684603362935; 17.3184403898301; 17.3683463138234; 17.4181783927569; 17.4679369096204; 17.7156358640698; 18.2056751785154; 18.68879035749; 19.1652344664982; 19.6352478927873; 20.099059103658; 20.5568853601578; 21.0089333871663; 21.4554000014051; 21.8964726992323; 22.3323302062775; 22.7631429910631; 23.1890737447833; 23.6102778293792; 24.0269036959922; 24.4390932757886; 24.8469823450548; 25.2507008663546; 25.6503733074314; 26.0461189394339; 26.4380521159325; 26.8262825340948; 27.2109154792923; 27.5920520543128; 27.969789394274; 28.3442208682439; 28.7154362685068; 29.0835219883356; 29.4485611890712; 29.8106339572472; 30.1698174524429; 30.5261860464955; 30.8798114546565; 31.2307628592324; 31.5791070262086] s*μPa.

Dry air thermal conductivity vector – Vector of thermal conductivity values for table lookup vector

The vector of thermal conductivity values of dry air, to be used for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size.

The default value is [19.8808489374933; 20.4162454629695; 21.2248743590758; 22.0232452890903; 22.8117314495857; 23.5906913619588; 23.9767075412237;

24.053639990175; 24.1304826682191; 24.2072359051629; 24.2839000294007;
 24.3604753679152; 24.4369622462801; 24.5133609886625; 24.5896719178249;
 24.6658953551279; 24.7420316205333; 25.1214164701832; 25.8738283029331;
 26.6180150229276; 27.3542674377332; 28.0828634735341; 28.8040686837224;
 29.5181367785657; 30.2253101618423; 30.9258204644224; 31.6198890677835;
 32.3077276126633; 32.9895384896798; 33.6655153099498; 34.33584335461;
 35.0007000027879; 35.6602551380254; 36.3146715334901; 36.9641052165342;
 37.6087058133173; 38.248616874307; 38.8839761815308; 39.5149160384798;
 40.1415635435708; 40.7640408480637; 41.3824653993118; 41.9969501701893;
 42.6076038755117; 43.2145311762238; 43.8178328720941; 44.4176060836121;
 45.013944423749; 45.6069381602018; 46.1966743687041; 46.783237077953;
 47.3667074066625] mW/m/K.

Minimum valid pressure — Lowest pressure allowed

1 kPa (default)

Lowest pressure allowed in the moist air network. The simulation issues an error when pressure is out of range.

Maximum valid pressure — Highest pressure allowed

inf MPa (default)

Highest pressure allowed in the moist air network. The simulation issues an error when pressure is out of range.

Minimum valid temperature — Lowest temperature allowed

-56.55 degC (default)

Lowest temperature allowed in the moist air network. The simulation issues an error when temperature is out of range.

Maximum valid temperature — Highest temperature allowed

350 degC (default)

Highest temperature allowed in the moist air network. The simulation issues an error when temperature is out of range.

Atmospheric pressure — Absolute pressure of the environment

0.101325 MPa (default)

Absolute pressure of the environment.

Atmospheric temperature — Absolute temperature of the environment

20 degC (default)

Absolute temperature of the environment.

Water Vapor

Water vapor specific gas constant — Universal gas constant divided by molar mass of water vapor

461.523 J/kg/K (default)

Universal gas constant divided by molar mass of water vapor.

Water vapor saturation pressure vector – Vector of vapor saturation values for table lookup

vector

Vector of vapor saturation values for water as a function of temperature, to be used for one-dimensional table lookup. Moist air becomes saturated when the partial pressure of water vapor is equal to the water vapor saturation pressure. Therefore, condensation may occur. The vector size must be the same as the temperature vector size. Dry air, water vapor, and trace gas properties share the same temperature vector.

The default value is [1.71168953425982e-06; 3.93770601979424e-06;

1.28411717710344e-05;

3.80051394873809e-05; .00010323902900209; .000259873810798063; .00040174102216384; .000437454836117844; .000476040820618202; .000517704666185737; .000562664883741207; .000611212677444345; .000657088049117899; .000705987905898279; .000758082381149132; .000813549384183233; .000872574861129522; .00122818386934022; .0023392147667769; .00424668834054807; .00738442748706953; .0123512704340234; .0199458019246787; .0312006356960619; .0474147199263783; .0701823607447713; .10141797792131; .143375967241115; .198665399739302; .270259606559999; .361500961984849; .476101381081492; .618139196722055; .792053183687694; 1.0026345688121; 1.25501792086105; 1.55467186826983; 1.90739066433297; 2.31928772772542; 2.79679245576864; 3.34665187151016; 3.97593907083533; 4.69207105435535; 5.5028394740883; 6.41645928168037; 7.4416425436169; 8.58770832955728; 9.86474556030261; 11.2838558865497; 12.8575218898034; 14.6001810568052; 16.5291642526045] MPa.

Water specific enthalpy of vaporization vector – Vector of specific enthalpy of vaporization values for table lookup

vector

Vector of the difference between the specific enthalpy of saturated water vapor and the specific enthalpy of saturated liquid water as a function of temperature, to be used for one-dimensional table lookup. The vector size must be the same as the temperature vector size. Dry air, water vapor, and trace gas properties share the same temperature vector.

The default value is [2836.88241275372; 2837.81392500514; 2838.63937175807; 2838.7309929628; 2838.06905313927; 2836.62597341095; 2835.60023952573; 2835.37006381376; 2835.13143158077; 2834.88429145066; 2834.62859062897; 2500.93420564316; 2498.55329907119; 2496.17495082036; 2493.79885912205; 2491.42474723191; 2489.05236104642; 2477.20875029194; 2453.54955988604; 2429.83856603313; 2406.00136954922; 2381.97406342174; 2357.69101156389; 2333.08088387814; 2308.06565480412; 2282.56034405021; 2256.47287422313; 2229.70428017508; 2202.14968030993; 2173.69998896199; 2144.24368406447; 2113.66758247452; 2081.85585110571; 2048.68725710494; 2014.03141249899; 1977.7449923284; 1939.66849592886; 1899.62342931686; 1857.40930169838; 1812.79975416601; 1765.53721731251; 1715.32525772152; 1661.81700477787; 1604.59703745016; 1543.1534616334; 1476.83692476464; 1404.80240352424; 1325.92091736114; 1238.61667822208; 1140.5102987018; 1027.62017777647; 892.733785613825] kJ/kg.

Water vapor specific enthalpy vector – Vector of specific enthalpy values for table lookup

vector

The vector of specific enthalpy values of water vapor, to be used for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size. Dry air, water vapor, and trace gas properties share the same temperature vector.

The default value is [2396.55944251649; 2408.68643343608; 2427.1988031141; 2445.702165897; 2464.18429108356; 2482.62529466839; 2491.82135326629; 2493.6580151792; 2495.49372578218; 2497.32843835745; 2499.16210446923; 2500.99462758899; 2502.83092214066; 2504.6665223621; 2506.50140563013; 2508.335548891; 2510.16892865793; 2519.32352241995; 2537.56068088674; 2555.67742615292; 2573.6403223998; 2591.4109179101; 2608.9455037701; 2626.19492016262; 2643.10444358428; 2659.61377634497; 2675.58278696023; 2696.1846256545; 2716.49989553741; 2736.6235210957; 2756.61216047011; 2776.50561773853; 2796.33378922508; 2816.11979583049; 2835.88181423877; 2855.63430574511; 2875.38889327595; 2895.15500577172; 2914.94035964689; 2934.75132290228; 2954.59319330244; 2974.47041285505; 2994.38673458964; 3014.34535328009; 3034.34900867103; 3054.40006755764; 3074.50058946898; 3094.65237953784; 3114.85703128106; 3135.11596137801; 3155.43043805905; 3175.80160435813] kJ/kg.

Water vapor dynamic viscosity vector – Vector of dynamic viscosity values for table lookup

vector

The vector of dynamic viscosity values of water vapor, to be used for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size. Dry air, water vapor, and trace gas properties share the same temperature vector.

The default value is [6.81365662228272; 7.04953750742707; 7.40970098298307; 7.76991278093907; 8.13017290129507; 8.49048134405107; 8.67065368632907; 8.70668960445667; 8.74272600580827; 8.77876289038387; 8.81480025818347; 8.85083810920707; 8.88687644345467; 8.92291526092627; 8.95895456162187; 8.99499434554147; 9.03103461268507; 9.21124319676307; 9.57169660671907; 9.93219833907507; 10.2927483938311; 10.6533467709871; 11.0139934705431; 11.3746884924991; 11.7354318368551; 12.0962235036111; 12.4570634927671; 12.8179518043231; 13.1788884382791; 13.5398733946351; 13.9009066733911; 14.2619882745471; 14.6231181981031; 14.9842964440591; 15.3455230124151; 15.7067979031711; 16.0681211163271; 16.4294926518831; 16.7909125098391; 17.1523806901951; 17.5138971929511; 17.8754620181071; 18.2370751656631; 18.5987366356191; 18.9604464279751; 19.3222045427311; 19.6840109798871; 20.0458657394431; 20.4077688213991; 20.7697202257551; 21.1317199525111; 21.4937680016671] s* μ Pa.

Water vapor thermal conductivity vector – Vector of thermal conductivity values for table lookup

vector

The vector of thermal conductivity values of water vapor, to be used for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size. Dry air, water vapor, and trace gas properties share the same temperature vector.

The default value is [11.46288215976; 11.941997488935; 12.676358586935; 13.414208884935; 14.155548382935; 14.900377080935; 15.274099879935; 15.348949115735; 15.423833243535; 15.498752263335; 15.573706175135;

15.648694978935; 15.723718674735; 15.798777262535; 15.873870742335;
 15.948999114135; 16.024162377935; 16.400502076935; 17.155798374935;
 17.914583872935; 18.676858570935; 19.442622468935; 20.211875566935;
 20.984617864935; 21.760849362935; 22.540570060935; 23.323779958935;
 24.110479056935; 24.900667354935; 25.694344852935; 26.491511550935;
 27.292167448935; 28.096312546935; 28.903946844935; 29.715070342935;
 30.529683040935; 31.347784938935; 32.169376036935; 32.994456334935;
 33.823025832935; 34.655084530935; 35.490632428935; 36.329669526935;
 37.172195824935; 38.018211322935; 38.867716020935; 39.720709918935;
 40.577193016935; 41.437165314935; 42.300626812935; 43.167577510935;
 44.038017408935] mW/m/K.

Water vapor diffusivity in air – Smoothing coefficient for flow reversals

25 mm²/s (default)

Coefficient for the diffusive flux of water vapor in the moist air mixture due to the difference in concentration of water vapor. During mixture flow reversal, the water vapor mass flow rate at the ports smoothly transitions between the upstream and downstream value based on the diffusivity.

Trace Gas

Trace gas model – Select how to model trace gas in the air mixture

None (default) | Track mass fraction only | Track mass fraction and gas properties

Select how the block models the amount of trace gas in the air mixture:

- **None** — No trace gas is present. The moist air mixture consists of only dry air and water vapor. Any nonzero values of trace gas level in parameters and variable targets of the blocks connected to the circuit are ignored. All connections to the trace gas source blocks in the model are also ignored. Therefore, you do not need to modify your model based on the trace gas modeling selection.
- **Track mass fraction only** — The trace gas level can be nonzero and vary during simulation. However, the amount of trace gas is assumed to be small enough to have a negligible impact on the fluid properties of the moist air mixture.
- **Track mass fraction and gas properties** — The trace gas level can be nonzero and vary during simulation. The fluid properties of the moist air mixture depend on the amount of trace gas in the mixture.

For more information on the impact of these options on block and system equations, see “Trace Gas Modeling Options”.

Trace gas specific gas constant – Universal gas constant divided by molar mass of trace gas

188.923 J/kg/K (default)

Universal gas constant divided by molar mass of trace gas.

Dependencies

Enabled when the **Trace gas model** parameter is set to Track mass fraction only or Track mass fraction and gas properties.

Trace gas specific enthalpy vector – Vector of specific enthalpy values for table lookup

vector

The vector of specific enthalpy values of trace gas, to be used for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size. Dry air, water vapor, and trace gas properties share the same temperature vector.

The default value is [439.555216260064; 444.670268200251; 452.538618847003; 460.487124258496; 468.522588768941; 476.649415958793; 480.74798538725; 481.57054318762; 482.394052398751; 483.218514288831; 484.043930029015; 484.870300696555; 485.697627277859; 486.525910671489; 487.355151691079; 488.185351068185; 489.016509455051; 493.186704139431; 501.599194953048; 510.107689961685; 518.711611080182; 527.410026793075; 536.201764778798; 545.085474203452; 554.059675505208; 563.122805485367; 572.27325096919; 581.509373027928; 590.829525105886; 600.232066705818; 609.715373581392; 619.277845178105; 628.917909922924; 638.634028838794; 648.424697859328; 658.288449140109; 668.223851601449; 678.229510889093; 688.304068901113; 698.44620299875; 708.654624994591; 718.928079991818; 729.265345132425; 739.665228299475; 750.126566808087; 760.648226111439; 771.229098541268; 781.868102096908; 792.564179292476; 803.316296068366; 814.123440770426; 824.984623198037] kJ/kg.

Dependencies

Enabled when the **Trace gas model** parameter is set to Track mass fraction and gas properties.

Trace gas dynamic viscosity vector – Vector of dynamic viscosity values for table lookup

vector

The vector of dynamic viscosity values of trace gas, to be used for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size. Dry air, water vapor, and trace gas properties share the same temperature vector.

The default value is [10.8921054191698; 11.2215357085649; 11.7233300740382; 12.2234238147811; 12.7215124453467; 13.2173328890673; 13.4643203184773; 13.5136406274989; 13.562934752385; 13.6122025024997; 13.6614436900741; 13.7106581301692; 13.7598456406398; 13.8090060420983; 13.8581391578796; 13.9072448140062; 13.9563228391538; 14.2012926853717; 14.6890687376044; 15.1738428124161; 15.6554929043332; 16.1339159800463; 16.6090257998657; 17.0807510097148; 17.5490334645512; 18.0138267508952; 18.4750948814453; 18.9328111390399; 19.3869570507278; 19.8375214755778; 20.2844997922255; 20.7278931741254; 21.1677079421287; 21.6039549853963; 22.036649242835; 22.4658092382469; 22.8914566632375; 23.3136160026627; 23.7323141980278; 24.1475803447965; 24.5594454200432; 24.9679420372927; 25.3731042257492; 25.774967231431; 26.1735673380029; 26.5689417053362; 26.9611282240413; 27.350165384404; 27.73609215832; 28.118947892972; 28.4987722151178; 28.8756049449796] s* μ Pa.

Dependencies

Enabled when the **Trace gas model** parameter is set to Track mass fraction and gas properties.

Trace gas thermal conductivity vector – Vector of thermal conductivity values for table lookup

vector

The vector of thermal conductivity values of trace gas, to be used for one-dimensional table lookup based on the corresponding temperature value. The vector size must be the same as the temperature vector size. Dry air, water vapor, and trace gas properties share the same temperature vector.

The default value is [10.489430678843; 10.9507672150864; 11.666197960308; 12.392559739764; 13.1299519771612; 13.8778409888416; 14.2554325416312; 14.3312274189707; 14.4071122346908; 14.4830860227061; 14.5591478179508; 14.6352966574703; 14.7115315815553; 14.787851634907; 14.8642558678259; 14.9407433374142; 15.0173131087819; 15.401364480906; 16.1751373988038; 16.9558134611595; 17.7432208682398; 18.5366976599676; 19.3360988502003; 20.1412225692689; 20.9517507672929; 21.7672310558863; 22.5871115634685; 23.4107980865695; 24.2377021890869; 25.0672663781241; 25.8989689893876; 26.732318357514; 27.5668446164644; 28.4020933015927; 29.2376215872816; 30.0729965057542; 30.9077942826816; 31.7416001931033; 32.5740086297993; 33.4046232539953; 34.2330571812868; 35.0589331874899; 35.8818839295189; 36.701552179391; 37.5175910702657; 38.3296643536716; 39.1374466671854; 39.9406238119064; 40.7388930391313; 41.5319633456945; 42.3195557774861; 43.1014037407076] mW/m/K.

Dependencies

Enabled when the **Trace gas model** parameter is set to Track mass fraction and gas properties.

Trace gas diffusivity in air – Smoothing coefficient for flow reversals

16 mm²/s (default)

Coefficient for the diffusive flux of trace gas in the moist air mixture due to the difference in concentration of trace gas. During mixture flow reversal, the trace gas mass flow rate at the ports smoothly transitions between the upstream and downstream value based on the diffusivity.

Dependencies

Enabled when the **Trace gas model** parameter is set to Track mass fraction only or Track mass fraction and gas properties.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Moisture & Trace Gas Cap (MA)

(To be removed) Moist air source port terminator

Note The Moisture & Trace Gas Cap (MA) block will be removed in a future release. For more information, see “Compatibility Considerations”.

Library: Simscape / Foundation Library / Moist Air / Sources /
Moisture & Trace Gas Sources



Description

Use Moisture & Trace Gas Cap (MA) block to terminate unused moist air source ports on other blocks.

Blocks with a finite moist air volume contain an internal node that lets you model moisture and trace gas levels inside the component. This internal node belongs to the moist air source domain. The corresponding port is named **S**. If no moisture or trace gas is injected or extracted from a block with moist air volume, connect its port **S** to a Moisture & Trace Gas Cap (MA) block.

Ports

Conserving

S — Terminate unused moist air source ports on other blocks

moist air source

Connect this port to an unused port **S** of another block.

Compatibility Considerations

Hide unused moist air source ports

Not recommended starting in R2019b

Starting in R2019b, all Moist Air library blocks with a finite moist air volume have a new parameter, **Moisture and trace gas source**, which controls the visibility of port **S** and provides options for modeling moisture and trace gas levels inside the component. In previous releases, you needed the Moisture & Trace Gas Cap (MA) block to cap unused ports **S** in the model. Now, ports **S** should be exposed only when in use.

Currently, legacy models using the Moisture & Trace Gas Cap (MA) block work the same as in previous releases. However, this block will be removed in the future. To update your legacy models, in each block connected to a Moisture & Trace Gas Cap (MA) block, set the **Moisture and trace gas source** parameter to **None**. Then delete the Moisture & Trace Gas Cap (MA) blocks and unused connection lines.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Cap (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Moisture Source (MA)

Inject or extract moisture at a constant rate

Library: Simscape / Foundation Library / Moist Air / Sources /
Moisture & Trace Gas Sources



Description

The Moisture Source (MA) block represents a constant source or sink of moisture for the connected moist air volume. A positive or negative moisture mass flow rate results in moisture being added or removed, respectively.

You can add moisture as water vapor or liquid water. For water vapor, the energy associated with the added or removed moisture is

$$\Phi_S = \begin{cases} \dot{m}_{\text{specified}} \cdot h_w(T_{\text{specified}}), & \text{if } \dot{m}_{\text{specified}} \geq 0 \\ \dot{m}_{\text{specified}} \cdot h_w(T_S), & \text{if } \dot{m}_{\text{specified}} < 0 \end{cases}$$

where:

- $\dot{m}_{\text{specified}}$ is the water vapor mass flow rate specified by the **Moisture mass flow rate** parameter.
- h_w is the water vapor specific enthalpy.
- $T_{\text{specified}}$ is the temperature of added moisture, as specified by the block parameters. The block uses this value to evaluate the specific enthalpy of the added moisture only. The specific enthalpy of removed moisture is based on the temperature of the connected moist air volume.
- T_S is the temperature at port **S**, which is the same as the temperature of the connected moist air volume.

For liquid water, the energy associated with the added or removed moisture is

$$\Phi_S = \begin{cases} \dot{m}_{\text{specified}} \cdot (h_w(T_{\text{specified}}) - \Delta h_{\text{vap}}(T_{\text{specified}})), & \text{if } \dot{m}_{\text{specified}} \geq 0 \\ \dot{m}_{\text{specified}} \cdot (h_w(T_S) - \Delta h_{\text{vap}}(T_S)), & \text{if } \dot{m}_{\text{specified}} < 0 \end{cases}$$

where Δh_{vap} is the water specific enthalpy of vaporization.

Port **S** is a moist air source conserving port. Connect this port to port **S** of a block with finite moist air volume to add or remove moisture through that block. For more information, see “Using Moisture and Trace Gas Sources”.

Ports

Conserving

S – Inject or extract moisture

moist air source

Connect this port to port **S** of a block with finite moist air volume to add or remove moisture through that block.

Parameters

Moisture added or removed — Select whether the source adds or removes moisture as water vapor or liquid water

Vapor (default) | Liquid

Select whether the source adds or removes moisture as water vapor or liquid water:

- **Vapor** — The enthalpy of the added or removed moisture corresponds to the enthalpy of water vapor, which is greater than that of liquid water.
- **Liquid** — The enthalpy of the added or removed moisture corresponds to the enthalpy of liquid water, which is less than that of water vapor.

Rate of added moisture — Constant mass flow rate through the source

0 kg/s (default)

Water vapor mass flow rate through the source. A positive value adds moisture to the connected moist air volume. A negative value extracts moisture from that volume.

Added moisture temperature specification — Select specification method for the temperature of added moisture

Atmospheric temperature (default) | Specified temperature

Select a specification method for the moisture temperature:

- **Atmospheric temperature** — Use the atmospheric temperature, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified temperature** — Specify a value by using the **Temperature of added moisture** parameter.

Temperature of added moisture — Moisture temperature

293.15 K (default)

Enter the desired temperature of added moisture. This temperature remains constant during simulation. The block uses this value to evaluate the specific enthalpy of the added moisture only. The specific enthalpy of removed moisture is based on the temperature of the connected moist air volume.

Dependencies

Enabled when the **Added moisture temperature specification** parameter is set to **Specified temperature**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Moisture Source (MA)

Topics

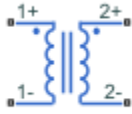
“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Mutual Inductor

Mutual inductor in electrical systems



Library

Electrical Elements

Description

The Mutual Inductor block models a mutual inductor, described with the following equations:

$$V1 = L1 \frac{dI1}{dt} + M \frac{dI2}{dt}$$

$$V2 = L2 \frac{dI2}{dt} + M \frac{dI1}{dt}$$

$$M = k\sqrt{L1 \cdot L2}$$

where

$V1$	Voltage across winding 1
$V2$	Voltage across winding 2
$I1$	Current flowing into the + terminal of winding 1
$I2$	Current flowing into the + terminal of winding 2
$L1, L2$	Winding self-inductances
M	Mutual inductance
k	Coefficient of coupling, $0 < k < 1$
t	Time

This block can be used to represent an AC transformer. If inductance and mutual inductance terms are not important in a model, or are unknown, you can use the Ideal Transformer block instead.

The two electrical networks connected to the primary and secondary windings must each have their own Electrical Reference block.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Inductance L1

Self-inductance of the first winding. The default value is 10 H.

Inductance L2

Self-inductance of the second winding. The default value is 0.1 H.

Coefficient of coupling

Coefficient of coupling, which defines the mutual inductance. The parameter value should be greater than zero and less than 1. The default value is 0.9.

Ports

The block has four electrical conserving ports. Polarity is indicated by the + and - signs. Ports labeled 1+ and 1- are connected to the primary winding. Ports labeled 2+ and 2- are connected to the secondary winding.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Ideal Transformer

Introduced in R2007a

Op-Amp

Ideal operational amplifier

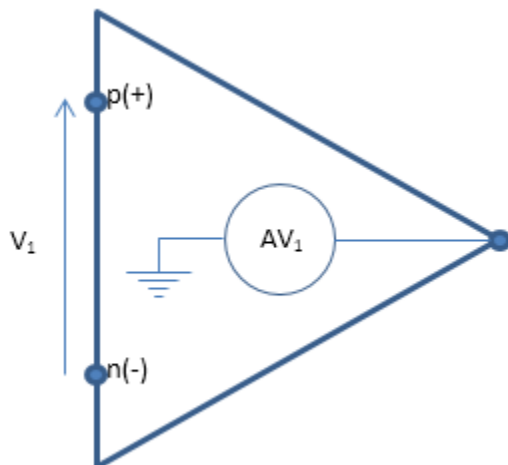


Library

Electrical Elements

Description

The Op-Amp block models an ideal operational amplifier (op-amp). The figure shows the implementation schematic.



The block implementation is based on the following assumptions:

- The ideal op-amp gain A is assumed to be infinite
- Then, for finite output, must have $V1 = 0$
- Ideal op-amp also implies current from p to n is zero ($i1 = 0$)

These assumptions result in the following equations for the block:

```
equations
  v1 == p.v - n.v;
  v1 == 0;
  i1 == 0;
end
```

For more information, click the **Source code** link in the block dialog box.

You can initialize the **Current into output node** variable prior to simulation. For more information, see “Block-Level Variable Initialization”.

Ports

The block has three electrical conserving ports.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

Introduced in R2007a

Open Circuit

Electrical port terminator that draws no current

Library: Simscape / Foundation Library / Electrical / Electrical Elements



Description

The Open Circuit block represents an electrical terminal that draws no current.

You can use this block to terminate electrical ports on other blocks that you want to leave open circuit. This is not necessary because if you leave a conserving port unconnected, the physical network sets all the Through variables at that port to 0. However, terminator blocks improve diagram readability.

You can also use this block to set the initial absolute voltage at a node.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

V – Zero current

electrical

Electrical conserving port that draws no current.

Compatibility Considerations

Capping unconnected ports is no longer required

Behavior changed in R2019b

This block is no longer required. Starting in R2019b, the restriction that disallowed unconnected conserving ports in Simscape models has been lifted. Now, if you leave a conserving port unconnected, the physical network sets all the Through variables at this port to 0. However, you can still use terminator blocks to improve diagram readability.

There are no plans to remove the terminator blocks. All existing models that use these blocks work the same as in previous releases.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Cap (2P) | Cap (G) | Cap (MA) | Cap (TL) | Hydraulic Cap | Perfect Insulator | Rotational Free End | Translational Free End

Introduced in R2012b

Perfect Insulator

Thermal element with perfect insulation and no thermal mass

Library: Simscape / Foundation Library / Thermal / Thermal Elements



Description

The Perfect Insulator block represents a thermal element with perfect insulation and no thermal mass.

You can use this block as an insulation for thermal ports to prevent heat exchange with the environment and to model an adiabatic process. This is not necessary because if you leave a conserving port unconnected, the physical network sets all the Through variables at that port to 0. However, terminator blocks improve diagram readability.

You can also use this block to set the initial temperature at a node.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

A — Zero heat flow

thermal

Thermal conserving port with zero heat flow.

Compatibility Considerations

Capping unconnected ports is no longer required

Behavior changed in R2019b

This block is no longer required. Starting in R2019b, the restriction that disallowed unconnected conserving ports in Simscape models has been lifted. Now, if you leave a conserving port unconnected, the physical network sets all the Through variables at this port to 0. However, you can still use terminator blocks to improve diagram readability.

There are no plans to remove the terminator blocks. All existing models that use these blocks work the same as in previous releases.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Cap (2P) | Cap (G) | Cap (MA) | Cap (TL) | Hydraulic Cap | Open Circuit | Rotational Free End | Translational Free End

Introduced in R2013a

Pipe (2P)

Rigid conduit for fluid flow in two-phase fluid systems



Library

Two-Phase Fluid/Elements

Description

The Pipe (2P) block models the flow dynamics of a two-phase fluid inside a rigid pipe. The dynamic compressibility and thermal capacity of the fluid are assumed non-negligible. The two-phase fluid conserving ports A and B represent the pipe inlets. The thermal conserving port H represents the pipe wall, through which heat transfer with the pipe surroundings occurs.

Fluid Inertia

The block provides an option to model fluid inertia, the resistance to sudden changes in mass flow rate. By default, fluid inertia modeling is turned off. This setting is appropriate when the pressure forces driving the flow far exceed the inertial forces acting on the flow.

The default setting reduces computational costs and is recommended for most models. However, fluid inertia can become important if the mass flow rate changes rapidly. In such cases, turning fluid inertia modeling on can help improve simulation accuracy.

Energy Balance

Energy conservation in the pipe is observed through the equation:

$$M\dot{u}_I + (\dot{m}_A + \dot{m}_B)u_I = \phi_A + \phi_B + Q_H,$$

where:

- M is the fluid mass inside the pipe.
- u_I is the specific internal energy of the fluid inside the pipe.
- ϕ_A is the energy flow rate into the pipe through port A.
- ϕ_B is the energy flow rate into the pipe through port B.
- Q_H is the heat flow rate into the pipe through the pipe wall, represented by port H.

Heat Flow Rate

Heat transfer between the pipe wall and the internal fluid volume is modeled as a convective process, with the heat flow rate computed as:

$$Q_H = h_{\text{coeff}} S_{\text{surf}} (T_H - T_I),$$

where:

- h_{coeff} is the average heat transfer coefficient in the pipe.
- S_{surf} is the pipe surface area.
- T_H is the pipe wall temperature.
- T_I is the temperature of the fluid in the pipe.

The calculation of the heat transfer coefficient depends on the fluid phase. In the subcooled liquid and superheated vapor phases, the coefficient is:

$$h_{\text{coeff}}^* = \frac{k_I^* \text{Nu}^*}{D_h},$$

where the asterisk denotes a value specific to the phase considered (liquid or vapor) and:

- Nu is the average Nusselt number in the pipe.
- k_I is the average thermal conductivity in the pipe.
- D_h is the hydraulic diameter of the pipe (that which a cross section of general shape would have if it were made circular).

In a two-phase mixture, the same coefficient is:

$$h_{\text{coeff}}^M = \frac{k_{I,SL}^M \text{Nu}^M}{D_h},$$

where the subscript M denotes a value specific to the two-phase mixture and the SL subscript indicates a value obtained for the saturated liquid.

Nusselt Number

In laminar flows, the Nusselt number is assumed constant and equal to the value specified in the block dialog box. The laminar flow Nusselt number applies when the Reynolds number is smaller than the value entered for the **Laminar flow upper Reynolds number limit** parameter.

The turbulent flow Nusselt number applies when the Reynolds number is greater than the value entered for the **Laminar flow upper Reynolds number limit** parameter. In the transitional region between laminar and turbulent flow, a cubic polynomial function blends the two Nusselt numbers. This blending ensures a smooth transition between flow regimes.

In the liquid and vapor phases, the Nusselt number for turbulent flow follows from the Gnielinski correlation:

$$\text{Nu}^* = \frac{\frac{f}{8} (\text{Re}^* - 1000) \text{Pr}_I^*}{1 + 12.7 \sqrt{\frac{f}{8}} (\text{Pr}_I^{*2/3} - 1)},$$

where, as before, the asterisk denotes the phase considered and:

- f is the friction factor of the pipe.
- Re is the Reynolds number.

- Pr_1 is the Prandtl number.

The friction factor is calculated as:

$$f = \left\{ -1.8 \log_{10} \left[\frac{6.9}{Re^*} + \left(\frac{\epsilon_r}{3.7} \right)^{1.11} \right] \right\}^{-2},$$

where ϵ_r is the roughness of the pipe. The Reynolds number is calculated as:

$$Re^* = \frac{\dot{m}_{Avg} |D_h v_1^*}{S \nu_1^*},$$

where the subscript Avg denotes an average value between the ports and:

- S is the cross-sectional area of the pipe.
- v_1 is the specific volume.
- ν_1 is the kinematic viscosity.

In the two-phase mixture, the Nusselt number for turbulent flow follows from the Cavallini and Zecchin correlation:

$$Nu^M = 0.05 \left[\left(1 - x_1 + x_1 \sqrt{\frac{v_{SV}}{v_{SL}}} \right) Re_{SL} \right]^{0.8} Pr_{SL}^{0.33},$$

where the subscript SL denotes a value for saturated liquid, the SV subscript a value for saturated vapor, and:

- x_1 is the vapor quality.
- v is the specific volume.

The Reynolds number of the saturated liquid is calculated as:

$$Re_{SL} = \frac{\dot{m}_{Avg} |D_h v_{SL}}{S \nu_{SL}},$$

Mass Balance

Mass conservation in the pipe is observed through the equation:

$$\left[\left(\frac{\partial \rho}{\partial p} \right)_u \dot{p}_I + \left(\frac{\partial \rho}{\partial u} \right)_p \dot{u}_I \right] V = \dot{m}_A + \dot{m}_B + \epsilon_M,$$

where:

- ρ is the fluid density.
- p_1 is the pressure inside the pipe.
- V is the volume of fluid in the pipe.
- \dot{m}_A is the mass flow rate into the pipe through port A.
- \dot{m}_B is the mass flow rate into the pipe through port B.
- ϵ_M is a correction term that accounts for the smoothing of the density partial derivatives across phase transition boundaries.

The block blends the density partial derivatives of the various domains using a cubic polynomial function. At a vapor quality of 0-0.1, this function blends the derivatives of the subcooled liquid and two-phase mixture domains. At a vapor quality of 0.9-1, it blends those of the two-phase mixture and superheated vapor domains. The correction term in the mass conservation equation,

$$\epsilon_M = \frac{M - V/v_I}{\tau},$$

is added to correct for the numerical errors introduced by the cubic polynomial function, with:

- M as the fluid mass in the pipe, computed from the equation:

$$\dot{M} = \dot{m}_A + \dot{m}_B,$$

- v_I as the specific volume of the fluid in the pipe.
- τ as the phase-change time constant—the characteristic duration of a phase-change event. This constant ensures that phase changes do not occur instantaneously, effectively introducing a time lag whenever they occur.

Momentum Balance

The momentum balance equations are defined separately for each half pipe section. In the half pipe adjacent to port A:

$$p_A - p_I = \frac{\dot{m}_A}{S} \left| \frac{\dot{m}_A}{S} (v_I - v_A) \right| + F_{visc,A} + I_A,$$

where:

- p_A is the pressure at port A.
- S is the cross-sectional area of the pipe.
- v_A is the specific volume of the fluid at port A.
- $F_{visc,A}$ is the viscous friction force in the half pipe adjacent to port A.
- I_A is the fluid inertia at port A:

$$I_A = \ddot{m}_A \frac{L}{2S}$$

The parameter L is the pipe length.

In the half pipe adjacent to port B:

$$p_B - p_I = \frac{\dot{m}_B}{S} \left| \frac{\dot{m}_B}{S} (v_I - v_B) \right| + F_{visc,B} + I_B,$$

where:

- p_B is the pressure at port B.
- v_B is the specific volume of the fluid at port B.
- $F_{visc,B}$ is the viscous friction force in the half pipe adjacent to port B.
- I_B is the fluid inertia at port B:

$$I_B = \ddot{m}_B \frac{L}{2S}$$

The fluid inertia terms, I_A and I_B , are zero when the **Fluid inertia** parameter is set to Off. The calculation of the viscous friction forces, $F_{\text{visc},A}$ and $F_{\text{visc},B}$ depends on the flow regime, laminar or turbulent.

Viscous Friction Force in Laminar Flows

In the laminar regime—that is, when the Reynolds number is smaller than the **Laminar flow upper Reynolds number limit** value specified in the block dialog box—the viscous friction force in the half pipe adjacent to port A is

$$F_{\text{visc},A}^{\text{laminar}} = \frac{f_{\text{shape}} L_{\text{eff}} \nu_I \dot{m}_A}{4D_h^2 S},$$

while in the half pipe adjacent to port B it is

$$F_{\text{visc},B}^{\text{laminar}} = \frac{f_{\text{shape}} L_{\text{eff}} \nu_I \dot{m}_B}{4D_h^2 S},$$

where:

- f_{shape} is the pipe shape factor.
- L_{eff} is the effective pipe length—the sum of the pipe length and the aggregate equivalent length of local resistances.
- D_h is the hydraulic diameter of the pipe.

Viscous Friction Force in Turbulent Flows

In the turbulent regime—that is, when the Reynolds number is greater than the **Turbulent flow lower Reynolds number limit** value specified in the block dialog box—the viscous friction force in the half pipe adjacent to port A is

$$F_{\text{visc},A}^{\text{turbulent}} = \frac{\dot{m}_A |\dot{m}_A| f_A L_{\text{eff}} \nu_I}{4D_H S^2},$$

while in the half pipe adjacent to port B it is

$$F_{\text{visc},B}^{\text{turbulent}} = \frac{\dot{m}_B |\dot{m}_B| f_B L_{\text{eff}} \nu_I}{4D_H S^2},$$

where:

- f_A is the Darcy friction factor for turbulent flow in the half pipe adjacent to port A.
- f_B is the Darcy friction factor for turbulent flow in the half pipe adjacent to port B.

The Darcy friction factor for turbulent flow in the half pipe adjacent to port A follows from the Haaland equation as

$$f_A = \frac{1}{\left\{ -1.8 \log_{10} \left[\frac{6.9}{Re_A} + \left(\frac{\epsilon_r}{3.7} \right)^{1.11} \right] \right\}^2},$$

and in the half pipe adjacent to port B as

$$f_B = \frac{1}{\left\{ -1.8 \log_{10} \left[\frac{6.9}{Re_B} + \left(\frac{\epsilon_r}{3.7} \right)^{1.11} \right] \right\}^2},$$

where:

- ϵ_r is the relative roughness of the pipe.
- Re_A is the Reynolds number in the half pipe adjacent to port A,

$$Re_A = \frac{|\dot{m}_A| D_h v_I}{S \nu_I}.$$

- Re_B is the Reynolds number in the half pipe adjacent to port B,

$$Re_B = \frac{|\dot{m}_B| D_h v_I}{S \nu_I}.$$

A cubic polynomial function is used to blend the friction losses in the transition region between laminar flow and turbulent flow.

Assumptions and Limitations

- The pipe wall is rigid.
- The flow is fully developed.
- The effect of gravity is negligible.
- Heat transfer is calculated with respect to the temperature of the fluid volume in the pipe. To model temperature gradient due to heat transfer along a long pipe, connect multiple Pipe (2P) blocks in series.

Parameters

Geometry

Pipe length

Distance between the pipe inlet and outlet. The default value is 5 m.

Cross-sectional area

Internal pipe area normal to the direction of flow. This area is constant along the length of the pipe. The default value is 0.01 m^2 .

Hydraulic diameter

Diameter of an equivalent pipe with a circular cross section. In a cylindrical pipe, the hydraulic diameter is the same as its actual diameter. The default value is 0.1 m .

Friction and Heat Transfer

Aggregate equivalent length of local resistances

Pressure loss due to local resistances such as bends, inlets, and fittings, expressed as the equivalent length of these resistances. The default value is 0.1 m .

Internal surface absolute roughness

Average height of all surface defects on the internal surface of the pipe. This parameter enables the calculation of the friction factor in the turbulent flow regime. The default value is $1.5e-5$ m.

Laminar flow upper Reynolds number limit

Largest value of the Reynolds number corresponding to fully developed laminar flow. As the Reynolds number rises above this limit, the flow gradually transitions from laminar to turbulent. The default value is 2000.

Turbulent flow lower Reynolds number limit

Smallest value of the Reynolds number corresponding to fully developed turbulent flow. As the Reynolds number falls below this limit, flow gradually transitions from turbulent to laminar. The default value is 4000.

Shape factor for laminar flow viscous friction

Semi-empirical parameter encoding the effect of pipe geometry on the viscous friction losses incurred in the laminar regime. The appropriate value to use depends on the cross-sectional shape of the pipe.

Typical values include 56 for a square cross section, 62 for a rectangular cross section, and 96 for a concentric annulus cross section [1 on page 1-381]. The default value, corresponding to a circular cross section, is 64.

Nusselt number for laminar flow heat transfer

Proportionality constant between convective and conductive heat transfer in the laminar regime. This parameter enables the calculation of convective heat transfer in laminar flows. Its value changes with the pipe cross-sectional area and thermal boundary conditions, e.g., constant temperature or constant heat flux at the pipe wall. The default value, corresponding to a circular pipe cross section, is 3.66.

Effects and Initial Conditions**Fluid inertia**

Option to model fluid inertia, the resistance of the fluid to rapid acceleration. The default is Off.

Initial fluid energy specification

Thermodynamic variable in terms of which to define the initial conditions of the component. The default setting is Temperature.

Initial pressure

Pressure in the chamber at the start of simulation, specified against absolute zero. The default value is 0.101325 MPa.

Initial temperature

Temperature in the chamber at the start of simulation, specified against absolute zero. This parameter is active when the **Initial fluid energy specification** option is set to Temperature. The default value is 293.15 K.

Initial vapor quality

Mass fraction of vapor in the chamber at the start of simulation. This parameter is active when the **Initial fluid energy specification** option is set to Vapor quality. The default value is 0.5.

Initial vapor void fraction

Volume fraction of vapor in the chamber at the start of simulation. This parameter is active when the **Initial fluid energy specification** option is set to Vapor void fraction. The default value is 0.5.

Initial specific enthalpy

Specific enthalpy of the fluid in the chamber at the start of simulation. This parameter is active when the **Initial fluid energy specification** option is set to **Specific enthalpy**. The default value is 1500 kJ/kg.

Initial specific internal energy

Specific internal energy of the fluid in the chamber at the start of simulation. This parameter is active when the **Initial fluid energy specification** option is set to **Specific internal energy**. The default value is 1500 kJ/kg.

Phase change time constant

Characteristic duration of a phase-change event. This constant introduces a time lag into the transition between phases. The default value is 0.1 s.

Ports

The block has two two-phase fluid conserving ports, A and B. Port H is a thermal conserving port representing the pipe wall through which heat exchange occurs.

References

[1] White, F.M., *Viscous Fluid Flow*, McGraw-Hill, 1991

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

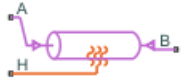
Local Restriction (2P) | Variable Local Restriction (2P)

Introduced in R2015b

Pipe (G)

Rigid conduit for gas flow

Library: Simscape / Foundation Library / Gas / Elements



Description

The Pipe (G) block models pipe flow dynamics in a gas network. The block accounts for viscous friction losses and convective heat transfer with the pipe wall. The pipe contains a constant volume of gas. The pressure and temperature evolve based on the compressibility and thermal capacity of this gas volume. Choking occurs when the outlet reaches the sonic condition.

Caution Gas flow through this block can choke. If a Mass Flow Rate Source (G) block or a Controlled Mass Flow Rate Source (G) block connected to the Pipe (G) block specifies a greater mass flow rate than the possible choked mass flow rate, you get a simulation error. For more information, see “Choked Flow”.

Mass Balance

Mass conservation relates the mass flow rates to the dynamics of the pressure and temperature of the internal node representing the gas volume:

$$\frac{\partial M}{\partial p} \cdot \frac{dp_I}{dt} + \frac{\partial M}{\partial T} \cdot \frac{dT_I}{dt} = \dot{m}_A + \dot{m}_B$$

where:

- $\frac{\partial M}{\partial p}$ is the partial derivative of the mass of the gas volume with respect to pressure at constant temperature and volume.
- $\frac{\partial M}{\partial T}$ is the partial derivative of the mass of the gas volume with respect to temperature at constant pressure and volume.
- p_I is the pressure of the gas volume.
- T_I is the temperature of the gas volume.
- t is time.
- \dot{m}_A and \dot{m}_B are mass flow rates at ports A and B, respectively. Flow rate associated with a port is positive when it flows into the block.

Energy Balance

Energy conservation relates the energy and heat flow rates to the dynamics of the pressure and temperature of the internal node representing the gas volume:

$$\frac{\partial U}{\partial p} \cdot \frac{dp_I}{dt} + \frac{\partial U}{\partial T} \cdot \frac{dT_I}{dt} = \Phi_A + \Phi_B + Q_H$$

where:

- $\frac{\partial U}{\partial p}$ is the partial derivative of the internal energy of the gas volume with respect to pressure at constant temperature and volume.
- $\frac{\partial U}{\partial T}$ is the partial derivative of the internal energy of the gas volume with respect to temperature at constant pressure and volume.
- Φ_A and Φ_B are energy flow rates at ports **A** and **B**, respectively.
- Q_H is heat flow rate at port **H**.

Partial Derivatives for Perfect and Semiperfect Gas Models

The partial derivatives of the mass M and the internal energy U of the gas volume, with respect to pressure and temperature at constant volume, depend on the gas property model. For perfect and semiperfect gas models, the equations are:

$$\frac{\partial M}{\partial p} = V \frac{\rho_I}{p_I}$$

$$\frac{\partial M}{\partial T} = -V \frac{\rho_I}{T_I}$$

$$\frac{\partial U}{\partial p} = V \left(\frac{h_I}{ZRT_I} - 1 \right)$$

$$\frac{\partial U}{\partial T} = V \rho_I \left(c_{pI} - \frac{h_I}{T_I} \right)$$

where:

- ρ_I is the density of the gas volume.
- V is the volume of gas.
- h_I is the specific enthalpy of the gas volume.
- Z is the compressibility factor.
- R is the specific gas constant.
- c_{pI} is the specific heat at constant pressure of the gas volume.

Partial Derivatives for Real Gas Model

For real gas model, the partial derivatives of the mass M and the internal energy U of the gas volume, with respect to pressure and temperature at constant volume, are:

$$\frac{\partial M}{\partial p} = V \frac{\rho_I}{\beta_I}$$

$$\frac{\partial M}{\partial T} = -V \rho_I \alpha_I$$

$$\frac{\partial U}{\partial p} = V \left(\frac{\rho_I h_I}{\beta_I} - T_I \alpha_I \right)$$

$$\frac{\partial U}{\partial T} = V \rho_I (c_{pI} - h_I \alpha_I)$$

where:

- β is the isothermal bulk modulus of the gas volume.
- α is the isobaric thermal expansion coefficient of the gas volume.

Momentum Balance

The momentum balance for each half of the pipe models the pressure drop due to momentum flux and viscous friction:

$$p_A - p_I = \left(\frac{\dot{m}_A}{S} \right)^2 \cdot \left(\frac{1}{\rho_I} - \frac{1}{\rho_A} \right) + \Delta p_{AI}$$

$$p_B - p_I = \left(\frac{\dot{m}_B}{S} \right)^2 \cdot \left(\frac{1}{\rho_I} - \frac{1}{\rho_B} \right) + \Delta p_{BI}$$

where:

- p is the gas pressure at port **A**, port **B**, or internal node **I**, as indicated by the subscript.
- ρ is the density at port **A**, port **B**, or internal node **I**, as indicated by the subscript.
- S is the cross-sectional area of the pipe.
- Δp_{AI} and Δp_{BI} are pressure losses due to viscous friction.

The heat exchanged with the pipe wall through port **H** is added to the energy of gas volume represented by the internal node via the energy conservation equation (see "Energy Balance" on page 1-382). Therefore, the momentum balances for each half of the pipe, between port **A** and the internal node and between port **B** and the internal node, are assumed to be adiabatic processes. The adiabatic relations are:

$$h_A + \frac{1}{2} \left(\frac{\dot{m}_A}{\rho_A S} \right)^2 = h_I + \frac{1}{2} \left(\frac{\dot{m}_A}{\rho_I S} \right)^2$$

$$h_B + \frac{1}{2} \left(\frac{\dot{m}_B}{\rho_B S} \right)^2 = h_I + \frac{1}{2} \left(\frac{\dot{m}_B}{\rho_I S} \right)^2$$

where h is the specific enthalpy at port **A**, port **B**, or internal node **I**, as indicated by the subscript.

The pressure losses due to viscous friction, Δp_{AI} and Δp_{BI} , depend on the flow regime. The Reynolds numbers for each half of the pipe are defined as:

$$\text{Re}_A = \frac{|\dot{m}_A| \cdot D_h}{S \cdot \mu_I}$$

$$\text{Re}_B = \frac{|\dot{m}_B| \cdot D_h}{S \cdot \mu_I}$$

where:

- D_h is the hydraulic diameter of the pipe.
- μ_I is the dynamic viscosity at internal node.

If the Reynolds number is less than the **Laminar flow upper Reynolds number limit** parameter value, then the flow is in the laminar flow regime. If the Reynolds number is greater than the **Turbulent flow lower Reynolds number limit** parameter value, then the flow is in the turbulent flow regime.

In the laminar flow regime, the pressure losses due to viscous friction are:

$$\Delta p_{AI_{lam}} = f_{shape} \frac{\dot{m}_A \cdot \mu_I}{2\rho_I \cdot D_h^2 \cdot S} \cdot \frac{L + L_{eqv}}{2}$$

$$\Delta p_{BI_{lam}} = f_{shape} \frac{\dot{m}_B \cdot \mu_I}{2\rho_I \cdot D_h^2 \cdot S} \cdot \frac{L + L_{eqv}}{2}$$

where:

- f_{shape} is the **Shape factor for laminar flow viscous friction** parameter value.
- L_{eqv} is the **Aggregate equivalent length of local resistances** parameter value.

In the turbulent flow regime, the pressure losses due to viscous friction are:

$$\Delta p_{AI_{tur}} = f_{DarcyA} \frac{\dot{m}_A \cdot |\dot{m}_A|}{2\rho_I \cdot D_h \cdot S^2} \cdot \frac{L + L_{eqv}}{2}$$

$$\Delta p_{BI_{tur}} = f_{DarcyB} \frac{\dot{m}_B \cdot |\dot{m}_B|}{2\rho_I \cdot D_h \cdot S^2} \cdot \frac{L + L_{eqv}}{2}$$

where f_{Darcy} is the Darcy friction factor at port **A** or **B**, as indicated by the subscript.

The Darcy friction factors are computed from the Haaland correlation:

$$f_{DarcyA} = \left[-1.8 \log \left(\frac{6.9}{\text{Re}_A} + \left(\frac{\varepsilon_{rough}}{3.7D_h} \right)^{1.11} \right) \right]^{-2}$$

$$f_{DarcyB} = \left[-1.8 \log \left(\frac{6.9}{\text{Re}_B} + \left(\frac{\varepsilon_{rough}}{3.7D_h} \right)^{1.11} \right) \right]^{-2}$$

where ε_{rough} is the **Internal surface absolute roughness** parameter value.

When the Reynolds number is between the **Laminar flow upper Reynolds number limit** and the **Turbulent flow lower Reynolds number limit** parameter values, the flow is in transition between laminar flow and turbulent flow. The pressure losses due to viscous friction during the transition region follow a smooth connection between those in the laminar flow regime and those in the turbulent flow regime.

Convective Heat Transfer

The convective heat transfer equation between the pipe wall and the internal gas volume is:

$$Q_H = Q_{conv} + \frac{k_I S_{surf}}{D_h} (T_H - T_I)$$

S_{surf} is the pipe surface area, $S_{surf} = 4SL/D_h$. Assuming an exponential temperature distribution along the pipe, the convective heat transfer is

$$Q_{conv} = |\dot{m}_{avg}| c_{p_{avg}} (T_H - T_{in}) \left(1 - e^{-\frac{h_{coeff} S_{surf}}{|\dot{m}_{avg}| c_{p_{avg}}}} \right)$$

where:

- T_{in} is the inlet temperature depending on flow direction.
- $\dot{m}_{avg} = (\dot{m}_A - \dot{m}_B)/2$ is the average mass flow rate from port **A** to port **B**.
- $c_{p_{avg}}$ is the specific heat evaluated at the average temperature.

The heat transfer coefficient, h_{coeff} , depends on the Nusselt number:

$$h_{coeff} = Nu \frac{k_{avg}}{D_h}$$

where k_{avg} is the thermal conductivity evaluated at the average temperature. The Nusselt number depends on the flow regime. The Nusselt number in the laminar flow regime is constant and equal to the value of the **Nusselt number for laminar flow heat transfer** parameter. The Nusselt number in the turbulent flow regime is computed from the Gnielinski correlation:

$$Nu_{tur} = \frac{\frac{f_{Darcy}}{8} (Re_{avg} - 1000) Pr_{avg}}{1 + 12.7 \sqrt{\frac{f_{Darcy}}{8} (Pr_{avg}^{2/3} - 1)}}$$

where Pr_{avg} is the Prandtl number evaluated at the average temperature. The average Reynolds number is

$$Re_{avg} = \frac{|\dot{m}_{avg}| D_h}{S \mu_{avg}}$$

where μ_{avg} is the dynamic viscosity evaluated at the average temperature. When the average Reynolds number is between the **Laminar flow upper Reynolds number limit** and the **Turbulent flow lower Reynolds number limit** parameter values, the Nusselt number follows a smooth transition between the laminar and turbulent Nusselt number values.

Choked Flow

The choked mass flow rates out of the pipe at ports **A** and **B** are:

$$\dot{m}_{A_{choked}} = \rho_A \cdot a_A \cdot S$$

$$\dot{m}_{B_{choked}} = \rho_B \cdot a_B \cdot S$$

where a_A and a_B is the speed of sound at ports **A** and **B**, respectively.

The unchoked pressure at port **A** or **B** is the value of the corresponding Across variable at that port:

$$p_{A_{unchoked}} = A.p$$

$$p_{B_{unchoked}} = B.p$$

The choked pressures at ports **A** and **B** are obtained by substituting the choked mass flow rates into the momentum balance equations for the pipe:

$$p_{A_{choked}} = p_I + \left(\frac{\dot{m}_{A_{choked}}}{S} \right)^2 \cdot \left(\frac{1}{\rho_I} - \frac{1}{\rho_A} \right) + \Delta p_{AI_{choked}}$$

$$p_{B_{choked}} = p_I + \left(\frac{\dot{m}_{B_{choked}}}{S} \right)^2 \cdot \left(\frac{1}{\rho_I} - \frac{1}{\rho_B} \right) + \Delta p_{BI_{choked}}$$

$\Delta p_{AI_{choked}}$ and $\Delta p_{BI_{choked}}$ are the pressure losses due to viscous friction, assuming that the choking has occurred. They are computed similar to Δp_{AI} and Δp_{BI} , with the mass flow rates at ports **A** and **B** replaced by the choked mass flow rate values.

Depending on whether choking has occurred, the block assigns either the choked or unchoked pressure value as the actual pressure at the port. Choking can occur at the pipe outlet, but not at the pipe inlet. Therefore, if $p_{A_{unchoked}} \geq p_I$, then port **A** is an inlet and $p_A = p_{A_{unchoked}}$. If $p_{A_{unchoked}} < p_I$, then port **A** is an outlet and

$$p_A = \begin{cases} p_{A_{unchoked}}, & \text{if } p_{A_{unchoked}} \geq p_{A_{choked}} \\ p_{A_{choked}}, & \text{if } p_{A_{unchoked}} < p_{A_{choked}} \end{cases}$$

Similarly, if $p_{B_{unchoked}} \geq p_I$, then port **B** is an inlet and $p_B = p_{B_{unchoked}}$. If $p_{B_{unchoked}} < p_I$, then port **B** is an outlet and

$$p_B = \begin{cases} p_{B_{unchoked}}, & \text{if } p_{B_{unchoked}} \geq p_{B_{choked}} \\ p_{B_{choked}}, & \text{if } p_{B_{unchoked}} < p_{B_{choked}} \end{cases}$$

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Gas Volume”.

Assumptions and Limitations

- The pipe wall is perfectly rigid.
- The flow is fully developed. Friction losses and heat transfer do not include entrance effects.
- The effect of gravity is negligible.
- Fluid inertia is negligible.
- This block does not model supersonic flow.

Ports

Conserving

A — Inlet or outlet

gas

Gas conserving port associated with the inlet or outlet of the pipe. This block has no intrinsic directionality.

B — Inlet or outlet

gas

Gas conserving port associated with the inlet or outlet of the pipe. This block has no intrinsic directionality.

H — Temperature of pipe wall

thermal

Thermal conserving port associated with the temperature of the pipe wall. This temperature may differ from the temperature of the gas volume.

Parameters

Geometry

Pipe length — The length of the pipe

5 m (default)

The length of the pipe along the direction of flow.

Cross-sectional area — The internal area of the pipe

0.01 m² (default)

The internal area of the pipe normal to the direction of the flow.

Hydraulic diameter — Diameter of an equivalent cylindrical pipe with the same cross-sectional area

0.1 m (default)

Diameter of an equivalent cylindrical pipe with the same cross-sectional area.

Friction and Heat Transfer

Aggregate equivalent length of local resistances — The combined length of all local resistances present in the pipe

0.1 m (default)

The combined length of all local resistances present in the pipe. Local resistances include bends, fittings, armatures, and pipe inlets and outlets. The effect of the local resistances is to increase the effective length of the pipe segment. This length is added to the geometrical pipe length only for friction calculations. The gas volume depends only on the pipe geometrical length, defined by the **Pipe length** parameter.

Internal surface absolute roughness — Average depth of all surface defects on the internal surface of the pipe

15e-6 m (default)

Average depth of all surface defects on the internal surface of the pipe, which affects the pressure loss in the turbulent flow regime.

Laminar flow upper Reynolds number limit — The Reynolds number above which flow begins to transition from laminar to turbulent

2000 (default)

The Reynolds number above which flow begins to transition from laminar to turbulent. This number equals the maximum Reynolds number corresponding to fully developed laminar flow.

Turbulent flow lower Reynolds number limit — The Reynolds number below which flow begins to transition from turbulent to laminar

4000 (default)

The Reynolds number below which flow begins to transition from turbulent to laminar. This number equals to the minimum Reynolds number corresponding to fully developed turbulent flow.

Shape factor for laminar flow viscous friction — Effect of pipe geometry on the viscous friction losses

64 (default)

Dimensionless factor that encodes the effect of pipe cross-sectional geometry on the viscous friction losses in the laminar flow regime. Typical values are 64 for a circular cross section, 57 for a square cross section, 62 for a rectangular cross section with an aspect ratio of 2, and 96 for a thin annular cross section [1].

Nusselt number for laminar flow heat transfer — Ratio of convective to conductive heat transfer

3.66 (default)

Ratio of convective to conductive heat transfer in the laminar flow regime. Its value depends on the pipe cross-sectional geometry and pipe wall thermal boundary conditions, such as constant temperature or constant heat flux. Typical value is 3.66, for a circular cross section with constant wall temperature [2].

References

[1] White, F. M., *Fluid Mechanics*. 7th Ed, Section 6.8. McGraw-Hill, 2011.

[2] Cengel, Y. A., *Heat and Mass Transfer - A Practical Approach*. 3rd Ed, Section 8.5. McGraw-Hill, 2007.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Local Restriction (G)

Topics

“Modeling Gas Systems”

Introduced in R2016b

Pipe (IL)

Rigid conduit for fluid flow in isothermal liquid systems

Library: Simscape / Foundation Library / Isothermal Liquid / Elements



Description

The Pipe (IL) block models pipe flow dynamics in an isothermal liquid network. The block accounts for viscous friction losses, and can also account for dynamic compressibility and fluid inertia.

The pipe contains a constant volume of liquid. The liquid experiences pressure losses due to viscous friction, following the Darcy-Weisbach equation.

Pipe Effects

The block lets you include dynamic compressibility and fluid inertia effects. Turning on each of these effects can improve model fidelity at the cost of increased equation complexity and potentially increased simulation cost:

- When dynamic compressibility is off, the liquid is assumed to spend negligible time in the pipe volume. Therefore, there is no accumulation of mass in the pipe, and mass inflow equals mass outflow. This is the simplest option. It is appropriate when the liquid mass in the pipe is a negligible fraction of the total liquid mass in the system.
- When dynamic compressibility is on, an imbalance of mass inflow and mass outflow can cause liquid to accumulate or diminish in the pipe. As a result, pressure in the pipe volume can rise and fall dynamically, which provides some compliance to the system and modulates rapid pressure changes. This is the default option.
- If dynamic compressibility is on, you can also turn on fluid inertia. This effect results in additional flow resistance, besides the resistance due to friction. This additional resistance is proportional to the rate of change of mass flow rate. Accounting for fluid inertia slows down rapid changes in flow rate but can also cause the flow rate to overshoot and oscillate. This option is appropriate in a very long pipe. Turn on fluid inertia and connect multiple pipe segments in series to model the propagation of pressure waves along the pipe, such as in the water hammer phenomenon.

Mass Balance

The mass conservation equation for the pipe is

$$\dot{m}_A + \dot{m}_B = \begin{cases} 0, & \text{if fluid dynamic compressibility is off} \\ \frac{\dot{p}_I}{\beta_I} \rho_I V, & \text{if fluid dynamic compressibility is on} \end{cases}$$

where:

- \dot{m}_A and \dot{m}_B are the mass flow rates through ports **A** and **B**.
- V is the pipe fluid volume.
- p_I is the pressure inside the pipe.
- ρ_I is the fluid density inside the pipe.
- β_I is the fluid bulk modulus inside the pipe.

The fluid can be a mixture of pure liquid and a small amount of entrained air, as specified by the Isothermal Liquid Properties (IL) block connected to the circuit. Equations used to compute ρ_I and β_I , as well as port densities ρ_A and ρ_B in the viscous friction pressure loss equations for each half pipe, depend on the selected isothermal liquid model. For detailed information, see “Isothermal Liquid Modeling Options”.

Momentum Balance

The table shows the momentum conservation equations for each half pipe.

For half pipe adjacent to port A	$p_A - p_I = \begin{cases} \Delta p_{v,A}, & \text{if fluid inertia is off} \\ \Delta p_{v,A} + \frac{L}{2S} \dot{m}_A, & \text{if fluid inertia is on} \end{cases}$
For half pipe adjacent to port B	$p_B - p_I = \begin{cases} \Delta p_{v,B}, & \text{if fluid inertia is off} \\ \Delta p_{v,B} + \frac{L}{2S} \dot{m}_B, & \text{if fluid inertia is on} \end{cases}$

In the equations:

- p , p_A , and p_B are the liquid pressures at port **A** and port **B**, respectively.
- $\Delta p_{v,A}$ and $\Delta p_{v,B}$ are the viscous friction pressure losses between the pipe volume center and ports **A** and **B**.
- L is the pipe length.
- S is the pipe cross-sectional area.

Viscous Friction Pressure Losses

The table shows the viscous friction pressure loss equations for each half pipe.

For half pipe adjacent to port A	$\Delta p_{v,A} = \begin{cases} \lambda \mu \left(\frac{L + L_{eq}}{2} \right) \frac{\dot{m}_A}{2\rho_I D_h^2 S}, & \text{if } Re_A < Re_{lam} \\ f_A \left(\frac{L + L_{eq}}{2} \right) \frac{\dot{m}_A \dot{m}_A }{2\rho_I D_h S^2}, & \text{if } Re_A > Re_{tur} \end{cases}$
For half pipe adjacent to port B	$\Delta p_{v,B} = \begin{cases} \lambda \mu \left(\frac{L + L_{eq}}{2} \right) \frac{\dot{m}_B}{2\rho_I D_h^2 S}, & \text{if } Re_B < Re_{lam} \\ f_B \left(\frac{L + L_{eq}}{2} \right) \frac{\dot{m}_B \dot{m}_B }{2\rho_I D_h S^2}, & \text{if } Re_B \geq Re_{tur} \end{cases}$

In the equations:

- λ is the pipe shape factor, used to calculate the Darcy friction factor in the laminar regime.

- μ is the dynamic viscosity of the liquid in the pipe.
- L_{eq} is the aggregate equivalent length of the local pipe resistances.
- D_h is the hydraulic diameter of the pipe.
- f_A and f_B are the Darcy friction factors in the pipe halves adjacent to ports **A** and **B**.
- Re_A and Re_B are the Reynolds numbers at ports **A** and **B**.
- Re_{lam} is the Reynolds number above which the flow transitions to turbulent.
- Re_{tur} is the Reynolds number below which the flow transitions to laminar.

When the Reynolds number is between Re_{lam} and Re_{tur} , the flow is in transition between laminar flow and turbulent flow. The pressure losses due to viscous friction during the transition region follow a smooth connection between those in the laminar flow regime and those in the turbulent flow regime.

The block computes the Reynolds numbers at ports **A** and **B** based on the mass flow rate through the appropriate port:

$$Re = \frac{|\dot{m}|D_h}{\mu S}$$

The Darcy friction factors follow from the Haaland approximation for the turbulent regime:

$$f = \frac{1}{\left(-1.8 \log_{10} \left(\frac{6.9}{Re} + \left(\frac{1}{3.7} \frac{r}{D_h} \right)^{1.11} \right)\right)^2}$$

where:

- f is the Darcy friction factor.
- r is the pipe surface roughness.

Assumptions and Limitations

- The pipe wall is rigid.
- The flow is fully developed.
- The effect of gravity is negligible.

Ports

Conserving

A – Inlet or outlet

isothermal liquid

Isothermal liquid conserving port associated with the inlet or outlet of the pipe. This block has no intrinsic directionality.

B – Inlet or outlet

isothermal liquid

Isothermal liquid conserving port associated with the inlet or outlet of the pipe. This block has no intrinsic directionality.

Parameters

Geometry

Pipe length — The length of the pipe

5 m (default) | positive scalar

The length of the pipe along the direction of flow.

Cross-sectional area — The internal area of the pipe

0.01 m² (default) | positive scalar

The internal area of the pipe normal to the direction of the flow.

Hydraulic diameter — Diameter of an equivalent cylindrical pipe with the same cross-sectional area

0.1128 m (default) | positive scalar

Diameter of an equivalent cylindrical pipe with the same cross-sectional area.

Friction

Aggregate equivalent length of local resistances — The combined length of all local resistances present in the pipe

1 m (default) | positive scalar

The combined length of all local resistances present in the pipe. Local resistances include bends, fittings, armatures, and pipe inlets and outlets. The effect of the local resistances is to increase the effective length of the pipe segment. This length is added to the geometrical pipe length only for friction calculations. The liquid volume inside the pipe depends only on the pipe geometrical length, defined by the **Pipe length** parameter.

Internal surface absolute roughness — Average depth of all surface defects on the internal surface of the pipe

15e-6 m (default) | positive scalar

Average depth of all surface defects on the internal surface of the pipe, which affects the pressure loss in the turbulent flow regime.

Laminar flow upper Reynolds number limit — The Reynolds number above which flow begins to transition from laminar to turbulent

2000 (default) | scalar greater than 1

The Reynolds number above which flow begins to transition from laminar to turbulent. This number equals the maximum Reynolds number corresponding to fully developed laminar flow.

Turbulent flow lower Reynolds number limit — The Reynolds number below which flow begins to transition from turbulent to laminar

4000 (default) | scalar greater than **Laminar flow upper Reynolds number limit** value

The Reynolds number below which flow begins to transition from turbulent to laminar. This number equals to the minimum Reynolds number corresponding to fully developed turbulent flow.

Laminar friction constant for Darcy friction factor – Effect of pipe geometry on the viscous friction losses

64 (default) | positive scalar

Dimensionless factor that encodes the effect of pipe cross-sectional geometry on the viscous friction losses in the laminar flow regime. Typical values are 64 for a circular cross section, 57 for a square cross section, 62 for a rectangular cross section with an aspect ratio of 2, and 96 for a thin annular cross section [1].

Effects and Initial Conditions

Fluid dynamic compressibility – Select whether to model fluid dynamic compressibility

On (default) | Off

Select whether to account for the dynamic compressibility of the liquid. Dynamic compressibility gives the liquid density a dependence on pressure, impacting the transient response of the system at small time scales.

Fluid inertia – Select whether to model fluid inertia

Off (default) | On

Select whether to account for the flow inertia of the liquid. Flow inertia gives the liquid a resistance to changes in mass flow rate.

Dependencies

Enabled when the **Fluid dynamic compressibility** parameter is set to On.

Initial liquid pressure – Liquid pressure at time zero

0.101325 MPa (default) | positive scalar

Liquid pressure in the pipe at the start of simulation.

Dependencies

Enabled when the **Fluid dynamic compressibility** parameter is set to On.

Initial mass flow rate from port A to port B – Mass flow rate at time zero

0 kg/s (default) | scalar

Mass flow rate from port **A** to port **B** at time zero.

Dependencies

Enabled when the **Fluid inertia** parameter is set to On.

References

[1] White, F. M., *Fluid Mechanics*. 7th Ed, Section 6.8. McGraw-Hill, 2011.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Laminar Leakage (IL) | Local Restriction (IL)

Topics

“Modeling Isothermal Liquid Systems”

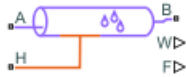
“Isothermal Liquid Modeling Options”

Introduced in R2020a

Pipe (MA)

Rigid conduit for moist air flow

Library: Simscape / Foundation Library / Moist Air / Elements



Description

The Pipe (MA) block models pipe flow dynamics in a moist air network due to viscous friction losses and convective heat transfer with the pipe wall. The pipe contains a constant volume of moist air. The pressure and temperature evolve based on the compressibility and thermal capacity of this moist air volume. Liquid water condenses out of the moist air volume when it reaches saturation. Choked flow occurs when the outlet reaches sonic condition.

Caution Air flow through this block can choke. If a Mass Flow Rate Source (MA) block or a Controlled Mass Flow Rate Source (MA) block connected to the Pipe (MA) block specifies a greater mass flow rate than the possible choked mass flow rate, the simulation generates an error. For more information, see “Choked Flow”.

The block equations use these symbols. Subscripts *a*, *w*, and *g* indicate the properties of dry air, water vapor, and trace gas, respectively. Subscript *ws* indicates water vapor at saturation. Subscripts *A*, *B*, *H*, and *S* indicate the appropriate port. Subscript *I* indicates the properties of the internal moist air volume.

\dot{m}	Mass flow rate
Φ	Energy flow rate
Q	Heat flow rate
p	Pressure
ρ	Density
R	Specific gas constant
V	Volume of moist air inside the pipe
c_v	Specific heat at constant volume
c_p	Specific heat at constant pressure
h	Specific enthalpy
u	Specific internal energy
x	Mass fraction (x_w is specific humidity, which is another term for water vapor mass fraction)
y	Mole fraction
φ	Relative humidity
r	Humidity ratio

T	Temperature
t	Time

Mass and Energy Balance

The net flow rates into the moist air volume inside the pipe are

$$\begin{aligned}\dot{m}_{net} &= \dot{m}_A + \dot{m}_B - \dot{m}_{condense} + \dot{m}_{wS} + \dot{m}_{gS} \\ \Phi_{net} &= \Phi_A + \Phi_B + Q_H - \Phi_{condense} + \Phi_S \\ \dot{m}_{w,net} &= \dot{m}_{wA} + \dot{m}_{wB} - \dot{m}_{condense} + \dot{m}_{wS} \\ \dot{m}_{g,net} &= \dot{m}_{gA} + \dot{m}_{gB} + \dot{m}_{gS}\end{aligned}$$

where:

- $\dot{m}_{condense}$ is the rate of condensation.
- $\Phi_{condense}$ is the rate of energy loss from the condensed water.
- Φ_S is the rate of energy added by the sources of moisture and trace gas. \dot{m}_{wS} and \dot{m}_{gS} are the mass flow rates of water and gas, respectively, through port **S**. The values of \dot{m}_{wS} , \dot{m}_{gS} , and Φ_S are determined by the moisture and trace gas sources connected to port **S** of the pipe.

Water vapor mass conservation relates the water vapor mass flow rate to the dynamics of the moisture level in the internal moist air volume:

$$\frac{dx_{wI}}{dt} \rho_I V + x_{wI} \dot{m}_{net} = \dot{m}_{w,net}$$

Similarly, trace gas mass conservation relates the trace gas mass flow rate to the dynamics of the trace gas level in the internal moist air volume:

$$\frac{dx_{gI}}{dt} \rho_I V + x_{gI} \dot{m}_{net} = \dot{m}_{g,net}$$

Mixture mass conservation relates the mixture mass flow rate to the dynamics of the pressure, temperature, and mass fractions of the internal moist air volume:

$$\left(\frac{1}{p_I} \frac{dp_I}{dt} - \frac{1}{T_I} \frac{dT_I}{dt} \right) \rho_I V + \frac{R_a - R_w}{R_I} (\dot{m}_{w,net} - x_w \dot{m}_{net}) + \frac{R_a - R_g}{R_I} (\dot{m}_{g,net} - x_g \dot{m}_{net}) = \dot{m}_{net}$$

Finally, energy conservation relates the energy flow rate to the dynamics of the pressure, temperature, and mass fractions of the internal moist air volume:

$$\rho_I c_{vI} V \frac{dT_I}{dt} + (u_{wI} - u_{aI}) (\dot{m}_{w,net} - x_w \dot{m}_{net}) + (u_{gI} - u_{aI}) (\dot{m}_{g,net} - x_g \dot{m}_{net}) + u_I \dot{m}_{net} = \Phi_{net}$$

The equation of state relates the mixture density to the pressure and temperature:

$$p_I = \rho_I R_I T_I$$

The mixture specific gas constant is

$$R_I = x_{aI} R_a + x_{wI} R_w + x_{gI} R_g$$

Momentum Balance

The momentum balance for each half of the pipe models the pressure drop due to momentum flux and viscous friction:

$$p_A - p_I = \left(\frac{\dot{m}_A}{S}\right)^2 \cdot \left(\frac{T_I}{\rho_I} - \frac{T_A}{\rho_A}\right) R_I + \Delta p_{AI}$$

$$p_B - p_I = \left(\frac{\dot{m}_B}{S}\right)^2 \cdot \left(\frac{T_I}{\rho_I} - \frac{T_B}{\rho_B}\right) R_I + \Delta p_{BI}$$

where:

- p is the pressure at port **A**, port **B**, or internal node I, as indicated by the subscript.
- ρ is the density at port **A**, port **B**, or internal node I, as indicated by the subscript.
- S is the cross-sectional area of the pipe.
- Δp_{AI} and Δp_{BI} are pressure losses due to viscous friction.

The pressure losses due to viscous friction, Δp_{AI} and Δp_{BI} , depend on the flow regime. The Reynolds numbers for each half of the pipe are defined as:

$$\text{Re}_A = \frac{|\dot{m}_A| \cdot D_h}{S \cdot \mu_I}$$

$$\text{Re}_B = \frac{|\dot{m}_B| \cdot D_h}{S \cdot \mu_I}$$

where:

- D_h is the hydraulic diameter of the pipe.
- μ_I is the dynamic viscosity at the internal node.

If the Reynolds number is less than the value of the **Laminar flow upper Reynolds number limit** parameter, then the flow is in the laminar flow regime. If the Reynolds number is greater than the value of the **Turbulent flow lower Reynolds number limit** parameter, then the flow is in the turbulent flow regime.

In the laminar flow regime, the pressure losses due to viscous friction are:

$$\Delta p_{AI_{lam}} = f_{shape} \frac{\dot{m}_A \cdot \mu_I}{2\rho_I \cdot D_h^2 \cdot S} \cdot \frac{L + L_{eqv}}{2}$$

$$\Delta p_{BI_{lam}} = f_{shape} \frac{\dot{m}_B \cdot \mu_I}{2\rho_I \cdot D_h^2 \cdot S} \cdot \frac{L + L_{eqv}}{2}$$

where:

- f_{shape} is the value of the **Shape factor for laminar flow viscous friction** parameter.
- L_{eqv} is the value of the **Aggregate equivalent length of local resistances** parameter.

In the turbulent flow regime, the pressure losses due to viscous friction are:

$$\Delta p_{AItur} = f_{DarcyA} \frac{\dot{m}_A \cdot |\dot{m}_A|}{2\rho_I \cdot D_h \cdot S^2} \cdot \frac{L + L_{eqv}}{2}$$

$$\Delta p_{BItur} = f_{DarcyB} \frac{\dot{m}_B \cdot |\dot{m}_B|}{2\rho_I \cdot D_h \cdot S^2} \cdot \frac{L + L_{eqv}}{2}$$

where f_{Darcy} is the Darcy friction factor at port **A** or **B**, as indicated by the subscript.

The Darcy friction factors are computed from the Haaland correlation:

$$f_{DarcyA} = \left[-1.8 \log \left(\frac{6.9}{Re_A} + \left(\frac{\varepsilon_{rough}}{3.7D_h} \right)^{1.11} \right) \right]^{-2}$$

$$f_{DarcyB} = \left[-1.8 \log \left(\frac{6.9}{Re_B} + \left(\frac{\varepsilon_{rough}}{3.7D_h} \right)^{1.11} \right) \right]^{-2}$$

where ε_{rough} is the value of the **Internal surface absolute roughness** parameter.

When the Reynolds number is between the **Laminar flow upper Reynolds number limit** and the **Turbulent flow lower Reynolds number limit** parameter values, the flow is in transition between laminar flow and turbulent flow. The pressure losses due to viscous friction during the transition region follow a smooth connection between those in the laminar flow regime and those in the turbulent flow regime.

The heat exchanged with the pipe wall through port **H** is added to the energy of the moist air volume represented by the internal node via the energy conservation equation (see "Mass and Energy Balance" on page 1-398). Therefore, the momentum balances for each half of the pipe, between port **A** and the internal node and between port **B** and the internal node, are assumed to be adiabatic processes. The adiabatic relations are:

$$h_A - h_I = \left(\frac{R_I \dot{m}_A}{S} \right)^2 \left[\left(\frac{T_I}{p_I} \right)^2 - \left(\frac{T_A}{p_A} \right)^2 \right]$$

$$h_B - h_I = \left(\frac{R_I \dot{m}_B}{S} \right)^2 \left[\left(\frac{T_I}{p_I} \right)^2 - \left(\frac{T_B}{p_B} \right)^2 \right]$$

where h is the specific enthalpy at port **A**, port **B**, or internal node I, as indicated by the subscript.

Convective Heat Transfer

The convective heat transfer equation between the pipe wall and the internal moist air volume is:

$$Q_H = Q_{conv} + \frac{k_I S_{surf}}{D_h} (T_H - T_I)$$

S_{surf} is the pipe surface area, $S_{surf} = 4SL/D_h$. Assuming an exponential temperature distribution along the pipe, the convective heat transfer is

$$Q_{conv} = |\dot{m}_{avg}| c_{pavg} (T_H - T_{in}) \left(1 - e^{-\frac{h_{coeff} S_{surf}}{|\dot{m}_{avg}| c_{pavg}}} \right)$$

where:

- T_{in} is the inlet temperature depending on flow direction.
- $\dot{m}_{avg} = (\dot{m}_A - \dot{m}_B)/2$ is the average mass flow rate from port **A** to port **B**.
- $c_{p_{avg}}$ is the specific heat evaluated at the average temperature.

The heat transfer coefficient, h_{coeff} , depends on the Nusselt number:

$$h_{coeff} = Nu \frac{k_{avg}}{D_h}$$

where k_{avg} is the thermal conductivity evaluated at the average temperature. The Nusselt number depends on the flow regime. The Nusselt number in the laminar flow regime is constant and equal to the value of the **Nusselt number for laminar flow heat transfer** parameter. The Nusselt number in the turbulent flow regime is computed from the Gnielinski correlation:

$$Nu_{tur} = \frac{\frac{f_{Darcy}}{8} (Re_{avg} - 1000) Pr_{avg}}{1 + 12.7 \sqrt{\frac{f_{Darcy}}{8}} (Pr_{avg}^{2/3} - 1)}$$

where Pr_{avg} is the Prandtl number evaluated at the average temperature. The average Reynolds number is

$$Re_{avg} = \frac{|\dot{m}_{avg}| D_h}{S \mu_{avg}}$$

where μ_{avg} is the dynamic viscosity evaluated at the average temperature. When the average Reynolds number is between the **Laminar flow upper Reynolds number limit** and the **Turbulent flow lower Reynolds number limit** parameter values, the Nusselt number follows a smooth transition between the laminar and turbulent Nusselt number values.

Saturation and Condensation

When the moist air volume reaches saturation, condensation may occur. The specific humidity at saturation is

$$x_{wsI} = \varphi_{ws} \frac{R_I}{R_w} \frac{p_{wsI}}{p_I}$$

where:

- φ_{ws} is the relative humidity at saturation (typically 1).
- p_{wsI} is the water vapor saturation pressure evaluated at T_I .

The rate of condensation is

$$\dot{m}_{condense} = \begin{cases} 0, & \text{if } x_{wI} \leq x_{wsI} \\ \frac{x_{wI} - x_{wsI}}{\tau_{condense}} \rho_I V, & \text{if } x_{wI} > x_{wsI} \end{cases}$$

where $\tau_{condense}$ is the value of the **Condensation time constant** parameter.

The condensed water is subtracted from the moist air volume, as shown in the conservation equations. The energy associated with the condensed water is

$$\Phi_{condense} = \dot{m}_{condense}(h_{wI} - \Delta h_{vapI})$$

where Δh_{vapI} is the specific enthalpy of vaporization evaluated at T_I .

Other moisture and trace gas quantities are related to each other as follows:

$$\phi_{wI} = \frac{y_{wI}P_I}{p_{wsI}}$$

$$y_{wI} = \frac{x_{wI}R_w}{R_I}$$

$$r_{wI} = \frac{x_{wI}}{1 - x_{wI}}$$

$$y_{gI} = \frac{x_{gI}R_g}{R_I}$$

$$x_{aI} + x_{wI} + x_{gI} = 1$$

Choked Flow

The unchoked pressure at port **A** or **B** is the value of the corresponding Across variable at that port:

$$p_{A_{unchoked}} = A \cdot p$$

$$p_{B_{unchoked}} = B \cdot p$$

However, the port pressure variables used in the momentum balance equations, p_A and p_B , do not necessarily coincide with the pressure across variables $A \cdot p$ and $B \cdot p$ because the pipe outlet may choke. Choked flow occurs when the downstream pressure is sufficiently low. At that point, the flow depends only on the conditions at the inlet. Therefore, when choked, the outlet pressure (p_A or p_B , whichever is the outlet) cannot decrease further even if the pressure downstream, represented by $A \cdot p$ or $B \cdot p$, continues to decrease.

Choking can occur at the pipe outlet, but not at the pipe inlet. Therefore, if port **A** is the inlet, then $p_A = A \cdot p$. If port **A** is the outlet, then

$$p_A = \begin{cases} A \cdot p, & \text{if } A \cdot p \geq p_{A_{choked}} \\ p_{A_{choked}}, & \text{if } A \cdot p < p_{A_{choked}} \end{cases}$$

Similarly, if port **B** is the inlet, then $p_B = B \cdot p$. If port **B** is the outlet, then

$$p_B = \begin{cases} B \cdot p, & \text{if } B \cdot p \geq p_{B_{choked}} \\ p_{B_{choked}}, & \text{if } B \cdot p < p_{B_{choked}} \end{cases}$$

The choked pressures at ports **A** and **B** are derived from the momentum balance by assuming the outlet velocity is equal to the speed of sound:

$$p_{A_{choked}} - p_I = p_{A_{choked}} \left(\frac{p_{A_{choked}} T_I}{p_I T_A} - 1 \right) \frac{c_{pA}}{c_{vI}} + \Delta p_{AI}$$

$$p_{B_{choked}} - p_I = p_{B_{choked}} \left(\frac{p_{B_{choked}} T_I}{p_I T_B} - 1 \right) \frac{c_{pB}}{c_{vI}} + \Delta p_{BI}$$

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Moist Air Volume”.

Assumptions and Limitations

- The pipe wall is perfectly rigid.
- The flow is fully developed. Friction losses and heat transfer do not include entrance effects.
- The effect of gravity is negligible.
- Fluid inertia is negligible.
- This block does not model supersonic flow.

Ports

Output

W — Rate of condensation measurement, kg/s

physical signal

Physical signal output port that measures the rate of condensation in the pipe.

F — Vector physical signal containing pressure, temperature, humidity, and trace gas levels data

physical signal vector

Physical signal output port that outputs a vector signal. The vector contains the pressure (in Pa), temperature (in K), moisture level, and trace gas level measurements inside the component. Use the Measurement Selector (MA) block to unpack this vector signal.

Conserving

A — Inlet or outlet

moist air

Moist air conserving port associated with the inlet or outlet of the pipe. This block has no intrinsic directionality.

B — Inlet or outlet

moist air

Moist air conserving port associated with the inlet or outlet of the pipe. This block has no intrinsic directionality.

H — Temperature of pipe wall

thermal

Thermal conserving port associated with the temperature of the pipe wall. The block includes the convective heat transfer between the moist air mixture inside the pipe and the pipe wall.

S — Inject or extract moisture and trace gas

moist air source

Connect this port to port **S** of a block from the Moisture & Trace Gas Sources library to add or remove moisture and trace gas. For more information, see “Using Moisture and Trace Gas Sources”.

Dependencies

This port is visible only if you set the **Moisture and trace gas source** parameter to Controlled.

Parameters

Main

Pipe length — Length of the pipe

5 m (default)

Length of the pipe along the direction of flow.

Cross-sectional area — Internal area of the pipe

0.01 m² (default)

Internal area of the pipe normal to the direction of the flow.

Hydraulic diameter — Diameter of an equivalent cylindrical pipe with the same cross-sectional area

0.1 m (default)

Diameter of an equivalent cylindrical pipe with the same cross-sectional area.

Friction and Heat Transfer

Aggregate equivalent length of local resistances — Combined length of all local resistances present in the pipe

0.1 m (default)

Combined length of all local resistances present in the pipe. Local resistances include bends, fittings, armatures, and pipe inlets and outlets. The effect of the local resistances is to increase the effective length of the pipe segment. This length is added to the geometrical pipe length only for friction calculations. The moist air volume depends only on the pipe geometrical length, defined by the **Pipe length** parameter.

Internal surface absolute roughness — Average depth of all surface defects on the internal surface of the pipe

15e-6 m (default)

Average depth of all surface defects on the internal surface of the pipe, which affects the pressure loss in the turbulent flow regime.

Laminar flow upper Reynolds number limit — Reynolds number above which flow begins to transition from laminar to turbulent

2000 (default)

Reynolds number above which flow begins to transition from laminar to turbulent. This number equals the maximum Reynolds number corresponding to the fully developed laminar flow.

Turbulent flow lower Reynolds number limit — Reynolds number below which flow begins to transition from turbulent to laminar

4000 (default)

Reynolds number below which flow begins to transition from turbulent to laminar. This number equals to the minimum Reynolds number corresponding to the fully developed turbulent flow.

Shape factor for laminar flow viscous friction — Effect of pipe geometry on the viscous friction losses

64 (default)

Dimensionless factor that encodes the effect of pipe cross-sectional geometry on the viscous friction losses in the laminar flow regime. Typical values are 64 for a circular cross section, 57 for a square cross section, 62 for a rectangular cross section with an aspect ratio of 2, and 96 for a thin annular cross section [1].

Nusselt number for laminar flow heat transfer — Ratio of convective to conductive heat transfer

3.66 (default)

Ratio of convective to conductive heat transfer in the laminar flow regime. Its value depends on the pipe cross-sectional geometry and pipe wall thermal boundary conditions, such as constant temperature or constant heat flux. Typical value is 3.66, for a circular cross section with constant wall temperature [2].

Moisture and Trace Gas**Relative humidity at saturation — Relative humidity above which condensation occurs**

1 (default)

Relative humidity above which condensation occurs.

Condensation time constant — Time scale for condensation

1e-3 s (default)

Characteristic time scale at which an oversaturated moist air volume returns to saturation by condensing out excess moisture.

Moisture and trace gas source — Model moisture and trace gas levels

None (default) | Constant | Controlled

This parameter controls visibility of port **S** and provides these options for modeling moisture and trace gas levels inside the component:

- **None** — No moisture or trace gas is injected into or extracted from the block. Port **S** is hidden. This is the default.
- **Constant** — Moisture and trace gas are injected into or extracted from the block at a constant rate. The same parameters as in the Moisture Source (MA) and Trace Gas Source (MA) blocks become available in the **Moisture and Trace Gas** section of the block interface. Port **S** is hidden.
- **Controlled** — Moisture and trace gas are injected into or extracted from the block at a time-varying rate. Port **S** is exposed. Connect the Controlled Moisture Source (MA) and Controlled Trace Gas Source (MA) blocks to this port.

Moisture added or removed — Select whether the block adds or removes moisture as water vapor or liquid water

Vapor (default) | Liquid

Select whether the block adds or removes moisture as water vapor or liquid water:

- **Vapor** — The enthalpy of the added or removed moisture corresponds to the enthalpy of water vapor, which is greater than that of liquid water.
- **Liquid** — The enthalpy of the added or removed moisture corresponds to the enthalpy of liquid water, which is less than that of water vapor.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Rate of added moisture — Constant mass flow rate through the block

0 kg/s (default)

Water vapor mass flow rate through the block. A positive value adds moisture to the connected moist air volume. A negative value extracts moisture from that volume.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Added moisture temperature specification — Select specification method for the temperature of added moisture

Atmospheric temperature (default) | Specified temperature

Select a specification method for the moisture temperature:

- **Atmospheric temperature** — Use the atmospheric temperature, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified temperature** — Specify a value by using the **Temperature of added moisture** parameter.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Temperature of added moisture — Moisture temperature

293.15 K (default)

Enter the desired temperature of added moisture. This temperature remains constant during simulation. The block uses this value to evaluate the specific enthalpy of the added moisture only. The specific enthalpy of removed moisture is based on the temperature of the connected moist air volume.

Dependencies

Enabled when the **Added moisture temperature specification** parameter is set to Specified temperature.

Rate of added trace gas — Constant mass flow rate through the block

0 kg/s (default)

Trace gas mass flow rate through the block. A positive value adds trace gas to the connected moist air volume. A negative value extracts trace gas from that volume.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Added trace gas temperature specification — Select specification method for the temperature of added trace gas

Atmospheric temperature (default) | Specified temperature

Select a specification method for the trace gas temperature:

- **Atmospheric temperature** — Use the atmospheric temperature, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified temperature** — Specify a value by using the **Temperature of added trace gas** parameter.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Temperature of added trace gas — Trace gas temperature

293.15 K (default)

Enter the desired temperature of added trace gas. This temperature remains constant during simulation. The block uses this value to evaluate the specific enthalpy of the added trace gas only. The specific enthalpy of removed trace gas is based on the temperature of the connected moist air volume.

Dependencies

Enabled when the **Added trace gas temperature specification** parameter is set to Specified temperature.

References

[1] White, F. M., *Fluid Mechanics*. 7th Ed, Section 6.8. McGraw-Hill, 2011.

[2] Cengel, Y. A., *Heat and Mass Transfer - A Practical Approach*. 3rd Ed, Section 8.5. McGraw-Hill, 2007.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Local Restriction (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Pipe (TL)

Rigid conduit for fluid flow in thermal liquid systems

Library: Simscape / Foundation Library / Thermal Liquid / Elements



Description

The Pipe (TL) block represents a pipeline segment with a fixed volume of liquid. The liquid experiences pressure losses due to viscous friction and heat transfer due to convection between the fluid and the pipe wall. Viscous friction follows from the Darcy-Weisbach equation, while the heat exchange coefficient follows from Nusselt number correlations.

Pipe Effects

The block lets you include dynamic compressibility and fluid inertia effects. Turning on each of these effects can improve model fidelity at the cost of increased equation complexity and potentially increased simulation cost:

- When dynamic compressibility is off, the liquid is assumed to spend negligible time in the pipe volume. Therefore, there is no accumulation of mass in the pipe, and mass inflow equals mass outflow. This is the simplest option. It is appropriate when the liquid mass in the pipe is a negligible fraction of the total liquid mass in the system.
- When dynamic compressibility is on, an imbalance of mass inflow and mass outflow can cause liquid to accumulate or diminish in the pipe. As a result, pressure in the pipe volume can rise and fall dynamically, which provides some compliance to the system and modulates rapid pressure changes. This is the default option.
- If dynamic compressibility is on, you can also turn on fluid inertia. This effect results in additional flow resistance, besides the resistance due to friction. This additional resistance is proportional to the rate of change of mass flow rate. Accounting for fluid inertia slows down rapid changes in flow rate but can also cause the flow rate to overshoot and oscillate. This option is appropriate in a very long pipe. Turn on fluid inertia and connect multiple pipe segments in series to model the propagation of pressure waves along the pipe, such as in the water hammer phenomenon.

Mass Balance

The mass conservation equation for the pipe is

$$\dot{m}_A + \dot{m}_B = \begin{cases} 0, & \text{if fluid dynamic compressibility is 'off'} \\ V\rho\left(\frac{1}{\beta}\frac{dp}{dt} + \alpha\frac{dT}{dt}\right), & \text{if fluid dynamic compressibility is 'on'} \end{cases}$$

where:

- \dot{m}_A and \dot{m}_B are the mass flow rates through ports **A** and **B**.
- V is the pipe fluid volume.
- ρ is the thermal liquid density in the pipe.

- β is the isothermal bulk modulus in the pipe.
- α is the isobaric thermal expansion coefficient in the pipe.
- p is the thermal liquid pressure in the pipe.
- T is the thermal liquid temperature in the pipe.

Momentum Balance

The table shows the momentum conservation equations for each half pipe.

For half pipe adjacent to port A	$p_A - p = \begin{cases} \Delta p_{v,A}, & \text{if fluid inertia is off} \\ \Delta p_{v,A} + \frac{L}{2S} \ddot{m}_A, & \text{if fluid inertia is on} \end{cases}$
For half pipe adjacent to port B	$p_B - p = \begin{cases} \Delta p_{v,B}, & \text{if fluid inertia is off} \\ \Delta p_{v,B} + \frac{L}{2S} \ddot{m}_B, & \text{if fluid inertia is on} \end{cases}$

In the equations:

- S is the pipe cross-sectional area.
- p , p_A , and p_B are the liquid pressures in the pipe, at port **A** and port **B**.
- $\Delta p_{v,A}$ and $\Delta p_{v,B}$ are the viscous friction pressure losses between the pipe volume center and ports **A** and **B**.

Viscous Friction Pressure Losses

The table shows the viscous friction pressure loss equations for each half pipe.

For half pipe adjacent to port A	$\Delta p_{v,A} = \begin{cases} \lambda \nu \left(\frac{L + L_{eq}}{2} \right) \frac{\dot{m}_A}{2D^2 S}, & \text{if } Re_A < Re_t \\ f_A \left(\frac{L + L_{eq}}{2} \right) \frac{\dot{m}_A \dot{m}_A }{2\rho D S^2}, & \text{if } Re_A \geq Re_t \end{cases}$
For half pipe adjacent to port B	$\Delta p_{v,B} = \begin{cases} \lambda \nu \left(\frac{L + L_{eq}}{2} \right) \frac{\dot{m}_B}{2D^2 S}, & \text{if } Re_B < Re_t \\ f_B \left(\frac{L + L_{eq}}{2} \right) \frac{\dot{m}_B \dot{m}_B }{2\rho D S^2}, & \text{if } Re_B \geq Re_t \end{cases}$

In the equations:

- λ is the pipe shape factor.
- ν is the kinematic viscosity of the thermal liquid in the pipe.
- L_{eq} is the aggregate equivalent length of the local pipe resistances.
- D is the hydraulic diameter of the pipe.
- f_A and f_B are the Darcy friction factors in the pipe halves adjacent to ports **A** and **B**.
- Re_A and Re_B are the Reynolds numbers at ports **A** and **B**.
- Re_t is the Reynolds number above which the flow transitions to turbulent.

- Re_t is the Reynolds number below which the flow transitions to laminar.

The Darcy friction factors follow from the Haaland approximation for the turbulent regime:

$$f = \frac{1}{\left[-1.8 \log_{10} \left(\frac{6.9}{Re} + \left(\frac{1}{3.7D} \right)^{1.11} \right) \right]^2},$$

where:

- f is the Darcy friction factor.
- r is the pipe surface roughness.

Energy Balance

The energy conservation equation for the pipe is

$$V \frac{d(\rho u)}{dt} = \Phi_A + \Phi_B + Q_H,$$

where:

- Φ_A and Φ_B are the total energy flow rates into the pipe through ports **A** and **B**.
- Q_H is the heat flow rate into the pipe through the pipe wall.

Wall Heat Flow Rate

The heat flow rate between the thermal liquid and the pipe wall is:

$$Q_H = Q_{conv} + \frac{kS_H}{D}(T_H - T),$$

where:

- Q_H is the net heat flow rate.
- Q_{conv} is the portion of the heat flow rate attributed to convection at nonzero flow rates.
- k is the thermal conductivity of the thermal liquid in the pipe.
- S_H is the surface area of the pipe wall, the product of the pipe perimeter and length.
- T_H is the temperature at the pipe wall.

Assuming an exponential temperature distribution along the pipe, the convective heat transfer is

$$Q_{conv} = |\dot{m}_{avg}| c_{p,avg} (T_H - T_{in}) \left(1 - \exp \left(- \frac{hA_H}{|\dot{m}_{avg}| c_{p,avg}} \right) \right),$$

where:

- $\dot{m}_{avg} = (\dot{m}_A - \dot{m}_B)/2$ is the average mass flow rate from port **A** to port **B**.
- $c_{p,avg}$ is the specific heat evaluated at the average temperature.
- T_{in} is the inlet temperature depending on flow direction.

The heat transfer coefficient, h_{coeff} , depends on the Nusselt number:

$$h_{coeff} = Nu \frac{k_{avg}}{D},$$

where k_{avg} is the thermal conductivity evaluated at the average temperature. The Nusselt number depends on the flow regime. The Nusselt number in the laminar flow regime is constant and equal to the **Nusselt number for laminar flow heat transfer** parameter value. The Nusselt number in the turbulent flow regime is computed from the Gnielinski correlation:

$$Nu_{tur} = \frac{\frac{f_{avg}}{8} (Re_{avg} - 1000) Pr_{avg}}{1 + 12.7 \sqrt{\frac{f_{avg}}{8}} (Pr_{avg}^{2/3} - 1)},$$

where f_{avg} is the Darcy friction factor at the average Reynolds number, Re_{avg} , and Pr_{avg} is the Prandtl number evaluated at the average temperature. The average Reynolds number is computed as:

$$Re_{avg} = \frac{|\dot{m}_{avg}| D}{S \mu_{avg}},$$

where μ_{avg} is the dynamic viscosity evaluated at the average temperature. When the average Reynolds number is between the **Laminar flow upper Reynolds number limit** and the **Turbulent flow lower Reynolds number limit** parameter values, the Nusselt number follows a smooth transition between the laminar and turbulent Nusselt number values.

Assumptions and Limitations

- The pipe wall is rigid.
- The flow is fully developed.
- The effect of gravity is negligible.

Ports

Conserving

A — Inlet or outlet

thermal liquid

Thermal liquid conserving port associated with the inlet or outlet of the pipe. This block has no intrinsic directionality.

B — Inlet or outlet

thermal liquid

Thermal liquid conserving port associated with the inlet or outlet of the pipe. This block has no intrinsic directionality.

H — Temperature of pipe wall

thermal

Thermal conserving port associated with the temperature of the pipe wall. This temperature may differ from the temperature of the thermal liquid inside the pipe.

Parameters

Geometry

Pipe length — The length of the pipe

5 m (default)

The length of the pipe along the direction of flow.

Cross-sectional area — The internal area of the pipe

0.01 m² (default)

The internal area of the pipe normal to the direction of the flow.

Hydraulic diameter — Diameter of an equivalent cylindrical pipe with the same cross-sectional area

0.1128 m (default)

Diameter of an equivalent cylindrical pipe with the same cross-sectional area.

Friction and Heat Transfer

Aggregate equivalent length of local resistances — The combined length of all local resistances present in the pipe

1 m (default)

The combined length of all local resistances present in the pipe. Local resistances include bends, fittings, armatures, and pipe inlets and outlets. The effect of the local resistances is to increase the effective length of the pipe segment. This length is added to the geometrical pipe length only for friction calculations. The liquid volume inside the pipe depends only on the pipe geometrical length, defined by the **Pipe length** parameter.

Internal surface absolute roughness — Average depth of all surface defects on the internal surface of the pipe

15e-6 m (default)

Average depth of all surface defects on the internal surface of the pipe, which affects the pressure loss in the turbulent flow regime.

Laminar flow upper Reynolds number limit — The Reynolds number above which flow begins to transition from laminar to turbulent

2000 (default)

The Reynolds number above which flow begins to transition from laminar to turbulent. This number equals the maximum Reynolds number corresponding to fully developed laminar flow.

Turbulent flow lower Reynolds number limit — The Reynolds number below which flow begins to transition from turbulent to laminar

4000 (default)

The Reynolds number below which flow begins to transition from turbulent to laminar. This number equals to the minimum Reynolds number corresponding to fully developed turbulent flow.

Shape factor for laminar flow viscous friction – Effect of pipe geometry on the viscous friction losses

64 (default)

Dimensionless factor that encodes the effect of pipe cross-sectional geometry on the viscous friction losses in the laminar flow regime. Typical values are 64 for a circular cross section, 57 for a square cross section, 62 for a rectangular cross section with an aspect ratio of 2, and 96 for a thin annular cross section [1].

Nusselt number for laminar flow heat transfer – Ratio of convective to conductive heat transfer

3.66 (default)

Ratio of convective to conductive heat transfer in the laminar flow regime. Its value depends on the pipe cross-sectional geometry and pipe wall thermal boundary conditions, such as constant temperature or constant heat flux. Typical value is 3.66, for a circular cross section with constant wall temperature [2].

Effects and Initial Conditions**Fluid dynamic compressibility – Select whether to model fluid dynamic compressibility**

On (default) | Off

Select whether to account for the dynamic compressibility of the liquid. Dynamic compressibility gives the liquid density a dependence on pressure and temperature, impacting the transient response of the system at small time scales.

Fluid inertia – Select whether to model fluid inertia

Off (default) | On

Select whether to account for the flow inertia of the liquid. Flow inertia gives the liquid a resistance to changes in mass flow rate.

Dependencies

Enabled when the **Fluid dynamic compressibility** parameter is set to On.

Initial liquid pressure – Liquid pressure at time zero

0.101325 MPa (default)

Liquid pressure in the pipe at the start of simulation.

Dependencies

Enabled when the **Fluid dynamic compressibility** parameter is set to On.

Initial liquid temperature – Liquid temperature at time zero

293.15 K (default)

Liquid temperature in the pipe at the start of simulation.

Initial mass flow rate from port A to port B – Mass flow rate at time zero

0 kg/s (default)

Mass flow rate from port **A** to port **B** at time zero.

Dependencies

Enabled when the **Fluid inertia** parameter is set to On.

References

[1] White, F. M., *Fluid Mechanics*. 7th Ed, Section 6.8. McGraw-Hill, 2011.

[2] Cengel, Y. A., *Heat and Mass Transfer - A Practical Approach*. 3rd Ed, Section 8.5. McGraw-Hill, 2007.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Local Restriction (TL)

Topics

“Heat Transfer in Insulated Oil Pipeline”

“Modeling Thermal Liquid Systems”

Introduced in R2013b

Pneumatic Absolute Reference

Reference connection to zero absolute pressure and temperature for pneumatic ports



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Pneumatic Absolute Reference block provides a pneumatic reference port at zero absolute pressure and temperature. Use this block with the Pneumatic Pressure & Temperature Sensor block to create Physical Signals corresponding to absolute pressure and temperature.

Ports

The block has one pneumatic conserving port, which is at zero absolute pressure and temperature.

See Also

Pneumatic Atmospheric Reference | Pneumatic Pressure & Temperature Sensor

Topics

“Grounding Rules”

Introduced in R2009b

Pneumatic Atmospheric Reference

Reference connection to ambient pressure and temperature for pneumatic ports



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Pneumatic Atmospheric Reference block provides a pneumatic reference port with pressure and temperature values set to the ambient temperature and pressure. The Gas Properties block, if present, specifies the values for ambient temperature and pressure for all pneumatic blocks in the circuit. If a pneumatic circuit does not contain a Gas Properties block, ambient temperature and pressure are set to default values of 293.15 K and 101,325 Pa. Use the Pneumatic Atmospheric Reference block with the Pneumatic Pressure Source block to model an ideal pressure source that takes atmospheric air and increases the pressure by a constant amount.

Ports

The block has one pneumatic conserving port.

See Also

[Gas Properties](#) | [Pneumatic Absolute Reference](#) | [Pneumatic Pressure Source](#)

Topics

“Grounding Rules”

Introduced in R2009b

Pneumatic Flow Rate Source

Ideal compressor with constant mass flow rate



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Pneumatic Flow Rate Source block represents an ideal compressor that maintains a specified mass flow rate regardless of the pressure difference. Use this block when delivery of an actual device is practically independent of the source pressure, for example, in positive displacement compressors. The compressor adds no heat. Block connections A and B correspond to the pneumatic inlet and outlet ports, respectively.

The block positive direction is from port A to port B. This means that the flow rate is positive if it flows from A to B. The pressure differential is determined as $p = p_A - p_B$ and is negative if pressure at the source outlet is greater than pressure at its inlet. The power generated by the source is negative if the source adds energy to the flow.

Warning Be careful when driving an orifice directly from a flow rate source. The choked flow condition limits the flow that is possible through an orifice as a function of upstream pressure and temperature. Hence the flow rate value produced by the flow rate source must be compatible with upstream pressure and temperature. Specifying a flow rate that is too high will result in an unsolvable set of equations.

Parameters

Mass flow rate

Specify the mass flow rate of the source. The default value is 0.001 kg/s.

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the source inlet.

B

Pneumatic conserving port associated with the source outlet.

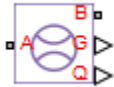
See Also

Controlled Pneumatic Flow Rate Source | Pneumatic Mass & Heat Flow Sensor

Introduced in R2009b

Pneumatic Mass & Heat Flow Sensor

Ideal mass flow and heat flow sensor



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Pneumatic Mass & Heat Flow Sensor block represents an ideal mass flow and heat flow sensor, that is, a device that converts mass flow rate and heat flow rate between the two pneumatic nodes into physical measurement signals G and Q, respectively.

The sensor positive direction is from port A to port B.

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the sensor inlet.

B

Pneumatic conserving port associated with the sensor outlet.

See Also

Controlled Pneumatic Flow Rate Source | PS-Simulink Converter | Pneumatic Flow Rate Source

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2009b

Pneumatic Piston Chamber

Translational pneumatic piston chamber based on ideal gas law



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Pneumatic Piston Chamber block models a pneumatic piston chamber based on the ideal gas law and assuming constant specific heats. Use this model as a building block for pneumatic translational actuators. The piston can exert force in one direction only, and the direction is set by the **Chamber orientation** parameter.

The continuity equation for the network representation of the piston chamber is

$$G = \frac{V_0 + A \cdot x}{RT} \left(\frac{dp}{dt} - \frac{p}{T} \frac{dT}{dt} \right) + \frac{A}{RT} \cdot p \cdot \frac{dx}{dt}$$

where

G	Mass flow rate at input port
V_0	Initial chamber volume
A	Piston effective area
x	Piston displacement
p	Absolute pressure in the chamber
R	Specific gas constant
T	Absolute gas temperature
t	Time

The energy equation is

$$q = \frac{c_v}{R} (V_0 + A \cdot x) \frac{dp}{dt} + \frac{c_p \cdot A}{R} p \frac{dx}{dt} - q_w$$

where

q	Heat flow due to gas inflow in the chamber (through the pneumatic port)
q_w	Heat flow through the chamber walls (through the thermal port)
c_v	Specific heat at constant volume
c_p	Specific heat at constant pressure

The force equation is

$$F = (p - p_a) \cdot A$$

where p_a is the atmospheric pressure acting on the outside of the piston.

Port A is the pneumatic conserving port associated with the chamber inlet. Port H is a thermal conserving port through which heat exchange with the environment takes place. Ports C and R are mechanical translational conserving ports associated with the piston case and rod, respectively. The gas flow and the heat flow are considered positive if they flow into the chamber.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Basic Assumptions and Limitations

- The gas is ideal.
- Specific heats at constant pressure and constant volume, c_p and c_v , are constant.

Parameters

Piston area

Specify the effective piston area. The default value is $.002 \text{ m}^2$.

Piston initial extension

Specify the initial offset of the piston from the cylinder cap. The default value is 0.

Dead volume

Specify the volume of gas in the chamber at zero piston position. The default value is $1\text{e-}5 \text{ m}^3$.

Chamber orientation

Specify the direction of force generation. The piston generates force in a positive direction if this parameter is set to 1 (the default). If you set this parameter to 2, the piston generates force in a negative direction.

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the chamber inlet.

H

Thermal conserving port through which heat exchange with the environment takes place.

R

Mechanical translational conserving port associated with the piston (rod).

C

Mechanical translational conserving port associated with the reference (case).

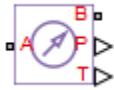
See Also

Constant Volume Pneumatic Chamber | Rotary Pneumatic Piston Chamber

Introduced in R2009b

Pneumatic Pressure & Temperature Sensor

Ideal pressure and temperature sensor



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Pneumatic Pressure & Temperature Sensor block represents an ideal pressure and temperature sensor, that is, a device that converts pressure differential and temperature differential measured between two pneumatic ports into physical measurement signals P and T, respectively.

The sensor positive direction is from port A to port B. This means that the sensor returns a positive pressure if the pressure at port A is greater than the pressure at port B. Similarly, the sensor returns a positive temperature if the temperature at port A is greater than the temperature at port B.

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the sensor inlet.

B

Pneumatic conserving port associated with the sensor outlet.

See Also

Controlled Pneumatic Pressure Source | PS-Simulink Converter | Pneumatic Pressure Source

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2009b

Pneumatic Pressure Source

Ideal compressor with constant pressure difference



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Pneumatic Pressure Source block represents an ideal compressor that maintains a specified pressure difference regardless of the flow rate. Use this block when pressure of an actual device is practically independent of the source flow rate, for example, in factory network outlets or large capacity receivers. The compressor adds no heat. Block connections A and B correspond to the pneumatic inlet and outlet ports, respectively.

A positive pressure difference results in the pressure at port B being higher than the pressure at port A.

Parameters

Pressure difference

Specify the pressure difference across the source. The default value is 0.

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the source inlet.

B

Pneumatic conserving port associated with the source outlet.

See Also

Controlled Pneumatic Pressure Source | Pneumatic Pressure & Temperature Sensor

Introduced in R2009b

Pneumatic Resistive Tube

Pneumatic pipe accounting for pressure loss and added heat due to flow resistance



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Pneumatic Resistive Tube block models the loss in pressure and heating due to viscous friction along a short stretch of pipe with circular cross section. Use this block with the Constant Volume Pneumatic Chamber block to build a model of a pneumatic transmission line.

The tube is simulated according to the following equations:

$$p_i - p_o = \begin{cases} \frac{RT_i}{p_i} \cdot \frac{32\mu L}{AD^2} \cdot G & \text{for } Re < Re_{lam} \text{ (laminar flow)} \\ f \cdot \frac{RT_i}{p_i} \cdot \frac{L}{D} \cdot \frac{G^2}{2A^2} & \text{for } Re > Re_{turb} \text{ (turbulent flow)} \end{cases}$$

where

p_i, p_o	Absolute pressures at the tube inlet and outlet, respectively. The inlet and outlet change depending on flow direction. For positive flow ($G > 0$), $p_i = p_A$, otherwise $p_i = p_B$.
T_i, T_o	Absolute gas temperatures at the tube inlet and outlet, respectively
G	Mass flow rate
μ	Gas viscosity
f	Friction factor for turbulent flow
D	Tube internal diameter
A	Tube cross-sectional area
L	Tube length

Re	Reynolds number
------	-----------------

The friction factor for turbulent flow is approximated by the Haaland function

$$f = \left(-1.8 \log_{10} \left(\frac{6.9}{Re} + \left(\frac{e}{3.7D} \right)^{1.11} \right) \right)^{-2}$$

where e is the surface roughness for the pipe material.

The Reynolds number is defined as:

$$Re = \rho v D / \mu$$

where ρ is the gas density and v is the gas velocity. Gas velocity is related to mass flow rate by

$$G = \rho v A$$

For flows between Re_{lam} and Re_{turb} , a linear blend is implemented between the flow predicted by the two equations.

In a real pipe, loss in kinetic energy due to friction is turned into added heat energy. However, the amount of heat is very small, and is neglected in the Pneumatic Resistive Tube block. Therefore, $q_i = q_o$, where q_i and q_o are the input and output heat flows, respectively.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see "Set Priority and Initial Target for Block Variables".

Basic Assumptions and Limitations

- The gas is ideal.
- The pipe has a circular cross section.
- The process is adiabatic, that is, there is no heat transfer with the environment.
- Gravitational effects can be neglected.
- The flow resistance adds no net heat to the flow.

Parameters

Tube internal diameter

Internal diameter of the tube. The default value is 0.01 m.

Tube length

Tube geometrical length. The default value is 10 m.

Aggregate equivalent length of local resistances

This parameter represents total equivalent length of all local resistances associated with the tube. You can account for the pressure loss caused by local resistances, such as bends, fittings, armature, inlet/outlet losses, and so on, by adding to the pipe geometrical length an aggregate equivalent length of all the local resistances. The default value is 0.

Internal surface roughness height

Roughness height on the tube internal surface. The parameter is typically provided in data sheets or manufacturer catalogs. The default value is 1.5×10^{-5} m, which corresponds to drawn tubing.

Reynolds number at laminar flow upper margin

Specifies the Reynolds number at which the laminar flow regime is assumed to start converting into turbulent flow. Mathematically, this value is the maximum Reynolds number at fully developed laminar flow. The default value is 2000.

Reynolds number at turbulent flow lower margin

Specifies the Reynolds number at which the turbulent flow regime is assumed to be fully developed. Mathematically, this value is the minimum Reynolds number at turbulent flow. The default value is 4000.

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the tube inlet for positive flow.

B

Pneumatic conserving port associated with the tube outlet for positive flow.

See Also

Constant Volume Pneumatic Chamber

Introduced in R2009b

Pressure & Internal Energy Sensor (2P)

Measure pressure and specific internal energy differences



Library

Two-Phase Fluid/Sensors

Description

The Pressure & Internal Energy Sensor (2P) block measures pressure and specific internal energy differences between two-phase fluid nodes. Ports A and B identify the nodes between which the measurement is taken.

The measured differences are positive if pressure and specific internal energy are greater at port A than at port B. Connect port B to an Absolute Reference (2P) block to measure absolute values instead—the differences with respect to absolute zero reference points.

Ports

The block has two two-phase fluid conserving ports, A and B. Physical signal port P outputs the pressure value. Physical signal port U outputs the specific internal energy value.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Absolute Reference (2P) | Controlled Pressure Source (2P) | Pressure Source (2P)

Introduced in R2015b

Pressure & Temperature Sensor (G)

Measure pressure and temperature differences

Library: Simscape / Foundation Library / Gas / Sensors



Description

The Pressure & Temperature Sensor (G) block represents an ideal sensor that measures pressure and temperature in a gas network. There is no mass or energy flow through the sensor.

The physical signal ports **P** and **T** report the pressure difference and the temperature difference, respectively, across the sensor. The measurements are positive when the values at port **A** are greater than the values at port **B**.

To measure the absolute pressure and absolute temperature at port **A**, connect port **B** to an Absolute Reference (G) block.

Temperature measurements are different from pressure measurements, because the energy flow rate depends on the flow direction. The temperature measurement represents the temperature upstream of the measured node. For example, if port **A** of the Pressure & Temperature Sensor (G) is connected to a node between pipe 1 and pipe 2 and the gas flows from pipe 1 to pipe 2, then the temperature measured at port **A** is the temperature of the gas volume in pipe 1. If the gas then switches direction and flows from pipe 2 to pipe 1, then the temperature measured at port **A** is the temperature of the gas volume in pipe 2. If the temperatures of the two gas volumes are different, then this will manifest as a change in the measured temperature when the flow direction switches. If there are two or more upstream flow paths merging at the node, then the temperature measurement at the node represents the weighted average based on the ideal mixing of the merging flow.

Ports

Output

P — Pressure measurement, Pa

physical signal

Physical signal output port for pressure difference measurement.

T — Temperature measurement, K

physical signal

Physical signal output port for temperature difference measurement.

Conserving

A — Sensor inlet

gas

Gas conserving port. The measurements are positive when the values at port **A** are greater than the values at port **B**.

B – Sensor outlet

gas

Gas conserving port. The measurements are positive when the values at port **A** are greater than the values at port **B**.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Absolute Reference (G) | Mass & Energy Flow Rate Sensor (G) | Thermodynamic Properties Sensor (G) | Volumetric Flow Rate Sensor (G)

Topics

“Modeling Gas Systems”

Introduced in R2016b

Pressure & Temperature Sensor (MA)

Measure pressure and temperature differences

Library: Simscape / Foundation Library / Moist Air / Sensors



Description

The Pressure & Temperature Sensor (MA) block represents an ideal sensor that measures pressure and temperature in a moist air network. There is no mass or energy flow through the sensor.

The physical signal ports **P** and **T** report the pressure difference and the temperature difference, respectively, across the sensor. The measurements are positive when the values at port **A** are greater than the values at port **B**.

To measure the absolute pressure and absolute temperature at port **A**, connect port **B** to an Absolute Reference (MA) block.

Temperature measurements are different from pressure measurements, because the energy flow rate depends on the flow direction. The temperature measurement represents the temperature upstream of the measured node.

For example, if port **A** of the Pressure & Temperature Sensor (MA) is connected to a node between pipe 1 and pipe 2 and the moist air flows from pipe 1 to pipe 2, then the temperature measured at port **A** is the temperature of the moist air volume in pipe 1. If the air flow then switches direction and flows from pipe 2 to pipe 1, then the temperature measured at port **A** is the temperature of the moist air volume in pipe 2.

If the temperatures of the two moist air volumes are different, then the measured temperature changes when the flow direction switches. If there are two or more upstream flow paths merging at the node, then the temperature measurement at the node represents the weighted average based on the ideal mixing of the merging flow.

Ports

Output

P — Pressure measurement, Pa

physical signal

Physical signal output port for pressure difference measurement.

T — Temperature measurement, K

physical signal

Physical signal output port for temperature difference measurement.

Conserving

A — Sensor inlet

moist air

Moist air conserving port. The measurements are positive when the values at port **A** are greater than the values at port **B**.

B — Sensor outlet

moist air

Moist air conserving port. The measurements are positive when the values at port **A** are greater than the values at port **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Absolute Reference (MA) | Mass & Energy Flow Rate Sensor (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Pressure & Temperature Sensor (TL)

Measure pressure and temperature differences

Library: Simscape / Foundation Library / Thermal Liquid / Sensors



Description

The Pressure & Temperature Sensor (TL) block represents an ideal sensor that measures pressure and temperature differences between two thermal liquid nodes.

Because pressure and temperature are Across variables, the sensor block must connect in parallel with the component being measured.

The physical signal ports **P** and **T** report the pressure difference and the temperature difference, respectively, across the sensor. The measurements are positive when the values at port **A** are greater than the values at port **B**. Connect the ports to PS-Simulink Converter blocks to transform the output physical signals into Simulink signals, for example, for plotting or additional data processing.

Ports

Output

P — Pressure measurement, Pa

physical signal

Physical signal output port for pressure difference measurement.

T — Temperature measurement, K

physical signal

Physical signal output port for temperature difference measurement.

Conserving

A — Sensor inlet

thermal liquid

Thermal liquid conserving port. The measurements are positive when the values at port **A** are greater than the values at port **B**.

B — Sensor outlet

thermal liquid

Thermal liquid conserving port. The measurements are positive when the values at port **A** are greater than the values at port **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mass & Energy Flow Rate Sensor (TL) | Thermodynamic Properties Sensor (TL) | Volumetric Flow Rate Sensor (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2013b

Pressure Sensor (IL)

Measure pressure differences, absolute pressure, or gauge pressure

Library: Simscape / Foundation Library / Isothermal Liquid / Sensors



Description

The Pressure Sensor (IL) block represents an ideal sensor that measures pressure in an isothermal liquid network. There is no mass flow through the sensor.

The **Pressure Measurements** parameter controls the measurement type and sensor configuration:

- **Pressure difference** — The physical signal port **P** reports the pressure difference across the sensor:

$$P = p_A - p_B,$$

where p_A and p_B are pressures at the sensor ports. The measurement is positive when the value at port **A** is greater than the value at port **B**.

- **Absolute pressure** — The physical signal port **Pa** reports the absolute pressure at port **A**:

$$Pa = p_A,$$



where p_A is pressure at port **A**, measured with respect to absolute zero. In this configuration, the sensor has only one isothermal liquid port.


- **Gauge pressure** — The physical signal port **Pg** reports the gauge pressure at port **A**:

$$Pg = p_A - p_{atm},$$

where p_A is pressure at port **A**, measured with respect to absolute zero, and p_{atm} is the atmospheric pressure specified by the Isothermal Liquid Properties (IL) block connected to the circuit. In this configuration, the sensor has only one isothermal liquid port.

The block icon changes depending on the value of the **Pressure Measurements** parameter.

Pressure Measurements	Block Icon
Pressure difference	
Absolute pressure	

Pressure Measurements	Block Icon
Gauge pressure	

Ports

Output

P – Pressure difference measurement, Pa

physical signal

Physical signal output port for the pressure difference measurement.

Dependencies

This port is visible if you set the **Pressure Measurements** parameter to Pressure difference.

Pa – Absolute pressure measurement, Pa

physical signal

Physical signal output port for the absolute pressure measurement.

Dependencies

This port is visible if you set the **Pressure Measurements** parameter to Absolute pressure.

Pg – Gauge pressure measurement, Pa

physical signal

Physical signal output port for the gauge pressure measurement.

Dependencies

This port is visible if you set the **Pressure Measurements** parameter to Gauge pressure.

Conserving

A – Sensor inlet

isothermal liquid

Isothermal liquid conserving port.

B – Sensor outlet

isothermal liquid

Isothermal liquid conserving port.

Dependencies

This port is visible if you set the **Pressure Measurements** parameter to Pressure difference.

Parameters

Pressure Measurements — Select whether the sensor measures pressure difference, absolute pressure, or gauge pressure

Pressure difference (default) | Absolute pressure | Gauge pressure

Select the type of measurements:

- **Pressure difference** — The sensors measures the pressure difference across the sensor. Selecting this option exposes the physical signal output port **P** and the second isothermal liquid port **B**. The measurement is positive when the value at port **A** is greater than the value at port **B**.
- **Absolute pressure** — The sensors measures the absolute pressure at port **A**. Selecting this option exposes the physical signal output port **Pa**.
- **Gauge pressure** — The sensors measures the gauge pressure at port **A**. Selecting this option exposes the physical signal output port **Pg**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Flow Rate Sensor (IL) | Liquid Properties Sensor (IL)

Topics

“Modeling Isothermal Liquid Systems”

“Isothermal Liquid Modeling Options”

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2020a

Pressure Source (2P)

Generate constant pressure differential



Library

Two-Phase Fluid/Sources

Description

The Pressure Source (2P) block generates a constant pressure differential in a two-phase fluid network branch. The source has two inlets, labeled **A** and **B**, with independently specified cross-sectional areas. By default, the source does isentropic work on the fluid, though the block provides the option to ignore this work.

The source is ideal. In other words, it maintains the specified pressure differential regardless of the mass flow rate produced through its ports. In addition, because the source is isentropic, there is no viscous friction between the ports and no heat exchange with the environment. Use this block to model an idealized pump or compressor or to set a boundary condition in a model.

Mass Balance

The volume of fluid in the source is considered negligible and is ignored in a model. There is no fluid accumulation between the ports and the sum of all mass flow rates into the source must therefore equal zero:

$$\dot{m}_A + \dot{m}_B = 0,$$

where \dot{m} denotes the mass flow rate into the source through a port. The block accepts as input the mass flow rate at port **A**. The flow is directed from port **A** to port **B** when the specified value is positive.

Energy Balance

By default, the source maintains the specified flow rate by performing isentropic work on the incoming fluid, though the block provides the option to ignore this term. The rate at which the source does work, if considered in the model, must equal the sum of the energy flow rates through the ports:

$$\phi_A + \phi_B + \phi_{\text{Work}} = 0,$$

where ϕ denotes the energy flow rate into the source through a port or by means of work. The energy flow rate due to work is equal to the power generated by the source. Its value is calculated from the specific total enthalpies at the ports:

$$\phi_{\text{Work}} = \dot{m}_A(h_A - h_B).$$

The specific total enthalpy h is defined as:

$$h_* = u_* + p_* v_* + \frac{1}{2} \left(\frac{\dot{m}_* v_*}{S} \right)^2,$$

where the asterisk denotes a port (**A** or **B**) and:

- u is specific internal energy.
- p is pressure.
- S is flow area.

The specific internal energy in the equation is obtained from the tabulated data of the Two-Phase Fluid Properties (2P) block. Its value is uniquely determined from the constraint that the work done by the source is isentropic. The specific entropy, a function of specific internal energy, must then have the same value at ports **A** and **B**:

$$s_A(p_A, u_A) = s_B(p_B, u_B),$$

where s is specific entropy. If the **Power added** parameter is set to **None**, the specific total enthalpies at the ports have the same value ($h_A = h_B$) and the work done by the source reduces to zero ($\phi_{\text{Work}} = 0$).

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

A – Inlet

two-phase fluid

Opening through fluid can enter and exit the source.

B – Inlet

two-phase fluid

Opening through fluid can enter and exit the source.

Parameters

Power added – Parameterization for the calculation of power

Isentropic power (default) | None

Parameterization for the calculation of power. Work is isentropic and its calculation is based on the assumptions of zero friction losses and zero heat exchange with the environment. Change to **None** to prevent the source from impacting the temperature of the fluid—for example, when using this block as a boundary condition in a model.

Pressure differential – Pressure gain from port A to port B

0 m³/s (default) | scalar with units of volume/time

Pressure differential between port **A** and port **B**. A positive value corresponds to a pressure that is greater at port **B** than at port **A**. The specified pressure differential is maintained no matter the mass flow rate produced through the ports.

Cross-sectional area at port A — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of volume/time

Area of the fluid opening normal to the direction of flow.

Cross-sectional area at port B — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of volume/time

Area of the fluid opening normal to the direction of flow.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Pressure Source (2P)

Introduced in R2015b

Pressure Source (G)

Generate constant pressure differential

Library: Simscape / Foundation Library / Gas / Sources



Description

The Pressure Source (G) block represents an ideal mechanical energy source in a gas network. The source can maintain a constant pressure differential across its ports regardless of the mass flow rate through the source. There is no flow resistance and no heat exchange with the environment. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

You can choose whether the source performs work on the gas flow:

- If the source is isentropic (**Power added** parameter is set to **Isentropic power**), then the isentropic relation depends on the gas property model.

Gas Model	Equations
Perfect gas	$\frac{(p_A)^{Z \cdot R/c_p}}{T_A} = \frac{(p_B)^{Z \cdot R/c_p}}{T_B}$
Semiperfect gas	$\int_{T_A}^{T_B} \frac{c_p(T)}{T} dT - Z \cdot R \cdot \ln(p_A) = \int_{T_B}^{T_B} \frac{c_p(T)}{T} dT - Z \cdot R \cdot \ln(p_B)$
Real gas	$s(T_A, p_A) = s(T_B, p_B)$

The power delivered to the gas flow is based on the specific total enthalpy associated with the isentropic process.

$$\Phi_{work} = -\dot{m}_A \left(h_A + \frac{w_A^2}{2} \right) - \dot{m}_B \left(h_B + \frac{w_B^2}{2} \right)$$

- If the source performs no work (**Power added** parameter is set to **None**), then the defining equation states that the specific total enthalpy is equal on both sides of the source. It is the same for all three gas property models.

$$h_A + \frac{w_A^2}{2} = h_B + \frac{w_B^2}{2}$$

The power delivered to the gas flow $\Phi_{work} = 0$.

The equations use these symbols:

c_p	Specific heat at constant pressure
-------	------------------------------------

h	Specific enthalpy
\dot{m}	Mass flow rate (flow rate associated with a port is positive when it flows into the block)
p	Pressure
R	Specific gas constant
s	Specific entropy
T	Temperature
w	Flow velocity
Z	Compressibility factor
Φ_{work}	Power delivered to the gas flow through the source

Subscripts A and B indicate the appropriate port.

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Conserving

A — Source inlet

gas

Gas conserving port. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

B — Source outlet

gas

Gas conserving port. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

Parameters

Power added — Select whether the source performs work

Isentropic power (default) | None

Select whether the source performs work on the gas flow:

- **Isentropic power** — The source performs isentropic work on the gas to maintain the specified pressure differential, regardless of the mass flow rate. Use this option to represent an idealized pump or compressor and properly account for the energy input and output, especially in closed-loop systems.
- **None** — The source performs no work on the flow, neither adding nor removing power, regardless of the pressure differential produced by the source. Use this option to set up the desired flow condition upstream of the system, without affecting the temperature of the flow.

Pressure differential — Constant pressure differential across the source

0 MPa (default)

Gas pressure differential across the ports of the source.

Cross-sectional area at port A — Area normal to flow path at port A

0.01 m² (default)

Area normal to flow path at port **A**.

Cross-sectional area at port B — Area normal to flow path at port B

0.01 m² (default)

Area normal to flow path at port **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Pressure Source (G)

Topics

“Modeling Gas Systems”

Introduced in R2016b

Pressure Source (IL)

Generate specified pressure differential in isothermal liquid network

Library: Simscape / Foundation Library / Isothermal Liquid / Sources



Description

The Pressure Source (IL) block represents an ideal mechanical energy source in an isothermal liquid network. The source can maintain specified pressure differential across its ports regardless of the mass flow rate through the source. There is no flow resistance and no heat exchange with the environment.

Ports **A** and **B** represent the source inlet and outlet. The input physical signal at port **P** specifies the pressure differential. Alternatively, you can specify a constant pressure differential as a block parameter. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

The block calculates the work performed on the fluid and includes the results in the simulation data log for information purposes:

$$W_{mech} = \dot{m} \frac{p_B - p_A}{\bar{\rho}}$$

$$\bar{\rho} = \frac{\rho_A + \rho_B}{2}$$

where:

- W_{mech} is the mechanical work performed by the source.
- \dot{m} is the mass flow rate generated by the source.
- p_A and p_B are pressures at ports **A** and **B**, respectively.
- $\bar{\rho}$ is the average fluid mixture density.
- ρ_A and ρ_B are fluid mixture density values at ports **A** and **B**, respectively. Equations used to compute the fluid mixture density depend on the selected isothermal liquid model. For detailed information, see “Isothermal Liquid Modeling Options”.

For information on viewing logged simulation data, see “Data Logging”.

Assumptions and Limitations

- There are no losses due to friction.
- There is no heat exchange with the environment.

Ports

Input

P — Pressure differential control signal, Pa

physical signal

Input physical signal that specifies the pressure differential across the source.

Dependencies

This port is visible if you set the **Source type** parameter to `Controlled`.

Conserving

A — Source inlet

isothermal liquid

Isothermal liquid conserving port. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

B — Source outlet

isothermal liquid

Isothermal liquid conserving port. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

Parameters

Source type — Select whether pressure differential can change during simulation

`Controlled` (default) | `Constant`

Select whether the pressure differential generated by the source can change during simulation:

- `Controlled` — The pressure differential is variable, controlled by an input physical signal. Selecting this option exposes the input port **P**.
- `Constant` — The flow rate is constant during simulation, specified by a block parameter. Selecting this option enables the **Pressure differential** parameter.

Pressure differential — Constant pressure differential across the source

0 MPa (default) | scalar

Desired pressure differential across the source.

Dependencies

Enabled when you set the **Source type** parameter to `Constant`.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Flow Rate Source (IL)

Topics

“Modeling Isothermal Liquid Systems”

“Isothermal Liquid Modeling Options”

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2020a

Pressure Source (MA)

Generate constant pressure differential

Library: Simscape / Foundation Library / Moist Air / Sources



Description

The Pressure Source (MA) block represents an ideal mechanical energy source in a moist air network. The source can maintain a constant pressure differential across its ports regardless of the mass flow rate through the source. There is no flow resistance and no heat exchange with the environment. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

The equations describing the source use these symbols.

c_p	Specific heat at constant pressure
h	Specific enthalpy
h_t	Specific total enthalpy
\dot{m}	Mass flow rate (flow rate associated with a port is positive when it flows into the block)
p	Pressure
ρ	Density
R	Specific gas constant
s	Specific entropy
T	Temperature
Φ_{work}	Power delivered to the moist air flow through the source

Subscripts A and B indicate the appropriate port.

Mass balance:

$$\dot{m}_A + \dot{m}_B = 0$$

$$\dot{m}_{wA} + \dot{m}_{wB} = 0$$

$$\dot{m}_{gA} + \dot{m}_{gB} = 0$$

Energy balance:

$$\Phi_A + \Phi_B + \Phi_{work} = 0$$

If the source performs no work (**Power added** parameter is set to None), then $\Phi_{work} = 0$.

If the source is isentropic (**Power added** parameter is set to Isentropic power), then

$$\Phi_{work} = \dot{m}_A(h_{tB} - h_{tA})$$

where

$$h_{tA} = h_A + \frac{1}{2} \left(\frac{\dot{m}_A}{\rho_A S_A} \right)^2$$

$$h_{tB} = h_B + \frac{1}{2} \left(\frac{\dot{m}_B}{\rho_B S_B} \right)^2$$

The mixture-specific enthalpies, $h_A = h(T_A)$ and $h_B = h(T_B)$, are constrained by the isentropic relation, that is, there is no change in entropy:

$$\int_{T_A}^{T_B} \frac{1}{T} dh(T) = R \ln \left(\frac{p_B}{p_A} \right)$$

The quantity specified by the **Pressure differential** parameter of the source is

$$p_B - p_A = \Delta p_{\text{specified}}$$

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Conserving

A — Source inlet

moist air

Moist air conserving port. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

B — Source outlet

moist air

Moist air conserving port. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

Parameters

Power added — Select whether the source performs work

Isentropic power (default) | None

Select whether the source performs work on the moist air flow:

- **Isentropic power** — The source performs isentropic work on the moist air to maintain the specified mass flow rate, regardless of the pressure differential. Use this option to represent an idealized pump or compressor and properly account for the energy input and output, especially in closed-loop systems.
- **None** — The source performs no work on the flow, neither adding nor removing power, regardless of the mass flow rate produced by the source. Use this option to set up the desired flow condition upstream of the system, without affecting the temperature of the flow.

Pressure differential — Constant pressure differential across the source

0 MPa (default)

Pressure differential of the moist air mixture across the ports of the source.

Cross-sectional area at port A — Area normal to flow path at port A

0.01 m² (default)

Area normal to flow path at port A.

Cross-sectional area at port B — Area normal to flow path at port B

0.01 m² (default)

Area normal to flow path at port B.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Pressure Source (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Pressure Source (TL)

Generate constant pressure differential

Library: Simscape / Foundation Library / Thermal Liquid / Sources



Description

The Pressure Source (TL) block represents an ideal mechanical energy source in a thermal liquid network. The source can maintain the specified pressure differential across its ports regardless of the mass flow rate through the source. There is no flow resistance and no heat exchange with the environment. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

The energy balance at the source is a function of the energy flow rates through ports **A** and **B** and the work done on the fluid:

$$\phi_A + \phi_B + \phi_{work} = 0,$$

where:

- ϕ_A is the energy flow rate into the source through port **A**.
- ϕ_B is the energy flow rate into the source through port **B**.
- ϕ_{work} is the isentropic work done on the fluid.

The isentropic work term is

$$\phi_{work} = \frac{\dot{m}(p_B - p_A)}{\rho_{avg}},$$

where:

- ϕ_{work} is the isentropic work done on the thermal liquid.
- p_A is the pressure at port **A**.
- p_B is the pressure at port **B**.
- ρ_{avg} is the average liquid density,

$$\rho_{avg} = \frac{\rho_A + \rho_B}{2}.$$

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Conserving

A — Source inlet

thermal liquid

Thermal liquid conserving port. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

B — Source outlet

thermal liquid

Thermal liquid conserving port. A positive pressure differential causes the pressure at port **B** to be greater than the pressure at port **A**.

Parameters

Pressure differential — Constant pressure differential across the source

0 Pa (default)

Desired pressure differential across the ports of the source.

Cross-sectional area at ports A and B — Area normal to flow path at inlet and outlet

0.01 m² (default)

Area normal to the direction of flow at the source inlet and outlet. The two cross-sectional areas are assumed identical.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Mass Flow Rate Source (TL) | Controlled Pressure Source (TL) | Controlled Volumetric Flow Rate Source (TL) | Mass Flow Rate Source (TL) | Volumetric Flow Rate Source (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2013b

Probe

Output block variables as signals during simulation

Library: Simscape / Utilities



Description

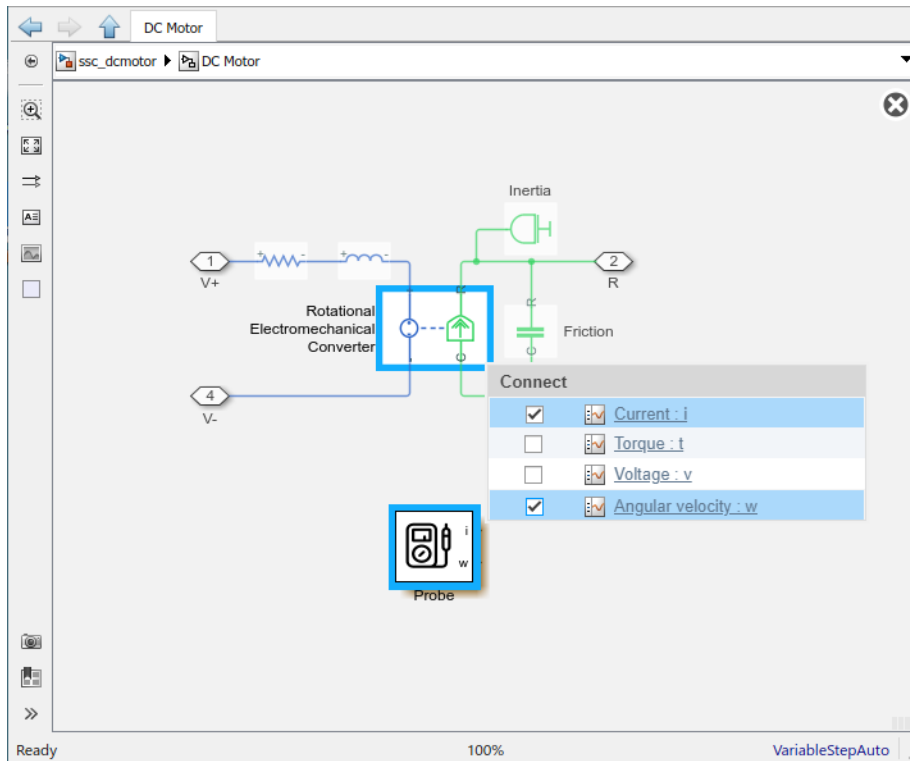
The Probe block lets you select variables from another block in the model and output them as Simulink signals. The following rules apply:

- The selected block must be at the same level of the model hierarchy as the Probe block.
- After selecting the block, you can select only the variables exposed on its **Variables** tab (that is, the same variables that can be used for block-level variable initialization).
- In case of conditional visibility (if a variable is exposed only in a subset of block parameterizations), this variable is always available for probing, regardless of whether it is currently exposed on the **Variables** tab.
- Each of the selected variables is output as a separate signal.
- The output signal unit matches the nominal unit of the variable. For more information, see “Specify Nominal Value-Unit Pairs for a Model” and “Units in Simulink”.
- The Probe block outputs Simulink signals. Therefore, you can connect it directly to Simulink blocks, like scopes or buses.
- You can attach (bind) a Probe block to only one block at a time. However, you can bind multiple Probe blocks to the same block in the model at the same time.

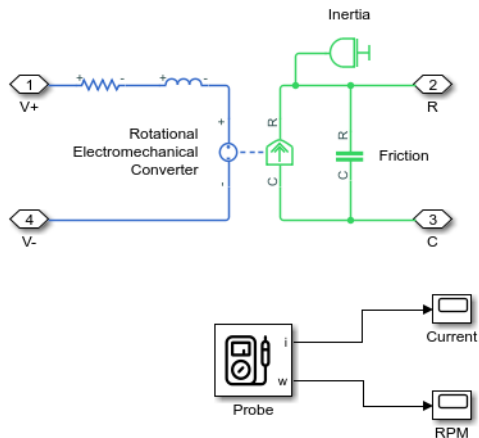
Working with the Block on the Model Canvas

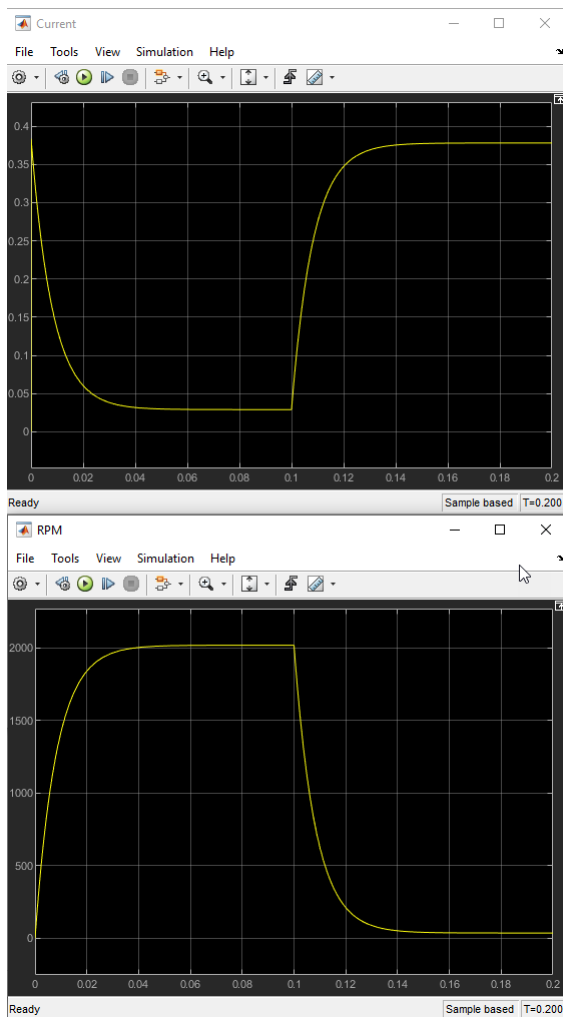
To bind variables to a Probe block:

- 1** Add the Probe block to a model at the desired level.
- 2** Double-click the Probe block to start the binding process.
- 3** Select a Simscape block at the same level of the model hierarchy.
- 4** From a context menu containing all variables available for the block initialization, select the variables to output. For example, if you selected a Rotational Electromechanical Converter block, the available variables include current, voltage, torque, and angular velocity.



- 5 To finish the binding process, click the X in the upper-right corner of the model canvas.
- 6 For each selected variable, the Probe block acquires an additional output port, with the port name matching the variable ID. (In this example, the port names are **i** and **w**.) Connect these ports to scopes or other blocks from the Simulink Sinks library to view the signal values during simulation.





To bind a Probe block to a different block in the model, or to change the selected variables, double-click the Probe block again and repeat the binding process. When binding to a different block, if the new block has variables with the same IDs as the ones previously bound, these variables are automatically selected again.

If you copy a Probe block together with the block that it is bound to, the connection is preserved and the same variables are automatically selected in the new pair.

Command Line Workflow

Instead of the interactive workflow on the model canvas, you can bind a Probe block and select variables by using these commands:

- `simscape.probe.setBoundsBlock(probeBlock, boundBlock)` — Binds *probeBlock* to *boundBlock*. *probeBlock* must be a valid full block path or a handle to a Probe block. *boundBlock* must be a valid full block path or a handle to another block at the same level of the model hierarchy as *probeBlock*. The command does not check whether *boundBlock* is a Simscape block or whether it has variables to probe.

`simscape.probe.setBoundsBlock(probeBlock, [])` returns the Probe block to the unbound state. Does not affect the selected variables.

- `simscape.probe.setVariables(probeBlock, variables)` — Selects variables for the *probeBlock* to output. *probeBlock* must be a valid full block path or a handle to a Probe block. *variables* must be a character vector, cell array of character vectors, or string array that specifies the variables. The character vectors or strings must be unique variable identifiers, lexicographically sorted. The command does not check whether the currently selected *boundBlock* contains these variables.

Instead of lexicographically sorting the variables, you can use this syntax:

```
I = simscape.probe.setVariables(probeBlock, variables, 'Sort', true)
```

The command then sorts the variables before applying them to *probeBlock*. Variable identifiers must still be unique. *I* returns the order in which the sorted *variables* appear as ports on the Probe block.

The programmatic equivalent of the interactive binding and variable selection shown in the example in “Working with the Block on the Model Canvas” on page 1-454 is:

```
simscape.probe.setBoundBlock('ssc_dcmotor/DC Motor/Probe','ssc_dcmotor/DC Motor/Rotational Electromechanical Converter');
simscape.probe.setVariables('ssc_dcmotor/DC Motor/Probe',{'i','w'});
```

As a result, the Probe block also has two output ports, **i** and **w**, bound to the **Current : i** and **Angular velocity : w** variables of the Rotational Electromechanical Converter block, respectively.

If you supply unsorted variables, the command returns the sorted order:

```
I = simscape.probe.setVariables('ssc_dcmotor/DC Motor/Probe', ['w', 'i'], 'Sort', true)
```

```
I =
```

```
     2     1
```

In this example, the second variable, *i*, appears as the first port on the Probe block, followed by the first variable, *w*.

You can use the `simscape.probe.getBoundBlock(probeBlock)` and `simscape.probe.getVariables(probeBlock)` commands, where *probeBlock* is a valid full block path or a handle to a Probe block, to return its bound block and variables, respectively.

Ports

Output

x — Value of selected variable during simulation

scalar | vector | matrix

Outputs the values of one selected variable during simulation as a Simulink signal. The port name matches the name of selected variable.

If you select more than one variable, the block acquires additional output ports.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Topics

“Decide How to Visualize Simulation Data”

Introduced in R2020a

PS Abs

Output absolute value of input physical signal

Library: Simscape / Foundation Library / Physical Signals / Nonlinear Operators



Description

The PS Abs block returns the absolute value of the input physical signal:

$$O = |I|$$

where

I	Physical signal at the input port
O	Physical signal at the output port

Untyped physical ports facilitate unit propagation. The output signal has the same physical unit as the input signal.

Ports

Input

I – Input physical signal, untyped

physical signal

Input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O – Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Sign

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2008a

PS Add

Add two physical signal inputs

Library: Simscape / Foundation Library / Physical Signals / Functions



Description

The PS Add block outputs the sum of two input physical signals:

$$O = I_1 + I_2$$

where

I_1	Physical signal at the first input port
I_2	Physical signal at the second input port
O	Physical signal at the output port

Physical signals at the two input ports must have commensurate units. The unit at the output port is determined by unit propagation rules. For more information, see “Physical Signal Unit Propagation”.

Ports

Input

I1 – Input physical signal, untyped

physical signal

First input physical signal to be added. In the block icon, this port is marked with a plus sign (+).

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

I2 – Input physical signal, untyped

physical signal

Second input physical signal to be added. In the block icon, this port is marked with a plus sign (+).

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O – Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

[PS Divide](#) | [PS Gain](#) | [PS Math Function](#) | [PS Product](#) | [PS Subtract](#)

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007a

PS Asynchronous Sample & Hold

Output sample-and-hold signal with external trigger

Library: Simscape / Foundation Library / Physical Signals / Discrete



Description

The PS Asynchronous Sample & Hold block sets the output signal, Y , equal to the input signal, 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.

The “Asynchronous PWM Voltage Source” example illustrates how you can use the PS Asynchronous Sample & Hold block to build components with more complex behaviors. For an alternative discrete-time implementation, see the “Discrete-Time PWM Voltage Source” example. The discrete-time version is better suited to fixed-step solvers and hardware-in-the-loop applications, whereas the asynchronous implementation is better suited to fast desktop simulation using variable-step solvers.

The signal unit at the output port is determined by unit propagation rules.

Ports

Input

U — Input physical signal, untyped


physical signal

Input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

T — Trigger physical signal, unitless

physical signal

Input physical signal that triggers the sample-and-hold action. In the block icon, this port is marked with the  (rising edge) sign.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

Y — Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Parameters

Initial output – Output at time zero

0 1 (default)

The value and unit of the output signal at time zero. The first edit box represents the signal value. The output of the block remains at this value until the block is triggered by a rising trigger signal becoming positive.

The second combo box represents the signal unit. By default, the unit is 1 (unitless). Specify a unit that is commensurate with the signal unit at the input port **U**. You can select a unit from the drop-down list or type the desired unit name, such as `rpm`, or a valid expression, such as `rad/s`. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Counter

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2011b

PS Ceil

Output the smallest integer larger than or equal to input physical signal

Library: Simscape / Foundation Library / Physical Signals / Nonlinear Operators



Description

The PS Ceil block rounds the input physical signal toward positive infinity, that is, to the nearest integer larger than or equal to the input value:

$$O = \text{ceil}(I)$$

where

I	Physical signal at the input port
O	Physical signal at the output port

Both the input and the output are physical signals. Untyped physical ports facilitate the signal size propagation. However, arguments passed to rounding functions (such as `ceil`) must be dimensionless. Therefore, the input signal must have the unit of 1 (unitless), and the output signal is also unitless.

Ports

Input

I – Input physical signal, untyped

physical signal

Input physical signal. Arguments passed to rounding functions (such as `ceil`) must be dimensionless. Therefore, the input signal does not have to be a scalar but must have the unit of 1 (unitless).

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O – Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Fix | PS Floor | PS Round | ceil

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2009a

PS Constant

Generate constant physical signal

Library: Simscape / Foundation Library / Physical Signals / Sources



Description

The PS Constant block generates a physical signal of a constant value. You specify the value and unit of the signal as the **Constant** parameter.

Ports

Output

0 — Output physical signal, typed by the block parameter value

physical signal

Output physical signal. The signal value and unit are determined by the value and unit of the **Constant** parameter.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Parameters

Constant — Signal value and unit

1 1 (default)

The value and unit of the output signal at port **0**. The first edit box represents the signal value. You can specify both positive and negative values.

The second combo box represents the unit of the output signal. By default, the unit is 1 (unitless). You can select a unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007b

PS Constant Delay

Delay input physical signal by specified time

Library: Simscape / Foundation Library / Physical Signals / Delays



Description

The PS Constant Delay block generates the output physical signal, **Y**, by delaying the input physical signal, **U**:

$$Y = U(t - \tau)$$

where τ is the delay time.

The delay time is constant throughout the simulation. You specify the value of the delay time as the **Delay time** parameter.

For the initial time interval, when $t \leq \text{StartTime} + \tau$, the block outputs the **Input history** parameter value. The unit specified for the **Input history** parameter must be commensurate with the unit of the output signal.

Note

- When simulating a model that contains blocks with delays, memory allocation for storing the data history is controlled by the **Delay memory budget [kB]** parameter in the Solver Configuration block. If this budget is exceeded, simulation errors out. You can adjust this parameter value based on your available memory resources.
 - For recommendation on how to linearize a model that contains blocks with delays, see “Linearizing with Simulink Linearization Blocks”.
-

Untyped physical ports facilitate unit propagation. The output signal has the same physical unit as the input signal.

Ports

Input

U — Input physical signal, untyped

physical signal

Input physical signal.

Output

Y — Output physical signal, untyped

physical signal

Output physical signal.

Parameters

Input history — Signal value and unit during the initial time interval

0.0 1 (default)

The output signal value and unit during the initial time interval, until the specified delay time elapses after the start of simulation. The first edit box represents the signal value.

The second combo box represents the signal unit, which must be commensurate with the unit of the output signal at port **Y**. By default, the unit is 1 (unitless). You can select a unit from the drop-down list or type the desired unit name, such as `rpm`, or a valid expression, such as `rad/s`. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Delay time — Delay time for the signal

1 s (default)

The delay for the signal, in units of time. The parameter value must be positive.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Variable Delay

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2012a

PS Constant Offset Estimator

Measure constant offset value of periodic signal

Library: Simscape / Foundation Library / Physical Signals / Periodic Operators



Description

The PS Constant Offset Estimator block measures the constant offset value of a periodic signal.

A signal is periodic if it completes a pattern within a measurable time frame, called a period, and repeats that pattern over identical subsequent periods. The signal base frequency is the number of periods per second.

The constant offset is the amount by which the average value of the periodic signal is not centered around the t -axis.

Both the input and the output are physical signals. Untyped physical ports facilitate unit propagation. The output signal has the same physical unit as the input signal.

To obtain meaningful results, run the simulation for at least one full time period of the signal.

Ports

Input

I – Input physical signal, untyped

physical signal

Input physical signal. The signal must be periodic.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O – Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Parameters

Base frequency – Periodic signal frequency

60 Hz (default)

Base frequency of the periodic signal.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Harmonic Estimator (Amplitude, Phase) | PS Harmonic Estimator (Real, Imaginary) | PS RMS Estimator

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2015b

PS Counter

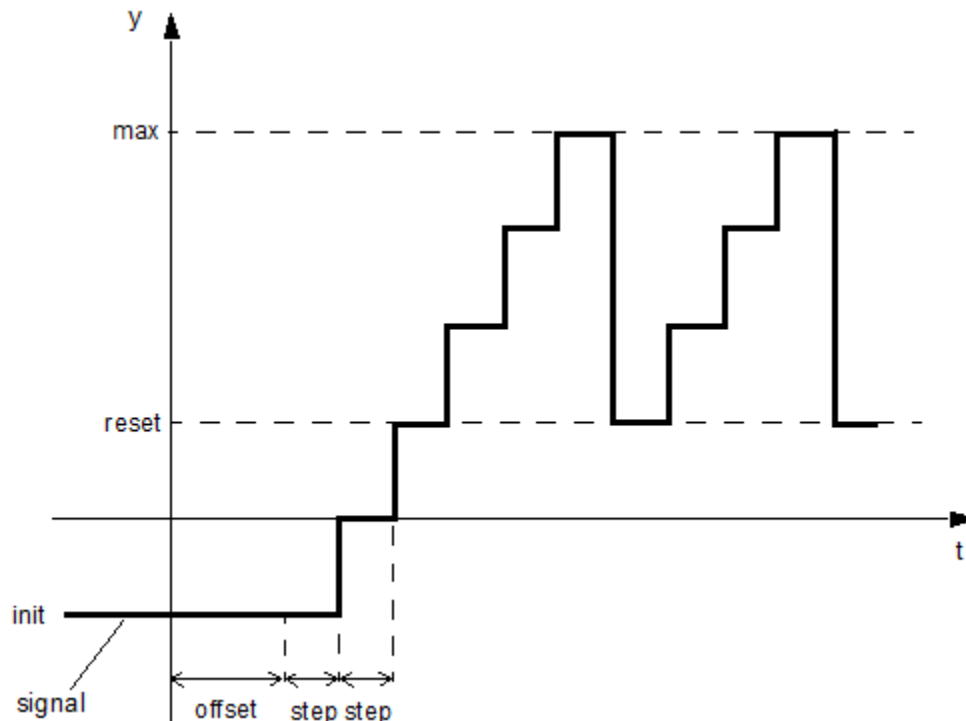
Increment output signal by 1 with every time step

Library: Simscape / Foundation Library / Physical Signals / Sources



Description

The PS Counter block increments the output signal, Y , by 1 with every time step repeatedly 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. The output signal generated by the block is shown in the following diagram.



If the initial time offset is specified, the block outputs the initial signal value $init$ until the simulation time reaches the $offset$ value, at which point the counting cycle starts. The block outputs the current value for one time step, then repeatedly increments the signal value by 1 and outputs it for one time step, until it reaches the maximum value max . The block outputs the max value for one time step, then returns to the $reset$ value, and the counting cycle starts again.

Use this block, in conjunction with other physical signal blocks, to model discrete behaviors.

The “Discrete-Time PWM Voltage Source” example illustrates how you can use the PS Counter block to build components with more complex behaviors. For an alternative asynchronous implementation,

see the “Asynchronous PWM Voltage Source” example. The discrete-time version is better suited to fixed-step solvers and hardware-in-the-loop applications, whereas the asynchronous implementation is better suited to fast desktop simulation using variable-step solvers.

Ports

Output

Y — Output physical signal, unitless

physical signal

Output physical signal.

Parameters

Sample time — Sample time interval

1 s (default)

The value and unit of the time *step* interval. The default *step* value is 1 s. To specify an initial time offset, enter the parameter value as [*step* , *offset*], otherwise the *offset* value is assumed to be 0.

Initial value — Output at time zero

0 (default)

The value of the output signal at the beginning of the first counting cycle. If you specify an initial time offset by using the **Sample time** parameter, the output of the block remains at this value until the simulation time reaches the *offset* value, after which the first counting cycle starts. The value must be an integer.

Reset value — Minimum output

0 (default)

The value of the output signal at the beginning of each counting cycle except the first one. The output of the block remains at this value for one time *step*, specified by the **Sample time** parameter. The value must be an integer.

Maximum value — Maximum output

intmax (default)

The value of the output signal at the end of the counting cycle. The output of the block remains at this value for one time *step*, specified by the **Sample time** parameter, at which point the signal returns to the **Reset value** and the cycle starts again. The value must be an integer. The default value is intmax (2147483647, the largest positive value that can be represented in the MATLAB software with a 32-bit integer).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Asynchronous Sample & Hold

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2012b

PS Dead Zone

Provide region of zero output for physical signals

Library: Simscape / Foundation Library / Physical Signals / Nonlinear Operators



Description

The PS Dead Zone block generates zero output when input signal falls within a specified region, called a dead zone. You can specify the lower and upper limits of the dead zone as block parameters. The block output depends on the input and dead zone:

- If the input is within the dead zone (greater than the lower limit and less than the upper limit), the output is zero.
- If the input is greater than or equal to the upper limit, the output is the input minus the upper limit.
- If the input is less than or equal to the lower limit, the output is the input minus the lower limit.

Both the input and the output are physical signals. Untyped physical ports facilitate unit propagation. The output signal has the same physical unit as the input signal.

Ports

Input

I – Input physical signal, untyped

physical signal

Input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

0 – Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Parameters

Upper limit – Dead zone upper bound

0.5 1 (default)

The upper limit, or end, of the dead zone.

The parameter unit must be commensurate with the unit of the input signal. You can select a unit from the drop-down list or type the desired unit name, such as `rpm`, or a valid expression, such as `rad/s`. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Lower limit — Dead zone lower bound

-0.5 1 (default)

The lower limit, or start, of the dead zone.

The parameter unit must be commensurate with the unit of the input signal. You can select a unit from the drop-down list or type the desired unit name, such as `rpm`, or a valid expression, such as `rad/s`. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Saturation

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007a

PS Divide

Compute element-wise division of two input physical signals

Library: Simscape / Foundation Library / Physical Signals / Functions



Description

The PS Divide block divides one physical signal input by another and outputs the quotient:

$$O = I_1 ./ I_2$$

where

I_1	Physical signal at the first input port (marked with the x sign)
I_2	Physical signal at the second input port (marked with the ÷ sign)
O	Physical signal at the output port

The signal unit at the output port is determined by unit propagation rules. For more information, see “Physical Signal Unit Propagation”.

Ports

Input

I1 – Numerator input, untyped

physical signal

First input physical signal, representing the numerator. In the block icon, this port is marked with the **x** sign.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

I2 – Denominator input, untyped

physical signal

Second input physical signal, representing the denominator. In the block icon, this port is marked with the **÷** sign.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O – Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Add | PS Gain | PS Math Function | PS Product | PS Subtract

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007a

PS Fix

Round input physical signal toward zero

Library: Simscape / Foundation Library / Physical Signals / Nonlinear Operators



Description

The PS Fix block rounds the input physical signal toward zero, that is, for a positive signal returns the nearest integer smaller than or equal to the input value, and for a negative signal returns the nearest integer larger than or equal to the input value:

$$O = \text{fix}(I)$$

where

I	Physical signal at the input port
O	Physical signal at the output port

Both the input and the output are physical signals. Untyped physical ports facilitate the signal size propagation. However, arguments passed to rounding functions (such as `fix`) must be dimensionless. Therefore, the input signal must have the unit of 1 (unitless), and the output signal is also unitless.

Ports

Input

I – Input physical signal, untyped

physical signal

Input physical signal. Arguments passed to rounding functions (such as `fix`) must be dimensionless. Therefore, the input signal does not have to be a scalar but must have the unit of 1 (unitless).

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O – Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Ceil | PS Floor | PS Round | fix

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2009a

PS Floor

Output the largest integer smaller than or equal to input physical signal

Library: Simscape / Foundation Library / Physical Signals / Nonlinear Operators



Description

The PS Floor block rounds the input physical signal toward negative infinity, that is, to the nearest integer smaller than or equal to the input value:

$$O = \text{floor}(I)$$

where

I	Physical signal at the input port
O	Physical signal at the output port

Both the input and the output are physical signals. Untyped physical ports facilitate the signal size propagation. However, arguments passed to rounding functions (such as `floor`) must be dimensionless. Therefore, the input signal must have the unit of 1 (unitless), and the output signal is also unitless.

Ports

Input

I — Input physical signal, untyped

physical signal

Input physical signal. Arguments passed to rounding functions (such as `floor`) must be dimensionless. Therefore, the input signal does not have to be a scalar but must have the unit of 1 (unitless).

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O — Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Ceil | PS Fix | PS Round | floor

Topics

“Physical Signal Unit Propagation”

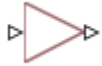
“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2009a

PS Gain

Multiply input physical signal by constant

Library: Simscape / Foundation Library / Physical Signals / Functions



Description

The PS Gain block performs element-wise multiplication of the input physical signal by a constant value (gain). You specify the gain as the **Gain** parameter value and unit.

The input signal value is multiplied by the value of the **Gain** parameter.

The signal unit at the output port is determined by unit propagation rules. For example, if the input signal is in N and the **Gain** parameter unit is m, then the output physical signal has the unit of N*m. For more information, see “Physical Signal Unit Propagation”.

Ports

Input

I – Input physical signal, untyped

physical signal

Input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O – Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Parameters

Gain – Multiplication coefficient

1 1 (default)

The multiplication coefficient. The first edit box represents the value of the coefficient. You can specify both positive and negative values.

The second combo box represents the unit of the multiplication coefficient. By default, the unit is 1 (unitless), which means that the output signal at port **O** has the same unit as the input signal. If you

specify a different unit for the **Gain** parameter, then the signal unit at the output port **O** is the input signal unit multiplied by the unit of the **Gain** parameter. You can select a unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Add | PS Divide | PS Math Function | PS Product | PS Subtract

Topics

“Physical Signal Unit Propagation”

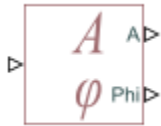
“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007a

PS Harmonic Estimator (Amplitude, Phase)

Measure harmonic amplitude and phase of periodic signal

Library: Simscape / Foundation Library / Physical Signals / Periodic Operators



Description

The PS Harmonic Estimator (Amplitude, Phase) block measures amplitude and phase of a single frequency component (harmonic) of a periodic signal.

A signal is periodic if it completes a pattern within a measurable time frame, called a period, and repeats that pattern over identical subsequent periods. The signal base frequency is the number of periods per second.

The block accepts a physical signal with one or more harmonics present. It outputs the phase and amplitude for the selected harmonic. Untyped physical ports facilitate unit propagation. The output signal for the amplitude has the same physical unit as the input signal. The output signal for phase has the unit of radians, or 1. For more information, see “Angular Units”.

To obtain meaningful results, run the simulation for at least one full time period of the signal.

Ports

Input

I — Periodic signal, untyped

physical signal

Input physical signal. The signal must be periodic.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

A — Amplitude of the harmonic signal, untyped

physical signal

Output physical signal corresponding to the harmonic amplitude.

Phi — Phase of the harmonic signal, rad

physical signal

Output physical signal corresponding to the harmonic phase. Although the physical output port is untyped, the signal unit is determined by the block equations. When connecting a PS-Simulink

Converter block to this port, set its **Output signal unit** parameter to 1, or to another value commensurate with angular units. For more information, see “Angular Units”.

Parameters

Base frequency — Periodic signal frequency

60 Hz (default)

Base frequency of the periodic signal.

Harmonic number — Number of harmonic

1 (default)

The number of harmonic within the periodic signal. The value must be a positive integer.

Minimum amplitude for phase detection — Minimum signal amplitude for measuring phase

0.001 1 (default)

The minimum amplitude necessary for measuring the harmonic phase. The parameter unit must be commensurate with the unit of the input signal. If the harmonic amplitude is less than this value, the output at port **Phi** is 0.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Constant Offset Estimator | PS Harmonic Estimator (Real, Imaginary) | PS RMS Estimator

Topics

“Physical Signal Unit Propagation”

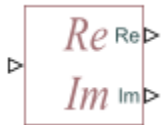
“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2015b

PS Harmonic Estimator (Real, Imaginary)

Measure real and imaginary parts of periodic signal harmonic

Library: Simscape / Foundation Library / Physical Signals / Periodic Operators



Description

The PS Harmonic Estimator (Real, Imaginary) block measures real and imaginary parts of a single frequency component (harmonic) of a periodic signal.

A signal is periodic if it completes a pattern within a measurable time frame, called a period, and repeats that pattern over identical subsequent periods. The signal base frequency is the number of periods per second.

The block accepts a physical signal with one or more harmonics present. It outputs the real and imaginary parts of the signal for the selected harmonic. Untyped physical ports facilitate unit propagation. The output signals at both ports have the same physical unit as the input signal.

To obtain meaningful results, run the simulation for at least one full time period of the signal.

Ports

Input

I — Periodic signal, untyped

physical signal

Input physical signal. The signal must be periodic.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

Re — Real part of the harmonic signal, untyped

physical signal

Output physical signal corresponding to the real part of the selected harmonic.

Im — Imaginary part of the harmonic signal, untyped

physical signal

Output physical signal corresponding to the imaginary part of the selected harmonic.

Parameters

Base frequency — Periodic signal frequency

60 Hz (default)

Base frequency of the periodic signal.

Harmonic number — Number of harmonic

1 (default)

The number of harmonic within the periodic signal. The value must be a positive integer.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Constant Offset Estimator | PS Harmonic Estimator (Amplitude, Phase) | PS RMS Estimator

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2015b

PS Integrator

Integrate physical signal

Library: Simscape / Foundation Library / Physical Signals / Linear Operators



Description

The PS Integrator block outputs the integral of its input at the current time step. The following equation represents the output of the block:

$$y(t) = \int_{t_0}^t u(t)dt + y_0$$

where

u	Physical signal at the input port I
y_0	Initial condition
y	Physical signal at the output port O
t	Time

The PS Integrator block is a dynamic system with one state, its output. The PS Integrator block's input is the state's time derivative:

$$x = y(t)$$

$$x_0 = y_0$$

$$\dot{x} = u(t)$$

The solver computes the output of the PS Integrator block at the current time step, using the current input value and the value of the state at the previous time step. To support this computational model, the PS Integrator block saves its output at the current time step for use by the solver to compute its output at the next time step. The block also provides the solver with an initial condition for use in computing the block's initial state at the beginning of a simulation run. The default value of the initial condition is 0 s. You can specify another value for the initial condition as a parameter on the block dialog box.

The block performs integration of the input signal over time, therefore the unit of the output signal equals the unit of the input signal multiplied by the unit of time, s. For example, if the input signal is in m/s, then the output signal is in m. The unit specified for the **Initial condition** parameter must be commensurate with the unit of the output signal.

Ports

Input

I — Input physical signal, u , untyped

physical signal

Input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

0 — Output physical signal, y , untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Parameters

Initial condition — Initial state for the integration

0 s (default)

Specify the initial condition for use in computing the block's initial state at the beginning of a simulation run.

The unit specified for the **Initial condition** parameter must match the unit of the output signal. By default, the input signal is unitless. Then the output signal is in s, and the **Initial condition** parameter unit is also s. If your input signal has a physical unit, multiply this unit by a unit of time to set the correct unit for the **Initial condition** parameter. For example, if the input signal is in m/s, then specify the **Initial condition** parameter unit as m.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see "Upgrading Models with Legacy Physical Signal Blocks".

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007a

PS Lookup Table (1D)

Approximate one-dimensional function using specified lookup method

Library: Simscape / Foundation Library / Physical Signals / Lookup Tables



Description

The PS Lookup Table (1D) block computes an approximation to some function $f=f(x)$ given data vectors x and f . Both the input and the output are physical signals.

The length of the x and f data vectors provided to this block must match. Also, the x data vector must be strictly monotonic, either increasing or decreasing.

You define the lookup table by specifying the **Table grid vector** parameter as a 1-by- n vector (x data vector) and the **Table values** parameter as a 1-by- n vector (f data vector). The block generates output based on the input values using the selected interpolation and extrapolation methods. You have a choice of two interpolation methods and two extrapolation methods. You also have an option for the block to issue an error if the input signal value is outside the table grid vector range.

Plotting Table Data

Plotting a lookup table lets you visualize the data before simulating the model, to make sure that the table is correct. The plot reflects tabulated data specified for the block, as well as the selected interpolation and extrapolation options.

To plot the data, right-click the block in your model and, from the context menu, select **Foundation Library > Plot Table**. For more information, see “Plot Lookup Tables”.

Ports

Input

x — Input value along the x-axis, untyped

physical signal

Input value along the x -axis.

Output

f — Output function value, untyped

physical signal

Output function value, based on applying the lookup table to the input value. The output signal unit is determined by the unit of the **Table values** parameter.

Parameters

Table grid vector — Vector of input values along the x-axis

[1, 2, 3, 4, 5] 1 (default)

Specify the vector of input values as a one-dimensional array. The input values vector must be strictly monotonic, either increasing or decreasing. The values can be nonuniformly spaced. For smooth interpolation, the vector must contain at least three values. For linear interpolation, two values are sufficient.

Parameter unit must be commensurate with the unit of the input signal at port **x**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as `rpm`, or a valid expression, such as `rad/s`. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Table values — Vector of output values along the f-axis

[0, 1, 2, 3, 4] 1 (default)

Specify the vector of output values as a one-dimensional array. The output values vector must be of the same size as the input values vector.

Parameter unit determines the unit of the output signal at port **f**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as `rpm`, or a valid expression, such as `rad/s`. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Interpolation method — Select the interpolation method

Linear (default) | Smooth

Select one of the following interpolation methods for approximating the output value when the input value is between two consecutive grid points:

- **Linear** — Uses a linear function. Select this option to get the best performance.
- **Smooth** — Uses a modified Akima interpolation algorithm. For details, see `tablelookup`. Select this option to produce a continuous curve with continuous first-order derivatives.

Extrapolation method — Select the extrapolation method

Linear (default) | Nearest | Error

Select one of the following extrapolation methods for determining the output value when the input value is outside the range specified in the argument list:

- **Linear** — Extends from the edge of the interpolation region linearly. The slope of the linear extrapolation is equal to the slope of the interpolated curve at the edge of the interpolation region.
- **Nearest** — Extends from the edge of the interpolation region as a constant. The value of the nearest extrapolation is equal to the value of the interpolated curve at the edge of the interpolation region. Select this option to produce an extrapolation that does not go above the highest point in the data or below the lowest point in the data.
- **Error** — Issues an error if the input signal is outside the range of the table. Select this option to avoid going into the extrapolation mode when you want your data to be within the table range.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Lookup Table (2D) | PS Lookup Table (3D) | PS Lookup Table (4D) | `tablelookup`

Topics

“Plot Lookup Tables”

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007a

PS Lookup Table (2D)

Approximate two-dimensional function using specified lookup method

Library: Simscape / Foundation Library / Physical Signals / Lookup Tables



Description

The PS Lookup Table (2D) block computes an approximation to some function $f=f(x_1, x_2)$ given the x_1 , x_2 , f data points. The two inputs and the output are physical signals.

You define the lookup table by specifying the **Table grid vector 1** parameter (vector of data points along the first axis), the **Table grid vector 2** parameter (vector of data points along the second axis), and the **2D array of table values** (array of output values). The block works on Cartesian mesh, i.e., function values must be specified at vertices of a rectangular array.

The x_1 and x_2 data vectors must be strictly monotonic, either increasing or decreasing. The array size of the tabulated function values must match the dimensions defined by the input vectors. That is, if the inputs are a 1-by- m vector and a 1-by- n vector, supply an m -by- n matrix of output values.

The block generates output based on the input grid lookup using the selected interpolation and extrapolation methods. You have a choice of two interpolation methods and two extrapolation methods. You also have an option for the block to issue an error if any of the input signal values is outside the respective table grid vector range.

Plotting Table Data

Plotting a lookup table lets you visualize the data before simulating the model, to make sure that the table is correct. The plot reflects tabulated data specified for the block, as well as the selected interpolation and extrapolation options.

To plot the data, right-click the block in your model and, from the context menu, select **Foundation Library > Plot Table**. For more information, see “Plot Lookup Tables”.

Ports

Input

x1 — Input value along the first axis, untyped

physical signal

Input value along the first axis.

x2 — Input value along the second axis, untyped

physical signal

Input value along the second axis.

Output

f – Output function value, untyped

physical signal

Output function value, based on applying the lookup table to the two input values. The output signal unit is determined by the unit of the **2D array of table values** parameter.

Parameters

Table grid vector 1 – Vector of input values along the first axis

[1, 2, 3, 4, 5] 1 (default)

Specify the vector of input values along the first axis as a 1-by-m array. The input values vector must be strictly monotonic, either increasing or decreasing. The values can be nonuniformly spaced. For smooth interpolation, the vector must contain at least three values. For linear interpolation, two values are sufficient.

Parameter unit must be commensurate with the unit of the input signal at port **x1**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Table grid vector 2 – Vector of input values along the second axis

[1, 2, 3, 4, 5] 1 (default)

Specify the vector of input values along the second axis as a 1-by-n array. The input values vector must be strictly monotonic, either increasing or decreasing. The values can be nonuniformly spaced. For smooth interpolation, the vector must contain at least three values. For linear interpolation, two values are sufficient.

Parameter unit must be commensurate with the unit of the input signal at port **x2**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

2D array of table values – Two-dimensional array of output values at input grid vertices

[0, 1, 2, 3, 4; 1, 2, 3, 4, 5; 2, 3, 4, 5, 6; 3, 4, 5, 6, 7; 4, 5, 6, 7, 8] 1 (default)

Specify the output values as an m-by-n matrix, defining the function values at the input grid vertices. The matrix size must match the dimensions defined by the input vectors.

Parameter unit determines the unit of the output signal at port **f**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Interpolation method – Select the interpolation method

Linear (default) | Smooth

Select one of the following interpolation methods for approximating the output value when the input value is between two consecutive grid points:

- **Linear** — Uses an extension of linear algorithm for multidimensional interpolation. The method performs linear interpolation first in $x1$ -direction and then in $x2$ -direction. Select this option to get the best performance.
- **Smooth** — Uses a modified Akima interpolation algorithm. For details, see `tablelookup`. Select this option to produce a continuous surface with continuous first-order derivatives.

Extrapolation method — Select the extrapolation method

Linear (default) | Nearest | Error

Select one of the following extrapolation methods for determining the output value when the input value is outside the range specified in the argument list:

- **Linear** — Extends from the edge of the interpolation region linearly. The slope of the linear extrapolation is equal to the slope of the interpolated surface at the edge of the interpolation region.
- **Nearest** — Extends from the edge of the interpolation region as a constant. The value of the nearest extrapolation is equal to the value of the interpolated surface at the edge of the interpolation region. Select this option to produce an extrapolation that does not go above the highest point in the data or below the lowest point in the data.
- **Error** — Issues an error if any of the input signals is outside the range of the table. Select this option to avoid going into the extrapolation mode when you want your data to be within the table range.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Lookup Table (1D) | PS Lookup Table (3D) | PS Lookup Table (4D) | `tablelookup`

Topics

“Plot Lookup Tables”

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007a

PS Lookup Table (3D)

Approximate three-dimensional function using specified lookup method

Library: Simscape / Foundation Library / Physical Signals / Lookup Tables



Description

The PS Lookup Table (3D) block computes an approximation to some function $f=f(x_1, x_2, x_3)$ given the x_1 , x_2 , x_3 , f data points. The three inputs and the output are physical signals.

You define the lookup table by specifying the **Table grid vector 1** parameter (vector of data points along the first axis), the **Table grid vector 2** parameter (vector of data points along the second axis), the **Table grid vector 3** parameter (vector of data points along the third axis), and the **3D array of table values** parameter (array of output values).

The x_1 , x_2 , and x_3 data vectors must be strictly monotonic, either increasing or decreasing. The array size of the tabulated function values must match the dimensions defined by the input vectors. That is, if the three input vectors have sizes 1-by- m , 1-by- n , and 1-by- p , respectively, supply an m -by- n -by- p array of output values.

The block generates output based on the input grid lookup using the selected interpolation and extrapolation methods. You have a choice of two interpolation methods and two extrapolation methods. You also have an option for the block to issue an error if any of the input signal values is outside the respective table grid vector range.

Ports

Input

x1 — Input value along the first axis, untyped

physical signal

Input value along the first axis.

x2 — Input value along the second axis, untyped

physical signal

Input value along the second axis.

x3 — Input value along the third axis, untyped

physical signal

Input value along the third axis.

Output

f — Output function value, untyped

physical signal

Output function value, based on applying the lookup table to the three input values. The output signal unit is determined by the unit of the **3D array of table values** parameter.

Parameters

Table grid vector 1 — Vector of input values along the first axis

[1, 2, 3] 1 (default)

Specify the vector of input values along the first axis as a 1-by-m array. The input values vector must be strictly monotonic, either increasing or decreasing. The values can be nonuniformly spaced. For smooth interpolation, the vector must contain at least three values. For linear interpolation, two values are sufficient.

Parameter unit must be commensurate with the unit of the input signal at port **x1**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Table grid vector 2 — Vector of input values along the second axis

[1, 2, 3, 4] 1 (default)

Specify the vector of input values along the second axis as a 1-by-n array. The input values vector must be strictly monotonic, either increasing or decreasing. The values can be nonuniformly spaced. For smooth interpolation, the vector must contain at least three values. For linear interpolation, two values are sufficient.

Parameter unit must be commensurate with the unit of the input signal at port **x2**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Table grid vector 3 — Vector of input values along the third axis

[1, 2, 3] 1 (default)

Specify the vector of input values along the third axis as a 1-by-p array. The input values vector must be strictly monotonic, either increasing or decreasing. The values can be nonuniformly spaced. For smooth interpolation, the vector must contain at least three values. For linear interpolation, two values are sufficient.

Parameter unit must be commensurate with the unit of the input signal at port **x3**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

3D array of table values — Three-dimensional array of output values at input grid vertices

ones(3, 4, 3) 1 (default)

Specify the output values as an m -by- n -by- p array, defining the function values at the input grid vertices. The output array size must match the dimensions defined by the input vectors.

Parameter unit determines the unit of the output signal at port **f**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Interpolation method — Select the interpolation method

Linear (default) | Smooth

Select one of the following interpolation methods for approximating the output value when the input value is between two consecutive grid points:

- **Linear** — Uses an extension of linear algorithm for multidimensional interpolation. The method performs linear interpolation first in $x1$ -direction, then in $x2$ -direction, and then in $x3$ -direction. Select this option to get the best performance.
- **Smooth** — Uses a modified Akima interpolation algorithm. For details, see `tablelookup`. Select this option to produce a continuous surface with continuous first-order derivatives.

Extrapolation method — Select the extrapolation method

Linear (default) | Nearest | Error

Select one of the following extrapolation methods for determining the output value when the input value is outside the range specified in the argument list:

- **Linear** — Extends from the edge of the interpolation region linearly. The slope of the linear extrapolation is equal to the slope of the interpolated surface at the edge of the interpolation region.
- **Nearest** — Extends from the edge of the interpolation region as a constant. The value of the nearest extrapolation is equal to the value of the interpolated surface at the edge of the interpolation region. Select this option to produce an extrapolation that does not go above the highest point in the data or below the lowest point in the data.
- **Error** — Issues an error if any of the input signals is outside the range of the table. Select this option to avoid going into the extrapolation mode when you want your data to be within the table range.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Lookup Table (1D) | PS Lookup Table (2D) | PS Lookup Table (4D) | `tablelookup`

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2016a

PS Lookup Table (4D)

Approximate four-dimensional function using specified lookup method

Library: Simscape / Foundation Library / Physical Signals / Lookup Tables



Description

The PS Lookup Table (4D) block computes an approximation to some function $f=f(x_1, x_2, x_3, x_4)$ given the x_1, x_2, x_3, x_4, f data points. The four inputs and the output are physical signals.

You define the lookup table by specifying the **Table grid vector 1** parameter (vector of data points along the first axis), the **Table grid vector 2** parameter (vector of data points along the second axis), the **Table grid vector 3** parameter (vector of data points along the third axis), the **Table grid vector 4** parameter (vector of data points along the fourth axis), and the **4D array of table values** parameter (array of output values).

The $x_1, x_2, x_3,$ and x_4 data vectors must be strictly monotonic, either increasing or decreasing. The array size of the tabulated function values must match the dimensions defined by the input vectors. That is, if the four input vectors have sizes 1-by- $m, 1$ -by- $n, 1$ -by- $p,$ and 1-by- $q,$ respectively, supply an m -by- n -by- p -by- q array of output values.

The block generates output based on the input grid lookup using the selected interpolation and extrapolation methods. You have a choice of two interpolation methods and two extrapolation methods. You also have an option for the block to issue an error if any of the input signal values is outside the respective table grid vector range.

Ports

Input

x1 – Input value along the first axis, untyped

physical signal

Input value along the first axis.

x2 – Input value along the second axis, untyped

physical signal

Input value along the second axis.

x3 – Input value along the third axis, untyped

physical signal

Input value along the third axis.

x4 — Input value along the fourth axis, untyped

physical signal

Input value along the fourth axis.

Output**f — Output function value, untyped**

physical signal

Output function value, based on applying the lookup table to the four input values. The output signal unit is determined by the unit of the **4D array of table values** parameter.

Parameters**Table grid vector 1 — Vector of input values along the first axis**

[1, 2, 3] (default)

Specify the vector of input values along the first axis as a 1-by-m array. The input values vector must be strictly monotonic, either increasing or decreasing. The values can be nonuniformly spaced. For smooth interpolation, the vector must contain at least three values. For linear interpolation, two values are sufficient.

Parameter unit must be commensurate with the unit of the input signal at port **x1**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Table grid vector 2 — Vector of input values along the second axis

[1, 2, 3, 4] (default)

Specify the vector of input values along the second axis as a 1-by-n array. The input values vector must be strictly monotonic, either increasing or decreasing. The values can be nonuniformly spaced. For smooth interpolation, the vector must contain at least three values. For linear interpolation, two values are sufficient.

Parameter unit must be commensurate with the unit of the input signal at port **x2**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Table grid vector 3 — Vector of input values along the third axis

[1, 2, 3] (default)

Specify the vector of input values along the third axis as a 1-by-p array. The input values vector must be strictly monotonic, either increasing or decreasing. The values can be nonuniformly spaced. For smooth interpolation, the vector must contain at least three values. For linear interpolation, two values are sufficient.

Parameter unit must be commensurate with the unit of the input signal at port **x3**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Table grid vector 4 — Vector of input values along the fourth axis

[1, 2, 3, 4, 5] (default)

Specify the vector of input values along the third axis as a 1-by-q array. The input values vector must be strictly monotonic, either increasing or decreasing. The values can be nonuniformly spaced. For smooth interpolation, the vector must contain at least three values. For linear interpolation, two values are sufficient.

Parameter unit must be commensurate with the unit of the input signal at port **x4**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

4D array of table values — Four-dimensional array of output values at input grid vertices

ones(3, 4, 3, 5) (default)

Specify the output values as an m-by-n-by-p-by-q array, defining the function values at the input grid vertices. The output array size must match the dimensions defined by the input vectors.

Parameter unit determines the unit of the output signal at port **f**. By default, the unit is 1 (unitless). You can select a different unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Interpolation method — Select the interpolation method

Linear (default) | Smooth

Select one of the following interpolation methods for approximating the output value when the input value is between two consecutive grid points:

- **Linear** — Uses an extension of linear algorithm for multidimensional interpolation. The method performs linear interpolation first in *x1*-direction, then in *x2*-direction, then in *x3*-direction, and then in *x4*-direction. Select this option to get the best performance.
- **Smooth** — Uses a modified Akima interpolation algorithm. For details, see `tablelookup`. Select this option to produce a continuous surface with continuous first-order derivatives.

Extrapolation method — Select the extrapolation method

Linear (default) | Nearest | Error

Select one of the following extrapolation methods for determining the output value when the input value is outside the range specified in the argument list:

- **Linear** — Extends from the edge of the interpolation region linearly. The slope of the linear extrapolation is equal to the slope of the interpolated surface at the edge of the interpolation region.
- **Nearest** — Extends from the edge of the interpolation region as a constant. The value of the nearest extrapolation is equal to the value of the interpolated surface at the edge of the interpolation region. Select this option to produce an extrapolation that does not go above the highest point in the data or below the lowest point in the data.
- **Error** — Issues an error if any of the input signals is outside the range of the table. Select this option to avoid going into the extrapolation mode when you want your data to be within the table range.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Lookup Table (1D) | PS Lookup Table (2D) | PS Lookup Table (3D) | `tablelookup`

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2016b

PS Math Function

Apply mathematical function to input physical signal

Library: Simscape / Foundation Library / Physical Signals / Functions



Description

The PS Math Function block applies a mathematical function to the value and unit of the input physical signal, u . The block output is the result of the operation of the function on the input. You can select one of the following functions from the **Function choice** parameter list.

Function	Description	Mathematical Expression
$\sin(u)$	Sine	$\sin(u)$
$\cos(u)$	Cosine	$\cos(u)$
$\exp(u)$	Exponential	e^u
$\log(u)$	Natural logarithm	$\ln(u)$
10^u	Power of base 10	10^u
$\log_{10}(u)$	Common (base 10) logarithm	$\log(u)$
u^2	Power 2	u^2
\sqrt{u}	Square root	$u^{0.5}$
$1/u$	Reciprocal	$1/u$
$\tanh(u)$	Hyperbolic tangent	$\tanh(u)$

The PS Math Function block 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{u} , the PS Math Function block issues an error if it receives negative input during simulation.

Note For u^2 , \sqrt{u} , and $1/u$ the unit of the output signal is the result of the operation of the function on the input signal unit. For all other functions in the list, the input signal must be unitless.

Ports

Input

I — Input physical signal, u , untyped

physical signal

Input port for the operand physical signal, u .

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

0 — Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Parameters

Function choice — Function to perform on the operand signal

`sin(u)` (default) | `cos(u)` | `exp(u)` | `log(u)` | `10^u` | `log10(u)` | `u^2` | `sqrt(u)` | `1/u` | `tanh(u)`

Select the function to perform. The block output is the result of the operation of the function on the value and unit of the input signal.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Add | PS Divide | PS Gain | PS Product | PS Subtract

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007b

PS Max

Output maximum of two input physical signals

Library: Simscape / Foundation Library / Physical Signals / Nonlinear Operators



Description

The PS Max block outputs the maximum of its two input physical signals:

$$O = \max(I_1, I_2)$$

where

I_1	Physical signal at the first input port
I_2	Physical signal at the second input port
O	Physical signal at the output port

Physical signals at the two input ports must have commensurate units. The unit at the output port is determined by unit propagation rules. For more information, see “Physical Signal Unit Propagation”.

Ports

Input

I1 – Input physical signal, untyped

physical signal

First input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

I2 – Input physical signal, untyped

physical signal

Second input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O – Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Min

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007b

PS Min

Output minimum of two input physical signals

Library: Simscape / Foundation Library / Physical Signals / Nonlinear Operators



Description

The PS Min block outputs the minimum of its two input physical signals:

$$O = \min(I_1, I_2)$$

where

I_1	Physical signal at the first input port
I_2	Physical signal at the second input port
O	Physical signal at the output port

Physical signals at the two input ports must have commensurate units. The unit at the output port is determined by unit propagation rules. For more information, see “Physical Signal Unit Propagation”.

Ports

Input

I1 – Input physical signal, untyped

physical signal

First input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

I2 – Input physical signal, untyped

physical signal

Second input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O – Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Max

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007b

PS Product

Multiply two physical signal inputs

Library: Simscape / Foundation Library / Physical Signals / Functions



Description

The PS Product block outputs the element-wise product of two input physical signals:

$$O = I_1 \cdot * I_2$$

where

I_1	Physical signal at the first input port
I_2	Physical signal at the second input port
O	Physical signal at the output port

The signal unit at the output port is determined by unit propagation rules. For more information, see “Physical Signal Unit Propagation”.

Ports

Input

I1 — Input physical signal, untyped

physical signal

First input physical signal to be added.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

I2 — Input physical signal, untyped

physical signal

Second input physical signal to be added.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O — Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Add | PS Divide | PS Gain | PS Math Function | PS Subtract

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007a

PS Random Number

Generate normally distributed random numbers for physical modeling

Library: Simscape / Foundation Library / Physical Signals / Sources



Description

The PS Random Number block generates normally (Gaussian) distributed random numbers. To generate uniformly distributed random numbers, use the PS Uniform Random Number block.

The block behavior is the same as the Simulink Random Number block (except that it generates a physical signal rather than a Simulink signal) and is based on the polar rejection method ([1], [2]).

You have an option to specify an initial time offset as part of the **Sample time** parameter. In this case, the block outputs 0 until the simulation time reaches the *offset* value, at which point the random sequence starts.

PS Random Number blocks that use the same seed and parameters generate a repeatable sequence. The seed resets to the specified value each time a simulation starts. By default, the block produces a sequence that has a mean of 0 and a variance of 1.

Ports

Output

Y — Output physical signal, unitless

physical signal

Output physical signal.

Parameters

Mean — Mean of the random numbers

0 (default)

The mean of the random numbers generated by the block.

Variance — Variance of the random numbers

1 (default)

The variance of the random numbers generated by the block.

Seed — Starting seed

0 (default)

The starting seed for the random number generator. Output is repeatable for a given seed. The seed must be an integer in the range of 0 to $(2^{32} - 1)$.

Sample time – Sample time interval

1 s (default)

The value of the time *step* interval. The default *step* value is 1 s. To specify an initial time offset, enter the parameter value as [*step*, *offset*], otherwise the *offset* value is assumed to be 0. The offset must be less than the step size.

Compatibility Considerations**Unit Propagation**

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

References

- [1] Bell, J. R. “Algorithm 334: Normal random deviates.” *Communications of the ACM*. Vol. 11, Number 7, 1968, p. 498.
- [2] Knop, R. “Remark on Algorithm 334 [G5]: normal random deviates.” *Communications of the ACM*. Vol. 12, Number 5, 1969, p. 281.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

PS Uniform Random Number

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2013a

PS Repeating Sequence

Output periodic piecewise linear signal

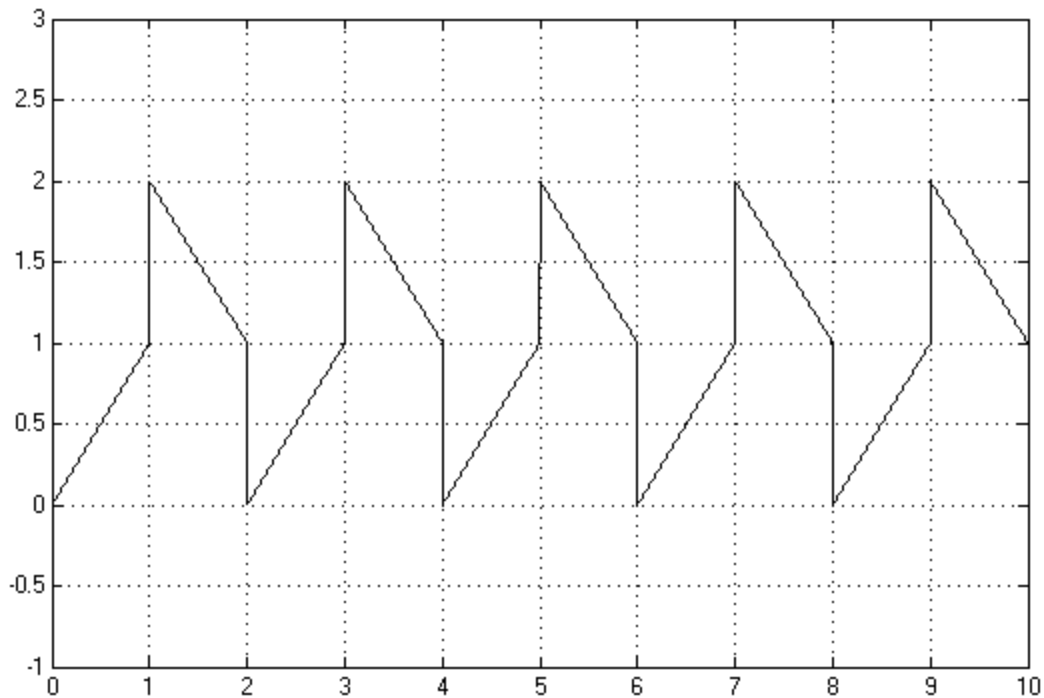
Library: Simscape / Foundation Library / Physical Signals / Sources



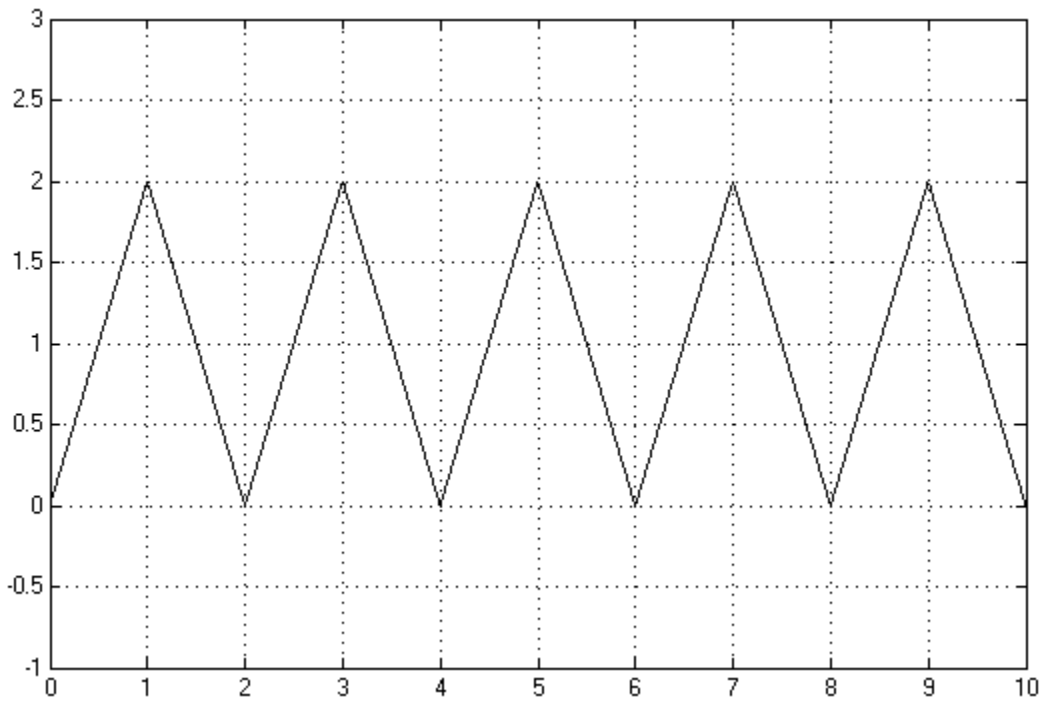
Description

The PS Repeating Sequence block outputs a periodic piecewise linear signal, y . 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. The number of segments must be no greater than 100. You specify how to connect the segments by choosing a signal type. For the same set of block parameter values, the resulting output signal will be different depending on the signal type:

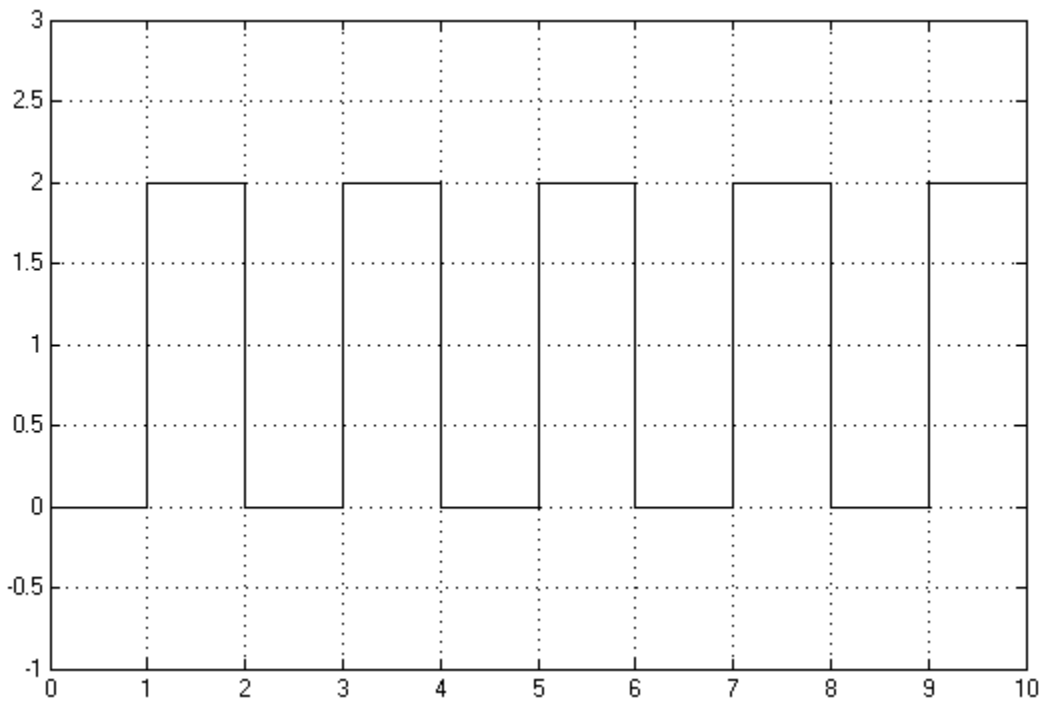
- **Discontinuous** — Each linear segment in the repeating sequence is defined by its duration, start value, and end value. If the end value of a segment is not the same as the start value of the next segment, they are connected by a vertical line.



- **Continuous** — Each linear segment in the repeating sequence is defined by its duration and start value. The end value of a segment is the same as the start value of the next segment.



- Discrete — Each linear segment in the repeating sequence is defined by its duration and start value. The end value of a segment is the same as its start value.

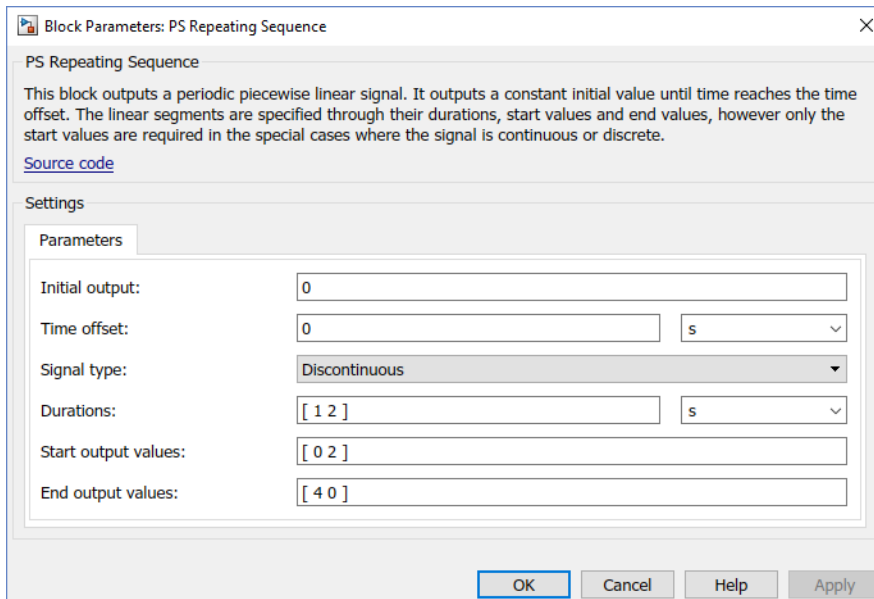


Use this block to generate various types of physical signals, such as pulse, sawtooth, stair, and so on.

Discontinuous Repeating Sequence

This example shows the mapping between the block parameter values and the resulting output signal.

Set the block parameters as shown:



Block Parameters: PS Repeating Sequence

PS Repeating Sequence

This block outputs a periodic piecewise linear signal. It outputs a constant initial value until time reaches the time offset. The linear segments are specified through their durations, start values and end values, however only the start values are required in the special cases where the signal is continuous or discrete.

[Source code](#)

Settings

Parameters

Initial output:

Time offset: s

Signal type:

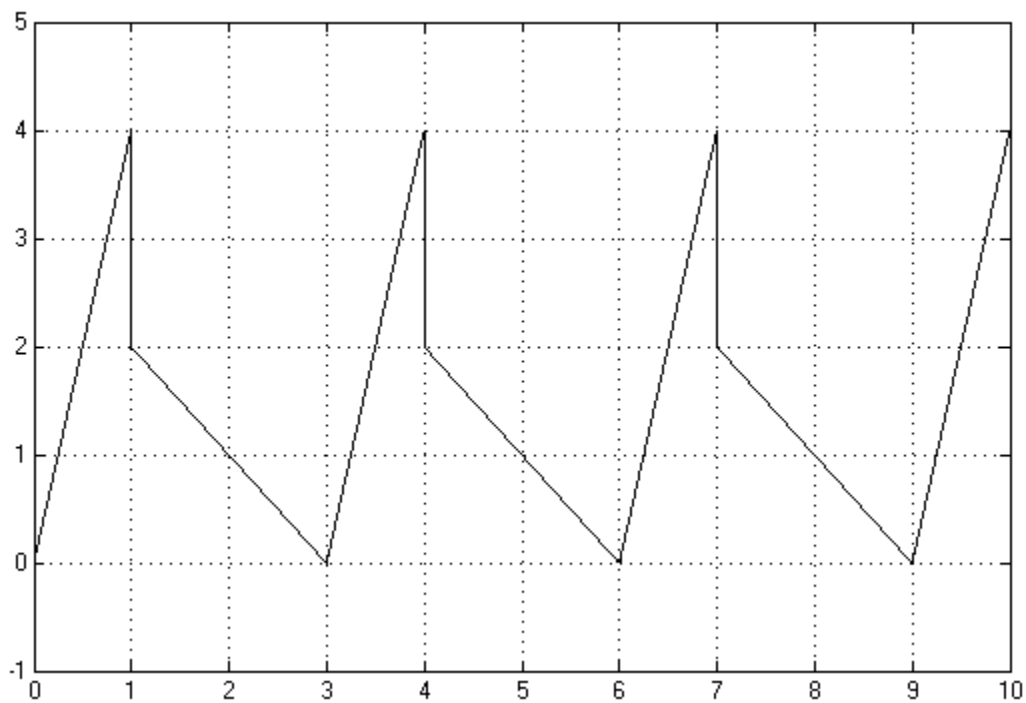
Durations: s

Start output values:

End output values:

OK Cancel Help Apply

The following plot shows the resulting block output.



The signal starts at 0 and consists of two linear segments. The duration of the first segment is 1 second, the segment starts at 0 and ends at 4. The signal is discontinuous, and the end value of the first segment is different than the start value of the second segment, therefore they are connected by a vertical line. The second segment starts at 2, lasts for 2 seconds, and ends at 0, after which the sequence repeats.

Ports

Output

Y — Output physical signal, unitless

physical signal

Output physical signal.

Parameters

Initial output — Output at time zero

0 (default)

The value of the output signal at time zero. The output of the block remains at this value until the simulation time reaches the **Time offset** value.

Time offset — Initial time offset

1 s (default)

The value of the initial time offset, before the start of the repeating sequence. During this time, the output of the block remains at the **Initial output** value.

Signal type — Type of generated signal

Discontinuous (default) | Continuous | Discrete

Select one of the following signal types:

- **Discontinuous** — For each linear segment in the repeating sequence, define its duration, start value, and end value. If the end value of a segment is not the same as the start value of the next segment, they are connected by a vertical line. This is the default method.
- **Continuous** — For each linear segment in the repeating sequence, define its duration and start value. The end value of a segment is the same as the start value of the next segment.
- **Discrete** — For each linear segment in the repeating sequence, define its duration and start value. The end value of a segment is the same as its start value.

Durations — Linear segment durations

1 s (default)

Specify the linear segment durations as a 1-by- n row vector, where n is the number of linear segments in the repeating sequence. n must be no greater than 100.

Example: [1 1] s specifies two linear segments, each lasting 1 second.

Start output values — Start values for each linear segment

1 (default)

Specify the start values of the output signal for each linear segment as a 1-by- n row vector, where n is the number of linear segments in the repeating sequence. The size of the vector must match the size of the **Durations** row vector.

Example: [0 2] specifies that the first of the two linear segments starts at 0, and the second one starts at 2.

End output values — End values for each linear segment

1 (default)

Specify the end values of the output signal for each linear segment as a 1-by- n row vector, where n is the number of linear segments in the repeating sequence. The size of the vector must match the size of the **Durations** row vector.

Dependencies

Enabled only when the **Signal type** parameter is set to **Discontinuous**. For other signal types, the end value of a segment is defined either by the start value of the next segment (**Continuous**) or the start value of the same segment (**Discrete**).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Counter

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2012b

PS RMS Estimator

Measure RMS value of periodic signal

Library: Simscape / Foundation Library / Physical Signals / Periodic Operators



Description

The PS RMS Estimator block measures the root mean square (RMS) value of a periodic signal.

A signal is periodic if it completes a pattern within a measurable time frame, called a period, and repeats that pattern over identical subsequent periods. The signal base frequency is the number of periods per second.

The block accepts a physical signal with one or more AC components present. It outputs an RMS value, where each frequency component is averaged over its time period.

Both the input and the output are physical signals. Untyped physical ports facilitate unit propagation. The output signal has the same physical unit as the input signal.

To obtain meaningful results, run the simulation for at least one full time period of the signal.

Ports

Input

I — Input physical signal, untyped

physical signal

Input physical signal. The signal must be periodic.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

0 — Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Parameters

Base frequency — Periodic signal frequency

60 Hz (default)

Base frequency of the periodic signal.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Constant Offset Estimator | PS Harmonic Estimator (Amplitude, Phase) | PS Harmonic Estimator (Real, Imaginary)

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2015b

PS Round

Round input physical signal toward nearest integer

Library: Simscape / Foundation Library / Physical Signals / Nonlinear Operators



Description

The PS Round block rounds the input physical signal toward the nearest integer:

$$O = \text{round}(I)$$

where

I	Physical signal at the input port
O	Physical signal at the output port

Both the input and the output are physical signals. Untyped physical ports facilitate the signal size propagation. However, arguments passed to rounding functions (such as `round`) must be dimensionless. Therefore, the input signal must have the unit of 1 (unitless), and the output signal is also unitless.

Ports

Input

I – Input physical signal, untyped

physical signal

Input physical signal. Arguments passed to rounding functions (such as `round`) must be dimensionless. Therefore, the input signal does not have to be a scalar but must have the unit of 1 (unitless).

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O – Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Ceil | PS Fix | PS Floor | round

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2012b

PS Saturation

Limit range of physical signal

Library: Simscape / Foundation Library / Physical Signals / Nonlinear Operators



Description

The PS Saturation block imposes upper and lower bounds on a physical signal. When the input signal is within the range specified by the **Lower limit** and **Upper limit** parameters, the input signal passes through unchanged. When the input signal is outside these bounds, the signal is clipped to the upper or lower bound.

When the **Lower limit** and **Upper limit** parameters are set to the same value, the block outputs that value.

Both the input and the output are physical signals. Untyped physical ports facilitate unit propagation. The output signal has the same physical unit as the input signal.

Ports

Input

I — Input physical signal, untyped

physical signal

Input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O — Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Parameters

Upper limit — Signal upper bound

0.5 1 (default)

The upper bound on the input signal. When the input signal to the PS Saturation block is above this value, the output of the block is clipped to this value.

The parameter unit must be commensurate with the unit of the input signal. You can select a unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Lower limit – Signal lower bound

-0.5 1 (default)

The lower bound on the input signal. When the input signal to the PS Saturation block is below this value, the output of the block is clipped to this value.

The parameter unit must be commensurate with the unit of the input signal. You can select a unit from the drop-down list or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Dead Zone

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007a

PS Sign

Output sign of input physical signal

Library: Simscape / Foundation Library / Physical Signals / Nonlinear Operators



Description

The PS Sign block returns the sign of the input physical signal:

- The output is 1 when the input is greater than zero.
- The output is 0 when the input is equal to zero.
- The output is -1 when the input is less than zero.

Both the input and the output are physical signals. Untyped physical ports facilitate the signal size propagation. The output signal is always unitless.

Ports

Input

I — Input physical signal, untyped

physical signal

Input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

0 — Output physical signal, untyped

physical signal

Output physical signal. The signal size is the same as the size of the input signal, but the unit is always 1 (unitless).

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Abs

Topics

“Physical Signal Unit Propagation”

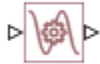
“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007b

PS Signal Specification

Specify size and unit of physical signal

Library: Simscape / Foundation Library / Physical Signals / Utilities



Description

The PS Signal Specification block lets you explicitly specify the size and unit of a physical signal. Use this block when the signal size and unit cannot be determined implicitly, based on port connections in the model.

The input signal at port **I** and the output signal at port **O** have the same size and unit. Both are determined by the value of the **Signal size and unit** parameter. Connect this block to untyped physical signal ports to specify the desired signal type.

Specify the signal size as `zeros(m, n)`. For example, a scalar is `zeros(1, 1)`.

The unit combo box includes a drop-down list of common units, such as those available in the Simulink-PS Converter and the PS-Simulink Converter block dialog boxes. It also contains a field where you can type a unit name or expression. For more information, see “How to Specify Units in Block Dialogs”.

Ports

Input

I — Input physical signal, typed by the block parameter value

physical signal

Input physical signal. The signal type (size and unit) is determined by the **Signal size and unit** parameter value.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O — Output physical signal, typed by the block parameter value

physical signal

Output physical signal. The signal type (size and unit) is determined by the **Signal size and unit** parameter value.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Parameters

Signal size and unit — Size and unit of the input and output physical signal

`zeros(1, 1)` 1 (default)

The size and unit of the physical signal, to be propagated to untyped physical signal ports connected to the block.

The first edit box represents the signal size. Specify it as `zeros(m, n)`. The default size, `zeros(1, 1)`, is scalar.

The second combo box represents the signal unit. By default, the unit is 1 (unitless). You can select a unit from the drop-down list or type the desired unit name, such as `rpm`, or a valid expression, such as `rad/s`. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2019a

PS Subtract

Compute simple subtraction of two input physical signals

Library: Simscape / Foundation Library / Physical Signals / Functions



Description

The PS Subtract block subtracts one physical signal input from another and outputs the difference:

$$O = I_1 - I_2$$

where

I_1	Physical signal at the first input port (marked with the plus sign)
I_2	Physical signal at the second input port (marked with the minus sign)
O	Physical signal at the output port

Physical signals at the two input ports must have commensurate units. The unit at the output port is determined by unit propagation rules. For more information, see “Physical Signal Unit Propagation”.

Ports

Input

I1 – Input physical signal, untyped

physical signal

First input physical signal. In the block icon, this port is marked with a plus sign (+).

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

I2 – Input physical signal, untyped

physical signal

Second input physical signal, to be subtracted from the first. In the block icon, this port is marked with a minus sign (-).

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

O – Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Add | PS Divide | PS Gain | PS Math Function | PS Product

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2007a

PS Switch

Single-pole double-throw switch controlled by external physical signal

Library: Simscape / Foundation Library / Physical Signals / Nonlinear Operators



Description

The PS Switch block compares the value of the physical signal presented at the second (middle) input port to the threshold value:

- If the control input value is greater than or equal to the threshold, the output is connected to the first input. This is the default connection shown in the block icon.
- If the control input value is less than the threshold, the output is connected to the third input.

The second (middle) input port is the control port and it never connects to the output.

All the inputs and the output are physical signals. Untyped physical ports facilitate unit propagation. Physical signals at the first and third input ports must have commensurate units. The unit at the output port is determined by unit propagation rules. For more information, see “Physical Signal Unit Propagation”.

Ports

Input

I1 — Input physical signal, untyped

physical signal

First input physical signal. This signal is output if the control signal value at port **I2** is greater than or equal to the **Threshold** parameter value.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

I2 — Control physical signal, untyped

physical signal

Second input physical signal. This is the control signal for the switch.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

I3 — Input physical signal, untyped

physical signal

Third input physical signal. This signal is output if the control signal value at port **I2** is less than the **Threshold** parameter value.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

0 — Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Parameters

Threshold — Threshold value for opening and closing the switch

0.5 1 (default)

The threshold value for opening and closing the switch. If the control physical signal, presented at the second (middle) input port **I2**, is greater than or equal to this value, then the output is connected to the first input. Otherwise, the output is connected to the third input.

The parameter unit must be commensurate with the unit of the control signal at port **I2**. You can select a unit from the drop-down list or type the desired unit name or expression. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Switch

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2009b

PS Terminator

Terminator for unconnected physical signal outputs

Library: Simscape / Foundation Library / Physical Signals / Sinks

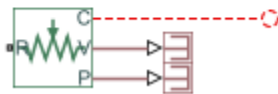


Description

Use the PS Terminator block to cap physical signal output ports that do not connect to other blocks.

Unlike conserving ports in physical modeling, or Simulink output ports, unconnected physical signal output ports do not generate warnings. However, you can use a PS Terminator block for clarity, to indicate that the signal was not inadvertently left unconnected.

The block icon size allows you to connect the PS Terminator block to an open port without rerouting the adjacent lines:



Ports

Input

I — Input physical signal, untyped

physical signal

Input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Compatibility Considerations

Unit Propagation

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Adiabatic Cup | Cap (TL) | Hydraulic Cap | Open Circuit | Perfect Insulator | Rotational Free End | Terminator | Translational Free End

Topics

“Physical Signal Unit Propagation”

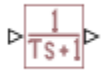
“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2014b

PS Transfer Function

Model low-pass and lead-lag filters

Library: Simscape / Foundation Library / Physical Signals / Linear Operators



Description

The PS Transfer Function block lets you model first-order low-pass filters or lead-lag filters. Modeling both the plant and controller in a Simscape network improves simulation efficiency. The PS Transfer Function block lets you easily parameterize a filter in the desired configuration:

- **Lag** — Model a first-order low-pass filter. You have to provide the lag time constant and the initial output as block parameters.
- **Lead-lag** — Model a lead-lag filter. You have to provide the lead time constant, the lag time constant, and the initial output as block parameters.

The block icon changes depending on the value of the **Transfer function type** parameter.

Restriction Type	Block Icon
Lag	
Lead-lag	

Untyped physical ports facilitate unit propagation. The output signal has the same physical unit as the input signal.

Assumptions and Limitations

- The block assumes unity low-frequency gain. To model non-unity gain, connect a PS Gain block in series with the PS Transfer Function block.

Ports

Input

U — Input physical signal, untyped

physical signal

Input physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Output

Y — Output physical signal, untyped

physical signal

Output physical signal.

The port name is not visible in the block icon, but you can see this name in the underlying source file (accessible by clicking the **Source code** link in the block dialog box).

Parameters

Transfer function type — Specify type of low-pass filter

Lag (default) | Lead-lag

Select the type of the low-pass filter:

- Lag — Model a first-order low-pass filter.
- Lead-lag — Model a lead-lag filter.

Lag time constant, T — Lag time constant for first-order low-pass filter

1 s (default) | positive scalar

The lag time constant for first-order low-pass filter.

Dependencies

Enabled when the **Transfer function type** parameter is set to Lag.

Lead time constant, T1 — Lead time constant for lead-lag filter

2 s (default) | positive scalar

The lead time constant for lead-lag filter.

Dependencies

Enabled when the **Transfer function type** parameter is set to Lead-lag.

Lag time constant, T2 — Lag time constant for lead-lag filter

1 s (default) | positive scalar

The lag time constant for lead-lag filter.

Dependencies

Enabled when the **Transfer function type** parameter is set to Lead-lag.

Initial output — Output signal value and unit at the start of simulation

0 1 (default) | scalar, vector, or matrix

The output signal value and unit at the beginning of a simulation run. The first edit box represents the signal value. This value can be a scalar, vector, or matrix, matching the size of the output signal at port Y.

The second combo box represents the signal unit, which must be commensurate with the unit of the output signal at port **Y**. By default, the unit is 1 (unitless). You can select a unit from the drop-down list or type the desired unit name, such as `rpm`, or a valid expression, such as `rad/s`. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Integrator

Topics

“Physical Signal Unit Propagation”

Introduced in R2020a

PS Uniform Random Number

Generate uniformly distributed random numbers for physical modeling

Library: Simscape / Foundation Library / Physical Signals / Sources



Description

The PS Random Number block generates uniformly distributed random numbers over the interval you specify. To generate normally (Gaussian) distributed random numbers, use the PS Random Number block.

The block behavior is the same as the Simulink Uniform Random Number block (except that it generates a physical signal rather than a Simulink signal).

You have an option to specify an initial time offset as part of the **Sample time** parameter. In this case, the block outputs 0 until the simulation time reaches the *offset* value, at which point the random sequence starts.

PS Uniform Random Number blocks that use the same seed and parameters generate a repeatable sequence. The seed resets to the specified value each time a simulation starts.

Ports

Output

Y — Output physical signal, unitless

physical signal

Output physical signal.

Parameters

Minimum — Minimum output value

0 (default)

The minimum output value generated by the block.

Maximum — Maximum output value

1 (default)

The maximum output value generated by the block.

Seed — Starting seed

0 (default)

The starting seed for the random number generator. Output is repeatable for a given seed. The seed must be an integer in the range of 0 to $(2^{32} - 1)$.

Sample time — Sample time interval

1 s (default)

The value of the time *step* interval. The default *step* value is 1 s. To specify an initial time offset, enter the parameter value as [*step*, *offset*], otherwise the *offset* value is assumed to be 0. The offset must be less than the step size.

Compatibility Considerations**Unit Propagation**

Behavior changed in R2019a

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

PS Random Number

Topics

“Physical Signal Unit Propagation”

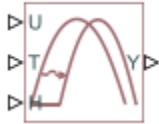
“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2013a

PS Variable Delay

Delay input physical signal by variable time

Library: Simscape / Foundation Library / Physical Signals / Delays



Description

The PS Variable Delay block generates the output physical signal, **Y**, by delaying the input physical signal, **U**:

$$Y = U(t - \tau)$$

where τ is the delay time, which can vary throughout the simulation. You supply the delay time as a signal through the input port **T**.

For the initial time interval, when $t \leq \text{StartTime} + \tau$, the block outputs the value of the signal supplied through the input port **H**.

Note

- When simulating a model that contains blocks with delays, memory allocation for storing the data history is controlled by the **Delay memory budget [kB]** parameter in the Solver Configuration block. If this budget is exceeded, simulation errors out. You can adjust this parameter value based on your available memory resources.
 - For recommendation on how to linearize a model that contains blocks with delays, see “Linearizing with Simulink Linearization Blocks”.
-

Untyped physical ports facilitate unit propagation. The output signal at port **Y** has the same physical unit as the input signal at port **U**. The signal unit at port **H** must be commensurate with the unit of the output signal.

The “Variable Transport Delay” example shows how you can model a variable transport delay using a custom block authored in Simscape language.

Ports

Input

U — Input physical signal, untyped
physical signal

Input physical signal that is being delayed.

T – Delay time, s

physical signal

Input physical signal that supplies the delay time.

H – Output during initial time interval, untyped

physical signal

Input physical signal that supplies the value to be output during the initial time interval, when time since the start of simulation is less than or equal to the delay time. The signal unit must be commensurate with the unit of the output signal at port **Y**.

Output**Y – Output physical signal, untyped**

physical signal

Output physical signal.

Parameters**Maximum delay time – Delay time for the signal**

10 s (default)

The upper limit for the delay time. Exceeding the maximum delay time during simulation results in a runtime error. The parameter value must be positive.

Compatibility Considerations**Unit Propagation***Behavior changed in R2019a*

Prior to R2019a, this block did not propagate physical units.

If your model contains legacy blocks without unit propagation, use the Upgrade Advisor to upgrade your blocks to the latest version. For more information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

PS Constant Delay

Topics

“Physical Signal Unit Propagation”

“Upgrading Models with Legacy Physical Signal Blocks”

Introduced in R2012a

PS-Simulink Converter

Convert physical signal into Simulink output signal





Library

Utilities

Description

The PS-Simulink Converter block converts a physical signal into a Simulink output signal. Use this block to connect outputs of a Physical Network diagram to Simulink scopes or other Simulink blocks.

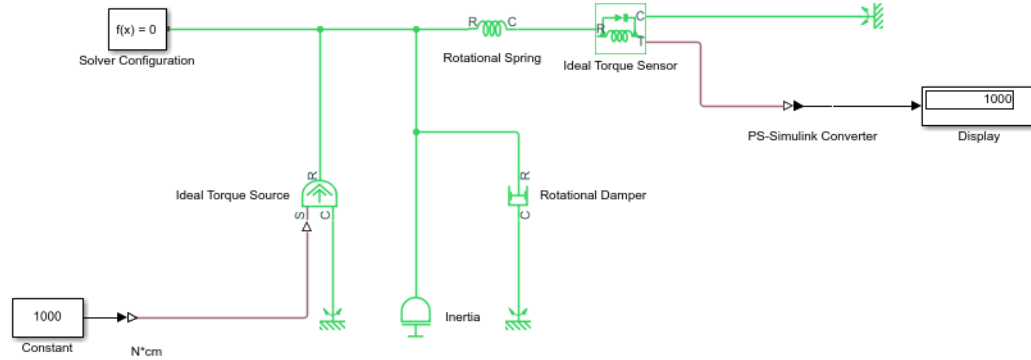
To convey signal conversion while taking up minimal canvas space, the block icon changes dynamically based on whether it is connected to other blocks.

When Block Is...	Block Icon
Unconnected	
Connected to other blocks	

The **Output signal unit** parameter lets you specify the desired units for the output signal. These units must be commensurate with the units of the input physical signal coming into the block. If you specify a desired output unit, the block applies a gain equal to the conversion factor before outputting the Simulink signal. For example, if the input physical signal coming into the block is displacement, in meters, and you set **Output signal unit** to mm, the block multiplies the value of the input signal by 10e3 before outputting it. If the output signal unit is the same as the input signal unit, no gain is applied.

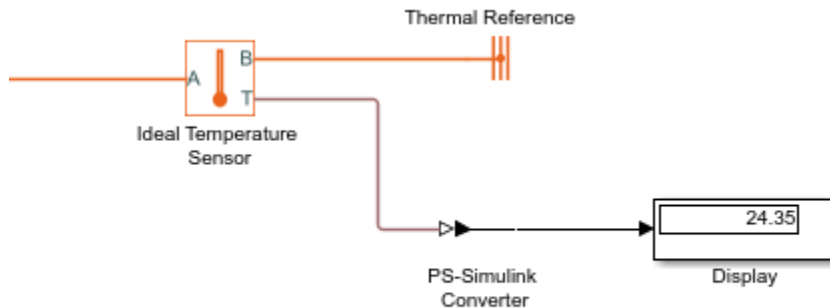
The default value of the **Output signal unit** parameter, `inherit`, automatically sets the unit at the block output port to match the unit of the input physical signal coming into the block, based on unit propagation rules. This way, you can easily connect a PS-Simulink Converter block to any signal, without worrying about setting the commensurate output unit.

In the diagram below, the input signal for the PS-Simulink Converter block is torque in N*m, and if you do not specify the output signal unit, the Display block shows the value of 10. If you change the **Output signal unit** parameter value in the PS-Simulink Converter block to N*cm, the torque value in the Display block changes to 1000, as shown in the diagram.



When the output signal is related to thermodynamic variables and contains units of temperature, you must decide whether affine conversion needs to be applied. For more information, see “When to Apply Affine Conversion”. Usually, if the output signal represents a relative temperature, that is, a change in temperature, you need to apply linear conversion, $\Delta T_{new} = L * \Delta T_{old}$ (the default method). However, if the output signal represents an absolute temperature, you need to apply affine conversion, $T_{new} = L * T_{old} + O$.

In the following diagram, the Display block shows the room temperature. If you want to display it in degrees Celsius, open the PS-Simulink Converter block, type degC in the **Output signal unit** field, and select the **Apply affine conversion** check box. The display reading is 24.35. However, if you leave the **Apply affine conversion** check box clear, the Display block would show 297.5.



Parameters

Output signal unit

Specify the desired units for the output signal. These units must be commensurate with the units of the input physical signal coming into the block. The system compares the units you specified with the actual units of the input physical signal and applies a gain equal to the conversion factor before outputting the Simulink signal. You can select a unit from the drop-down list, or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”. The default value is inherit, which means that the output unit matches the unit at the block input port and no gain is applied.

Apply affine conversion

This check box is applicable only for units that can be converted either with or without an affine offset, such as thermal units. For more information, see “Thermal Unit Conversions”.

Restricted Parameters

When your model is in Restricted editing mode, you cannot modify any of the block parameters.

Ports

The block has a physical signal input port, located on its left side, and a Simulink output port, located on its right side (in the block default orientation).

See Also

Simulink-PS Converter

Topics

“Working with Simulink Units”

Introduced in R2007a

Radiative Heat Transfer

Heat transfer by radiation



Library

Thermal Elements

Description

The Radiative Heat Transfer block represents a heat transfer by radiation between two bodies. The transfer is governed by the Stefan-Boltzmann law and is described with the following equation:

$$Q = k \cdot A \cdot (T_A^4 - T_B^4)$$

where

Q	Heat flow
k	Radiation coefficient
A	Emitting body surface area
T_A, T_B	Temperatures of the bodies

The radiation coefficient is determined by geometrical shapes, dimensions, and surface emissivity. For example, the radiation constant for the heat transfer between two parallel plates is computed as

$$k = \frac{\sigma}{\frac{1}{\varepsilon_1} + \frac{1}{\varepsilon_2} - 1}$$

where

σ	Stefan-Boltzmann constant
$\varepsilon_1, \varepsilon_2$	Surface emissivity for the emitting and receiving plate, respectively

Similarly, the radiation coefficient for concentric cylinders is determined with the formula

$$k = \frac{\sigma}{\frac{1}{\varepsilon_1} + \frac{1 - \varepsilon_2}{\varepsilon_2} \frac{r_1}{r_2}}$$

where r_1 and r_2 are the emitting and receiving cylinder radii, respectively. Reference [1 on page 1-549] contains formulas for a wide variety of shapes.

Connections A and B are thermal conserving ports associated with the emitting and receiving bodies, respectively. The block positive direction is from port A to port B. This means that the heat flow is positive if it flows from A to B.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Area

Radiating body area of heat transfer. The default value is 0.0001 m^2 .

Radiation coefficient

Radiation coefficient of the two bodies, based on their geometrical shapes, dimensions, and surface emissivity. See [1 on page 1-549] for more information. The default value is $4\text{e-}8 \text{ W/m}^2/\text{K}^4$.

Ports

The block has the following ports:

A

Thermal conserving port associated with body A.

B

Thermal conserving port associated with body B.

References

[1] Siegel, R. and J.R. Howell. *Thermal Radiation Heat Transfer*. New York: Taylor and Francis, 2002.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Conductive Heat Transfer | Convective Heat Transfer

Introduced in R2007b

Reluctance

Magnetic reluctance



Library

Magnetic Elements

Description

The Reluctance block models a magnetic reluctance, that is, a component that resists flux flow. The ratio of the magnetomotive force (mmf) across the component to the resulting flux that flows through the component is constant, and the ratio value is defined as the reluctance. Reluctance depends on the geometry of the section being modeled.

The block is based on the following equations:

$$MMF = \Phi \cdot \mathfrak{R}$$

$$\mathfrak{R} = \frac{g}{\mu_0 \cdot \mu_r \cdot A}$$

where

MMF	Magnetomotive force (mmf) across the component
Φ	Flux through the component
\mathfrak{R}	Reluctance
g	Thickness of the section being modeled, or length of air gap
μ_0	Permeability constant
μ_r	Relative permeability of the material
A	Cross-sectional area of the section being modeled

Connections N and S are magnetic conserving ports. The mmf across the reluctance is given by $MMF(N) - MMF(S)$, and the sign of the flux is positive when flowing through the device from N to S.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Thickness or length of section or gap

Thickness of the section being modeled, or length of air gap. The default value is 0.001 m.

Cross-sectional area

Area of the section being modeled. The default value is 0.01 m².

Relative permeability of material

Relative permeability of the section material. The default value is 1.

Ports

The block has the following ports:

N

Magnetic conserving port associated with the block North terminal.

S

Magnetic conserving port associated with the block South terminal.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

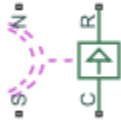
See Also

Fundamental Reluctance | Variable Reluctance

Introduced in R2010a

Reluctance Force Actuator

Magnetomotive device based on reluctance force



Library

Magnetic Elements

Description

The Reluctance Force Actuator block models a generic magnetomotive device based on reluctance force.

The block is based on the following equations:

$$F = -0.5 \cdot \Phi^2 \cdot \frac{d\mathfrak{R}}{dx}$$

$$\mathfrak{R}(x) = \frac{x}{\mu_0 \cdot \mu_r \cdot A}$$

$$u = dx$$

where

F	Reluctance force
Φ	Flux in the magnetic circuit
\mathfrak{R}	Reluctance
x	Thickness or length of the air gap
μ_0	Permeability constant
μ_r	Relative permeability of the material
A	Cross-sectional area of the section being modeled
u	Velocity

Connections N and S are magnetic conserving ports, and connections C and R are mechanical translational conserving ports. The magnetic force produced by the actuator acts to close the gap, therefore the resulting force is negative when it acts from C to R.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Basic Assumptions and Limitations

- The current excitation in the system is constant.
- Only axial reluctance is modeled.

Parameters

Initial air gap

Thickness or length of air gap at the beginning of simulation. The default value is 2 mm.

Minimum air gap

Minimal value of air gap, with the reluctance force acting to close the air gap. The parameter value has to be greater than 0. The default value is $1e-4$ mm.

Cross-sectional area

Area of the section being modeled. The default value is 0.01 m^2 .

Relative permeability of material

Relative permeability of the section material. The default value is 1.

Contact stiffness

Stiffness that models the hard stop at the minimum air gap position. The default value is $10e6$ N/m.

Contact damping

Damping that models the hard stop at the minimum air gap position. The default value is 500 N/(m/s).

Ports

The block has the following ports:

N

Magnetic conserving port associated with the block North terminal.

S

Magnetic conserving port associated with the block South terminal.

R

Mechanical translational conserving port associated with the rod.

C

Mechanical translational conserving port associated with the case.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Fundamental Reluctance | Reluctance | Variable Reluctance

Introduced in R2010a

Reservoir (2P)

Two-phase fluid reservoir at constant temperature and pressure



Library

Two-Phase Fluid/Elements

Description

The Reservoir (2P) block sets pressure and temperature boundary conditions in a two-phase fluid network. The reservoir is assumed infinite in size, causing its pressure and specific internal energy to remain constant.

Port A represents the reservoir inlet. The flow resistance between port A and the reservoir interior is assumed negligible. The pressure at port A is therefore equal to that inside the reservoir.

The specific internal energy at the reservoir inlet depends on the flow direction. If fluid leaves the reservoir, the fluid at port A is at the specific internal energy of the reservoir. The reservoir acts as an energy source. If fluid enters the reservoir, the fluid at port A is at the specific internal energy upstream of the reservoir. The reservoir acts as an energy sink.

This block also serves as a reference connection for the Pressure & Internal Energy Sensor (2P) block. In this case, the measured pressure and specific internal energy are relative to the reservoir pressure and specific internal energy.

Assumptions and Limitations

- The pressure and specific internal energy of the fluid are assumed constant in the reservoir.
- The flow resistance between port A and the reservoir interior is negligible. Pressure is the same at port A and in the reservoir interior.

Parameters

Reservoir pressure specification

Pressurization characteristics of the reservoir. Select **Atmospheric pressure** to set the reservoir pressure to the atmospheric pressure specified in the Two-Phase Fluid Properties (2P) block. Select **Specified pressure** to set the reservoir pressure to a different value. The default setting is **Atmospheric pressure**.

Reservoir pressure

Absolute pressure inside the reservoir. This pressure is fixed during simulation. This parameter is active only when the **Reservoir pressure specification** parameter is set to **Specified pressure**. The default value, 0.101325 MPa, corresponds to atmospheric pressure at mean sea level.

Reservoir energy specification

Thermodynamic variable to use in defining the conditions of the reservoir. The default setting is Temperature.

Reservoir temperature

Absolute temperature inside the reservoir. This temperature is fixed during simulation. This parameter is active when the **Reservoir energy specification** option is set to Temperature. The default value is 293.15 K.

Reservoir vapor quality

Mass fraction of vapor in the reservoir. This value is fixed during simulation. This parameter is active when the **Reservoir energy specification** option is set to Vapor quality. The default value is 0.5.

Reservoir vapor void fraction

Volume fraction of vapor in the reservoir. This value is fixed during simulation. This parameter is active when the **Reservoir energy specification** option is set to Vapor void fraction. The default value is 0.5.

Reservoir specific enthalpy

Specific enthalpy of the fluid in the reservoir. This value is fixed during simulation. This parameter is active when the **Reservoir energy specification** option is set to Specific enthalpy. The default value is 1500 kJ/kg.

Reservoir specific internal energy

Specific internal energy of the fluid in the reservoir. This value is fixed during simulation. This parameter is active when the **Reservoir energy specification** option is set to Specific internal energy. The default value is 1500 kJ/kg.

Cross-sectional area at port A

Flow area of the reservoir inlet, represented by port **A**. The flow area must be greater than zero. The default value is 0.01 m².

Ports

The block has one two-phase fluid conserving port, **A**, representing the reservoir inlet.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Constant Volume Chamber (2P)

Introduced in R2015b

Reservoir (G)

Boundary conditions for gas network at constant pressure and temperature

Library: Simscape / Foundation Library / Gas / Elements



Description

The Reservoir (G) block represents an infinite reservoir at fixed pressure and temperature. The reservoir and its inlet can be at atmospheric pressure or at a specified pressure. Port **A**, a gas conserving port, represents the reservoir inlet.

The volume of gas inside the reservoir is assumed infinite. Therefore, the flow is assumed quasi-steady.

Gas enters and leaves the reservoir at the reservoir pressure, but its temperature is determined by the direction of gas flow. If gas flows into the reservoir, its temperature is determined by the gas network upstream. The reservoir acts as a heat sink. If gas flows out of the reservoir, its temperature equals that of the reservoir. The reservoir acts as a heat source.

This block also functions as a reference point for pressure and temperature measurements in a gas network. These measurements are relative to the reservoir pressure and temperature, respectively. Connect the reservoir to port **B** of a Pressure and Temperature Sensor (G) block to measure relative pressure and temperature of a node connected to the **A** port of the sensor.

Assumptions and Limitations

- Reservoir pressure and temperature are constant.
- Gas in the reservoir is quasi-steady.

Ports

Conserving

A — Reservoir inlet

gas

Gas conserving port associated with the reservoir inlet.

Parameters

Reservoir pressure specification — Select specification method for reservoir pressure

Atmospheric pressure (default) | Specified pressure

Select a specification method for the reservoir pressure:

- **Atmospheric pressure** — Use the atmospheric pressure, specified by the Gas Properties (G) block connected to the circuit.

- Specified pressure — Specify a value by using the **Reservoir pressure** parameter.

Reservoir pressure — Pressure in the reservoir

0.101325 MPa (default)

Enter the desired pressure in the reservoir. This pressure remains constant during simulation.

Dependencies

Enabled when the **Reservoir pressure specification** parameter is set to Specified pressure.

Reservoir temperature — Temperature in the reservoir

293.15 K (default)

Enter the desired temperature in the reservoir. This temperature remains constant during simulation.

Cross-sectional area at port A — Area normal to flow path at the reservoir inlet

0.01 m² (default)

The cross-sectional area of the reservoir inlet, in the direction normal to gas flow path.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Reservoir (G)

Topics

“Modeling Gas Systems”

Introduced in R2016b

Reservoir (IL)

Isothermal liquid reservoir at constant or time-varying pressure



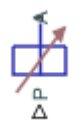
Library: Simscape / Foundation Library / Isothermal Liquid / Elements



Description

The Reservoir (IL) block represents an infinite reservoir of isothermal liquid. Use this block to set the boundary condition in an isothermal liquid network. The liquid in the reservoir can be at a constant pressure, either atmospheric or specified by a block parameter. You can also control the pressure inside the reservoir by using the physical signal input port **P**. Port **A**, an isothermal liquid conserving port, represents the reservoir inlet.

The block icon changes depending on the values of the **Reservoir type** and **Reservoir pressure specification** parameters.

Reservoir Type	Reservoir Pressure Specification	Block Icon
Constant	Atmospheric pressure	
	Specified pressure	
Controlled		

Assumptions and Limitations

- The volume of liquid inside the reservoir is assumed infinite. Therefore, the flow is assumed quasi-steady.
- There is no flow resistance between port **A** and the reservoir interior.

Ports

Input

P — Pressure control signal, Pa
physical signal

Physical signal port that provides the reservoir pressure control signal. The signal saturates when its value drops below the minimum valid pressure specified by the Isothermal Liquid Properties (IL) block connected to the circuit.

Dependencies

This port is visible only if you set the **Reservoir type** parameter to `Controlled`.

Conserving**A — Reservoir inlet**

isothermal liquid

Isothermal liquid conserving port associated with the reservoir inlet.

Parameters**Reservoir type — Specify whether reservoir pressure can change during simulation**

`Constant` (default) | `Controlled`

Select whether the reservoir pressure can change during simulation:

- `Constant` — The reservoir pressure remains constant during simulation. Port **P** is hidden.
- `Controlled` — The input physical signal at port **P** specifies the reservoir pressure, which can vary during simulation.

Reservoir pressure specification — Specification method for reservoir pressure

`Atmospheric pressure` (default) | `Specified pressure`

Specification method for the reservoir pressure:

- `Atmospheric pressure` — Use the atmospheric pressure specified by the Isothermal Liquid Properties (IL) block connected to the circuit.
- `Specified pressure` — Specify a value by using the **Reservoir pressure** parameter.

Dependencies

Enabled when the **Reservoir type** parameter is set to `Constant`.

Reservoir pressure — Pressure in reservoir

`0.101325` MPa (default) | positive scalar

Desired pressure in the reservoir. This pressure remains constant during simulation. The specified pressure cannot be greater than the atmospheric pressure specified by the Isothermal Liquid Properties (IL) block connected to the circuit.

Dependencies

Enabled when the **Reservoir pressure specification** parameter is set to `Specified pressure`.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Constant Volume Chamber (IL)

Topics

“Modeling Isothermal Liquid Systems”

“Isothermal Liquid Modeling Options”

Introduced in R2020a

Reservoir (MA)

Boundary conditions for moist air network at constant pressure, temperature, moisture, and trace gas levels

Library: Simscape / Foundation Library / Moist Air / Elements



Description

The Reservoir (MA) block sets boundary conditions in a moist air network. The volume of moist air inside the reservoir is assumed infinite. Therefore, the flow is assumed quasi-steady. Moist air leaves the reservoir at the reservoir pressure, temperature, specific humidity, and trace gas mass fraction. Moist air enters the reservoir at the reservoir pressure, but the temperature, specific humidity, and trace gas mass fraction are determined by the moist air network upstream.

You specify the reservoir pressure, temperature, amount of moisture, and amount of trace gas by entering block parameter values. Block parameters related to trace gas are ignored if **Trace gas model** in the Moist Air Properties (MA) block is set to None.

You can specify moisture as one of:

- Relative humidity, φ_w
- Specific humidity, x_w
- Water vapor mole fraction, y_w
- Humidity ratio, r_w

You can specify trace gas as one of:

- Trace gas mass fraction, x_g
- Trace gas mole fraction, y_g

These moisture and trace gas quantities are related to each other as follows:

$$\varphi_w = \frac{y_w P}{p_{ws}}$$

$$y_w = \frac{x_w R_w}{R}$$

$$r_w = \frac{x_w}{1 - x_w}$$

$$y_g = \frac{x_g R_g}{R}$$

$$x_a + x_w + x_g = 1$$

$$R = x_a R_a + x_w R_w + x_g R_g$$

where:

- p is pressure.
- R is specific gas constant.

Subscripts a , w , and g indicate the properties of dry air, water vapor, and trace gas, respectively. Subscript ws indicates water vapor at saturation.

Ports

Conserving

A — Reservoir inlet

moist air

Moist air conserving port associated with the reservoir inlet.

Parameters

Reservoir pressure specification — Select specification method for reservoir pressure

Atmospheric pressure (default) | Specified pressure

Select a specification method for the reservoir pressure:

- **Atmospheric pressure** — Use the atmospheric pressure, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified pressure** — Specify a value by using the **Reservoir pressure** parameter.

Reservoir pressure — Pressure in the reservoir

0.101325 MPa (default)

Enter the desired pressure in the reservoir. This pressure remains constant during simulation.

Dependencies

Enabled when the **Reservoir pressure specification** parameter is set to **Specified pressure**.

Reservoir temperature specification — Select specification method for reservoir temperature

Atmospheric temperature (default) | Specified temperature

Select a specification method for the reservoir temperature:

- **Atmospheric temperature** — Use the atmospheric temperature, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified temperature** — Specify a value by using the **Reservoir temperature** parameter.

Reservoir temperature — Temperature in the reservoir

293.15 K (default)

Enter the desired temperature in the reservoir. This temperature remains constant during simulation.

Dependencies

Enabled when the **Reservoir temperature specification** parameter is set to **Specified temperature**.

Reservoir moisture specification — Select the moisture property

Relative humidity (default) | Specific humidity | Mole fraction | Humidity ratio

Select a moisture property:

- Relative humidity — Specify moisture using the relative humidity.
- Specific humidity — Specify moisture using the specific humidity.
- Mole fraction — Specify moisture using the water vapor mole fraction.
- Humidity ratio — Specify moisture using the humidity ratio.

Reservoir relative humidity — Relative humidity in the reservoir

0.5 (default)

Enter the desired relative humidity in the reservoir. This value remains constant during simulation.

DependenciesEnabled when the **Reservoir moisture specification** parameter is set to Relative humidity.**Reservoir specific humidity — Specific humidity in the reservoir**

0.01 (default)

Enter the desired specific humidity in the reservoir. This value remains constant during simulation.

DependenciesEnabled when the **Reservoir moisture specification** parameter is set to Specific humidity.**Reservoir water vapor mole fraction — Water vapor mole fraction in the reservoir**

0.01 (default)

Enter the desired water vapor mole fraction in the reservoir. This value remains constant during simulation.

DependenciesEnabled when the **Reservoir moisture specification** parameter is set to Mole fraction.**Reservoir humidity ratio — Humidity ratio in the reservoir**

0.01 (default)

Enter the desired relative humidity in the reservoir. This value remains constant during simulation.

DependenciesEnabled when the **Reservoir moisture specification** parameter is set to Humidity ratio.**Reservoir trace gas specification — Select the trace gas property**

Mass fraction (default) | Mole fraction

Select a trace gas property:

- Mass fraction — Specify the trace gas mass fraction.
- Mole fraction — Specify the trace gas mole fraction.

Reservoir trace gas mass fraction — Trace gas mass fraction in the reservoir

0.001 (default)

Enter the desired trace gas mass fraction in the reservoir. This value remains constant during simulation.

Dependencies

Enabled when the **Reservoir trace gas specification** parameter is set to Mass fraction.

Reservoir trace gas mole fraction — Trace gas mole fraction in the reservoir

0.001 (default)

Enter the desired trace gas mole fraction in the reservoir. This value remains constant during simulation.

Dependencies

Enabled when the **Reservoir trace gas specification** parameter is set to Mole fraction.

Relative humidity at saturation — Relative humidity above which condensation occurs

1 (default)

Relative humidity above which condensation occurs. Amount of moisture in the reservoir must be less than saturation.

Cross-sectional area at port A — Area normal to flow path at the reservoir inlet

0.01 m² (default)

The cross-sectional area of the reservoir inlet.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Reservoir (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Reservoir (TL)

Thermal liquid reservoir at constant temperature and pressure

Library: Simscape / Foundation Library / Thermal Liquid / Elements



Description

The Reservoir (TL) block represents an infinite reservoir at fixed pressure and temperature. The reservoir and its inlet can be at atmospheric pressure or at a specified pressure. Port **A**, a thermal liquid conserving port, represents the reservoir inlet.

The inlet temperature depends on the direction of liquid flow. If liquid flows into the reservoir, the inlet temperature equals that of the upstream liquid and the reservoir acts as a heat sink. If liquid flows out of the reservoir, the inlet temperature equals that of the reservoir and the reservoir acts as a heat source.

To ensure a smooth temperature change at the reservoir inlet during liquid flow reversal, the block includes heat conduction along a length equal to the effective diameter of the inlet pipe. This diameter is a function of the specified cross-sectional area of the inlet pipe.

Assumptions and Limitations

- Reservoir pressure and temperature are constant.

Ports

Conserving

A — Reservoir inlet

thermal liquid

Thermal liquid conserving port associated with the reservoir inlet.

Parameters

Reservoir pressure specification — Specification method for reservoir pressure

Atmospheric pressure (default) | Specified pressure

Specification method for the reservoir pressure:

- **Atmospheric pressure** — Use the atmospheric pressure specified by a Thermal Liquid Settings (TL) or Thermal Liquid Properties (TL) block connected to the circuit.
- **Specified pressure** — Specify a value by using the **Reservoir pressure** parameter.

Reservoir pressure — Pressure in reservoir

0.101325 MPa (default)

Desired pressure in the reservoir. This pressure remains constant during simulation.

Dependencies

Enabled when the **Reservoir pressure specification** parameter is set to Specified pressure.

Reservoir temperature — Temperature in reservoir

293.15 K (default)

Desired temperature in the reservoir. This temperature remains constant during simulation.

Cross-sectional area at port A — Area normal to flow path at reservoir inlet

0.01 m² (default)

Cross-sectional area of the reservoir inlet pipe. The block uses this area to determine the characteristic length of the pipe along which heat conduction occurs.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Constant Volume Chamber (TL) | Controlled Reservoir (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2015a

Resistor

Linear resistor in electrical systems



Library

Electrical Elements

Description

The Resistor block models a linear resistor, described with the following equation:

$$V = I \cdot R$$

where

V	Voltage
I	Current
R	Resistance

Connections + and - are conserving electrical ports corresponding to the positive and negative terminals of the resistor, respectively. By convention, the voltage across the resistor is given by $V(+)$ - $V(-)$, and the sign of the current is positive when flowing through the device from the positive to the negative terminal. This convention ensures that the power absorbed by a resistor is always positive.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see "Set Priority and Initial Target for Block Variables".

Parameters

Resistance

Resistance, in ohms. The default value is 1 Ω .

Ports

The block has the following ports:

+

Electrical conserving port associated with the resistor positive terminal.

-

Electrical conserving port associated with the resistor negative terminal.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Thermal Resistor | Variable Resistor

Introduced in R2007a

Rotary Pneumatic Piston Chamber

Rotational pneumatic piston chamber based on ideal gas law



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Rotary Pneumatic Piston Chamber block models a pneumatic rotary piston chamber based on the ideal gas law and assuming constant specific heats. Use this model as a building block for pneumatic rotational actuators. The piston can generate torque in one direction only, and the direction is set by the **Chamber orientation** parameter.

The continuity equation for the network representation of the piston chamber is

$$G = \frac{V_0 + D \cdot \theta}{RT} \left(\frac{dp}{dt} - \frac{p}{T} \frac{dT}{dt} \right) + \frac{D}{RT} \cdot p \cdot \frac{d\theta}{dt}$$

where

G	Mass flow rate at input port
V_0	Initial chamber volume
D	Piston displacement (volume per unit angle)
Θ	Piston angle
p	Absolute pressure in the chamber
R	Specific gas constant
T	Absolute gas temperature
t	Time

The energy equation is

$$q = \frac{c_v}{R} (V_0 + D \cdot \theta) \frac{dp}{dt} + \frac{c_p \cdot D}{R} p \frac{d\theta}{dt} - q_w$$

where

q	Heat flow due to gas inflow in the chamber (through the pneumatic port)
q_w	Heat flow through the chamber walls (through the thermal port)
c_v	Specific heat at constant volume
c_p	Specific heat at constant pressure

The torque equation is

$$\tau = p \cdot D$$

Port A is the pneumatic conserving port associated with the chamber inlet. Port H is a thermal conserving port through which heat exchange with the environment takes place. Ports C and R are mechanical rotational conserving ports associated with the piston case and rod, respectively. The gas flow and the heat flow are considered positive if they flow into the chamber.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Basic Assumptions and Limitations

- The gas is ideal.
- Specific heats at constant pressure and constant volume, c_p and c_v , are constant.

Parameters

Displacement

Specify the effective piston displacement, as volume per unit angle. The default value is $.001 \text{ m}^3/\text{rad}$.

Initial angle

Specify the initial piston angle. The default value is 0 .

Dead volume

Specify the volume of gas in the chamber at zero piston position. The default value is $1e-5 \text{ m}^3$.

Chamber orientation

Specify the direction of torque generation. The piston generates torque in a positive direction if this parameter is set to 1 (the default). If you set this parameter to 2, the piston generates torque in a negative direction.

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the chamber inlet.

H

Thermal conserving port through which heat exchange with the environment takes place.

R

Mechanical rotational conserving port associated with the piston (rod).

C

Mechanical rotational conserving port associated with the reference (case).

See Also

Constant Volume Pneumatic Chamber | Pneumatic Piston Chamber

Introduced in R2009b

Rotational Damper

Viscous damper in mechanical rotational systems



Library

Mechanical Rotational Elements

Description

The Rotational Damper block represents an ideal mechanical rotational viscous damper described with the following equations:

$$T = D \cdot \omega$$

$$\omega = \omega_R - \omega_C$$

where

T	Torque transmitted through the damper
D	Damping (viscous friction) coefficient
ω	Relative angular velocity
ω_R, ω_C	Absolute angular velocities of terminals R and C, respectively

The block positive direction is from port R to port C. This means that the torque is positive if it acts in the direction from R to C.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Damping coefficient

Damping coefficient, defined by viscous friction. The default value is 0.001 N*m/(rad/s).

Ports

The block has the following ports:

R

Mechanical rotational conserving port.

C

Mechanical rotational conserving port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

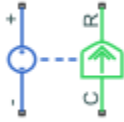
See Also

[Rotational Friction](#) | [Rotational Hard Stop](#) | [Rotational Spring](#)

Introduced in R2007a

Rotational Electromechanical Converter

Interface between electrical and mechanical rotational domains



Library

Electrical Elements

Description

The Rotational Electromechanical Converter block provides an interface between the electrical and mechanical rotational domains. It converts electrical energy into mechanical energy in the form of rotational motion, and vice versa. The converter is described with the following equations:

$$T = K \cdot I$$

$$V = K \cdot \omega$$

where

V	Voltage across the electrical ports of the converter
I	Current through the electrical ports of the converter
T	Torque
ω	Angular speed
K	Constant of proportionality

The Rotational Electromechanical Converter block represents a lossless electromechanical energy conversion, therefore the same constant of proportionality is used in both equations.

Connections + and - are conserving electrical ports corresponding to the positive and negative terminals of the converter, respectively. Connections C and R are conserving mechanical rotational ports. If the current flowing from the positive to the negative terminal is positive, then the resulting torque is positive acting from port C to port R. This direction can be altered by using a negative value for K .

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Constant of proportionality K

Constant of proportionality for electromechanical conversions. The default value is 0.1 V/(rad/s).

Ports

The block has the following ports:

- +
Electrical conserving port associated with the converter positive terminal.
- Electrical conserving port associated with the converter negative terminal.
- C
Mechanical rotational conserving port.
- R
Mechanical rotational conserving port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Translational Electromechanical Converter

Introduced in R2007a

Rotational Free End

Rotational port terminator with zero torque

Library: Simscape / Foundation Library / Mechanical / Rotational Elements



Description

The Rotational Free End block represents a mechanical rotational port that rotates freely, without torque.

You can use this block to terminate mechanical rotational ports on other blocks that you want to leave unconnected. This is not necessary because if you leave a conserving port unconnected, the physical network sets all the Through variables at that port to 0. However, terminator blocks improve diagram readability.

You can also use this block to set the initial rotational velocity at a node.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

W — Zero torque

mechanical rotational

Mechanical rotational conserving port that rotates freely, without torque.

Compatibility Considerations

Capping unconnected ports is no longer required

Behavior changed in R2019b

This block is no longer required. Starting in R2019b, the restriction that disallowed unconnected conserving ports in Simscape models has been lifted. Now, if you leave a conserving port unconnected, the physical network sets all the Through variables at this port to 0. However, you can still use terminator blocks to improve diagram readability.

There are no plans to remove the terminator blocks. All existing models that use these blocks work the same as in previous releases.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Cap (2P) | Cap (G) | Cap (MA) | Cap (TL) | Hydraulic Cap | Open Circuit | Perfect Insulator | Translational Free End

Introduced in R2012b

Rotational Friction

Friction in contact between rotating bodies

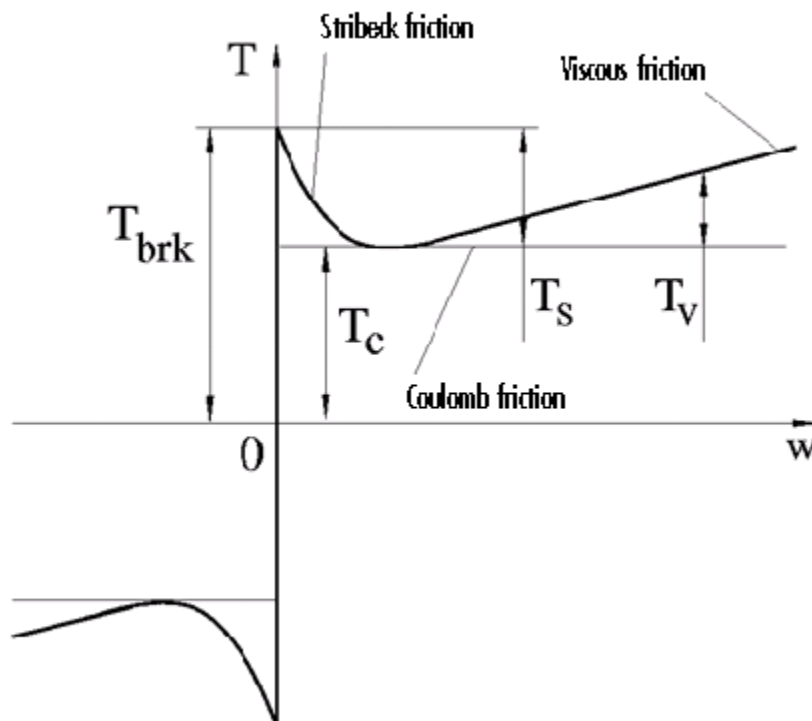


Library

Mechanical Rotational Elements

Description

The Rotational Friction block represents friction in contact between rotating bodies. The friction torque is simulated as a function of relative velocity and is assumed to be the sum of Stribeck, Coulomb, and viscous components, as shown in the following figure.



The Stribeck friction, T_s , is the negatively sloped characteristics taking place at low velocities (see [1] on page 1-700). The Coulomb friction, T_c , results in a constant torque at any velocity. The viscous friction, T_v , opposes motion with the torque directly proportional to the relative velocity. The sum of the Coulomb and Stribeck frictions at the vicinity of zero velocity is often referred to as the breakaway friction, T_{brk} . The friction is approximated with the following equations:

$$T = \sqrt{2}e^{-(T_{brk} - T_C)} \cdot \exp\left(-\left(\frac{\omega}{\omega_{St}}\right)^2\right) \cdot \frac{\omega}{\omega_{St}} + T_C \cdot \tanh\left(\frac{\omega}{\omega_{Coul}}\right) + f\omega$$

$$\omega_{St} = \omega_{brk}\sqrt{2}$$

$$\omega_{Coul} = \omega_{brk}/10$$

$$\omega = \omega_R - \omega_C$$

where

T	Friction torque
T_C	Coulomb friction torque
T_{brk}	Breakaway friction torque
ω_{brk}	Breakaway friction velocity
ω_{St}	Stribeck velocity threshold
ω_{Coul}	Coulomb velocity threshold
ω_R, ω_C	Absolute angular velocities of terminals R and C, respectively
ω	Relative velocity
f	Viscous friction coefficient

The exponential function used in the Stribeck portion of the force equation is continuous and decays at velocity magnitudes greater than the breakaway friction velocity.

The hyperbolic tangent function used in the Coulomb portion of the force equation ensures that the equation is smooth and continuous through $\omega = 0$, but quickly reaches its full value at nonzero velocities.

The block positive direction is from port R to port C. This means that if the port R velocity is greater than that of port C, the block transmits torque from R to C.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Breakaway friction torque

Breakaway friction torque, which is the sum of the Coulomb and the static frictions. It must be greater than or equal to the Coulomb friction torque value. The default value is 25 N*m.

Breakaway friction velocity

The velocity at which the Stribeck friction is at its peak. At this point, the sum of the Stribeck and Coulomb friction is the **Breakaway friction force**. The default value is 0.1 rad/s.

Coulomb friction torque

Coulomb friction torque, which is the friction that opposes rotation with a constant torque at any velocity. The default value is 20 N*m.

Viscous friction coefficient

Proportionality coefficient between the friction torque and the relative angular velocity. The parameter value must be greater than or equal to zero. The default value is 100 N*m/(rad/s).

Ports

The block has the following ports:

R

Mechanical rotational conserving port.

C

Mechanical rotational conserving port.

Examples

The “Mechanical Rotational System with Stick-Slip Motion” example illustrates the use of the Rotational Friction block in mechanical systems. The friction element is installed between the load and the velocity source, and there is a difference between the breakaway and the Coulomb frictions. As a result, stick-slip motion is developed in the regions of constant velocities.

References

[1] B. Armstrong, C.C. de Wit, *Friction Modeling and Compensation*, The Control Handbook, CRC Press, 1995

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Rotational Damper | Rotational Hard Stop | Rotational Spring

Introduced in R2007a

Rotational Hard Stop

Double-sided rotational hard stop

Library: Simscape / Foundation Library / Mechanical / Rotational Elements



Description

The Rotational Hard Stop block represents a double-sided mechanical rotational hard stop that restricts motion of a body between upper and lower bounds. Both ports of the block are of mechanical rotational type. The impact interaction between the slider and the stops is assumed to be elastic. The stop is implemented as a spring that comes into contact with the slider as the gap is cleared. The spring opposes slider penetration into the stop with the torque linearly proportional to this penetration. To account for energy dissipation and nonelastic effects, the damping is introduced as a block parameter, thus making it possible to account for energy loss.

The basic hard stop model, Full stiffness and damping applied at bounds, damped rebound, is described with the following equations:

$$T = \begin{cases} K_p \cdot (\varphi - g_p) + D_p \cdot \omega & \text{for } \varphi \geq g_p \\ 0 & \text{for } g_n < \varphi < g_p \\ K_n \cdot (\varphi - g_n) + D_n \cdot \omega & \text{for } \varphi \leq g_n \end{cases}$$

$$\omega = \frac{d\varphi}{dt}$$

where

- T is interaction torque between the slider and the case.
- g_p is the initial gap between the slider and upper bound.
- g_n is the initial gap between the slider and lower bound.
- φ is the slider angular position.
- K_p is contact stiffness at upper bound.
- K_n is contact stiffness at lower bound.
- D_p is damping coefficient at upper bound.
- D_n is damping coefficient at lower bound.
- ω is the slider angular velocity.
- t is time.

In the Full stiffness and damping applied at bounds, undamped rebound hard stop model, equations contain additional terms, $ge(\omega,0)$ and $le(\omega,0)$. These terms ensure that damping is not applied on the rebound.

$$T = \begin{cases} K_p \cdot (\varphi - g_p) + D_p \cdot \omega \cdot ge(\omega, 0) & \text{for } \varphi \geq g_p \\ 0 & \text{for } g_n < \varphi < g_p \\ K_n \cdot (\varphi - g_n) + D_n \cdot \omega \cdot le(\omega, 0) & \text{for } \varphi \leq g_n \end{cases}$$

Relational functions `ge` (greater or equal) and `le` (less or equal) do not generate zero crossings when angular velocity changes sign. For more information, see “Enabling and Disabling Zero-Crossing Conditions in Simscape Language”. However, the solver treats `ge` and `le` functions as nonlinear. Therefore, if `simscape.findNonlinearBlocks` indicates that the rest of your network is linear or switched linear, use the Full stiffness and damping applied at bounds, damped rebound model to improve performance.

The default hard stop model, Stiffness and damping applied smoothly through transition region, damped rebound, adds two transitional regions to the equations, one at each bound. While the slider travels through a transition region, the block smoothly ramps up the torque from zero to the full value. At the end of the transition region, the full stiffness and damping are applied. On the rebound, both stiffness and damping torques are smoothly decreased back to zero. These equations also use the `ge` and `le` relational functions, which do not produce zero crossings.

The block is oriented from R to C. This means that the block transmits torque from port R to port C when the gap is closed in the positive direction.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

R — Rod (slider)

mechanical rotational

Mechanical rotational conserving port associated with the slider that travels between stops installed on the case.

C — Case

mechanical rotational

Mechanical rotational conserving port associated with the case.

Parameters

Upper bound — Gap between slider and upper bound

0.1 rad (default)

Gap between the slider and the upper bound. The direction is specified with respect to the local coordinate system, with the slider located in the origin. A positive value of the parameter specifies the gap between the slider and the upper bound. A negative value sets the slider as penetrating into the upper bound.

Lower bound — Gap between slider and lower bound

-0.1 rad (default)

Gap between the slider and the lower bound. The direction is specified with respect to the local coordinate system, with the slider located in the origin. A negative value of the parameter specifies the gap between the slider and the lower bound. A positive value sets the slider as penetrating into the lower bound.

Contact stiffness at upper bound — Elasticity of collision at upper bound

1e6 N*m/rad (default)

This parameter specifies the elastic property of colliding bodies when the slider hits the upper bound. The greater the value of the parameter, the less the bodies penetrate into each other, the more rigid the impact becomes. Lesser value of the parameter makes contact softer, but generally improves convergence and computational efficiency.

Contact stiffness at lower bound — Elasticity of collision at lower bound

1e6 N*m/rad (default)

This parameter specifies the elastic property of colliding bodies when the slider hits the upper bound. The greater the value of the parameter, the less the bodies penetrate into each other, the more rigid the impact becomes. Lesser value of the parameter makes contact softer, but generally improves convergence and computational efficiency.

Contact damping at upper bound — Dissipating property at upper bound

0.01 N*m/(rad/s) (default)

This parameter specifies dissipating property of colliding bodies when the slider hits the upper bound. The greater the value of the parameter, the more energy dissipates during an interaction.

Contact damping at lower bound — Dissipating property at lower bound

0.01 N*m/(rad/s) (default)

This parameter specifies dissipating property of colliding bodies when the slider hits the lower bound. The greater the value of the parameter, the more energy dissipates during an interaction.

Hard stop model — Select the hard stop model

Stiffness and damping applied smoothly through transition region, damped rebound (default) | Full stiffness and damping applied at bounds, undamped rebound | Full stiffness and damping applied at bounds, damped rebound

Select the hard stop model:

- Stiffness and damping applied smoothly through transition region, damped rebound — Specify a transition region, in which the torque is scaled from zero. At the end of the transition region, the full stiffness and damping are applied. This model has damping applied on the rebound, but it is limited to the value of the stiffness torque. In this sense, damping can reduce or eliminate the torque provided by the stiffness, but never exceed it. All equations are smooth and produce no zero crossings.
- Full stiffness and damping applied at bounds, undamped rebound — This model has full stiffness and damping applied with impact at upper and lower bounds, with no damping on the rebound. Equations produce no zero crossings when velocity changes sign, but there is a position-based zero crossing at the bounds. Having no damping on rebound helps to push the slider past this position quickly. This model has nonlinear equations.

- **Full stiffness and damping applied at bounds, damped rebound** — This model has full stiffness and damping applied with impact at upper and lower bounds, with damping applied on the rebound as well. Equations are switched linear, but produce position-based zero crossings. Use this hard stop model if `simscape.findNonlinearBlocks` indicates that this is the block that prevents the whole network from being switched linear.

Transition region — Region where torque is ramped up

0.001 rad (default)

Region where the torque is ramped up from zero to the full value. At the end of the transition region, the full stiffness and damping are applied.

Dependencies

Enabled when the **Hard stop model** parameter is set to **Stiffness and damping applied smoothly through transition region, damped rebound**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Rotational Damper | Rotational Friction | Rotational Spring

Introduced in R2007a

Rotational Hydro-Mechanical Converter

Interface between hydraulic and mechanical rotational domains



Library

Hydraulic Elements

Description

The Rotational Hydro-Mechanical Converter block models an ideal transducer that converts hydraulic energy into mechanical energy, in the form of rotational motion of the converter shaft, and vice versa. Physically, the converter represents the main component of a single-acting rotary vane actuator. The compressibility option makes the converter account for dynamic variations of the fluid density.

Using this block as a basic element, you can build a large variety of rotary actuators by adding application-specific effects, such as leakage, friction, hard stops, and so on.

The converter is simulated according to the following equations:

$$q = \frac{d\left(\frac{\rho}{\rho_l^0} V\right)}{dt} = \frac{d\left(\frac{\rho}{\rho_l^0}\right)}{dt} V + \frac{\rho}{\rho_l^0} \cdot \varepsilon \cdot (\omega_S - \omega_C) \cdot D$$

$$T = \varepsilon \cdot p \cdot D$$

$$\rho = \begin{cases} \frac{\left(\frac{\alpha}{1-\alpha}\right)\rho_g^0 + \rho_l^0}{\left(\frac{\alpha}{1-\alpha}\right)\left(\frac{p_0}{p}\right)^{\frac{1}{\gamma}} + e^{-\frac{p-p_0}{\beta_l}}} & \text{if compressibility is on} \\ \rho_l^0 & \text{if compressibility is off} \end{cases}$$

where

q	Flow rate to the converter chamber
D	Converter displacement, or fluid volume needed to rotate the shaft per angle unit
ω_S	Converter shaft angular velocity
ω_C	Converter case angular velocity
T	Torque on the shaft
p	Gauge pressure of fluid in the converter chamber
V	Piston volume

α	Relative amount of trapped air
ρ_l^0	Fluid density at atmospheric conditions
ρ_g^0	Gas density at atmospheric conditions
p_0	Atmospheric pressure
γ	Specific heat ratio
β_l	Bulk modulus at atmospheric conditions and no gas
ε	Converter orientation with respect to the globally assigned positive direction. If pressure applied at port A generates torque in positive direction, ε equals 1. If pressure applied at port A generates torque in negative direction, ε equals -1.

The piston volume is computed according to

$$V = V_{dead} + D \cdot (\theta_0 + \theta)$$

$$\frac{d\theta}{dt} = \varepsilon \cdot (\omega_S - \omega_C)$$

where

V_{dead}	Chamber dead volume
θ_0	Shaft initial angle
θ	Shaft rotation from initial position

Port A is a hydraulic conserving port associated with the converter inlet. Ports S and C are mechanical rotational conserving ports associated with the shaft and the case of the converter, respectively. Pressure at port A generates torque in the direction specified by the **Converter orientation** parameter.

The block dialog box does not have a **Source code** link. To view the underlying component source, open the following files in the MATLAB editor:

- For incompressible converter implementation — `rotational_converter_incompressible.ssc`
- For compressible converter implementation — `rotational_converter_compressible.ssc`

Basic Assumptions and Limitations

The block simulates an ideal converter, with an option to account for fluid compressibility. Other effects, such as hard stops, inertia, or leakage, are modeled outside of the converter.

Parameters

Displacement

Effective converter displacement. The default value is $1.2e-4 \text{ m}^3/\text{rad}$.

Converter orientation

Specifies converter orientation with respect to the globally assigned positive direction. The converter can be installed in two different ways, depending upon whether it generates torque in the positive or in the negative direction when pressure is applied at its inlet. If pressure applied

at port A generates torque in negative direction, set the parameter to `Acts in negative direction`. The default value is `Acts in positive direction`.

Compressibility

Specifies whether fluid density is taken as constant or varying with pressure. The default value is `Off`, in which case the block models an ideal transducer. If you select `On`, the block dialog box displays additional parameters that let you model dynamic variations of the fluid density without adding any extra blocks.

Shaft initial angle

Initial offset of the piston from the cylinder cap. The default value is `0`.

Dead volume

Volume of fluid in the chamber at zero piston position. The default value is `1e-4 m^3`.

Specific heat ratio

Gas-specific heat ratio. The default value is `1.4`.

Initial pressure

Initial pressure in the chamber. This parameter specifies the initial condition for use in computing the block's initial state at the beginning of a simulation run. The default value is `0`.

Restricted Parameters

When your model is in Restricted editing mode, you cannot modify the following parameters:

- **Converter orientation**
- **Compressibility**

All other block parameters are available for modification.

Ports

The block has the following ports:

A

Hydraulic conserving port associated with the converter inlet.

S

Mechanical rotational conserving port associated with the shaft of the converter.

C

Mechanical rotational conserving port associated with the case of the converter.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Translational Hydro-Mechanical Converter

Introduced in R2007a

Rotational Inerter

Two-port inertia in mechanical rotational systems



Library

Mechanical Rotational Elements

Description

The Rotational Inerter block represents a device that has torque proportional to the rate of change of the relative angular velocity across the ports. It is essentially a two-port inertia that works on the velocity difference between the ports, not the absolute velocity.

Use this block in high performance suspension systems, to decouple weave and roll modes, or in applications where you need to model a passively tuned mass-spring-damper response.

The block is described with the following equations:

$$T = B \frac{d\omega}{dt}$$

$$\omega = \omega_R - \omega_C$$

where

T	Torque transmitted through the inerter
B	Rotational inertance
ω	Relative angular velocity
ω_R, ω_C	Absolute angular velocities at ports R and C, respectively

The block positive direction is from port R to port C. This means that the torque is positive if it acts in the direction from R to C.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Rotational inertance

Proportionality coefficient between the torque and the rate of change of the relative angular velocity across the ports. The default value is 1 kg*m/².

Ports

The block has the following ports:

R

Mechanical rotational conserving port associated with the rod.

C

Mechanical rotational conserving port associated with the case.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

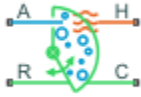
See Also

Inertia

Introduced in R2015b

Rotational Mechanical Converter (2P)

Interface between two-phase fluid and mechanical rotational networks



Library

Two-Phase Fluid/Elements

Description

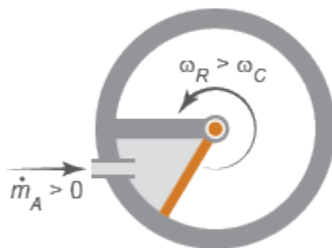
The Rotational Mechanical Converter (2P) block models an interface between two-phase fluid and mechanical rotational networks. The interface converts pressure in the fluid network into torque in the mechanical rotational network and vice versa.

This block enables you to model a rotary actuator powered by a two-phase fluid system. It does not, however, account for inertia, friction, or hard stops, common in rotary actuators. You can model these effects separately using Simscape blocks such as Inertia, Rotational Friction, and Rotational Hard Stop.

Port A represents the inlet through which fluid enters and exits the converter. Ports C and R represent the converter casing and moving interface, respectively. Port H represents the wall through which the converter exchanges heat with its surroundings.

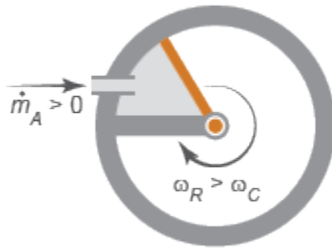
Torque Direction

The torque direction depends on the mechanical orientation of the converter. If the **Mechanical Orientation** parameter is positive, then a positive flow rate through the inlet tends to rotate the moving interface in the positive direction relative to the converter casing.



Positive Mechanical Orientation

If the **Mechanical Orientation** parameter is negative, then a positive mass flow rate through the inlet tends to rotate the moving interface in the negative direction relative to the converter casing.



Negative Mechanical Orientation

Flow and Thermal Resistances

The flow resistance between port A and the converter interior is assumed negligible. Pressure losses between the two is approximately zero. The pressure at port A is therefore equal to that in the converter:

$$p_A = p_I,$$

where:

- p_A is the pressure at port A.
- p_I is the pressure in the converter.

Similarly, the thermal resistance between port H and the converter interior is assumed negligible. The temperature gradient between the two is approximately zero. The temperature at port H is therefore equal to that in the converter:

$$T_H = T_I,$$

where:

- T_H is the temperature at port H.
- T_I is the temperature in the converter.

Fluid Volume

The volume of fluid in the converter is the sum of the dead and displaced fluid volumes. The dead volume is the amount of fluid left in the converter at a zero interface angle. This volume enables you to model the effects of dynamic compressibility and thermal capacity even when the interface is in its zero position.

The displacement volume is the amount of fluid added to the converter due to rotation of the moving interface. This volume increases with the interface angle. The total volume in the converter as a function of the interface angle is

$$V = V_{dead} + D_{vol}\theta_{int}\epsilon_{or},$$

where:

- V is the total volume of fluid in the converter.
- V_{dead} is the dead volume of the converter.

- D_{vol} is the displaced fluid volume per unit rotation of the interface.
- θ_{int} is the rotation angle of the moving interface.
- ϵ_{or} is an integer encoding the mechanical orientation of the converter:

$$\epsilon_{or} = \begin{cases} +1, & \text{if the mechanical orientation is positive} \\ -1, & \text{if the mechanical orientation is negative} \end{cases}$$

Force Balance

At equilibrium, the internal pressure in the converter counteracts the external pressure of its surroundings and the torque exerted by the mechanical network on the moving interface. This torque is the reverse of that applied by the fluid network. The torque balance in the converter is therefore

$$p_I D_{vol} = p_{atm} D_{vol} - t_{int} \epsilon_{or},$$

where:

- p_{atm} is the environmental pressure outside the converter.
- t_{int} is the magnitude of the torque exerted by the fluid network on the moving interface.

Energy Balance

The total energy in the converter can change due to energy flow through the inlet, heat flow through the converter wall, and work done on the mechanical network. The energy flow rate, given by the energy conservation equation, is therefore

$$\dot{E} = \phi_A + \phi_H - p_I D_{vol} \dot{\theta}_{int} \epsilon_{or},$$

where:

- E is the total energy of the fluid in the converter.
- ϕ_A is the energy flow rate into the converter through port A.
- ϕ_H is the heat flow rate into the converter through port H.

Taking the fluid kinetic energy in the converter to be negligible, the total energy of the fluid reduces to:

$$E = M u_I,$$

where:

- M is the fluid mass in the converter.
- u_I is the specific internal energy of the fluid in the converter.

Mass Balance

The fluid mass in the converter can change due to flow through the inlet, represented by port A. The mass flow rate, given by the mass conservation equation, is therefore

$$\dot{M} = \dot{m}_A,$$

where:

- \dot{m}_A is the mass flow rate into the converter through port A.

A change in fluid mass can accompany a change in fluid volume, due to rotation of the moving interface. It can also accompany a change in mass density, due to an evolving pressure or specific internal energy in the converter. The mass rate of change in the converter is then

$$\dot{M} = \left[\left(\frac{\partial \rho}{\partial p} \right)_u \dot{p}_I + \left(\frac{\partial \rho}{\partial u} \right)_p \dot{u}_I \right] V + \frac{D_{vol} \dot{\theta}_{int} \epsilon_{or}}{v_I},$$

where:

- $\left(\frac{\partial \rho}{\partial p} \right)_u$ is the partial derivative of density with respect to pressure at constant specific internal energy.
- $\left(\frac{\partial \rho}{\partial u} \right)_p$ is the partial derivative of density with respect to specific internal energy at constant pressure.
- v_I is the specific volume of the fluid in the converter.

The block blends the density partial derivatives of the various domains using a cubic polynomial function. At a vapor quality of 0–0.1, this function blends the derivatives of the subcooled liquid and two-phase mixture domains. At a vapor quality of 0.9–1, it blends those of the two-phase mixture and superheated vapor domains.

The smoothed density partial derivatives introduce into the original mass conservation equation undesirable numerical errors. To correct for these errors, the block adds the correction term

$$\epsilon_M = \frac{M - V/v_I}{\tau},$$

where:

- ϵ_M is the correction term.
- τ is the phase-change time constant—the characteristic duration of a phase change event. This constant ensures that phase changes do not occur instantaneously, effectively introducing a time lag whenever they occur.

The final form of the mass conservation equation is

$$\left[\left(\frac{\partial \rho}{\partial p} \right)_u \dot{p}_I + \left(\frac{\partial \rho}{\partial u} \right)_p \dot{u}_I \right] V + \frac{D_{vol} \dot{\theta}_{int} \epsilon_{or}}{v_I} = \dot{m}_A + \epsilon_M.$$

The block uses this equation to calculate the internal pressure in the converter given the mass flow rate through the inlet.

Assumptions and Limitations

- The converter walls are rigid. They do not deform under pressure.
- The flow resistance between port A and the converter interior is negligible. The pressure is the same at port A and in the converter interior.
- The thermal resistance between port H and the converter interior is negligible. The temperature is the same at port H and in the converter interior.

- The moving interface is perfectly sealed. No fluid leaks across the interface.
- Mechanical effects such as hard stops, inertia, and friction, are ignored.

Parameters

Parameters Tab

Mechanical orientation

Orientation of the moving interface relative to the two-phase fluid network. If the orientation is positive, then a positive flow into the converter results in a positive rotation of port R relative to port C. If the orientation is negative, then a positive flow into the converter results in a negative rotation of port R relative to port C. The default setting is **Positive**.

Interface volume displacement

Displaced fluid volume per unit rotation of the moving interface. The default value is $0.01 \text{ m}^3/\text{rad}$.

Interface initial rotation

Angle of the moving interface at the start of simulation. A zero angle corresponds to a total fluid volume in the converter equal to the specified dead volume. The default value is 0 rad .

This parameter must be greater than or equal to zero if the **Mechanical orientation** parameter is set to **Positive**. It must be smaller than or equal to zero if the **Mechanical orientation** parameter is set to **Negative**.

Dead volume

Volume of fluid left in the converter when the interface angle is zero. The dead volume enables the block to account for mass and energy storage in the converter even at a zero interface angle. The default value is $1e-5 \text{ m}^3$.

Cross-sectional area at port A

Flow area of the converter inlet, represented by port **A**. Pressure losses due to changes in flow area inside the converter are ignored. The default value is 0.01 m^2 .

Environment pressure specification

Pressure characteristics of the surrounding environment. Select **Atmospheric pressure** to set the environment pressure to the atmospheric pressure specified in the Two-Phase Fluid Properties (2P) block. Select **Specified pressure** to set the environment pressure to a different value. The default setting is **Atmospheric pressure**.

Environment pressure

Absolute pressure of the surrounding environment. The environment pressure acts against the internal pressure of the converter and affects the motion of the converter shaft. This parameter is active only when the **Environment pressure specification** parameter is set to **Specified pressure**. The default value, 0.101325 MPa , corresponds to atmospheric pressure at mean sea level.

Effects and Initial Conditions Tab

Initial fluid energy specification

Thermodynamic variable in terms of which to define the initial conditions of the component. The default setting is **Temperature**.

Initial pressure

Pressure in the chamber at the start of simulation, specified against absolute zero. The default value is 0.101325 MPa.

Initial temperature

Temperature in the chamber at the start of simulation, specified against absolute zero. This parameter is active when the **Initial fluid energy specification** option is set to Temperature. The default value is 293.15 K.

Initial vapor quality

Mass fraction of vapor in the chamber at the start of simulation. This parameter is active when the **Initial fluid energy specification** option is set to Vapor quality. The default value is 0.5.

Initial vapor void fraction

Volume fraction of vapor in the chamber at the start of simulation. This parameter is active when the **Initial fluid energy specification** option is set to Vapor void fraction. The default value is 0.5.

Initial specific enthalpy

Specific enthalpy of the fluid in the chamber at the start of simulation. This parameter is active when the **Initial fluid energy specification** option is set to Specific enthalpy. The default value is 1500 kJ/kg.

Initial specific internal energy

Specific internal energy of the fluid in the chamber at the start of simulation. This parameter is active when the **Initial fluid energy specification** option is set to Specific internal energy. The default value is 1500 kJ/kg.

Phase change time constant

Characteristic duration of a phase-change event. This constant introduces a time lag into the transition between phases. The default value is 0.1 s.

Ports

The block has the following ports:

A

Two-phase fluid conserving port associated with the converter inlet.

H

Thermal conserving port representing the converter surface through which heat exchange occurs.

R

Mechanical rotational conserving port associated with the converter rotor.

C

Mechanical rotational conserving port associated with the converter case.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Translational Mechanical Converter (2P)

Introduced in R2015b

Rotational Mechanical Converter (G)

Interface between gas and mechanical rotational networks

Library: Simscape / Foundation Library / Gas / Elements



Description

The Rotational Mechanical Converter (G) block models an interface between a gas network and a mechanical rotational network. The block converts gas pressure into mechanical torque and vice versa. It can be used as a building block for rotary actuators.

The converter contains a variable volume of gas. The pressure and temperature evolve based on the compressibility and thermal capacity of this gas volume. If **Mechanical orientation** is set to **Positive**, then an increase in the gas volume results in a positive rotation of port **R** relative to port **C**. If **Mechanical orientation** is set to **Negative**, then an increase in the gas volume results in a negative rotation of port **R** relative to port **C**.

Port **A** is the gas conserving port associated with the converter inlet. Port **H** is the thermal conserving port associated with the temperature of the gas inside the converter. Ports **R** and **C** are the mechanical rotational conserving ports associated with the moving interface and converter casing, respectively.

Mass Balance

Mass conservation equation is similar to that for the Constant Volume Chamber (G) block, with an additional term related to the change in gas volume:

$$\frac{\partial M}{\partial p} \cdot \frac{dp_I}{dt} + \frac{\partial M}{\partial T} \cdot \frac{dT_I}{dt} + \rho_I \frac{dV}{dt} = \dot{m}_A$$

where:

- $\frac{\partial M}{\partial p}$ is the partial derivative of the mass of the gas volume with respect to pressure at constant temperature and volume.
- $\frac{\partial M}{\partial T}$ is the partial derivative of the mass of the gas volume with respect to temperature at constant pressure and volume.
- p_I is the pressure of the gas volume. Pressure at port A is assumed equal to this pressure, $p_A = p_I$.
- T_I is the temperature of the gas volume. Temperature at port H is assumed equal to this temperature, $T_H = T_I$.
- ρ_I is the density of the gas volume.
- V is the volume of gas.

- t is time.
- \dot{m}_A is the mass flow rate at port **A**. Flow rate associated with a port is positive when it flows into the block.

Energy Balance

Energy conservation equation is also similar to that for the Constant Volume Chamber (G) block. The additional term accounts for the change in gas volume, as well as the pressure-volume work done by the gas on the moving interface:

$$\frac{\partial U}{\partial p} \cdot \frac{dp_I}{dt} + \frac{\partial U}{\partial T} \cdot \frac{dT_I}{dt} + \rho_I h_I \frac{dV}{dt} = \Phi_A + Q_H$$

where:

- $\frac{\partial U}{\partial p}$ is the partial derivative of the internal energy of the gas volume with respect to pressure at constant temperature and volume.
- $\frac{\partial U}{\partial T}$ is the partial derivative of the internal energy of the gas volume with respect to temperature at constant pressure and volume.
- Φ_A is the energy flow rate at port **A**.
- Q_H is the heat flow rate at port **H**.
- h_I is the specific enthalpy of the gas volume.

Partial Derivatives for Perfect and Semiperfect Gas Models

The partial derivatives of the mass M and the internal energy U of the gas volume, with respect to pressure and temperature at constant volume, depend on the gas property model. For perfect and semiperfect gas models, the equations are:

$$\frac{\partial M}{\partial p} = V \frac{\rho_I}{p_I}$$

$$\frac{\partial M}{\partial T} = -V \frac{\rho_I}{T_I}$$

$$\frac{\partial U}{\partial p} = V \left(\frac{h_I}{ZRT_I} - 1 \right)$$

$$\frac{\partial U}{\partial T} = V \rho_I \left(c_{pI} - \frac{h_I}{T_I} \right)$$

where:

- ρ_I is the density of the gas volume.
- V is the volume of gas.
- h_I is the specific enthalpy of the gas volume.
- Z is the compressibility factor.
- R is the specific gas constant.
- c_{pI} is the specific heat at constant pressure of the gas volume.

Partial Derivatives for Real Gas Model

For real gas model, the partial derivatives of the mass M and the internal energy U of the gas volume, with respect to pressure and temperature at constant volume, are:

$$\frac{\partial M}{\partial p} = V \frac{\rho_I}{\beta_I}$$

$$\frac{\partial M}{\partial T} = -V \rho_I \alpha_I$$

$$\frac{\partial U}{\partial p} = V \left(\frac{\rho_I h_I}{\beta_I} - T_I \alpha_I \right)$$

$$\frac{\partial U}{\partial T} = V \rho_I (c_{pI} - h_I \alpha_I)$$

where:

- β is the isothermal bulk modulus of the gas volume.
- α is the isobaric thermal expansion coefficient of the gas volume.

Gas Volume

The gas volume depends on the rotation of the moving interface:

$$V = V_{dead} + D_{int} \theta_{int} \varepsilon_{int}$$

where:

- V_{dead} is the dead volume.
- D_{int} is the interface volume displacement.
- θ_{int} is the interface rotation.
- ε_{int} is the mechanical orientation coefficient. If **Mechanical orientation** is Positive, $\varepsilon_{int} = 1$. If **Mechanical orientation** is Negative, $\varepsilon_{int} = -1$.

Torque Balance

Torque balance across the moving interface on the gas volume is

$$\tau_{int} = (p_{env} - p_I) D_{int} \varepsilon_{int}$$

where:

- τ_{int} is the torque from port **R** to port **C**.
- p_{env} is the environment pressure.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Gas Volume”.

Assumptions and Limitations

- The converter casing is perfectly rigid.
- There is no flow resistance between port **A** and the converter interior.
- There is no thermal resistance between port **H** and the converter interior.
- The moving interface is perfectly sealed.
- The block does not model mechanical effects of the moving interface, such as hard stop, friction, and inertia.

Ports

Conserving

A — Converter inlet

gas

Gas conserving port associated with the converter inlet.

H — Temperature inside converter

thermal

Thermal conserving port associated with the temperature of the gas inside the converter.

R — Rod

mechanical rotational

Mechanical rotational conserving port associated with the moving interface.

C — Case

mechanical rotational

Mechanical rotational conserving port associated with the converter casing.

Parameters

Mechanical orientation — Select the converter orientation

Positive (default) | Negative

Select the alignment of moving interface with respect to the converter gas volume:

- **Positive** — Increase in the gas volume results in a positive rotation of port **R** relative to port **C**.
- **Negative** — Increase in the gas volume results in a negative rotation of port **R** relative to port **C**.

Interface volume displacement — Displaced gas volume per unit rotation

0.01 m³/rad (default)

Displaced gas volume per unit rotation of the moving interface.

Interface initial rotation — Rotational offset of port R relative to port C at the start of simulation

0 rad (default)

Rotational offset of port **R** relative to port **C** at the start of simulation. A value of 0 corresponds to an initial gas volume equal to **Dead volume**.

Dependencies

- If **Mechanical orientation** is **Positive**, the parameter value must be greater than or equal to 0.
- If **Mechanical orientation** is **Negative**, the parameter value must be less than or equal to 0.

Dead volume — Volume of gas when the interface rotation is 0

1e-5 m³ (default)

Volume of gas when the interface rotation is 0.

Cross-sectional area at port A — Area normal to flow path at the converter inlet

0.01 m² (default)

The cross-sectional area of the converter inlet, in the direction normal to gas flow path.

Environment pressure specification — Select a specification method for the environment pressure

Atmospheric pressure (default) | Specified pressure

Select a specification method for the environment pressure:

- **Atmospheric pressure** — Use the atmospheric pressure, specified by the Gas Properties (G) block connected to the circuit.
- **Specified pressure** — Specify a value by using the **Environment pressure** parameter.

Environment pressure — Pressure outside the converter

0.101325 MPa (default)

Pressure outside the converter acting against the pressure of the converter gas volume. A value of 0 indicates that the converter expands into vacuum.

Dependencies

Enabled when the **Environment pressure specification** parameter is set to **Specified pressure**.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Constant Volume Chamber (G) | Translational Mechanical Converter (G)

Topics

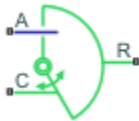
“Modeling Gas Systems”

Introduced in R2016b

Rotational Mechanical Converter (IL)

Interface between isothermal liquid and mechanical rotational networks

Library: Simscape / Foundation Library / Isothermal Liquid / Elements



Description

The Rotational Mechanical Converter (IL) block models an interface between an isothermal liquid network and a mechanical rotational network. The block converts isothermal liquid pressure into mechanical torque and vice versa. It can be used as a building block for rotary actuators.

The converter contains a variable volume of liquid. If **Model dynamic compressibility** is set to On, then the pressure evolves based on the dynamic compressibility of the liquid volume. The **Mechanical orientation** parameter lets you specify whether an increase in pressure moves port **R** away from or towards port **C**.

Port **A** is the isothermal liquid conserving port associated with the converter inlet. Ports **R** and **C** are the mechanical rotational conserving ports associated with the moving interface and converter casing, respectively.

Mass Balance

The mass conservation equations in the mechanical converter volume are

$$\dot{m}_A = \begin{cases} \varepsilon \rho_I D \omega, & \text{if fluid dynamic compressibility is off} \\ \varepsilon \rho_I D \omega + \frac{1}{\beta_I} \frac{dp_I}{dt} \rho_I V, & \text{if fluid dynamic compressibility is on} \end{cases}$$

$$\omega = \frac{d\theta}{dt}$$

$$V = V_{dead} + \varepsilon D \theta$$

where:

- \dot{m}_A is the mass flow rate into the converter through port **A**.
- ε is the mechanical orientation of the converter (1 if fluid pressure moves R away from C, -1 if fluid pressure moves R towards C).
- ρ_I is the fluid density inside the converter.
- β_I is the fluid bulk modulus inside the converter.
- D is the converter volume displacement, that is, fluid volume needed to rotate the shaft per angle unit.
- ω is the angular velocity of the converter interface.

- θ is the converter interface rotation.
- V is the liquid volume inside the converter.
- V_{dead} is the dead volume, that is, volume of liquid when the interface displacement is 0.
- p_I is the pressure inside the converter.

Equations used to compute the fluid mixture density and bulk modulus depend on the selected isothermal liquid model. For detailed information, see “Isothermal Liquid Modeling Options”.

Momentum Balance

The momentum conservation equation in the mechanical converter volume is

$$\tau = \varepsilon(p_{\text{env}} - p)D,$$

where:

- τ is the torque the liquid exerts on the converter interface.
- p_{env} is the environment pressure outside the converter.

Assumptions and Limitations

- Converter walls are perfectly rigid.
- The converter contains no mechanical hard stops. To include hard stops, use the Rotational Hard Stop block.
- The flow resistance between the inlet and the interior of the converter is negligible.
- The kinetic energy of the fluid in the converter is negligible.

Ports

Conserving

A – Converter inlet

isothermal liquid

Isothermal liquid conserving port associated with the converter inlet.

R – Rod

mechanical rotational

Mechanical rotational conserving port associated with the moving interface.

C – Case

mechanical rotational

Mechanical rotational conserving port associated with the converter casing.

Parameters

Mechanical orientation – Select the converter orientation

Fluid pressure moves R away from C (default) | Fluid pressure moves R towards C

Select the alignment of moving interface with respect to the fluid pressure:

- Fluid pressure moves R away from C — Increase in the fluid pressure results in a positive rotation of port **R** relative to port **C**.
- Fluid pressure moves R towards C — Increase in the fluid pressure results in a negative rotation of port **R** relative to port **C**.

Interface volume displacement — Displaced liquid volume per unit rotation

0.001 m³/rad (default) | positive scalar

Displaced liquid volume per unit rotation of the moving interface.

Interface initial rotation — Rotational offset of port R relative to port C at the start of simulation

0 rad (default) | scalar

Rotational offset of port **R** relative to port **C** at the start of simulation. A value of 0 corresponds to an initial liquid volume equal to **Dead volume**.

Dependencies

- If **Mechanical orientation** is Fluid pressure moves R away from C, the parameter value must be greater than or equal to 0.
- If **Mechanical orientation** is Fluid pressure moves R towards C, the parameter value must be less than or equal to 0.

Dead volume — Volume of liquid when the interface rotation is 0

1e-5 m³ (default) | positive scalar

Volume of liquid when the interface rotation is 0.

Cross-sectional area at port A — Area normal to flow path at the converter inlet

0.01 m² (default) | positive scalar

The cross-sectional area of the converter inlet, in the direction normal to the flow path.

Environment pressure specification — Select a specification method for the environment pressure

Atmospheric pressure (default) | Specified pressure

Select a specification method for the pressure outside the converter:

- Atmospheric pressure — Use the atmospheric pressure, specified by the Thermal Liquid Settings (TL) or Thermal Liquid Properties (TL) block connected to the circuit.
- Specified pressure — Specify a value by using the **Environment pressure** parameter.

Environment pressure — Pressure outside the converter

0.101325 MPa (default) | positive scalar

Pressure outside the converter acting against the pressure of the converter liquid volume. A value of 0 indicates that the converter expands into vacuum.

Dependencies

Enabled when the **Environment pressure specification** parameter is set to Specified pressure.

Fluid dynamic compressibility — Select whether to model fluid dynamic compressibility

On (default) | Off

Select whether to account for the dynamic compressibility of the liquid. Dynamic compressibility gives the liquid density a dependence on pressure and temperature, impacting the transient response of the system at small time scales.

Initial liquid pressure — Liquid pressure at time zero

0.101325 MPa (default) | positive scalar

Liquid pressure in the converter at the start of simulation.

Dependencies

Enabled when the **Fluid dynamic compressibility** parameter is set to On.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Translational Mechanical Converter (IL)

Topics

“Modeling Isothermal Liquid Systems”

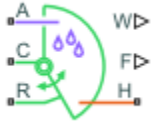
“Isothermal Liquid Modeling Options”

Introduced in R2020a

Rotational Mechanical Converter (MA)

Interface between moist air and mechanical rotational networks

Library: Simscape / Foundation Library / Moist Air / Elements



Description

The Rotational Mechanical Converter (MA) block models an interface between a moist air network and a mechanical rotational network. The block converts moist air pressure into mechanical torque and vice versa. You can use it as a building block for rotary actuators.

The converter contains a variable volume of moist air. The pressure and temperature evolve based on the compressibility and thermal capacity of this moist air volume. Liquid water condenses out of the moist air volume when it reaches saturation.

If **Mechanical orientation** is set to **Positive**, an increase in the moist air volume inside the converter results in a positive rotation of port **R** relative to port **C**. If **Mechanical orientation** is set to **Negative**, an increase in the moist air volume results in a negative rotation of port **R** relative to port **C**.

The block equations use these symbols. Subscripts **a**, **w**, and **g** indicate the properties of dry air, water vapor, and trace gas, respectively. Subscript **ws** indicates water vapor at saturation. Subscripts **A**, **H**, and **S** indicate the appropriate port. Subscript **I** indicates the properties of the internal moist air volume.

\dot{m}	Mass flow rate
Φ	Energy flow rate
Q	Heat flow rate
p	Pressure
ρ	Density
R	Specific gas constant
V	Volume of moist air inside the converter
c_v	Specific heat at constant volume
h	Specific enthalpy
u	Specific internal energy
x	Mass fraction (x_w is specific humidity, which is another term for water vapor mass fraction)
y	Mole fraction
φ	Relative humidity
r	Humidity ratio

T	Temperature
t	Time

The net flow rates into the moist air volume inside the converter are

$$\begin{aligned}\dot{m}_{net} &= \dot{m}_A - \dot{m}_{condense} + \dot{m}_{wS} + \dot{m}_{gS} \\ \Phi_{net} &= \Phi_A + Q_H - \Phi_{condense} + \Phi_S \\ \dot{m}_{w,net} &= \dot{m}_{wA} - \dot{m}_{condense} + \dot{m}_{wS} \\ \dot{m}_{g,net} &= \dot{m}_{gA} + \dot{m}_{gS}\end{aligned}$$

where:

- $\dot{m}_{condense}$ is the rate of condensation.
- $\Phi_{condense}$ is the rate of energy loss from the condensed water.
- Φ_S is the rate of energy added by the sources of moisture and trace gas. \dot{m}_{wS} and \dot{m}_{gS} are the mass flow rates of water and gas, respectively, through port **S**. The values of \dot{m}_{wS} , \dot{m}_{gS} , and Φ_S are determined by the moisture and trace gas sources connected to port **S** of the converter.

Water vapor mass conservation relates the water vapor mass flow rate to the dynamics of the moisture level in the internal moist air volume:

$$\frac{dx_{wI}}{dt} \rho_I V + x_{wI} \dot{m}_{net} = \dot{m}_{w,net}$$

Similarly, trace gas mass conservation relates the trace gas mass flow rate to the dynamics of the trace gas level in the internal moist air volume:

$$\frac{dx_{gI}}{dt} \rho_I V + x_{gI} \dot{m}_{net} = \dot{m}_{g,net}$$

Mixture mass conservation relates the mixture mass flow rate to the dynamics of the pressure, temperature, and mass fractions of the internal moist air volume:

$$\left(\frac{1}{p_I} \frac{dp_I}{dt} - \frac{1}{T_I} \frac{dT_I}{dt} \right) \rho_I V + \frac{R_a - R_w}{R_I} (\dot{m}_{w,net} - x_w \dot{m}_{net}) + \frac{R_a - R_g}{R_I} (\dot{m}_{g,net} - x_g \dot{m}_{net}) + \rho_I \dot{V} = \dot{m}_{net}$$

where \dot{V} is the rate of change of the converter volume.

Finally, energy conservation relates the energy flow rate to the dynamics of the pressure, temperature, and mass fractions of the internal moist air volume:

$$\rho_I c_{vI} V \frac{dT_I}{dt} + (u_{wI} - u_{aI}) (\dot{m}_{w,net} - x_w \dot{m}_{net}) + (u_{gI} - u_{aI}) (\dot{m}_{g,net} - x_g \dot{m}_{net}) + u_I \dot{m}_{net} = \Phi_{net} - p_I \dot{V}$$

The equation of state relates the mixture density to the pressure and temperature:

$$p_I = \rho_I R_I T_I$$

The mixture specific gas constant is

$$R_I = x_{aI} R_a + x_{wI} R_w + x_{gI} R_g$$

The converter volume depends on the rotation of the moving interface:

$$V = V_{dead} + D_{int}\theta_{int}\varepsilon_{int}$$

where:

- V_{dead} is the dead volume.
- D_{int} is the interface volume displacement.
- θ_{int} is the interface rotation.
- ε_{int} is the mechanical orientation coefficient. If **Mechanical orientation is Positive**, $\varepsilon_{int} = 1$. If **Mechanical orientation is Negative**, $\varepsilon_{int} = -1$.

The torque balance on the mechanical interface is

$$\tau_{int} = (p_{env} - p_I)D_{int}\varepsilon_{int}$$

where:

- τ_{int} is the torque from port **R** to port **C**.
- p_{env} is the environment pressure.

Flow resistance and thermal resistance are not modeled in the converter:

$$\begin{aligned} p_A &= p_I \\ T_H &= T_I \end{aligned}$$

When the moist air volume reaches saturation, condensation may occur. The specific humidity at saturation is

$$x_{wsI} = \varphi_{ws} \frac{R_I p_{wsI}}{R_w p_I}$$

where:

- φ_{ws} is the relative humidity at saturation (typically 1).
- p_{wsI} is the water vapor saturation pressure evaluated at T_I .

The rate of condensation is

$$\dot{m}_{condense} = \begin{cases} 0, & \text{if } x_{wI} \leq x_{wsI} \\ \frac{x_{wI} - x_{wsI}}{\tau_{condense}} \rho_I V, & \text{if } x_{wI} > x_{wsI} \end{cases}$$

where $\tau_{condense}$ is the value of the **Condensation time constant** parameter.

The condensed water is subtracted from the moist air volume, as shown in the conservation equations. The energy associated with the condensed water is

$$\Phi_{condense} = \dot{m}_{condense}(h_{wI} - \Delta h_{vapI})$$

where Δh_{vapI} is the specific enthalpy of vaporization evaluated at T_I .

Other moisture and trace gas quantities are related to each other as follows:

$$\phi_{wI} = \frac{y_{wI} p_I}{p_{wsI}}$$

$$y_{wI} = \frac{x_{wI} R_w}{R_I}$$

$$r_{wI} = \frac{x_{wI}}{1 - x_{wI}}$$

$$y_{gI} = \frac{x_{gI} R_g}{R_I}$$

$$x_{aI} + x_{wI} + x_{gI} = 1$$

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Moist Air Volume”.

Assumptions and Limitations

- The converter casing is perfectly rigid.
- Flow resistance between the converter inlet and the moist air volume is not modeled. Connect a Local Restriction (MA) block or a Flow Resistance (MA) block to port **A** to model pressure losses associated with the inlet.
- Thermal resistance between port **H** and the moist air volume is not modeled. Use Thermal library blocks to model thermal resistances between the moist air mixture and the environment, including any thermal effects of a chamber wall.
- The moving interface is perfectly sealed.
- The block does not model the mechanical effects of the moving interface, such as hard stops, friction, and inertia.

Ports

Output

W — Rate of condensation measurement, kg/s

physical signal

Physical signal output port that measures the rate of condensation in the converter.

F — Vector physical signal containing pressure, temperature, humidity, and trace gas levels data

physical signal vector

Physical signal output port that outputs a vector signal. The vector contains the pressure (in Pa), temperature (in K), moisture level, and trace gas level measurements inside the component. Use the Measurement Selector (MA) block to unpack this vector signal.

Conserving

A – Converter inlet

moist air

Moist air conserving port associated with the converter inlet.

H – Temperature inside converter

thermal

Thermal conserving port associated with the temperature of the moist air mixture inside the converter.

R – Rod

mechanical rotational

Mechanical rotational conserving port associated with the moving interface.

C – Case

mechanical rotational

Mechanical rotational conserving port associated with the converter casing.

S – Inject or extract moisture and trace gas

moist air source

Connect this port to port **S** of a block from the Moisture & Trace Gas Sources library to add or remove moisture and trace gas. For more information, see “Using Moisture and Trace Gas Sources”.

Dependencies

This port is visible only if you set the **Moisture and trace gas source** parameter to **Controlled**.

Parameters

Main

Mechanical orientation – Select the converter orientation

Positive (default) | Negative

Select the alignment of moving interface with respect to the volume of moist air inside the converter:

- **Positive** – Increase in the moist air volume results in a positive rotation of port **R** relative to port **C**.
- **Negative** – Increase in the moist air volume results in a negative rotation of port **R** relative to port **C**.

Interface volume displacement – Displaced moist air volume per unit rotation

0.01 m³/rad (default)

Displaced moist air volume per unit rotation of the moving interface.

Interface initial rotation – Rotational offset of port R relative to port C at the start of simulation

0 m (default)

Rotational offset of port **R** relative to port **C** at the start of simulation. A value of 0 corresponds to an initial moist air volume equal to **Dead volume**.

Dependencies

- If **Mechanical orientation** is **Positive**, the parameter value must be greater than or equal to 0.
- If **Mechanical orientation** is **Negative**, the parameter value must be less than or equal to 0.

Dead volume — Volume of moist air when the interface rotation is 0

1e-5 m³ (default)

Volume of moist air when the interface rotation is 0.

Cross-sectional area at port A — Area normal to flow path at the converter inlet

0.01 m² (default)

Cross-sectional area of the converter inlet, in the direction normal to the moist air flow path.

Environment pressure specification — Select a specification method for the environment pressure

Atmospheric pressure (default) | Specified pressure

Select a specification method for the environment pressure:

- **Atmospheric pressure** — Use the atmospheric pressure, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified pressure** — Specify a value by using the **Environment pressure** parameter.

Environment pressure — Pressure outside the converter

0.101325 MPa (default)

Pressure outside the converter acting against the pressure of the converter moist air volume. A value of 0 indicates that the converter expands into vacuum.

Dependencies

Enabled when the **Environment pressure specification** parameter is set to **Specified pressure**.

Moisture and Trace Gas**Relative humidity at saturation — Relative humidity above which condensation occurs**

1 (default)

Relative humidity above which condensation occurs.

Condensation time constant — Time scale for condensation

1e-3 s (default)

Characteristic time scale at which an oversaturated moist air volume returns to saturation by condensing out excess moisture.

Moisture and trace gas source — Model moisture and trace gas levels

None (default) | Constant | Controlled

This parameter controls visibility of port **S** and provides these options for modeling moisture and trace gas levels inside the component:

- **None** — No moisture or trace gas is injected into or extracted from the block. Port **S** is hidden. This is the default.
- **Constant** — Moisture and trace gas are injected into or extracted from the block at a constant rate. The same parameters as in the Moisture Source (MA) and Trace Gas Source (MA) blocks become available in the **Moisture and Trace Gas** section of the block interface. Port **S** is hidden.
- **Controlled** — Moisture and trace gas are injected into or extracted from the block at a time-varying rate. Port **S** is exposed. Connect the Controlled Moisture Source (MA) and Controlled Trace Gas Source (MA) blocks to this port.

Moisture added or removed — Select whether the block adds or removes moisture as water vapor or liquid water

Vapor (default) | Liquid

Select whether the block adds or removes moisture as water vapor or liquid water:

- **Vapor** — The enthalpy of the added or removed moisture corresponds to the enthalpy of water vapor, which is greater than that of liquid water.
- **Liquid** — The enthalpy of the added or removed moisture corresponds to the enthalpy of liquid water, which is less than that of water vapor.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Rate of added moisture — Constant mass flow rate through the block

0 kg/s (default)

Water vapor mass flow rate through the block. A positive value adds moisture to the connected moist air volume. A negative value extracts moisture from that volume.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Added moisture temperature specification — Select specification method for the temperature of added moisture

Atmospheric temperature (default) | Specified temperature

Select a specification method for the moisture temperature:

- **Atmospheric temperature** — Use the atmospheric temperature, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified temperature** — Specify a value by using the **Temperature of added moisture** parameter.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Temperature of added moisture — Moisture temperature

293.15 K (default)

Enter the desired temperature of added moisture. This temperature remains constant during simulation. The block uses this value to evaluate the specific enthalpy of the added moisture only. The specific enthalpy of removed moisture is based on the temperature of the connected moist air volume.

Dependencies

Enabled when the **Added moisture temperature specification** parameter is set to **Specified temperature**.

Rate of added trace gas — Constant mass flow rate through the block

0 kg/s (default)

Trace gas mass flow rate through the block. A positive value adds trace gas to the connected moist air volume. A negative value extracts trace gas from that volume.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to **Constant**.

Added trace gas temperature specification — Select specification method for the temperature of added trace gas

Atmospheric temperature (default) | Specified temperature

Select a specification method for the trace gas temperature:

- **Atmospheric temperature** — Use the atmospheric temperature, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified temperature** — Specify a value by using the **Temperature of added trace gas** parameter.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to **Constant**.

Temperature of added trace gas — Trace gas temperature

293.15 K (default)

Enter the desired temperature of added trace gas. This temperature remains constant during simulation. The block uses this value to evaluate the specific enthalpy of the added trace gas only. The specific enthalpy of removed trace gas is based on the temperature of the connected moist air volume.

Dependencies

Enabled when the **Added trace gas temperature specification** parameter is set to **Specified temperature**.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Translational Mechanical Converter (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Rotational Mechanical Converter (TL)

Interface between thermal liquid and mechanical rotational networks

Library: Simscape / Foundation Library / Thermal Liquid / Elements



Description

The Rotational Mechanical Converter (TL) block models an interface between a thermal liquid network and a mechanical rotational network. The block converts thermal liquid pressure into mechanical torque and vice versa. It can be used as a building block for rotary actuators.

The converter contains a variable volume of liquid. The temperature evolves based on the thermal capacity of this volume. If **Model dynamic compressibility** is set to **On**, then the pressure also evolves based on the dynamic compressibility of the liquid volume. If **Mechanical orientation** is set to **Positive**, then an increase in the liquid volume results in a positive rotation of port **R** relative to port **C**. If **Mechanical orientation** is set to **Negative**, then an increase in the liquid volume results in a negative rotation of port **R** relative to port **C**.

Port **A** is the thermal liquid conserving port associated with the converter inlet. Port **H** is the thermal conserving port associated with the temperature of the liquid inside the converter. Ports **R** and **C** are the mechanical rotational conserving ports associated with the moving interface and converter casing, respectively.

Mass Balance

The mass conservation equation in the mechanical converter volume is

$$\dot{m}_A = \varepsilon \rho D \Omega + \begin{cases} 0, & \text{if fluid dynamic compressibility is off} \\ V \rho \left(\frac{1}{\beta} \frac{dp}{dt} + \alpha \frac{dT}{dt} \right), & \text{if fluid dynamic compressibility is on} \end{cases}$$

where:

- \dot{m}_A is the liquid mass flow rate into the converter through port A.
- ε is the mechanical orientation of the converter (1 if positive, -1 if negative).
- ρ is the liquid mass density.
- D is the converter displacement.
- Ω is the angular velocity of the converter interface.
- V is the liquid volume inside the converter.
- β is the liquid bulk modulus inside the converter.
- α is the coefficient of thermal expansion of the liquid.
- p is the liquid pressure inside the converter.

- T is the liquid temperature inside the converter.

Momentum Balance

The momentum conservation equation in the mechanical converter volume is

$$\tau = -\varepsilon(p - p_{\text{Atm}})D,$$

where:

- τ is the torque the liquid exerts on the converter interface.
- p_{Atm} is the atmospheric pressure.

Energy Balance

The energy conservation equation in the mechanical converter volume is

$$\frac{d(\rho u V)}{dt} = \phi_A + Q_H - pD\varepsilon\Omega,$$

where:

- u is the liquid internal energy in the converter.
- ϕ_A is the total energy flow rate into the mechanical converter volume through port A.
- Q_H is the heat flow rate into the mechanical converter volume.

Assumptions and Limitations

- Converter walls are not compliant. They cannot deform regardless of internal pressure and temperature.
- The converter contains no mechanical hard stops. To include hard stops, use the Rotational Hard Stop block.
- The flow resistance between the inlet and the interior of the converter is negligible.
- The thermal resistance between the thermal port and the interior of the converter is negligible.
- The kinetic energy of the fluid in the converter is negligible.

Ports

Conserving

A — Converter inlet

thermal liquid

Thermal liquid conserving port associated with the converter inlet.

H — Temperature inside converter

thermal

Thermal conserving port associated with the temperature of the liquid inside the converter.

R — Rod

mechanical rotational

Mechanical rotational conserving port associated with the moving interface.

C – Case

mechanical rotational

Mechanical rotational conserving port associated with the converter casing.

Parameters**Main****Mechanical orientation – Select the converter orientation**

Positive (default) | Negative

Select the alignment of moving interface with respect to the converter liquid volume:

- **Positive** — Increase in the liquid volume results in a positive rotation of port **R** relative to port **C**.
- **Negative** — Increase in the liquid volume results in a negative rotation of port **R** relative to port **C**.

Interface volume displacement – Displaced liquid volume per unit rotation

$1.2e-4 \text{ m}^3/\text{rad}$ (default)

Displaced liquid volume per unit rotation of the moving interface.

Interface initial rotation – Rotational offset of port R relative to port C at the start of simulation

0 rad (default)

Rotational offset of port **R** relative to port **C** at the start of simulation. A value of 0 corresponds to an initial liquid volume equal to **Dead volume**.

Dependencies

- If **Mechanical orientation** is **Positive**, the parameter value must be greater than or equal to 0.
- If **Mechanical orientation** is **Negative**, the parameter value must be less than or equal to 0.

Dead volume – Volume of liquid when the interface rotation is 0

$1e-5 \text{ m}^3$ (default)

Volume of liquid when the interface rotation is 0.

Cross-sectional area at port A – Area normal to flow path at the converter inlet

0.01 m^2 (default)

The cross-sectional area of the converter inlet, in the direction normal to the flow path.

Environment pressure specification – Select a specification method for the environment pressure

Atmospheric pressure (default) | Specified pressure

Select a specification method for the pressure outside the converter:

- **Atmospheric pressure** — Use the atmospheric pressure, specified by the Thermal Liquid Settings (TL) or Thermal Liquid Properties (TL) block connected to the circuit.
- **Specified pressure** — Specify a value by using the **Environment pressure** parameter.

Environment pressure — Pressure outside the converter

0.101325 MPa (default)

Pressure outside the converter acting against the pressure of the converter liquid volume. A value of 0 indicates that the converter expands into vacuum.

Dependencies

Enabled when the **Environment pressure specification** parameter is set to Specified pressure.

Effects and Initial Conditions**Fluid dynamic compressibility — Select whether to model fluid dynamic compressibility**

On (default) | Off

Select whether to account for the dynamic compressibility of the liquid. Dynamic compressibility gives the liquid density a dependence on pressure and temperature, impacting the transient response of the system at small time scales.

Initial liquid pressure — Liquid pressure at time zero

0.101325 MPa (default)

Liquid pressure in the converter at the start of simulation.

Dependencies

Enabled when the **Fluid dynamic compressibility** parameter is set to On.

Initial liquid temperature — Liquid temperature at time zero

293.15 K (default)

Liquid temperature in the converter at the start of simulation.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Translational Mechanical Converter (TL)

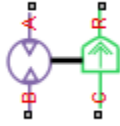
Topics

“Modeling Thermal Liquid Systems”

Introduced in R2013b

Rotational Pneumatic-Mechanical Converter

Interface between pneumatic and mechanical rotational domains



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Rotational Pneumatic-Mechanical Converter block provides an interface between the pneumatic and the mechanical rotational domains. Use it as a building block for modeling pneumatic pumps and motors.

The pneumatic flow rate and mechanical rotation are related by the following equations:

$$Q = D \cdot \omega$$

$$T = \begin{cases} D \cdot (p_A - p_B) \cdot \eta & \text{for } (p_A - p_B) \cdot \omega \geq 0 \\ D \cdot (p_A - p_B) / \eta & \text{for } (p_A - p_B) \cdot \omega < 0 \end{cases}$$

where

Q	Volumetric flow rate flowing from port A to port B
p_A	Pressure at port A
p_B	Pressure at port B
ω	Shaft angular rotational speed
T	Mechanical torque
D	Volumetric displacement per unit rotation
η	Converter efficiency

The torque equation depends on the direction of power flow, and is always such that the conversion results in some thermal losses.

From considering energy flow, the heat flow out (q_o) of the converter must equate to the heat flow in (q_i) minus mechanical work done. Therefore, the heat equations are:

$$q_i = |G| \cdot c_p \cdot T_i$$

$$q_o = \begin{cases} q_i - D \cdot (p_A - p_B) \cdot \omega \cdot \eta & \text{for } (p_A - p_B) \cdot \omega \geq 0 \\ q_i - D \cdot (p_A - p_B) \cdot \omega / \eta & \text{for } (p_A - p_B) \cdot \omega < 0 \end{cases}$$

where G is the mass flow rate.

If the pneumatic pressure drops from port A to port B, then the resulting torque is positive acting from the mechanical port C to port R.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see "Set Priority and Initial Target for Block Variables".

Basic Assumptions and Limitations

- Conversion efficiency is constant, that is, it does not depend on torque or speed.
- Gas flow rate is linearly dependent of pump speed.
- The process is adiabatic, that is, there is no heat transfer with the environment.
- Gravitational effects can be neglected.

Parameters

Displacement

Specify the effective piston displacement, as volume per unit angle. The default value is $.001 \text{ m}^3/\text{rad}$.

Efficiency

Specify the converter efficiency. The default value is 0.2 .

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the converter inlet.

B

Pneumatic conserving port associated with the converter outlet.

R

Mechanical rotational conserving port associated with the piston (rod).

C

Mechanical rotational conserving port associated with the reference (case).

See Also

Rotary Pneumatic Piston Chamber

Introduced in R2009b

Rotational Spring

Ideal spring in mechanical rotational systems



Library

Mechanical Rotational Elements

Description

The Rotational Spring block represents an ideal mechanical rotational linear spring, described with the following equations:

$$T = K \cdot \varphi$$

$$\varphi = \varphi_{init} + \varphi_R - \varphi_C$$

$$\omega = \frac{d\varphi}{dt}$$

where

T	Torque transmitted through the spring
K	Spring rate
φ	Relative displacement angle (spring deformation)
φ_{init}	Spring preliminary winding (spring offset)
φ_R, φ_C	Absolute angular displacements of terminals R and C, respectively
ω	Relative angular velocity
t	Time

The block positive direction is from port R to port C. This means that the torque is positive if it acts in the direction from R to C.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Spring rate

Spring rate. The default value is 10 N*m/rad.

Ports

The block has the following ports:

R

Mechanical rotational conserving port.

C

Mechanical rotational conserving port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

[Rotational Hard Stop](#) | [Rotational Damper](#) | [Rotational Friction](#)

Introduced in R2007a

Simscape Bus

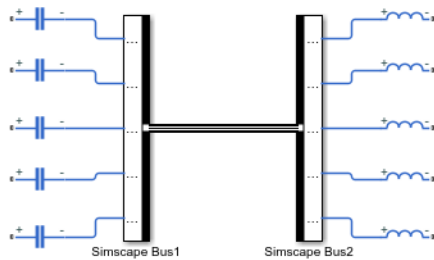
Bus for conserving connection lines

Library: Simscape / Utilities

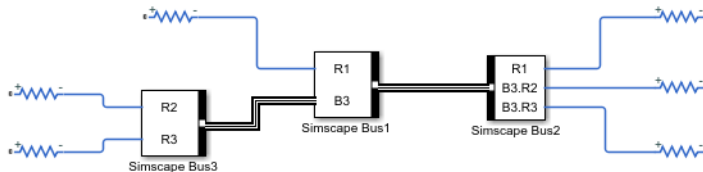


Description

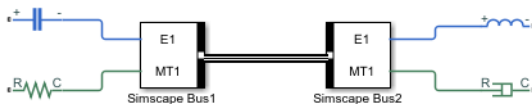
The Simscape Bus block bundles conserving connections into a Simscape Bus line. You can also use this block to access one or more connections from an existing Simscape Bus line. Physical connection lines are nondirectional, therefore, the block can serve as both a bus creator and a bus selector. Typically, a model would have two Simscape Bus blocks, facing each other, with their bus ports connected.



However, a model can also contain a hierarchy of Simscape Bus blocks.



Conserving connections bundled into a Simscape Bus line can belong to different domains.






The block bundles only the Simscape conserving connections, that is, nondirectional connection lines between Simscape conserving ports or Simscape Multibody ports. It does not work with other types of connections, such as physical signal lines or Simscape Electrical™ Specialized Power Systems connection lines. For more information on conserving connections, see “Physical Conserving Ports”.

The Simscape Bus block can have ports on two sides:

- Parent (bundle) side, indicated by the black ribbon. This side contains a single bus port, which bundles all the connections coming into the block on the children side.
- Children (elements) side, opposite the parent side. This side does not have any ports by default. You add ports by dragging the connection lines to that side of the block, or by using the block dialog box. Elements coming into the block on the children side can be either conserving connection lines or Simscape Bus lines. Every time you add a port on the children side, the corresponding connection is added to the connections list inside the block.

The names of the connections are shown on the block icon, next to the corresponding ports on the children side, and in the **Hierarchy Strings** list in the block dialog box. You can edit these names in the dialog box or directly on the icon, if needed, by clicking and typing a new name.

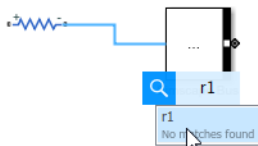
The block dialog box contains these buttons:

-  Add a new connection name to the **Hierarchy Strings** list. The default connection names are Connection1, Connection2, and so on. You can edit these names, as needed, by selecting a row, then clicking it again and typing the new name. When you add a connection using this button, a round conserving port appears on the children side of the block, similar to that of a Connection Port block. The domain type of this conserving port is defined once you connect it to a conserving port on another block. After that, you can connect only the same type of port to the corresponding connection on the second bus. You can also add connections directly on the canvas, by dragging a connection line from a conserving port on another block to the children side of the Simscape Bus block.
-  Delete the selected connection from the **Hierarchy Strings** list. For unconnected ports, you can also delete them or edit their names directly on the canvas.
-  Refresh the **Hierarchy Strings** list after adding or removing connections on the canvas.

Working with the Block on Model Canvas

To add connections to a Simscape Bus block:

- 1 Select a conserving port on another block. Or, to create a hierarchy of Simscape Bus blocks, select a bus port on the parent side of another Simscape Bus block.
- 2 Drag a connection line from the selected port to the children side of the first Simscape Bus block.
- 3 A new connection with the default name is automatically added to the block connection list. You can edit the connection name, as needed, in the upper blue field. When done editing, select the connection name in the lower blue field to create the connection.



When you delete a connection line, the corresponding port on the children side of the block is not deleted. You can reuse it by connecting another block.

To delete unused ports:

- Select a round conserving port on the children side of a Simscape Bus block. Make sure to select just the port, not the whole block.
- From the ellipsis menu, select **Delete Port**. Deleting a port also deletes the corresponding name in the block connection list.

Ports

Conserving

Bus — Simscape Bus port

bus

Port connected to a Simscape Bus line, which represents a bundle of conserving connection lines. The bundle contains all the lines connected to the block on the opposite side.



This opposite side, by default, does not have ports. You add ports by dragging the connection lines to that side of the block, or by using the block dialog box. Elements coming into the block on the children side can be either conserving connection lines or Simscape Bus lines. Every time you add a port on the children side, the corresponding connection is added to the connections list inside the block.

Parameters

Hierarchy Strings — List of connection names

empty (default) | connection names

List of connection names corresponding to the conserving lines that are connected to the Simscape Bus block. By default, this list is empty. Every time you add a connection, a conserving port appears

on the children side of the block. Use the  or  buttons in the block dialog box to add or remove connections.

You can also add and remove connections directly on the model canvas, as described in “Working with the Block on Model Canvas” on page 1-626.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Bus Creator | Bus Selector | Connection Port

Topics

“Types of Composite Signals”

“Build and Edit a Model Interactively”

Introduced in R2018b

Simscape Component

Deploy Simscape language component as custom block in model diagram



Library

Utilities

Description

The Simscape Component block lets you generate a Simscape block directly from a textual component file, skipping the library build process.

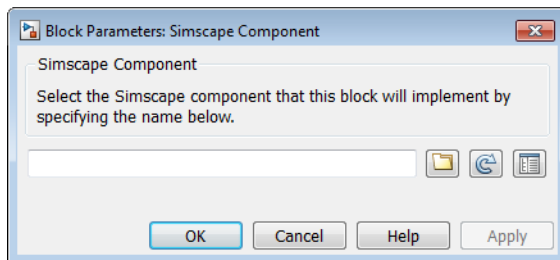
You can use the `ssc_build` command to generate a custom block library from a complete package of Simscape component files. However, you can also do on-the-fly conversion of a component file directly into a custom block in your model by using a Simscape Component block.

To deploy a component file directly into a block diagram:

- 1 Open the Simscape > Utilities library and add the Simscape Component block to your model. At first, the block does not point to any component file. Therefore, it does not have any ports, and the block icon states it is *Unspecified*.



- 2 Double-click the block to open the source file selector dialog box.



- 3 Browse to the desired Simscape language component file. The file does not have to be in a package. However, the directory where the file resides has to be on the MATLAB path. If the file resides in a package, then the package parent directory must be on the MATLAB path.
- 4 If you selected a file that is not on the MATLAB path, a File Not On Path dialog opens. Click **Add**.
- 5 Click **OK** to close the file selector dialog box. The block icon changes, and the block acquires the ports, parameters, and variables based on the selected source component.

If you now double-click the block, its dialog box has all the same elements as if it was generated from the component file through a library build process: name, description, parameters, variables, and a link to **Source code**. However, the block dialog box has an additional **Choose source** button that lets you point the block to a different component file.

Source File Selector Dialog Box

When a Simscape Component block points to a valid Simscape language component file, its dialog box has the name, description, parameters, and variables defined by that source file. The source file selector dialog box opens in the following cases:

- When you add a new Simscape Component block to your model, to let you select the source component file.
- When you click the **Choose source** button in an existing Simscape Component block dialog, to let you select a different source component file.
- When you open a model with an unresolved Simscape Component block (because the underlying source component is no longer on the path) and double-click the block, to let you resolve the source location or select a different source component file.

A source component file can be located:

- In the current working directory
- In a directory on the MATLAB path
- In a package, with the package parent directory being on the MATLAB path




To specify the source file, you can either browse to it or type its name directly into the name field of the selector dialog box:

- If you type the name of the file, it must satisfy the location requirements, listed above. If the file resides in a package, you must provide the file name relative to the package root. If these conditions are not met when you type the file name, the source location cannot be resolved, and you get an error message.
- If you browse to the source file, you can select any Simscape source or protected file (that is, any file with the `.ssc` or `.sscp` extension). However, if the selected file does not satisfy the location requirements, a File Not On Path dialog opens. Click **Add** to add the appropriate directory to the MATLAB path and resolve the source location. If you click **Cancel**, the source location cannot be resolved and the source selection process is cancelled.

When the source location is successfully resolved, the name of the source file appears in the text field of the source file selector dialog box. The block name, description (if available), and the link to source code appear in the preview pane. Click **OK** or **Apply** to point the Simscape Component block to the selected source file:

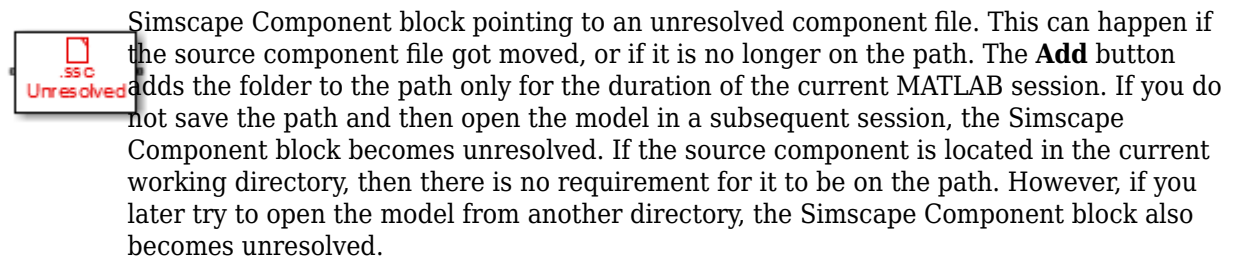
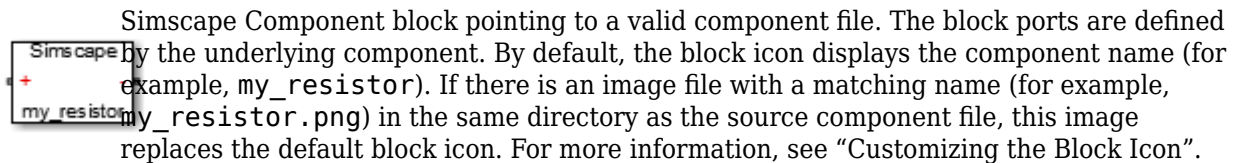
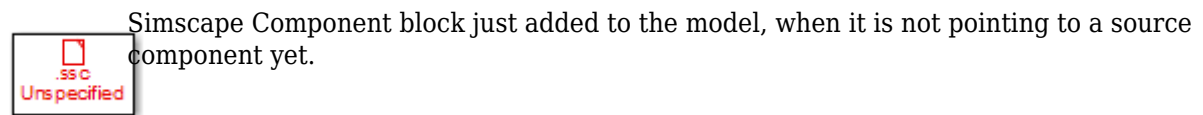
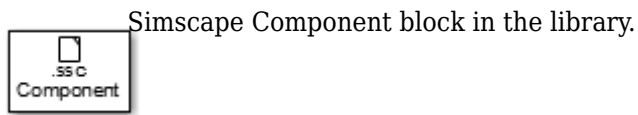
- Clicking **Apply** updates the block icon, closes the file selector dialog box, and opens the block dialog box, which contains the parameters and variables.
- Clicking **OK** updates the block icon and closes the file selector dialog box, but does not open the block dialog box. Double-click the block if you wish to view or modify the block parameters and variables.

The buttons next to the text field in the source file selector dialog box perform the following actions:

-  Opens a file browser, to let you select the source component file. By default, the browser displays only the files with the `.ssc` or `.sscp` extension. If the Simscape Component block currently points to a source file, and the source location is successfully resolved, the browser opens in the directory where that source file resides. Otherwise, the browser opens in the current working directory.
-  Refreshes the preview pane of the source file selector dialog box. If you type the source component name directly into the text field of the selector dialog box, the preview pane does not automatically update. If you want to preview the block name, description, or source code of the source component before finalizing the selection, click this button.
-  Opens the block dialog box, which contains the parameters and variables based on the selected component. This button and the **Choose source** button let you toggle between the file selector and the block dialog box for the Simscape Component block.

Block Icon Appearance

The Simscape Component block icon appearance changes depending on the block state.



Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Topics

- “Deploy a Component File in Block Diagram”
- “Switch Between Different Source Components”

“Prototype a Component and Get Instant Feedback”
“Selecting Component File Directly from Block”
“Customizing the Block Name and Appearance”

Introduced in R2016a

Simulink-PS Converter

Convert Simulink input signal into physical signal





Library

Utilities





Description

The Simulink-PS Converter block converts the input Simulink signal into a physical signal. Use this block to connect Simulink sources or other Simulink blocks to the inputs of a Physical Network diagram.

To convey signal conversion while taking up minimal canvas space, the block icon changes dynamically based on whether it is connected to other blocks.

When Block Is...	Block Icon
Unconnected	
Connected to other blocks	

The block icon also changes based on the value of the **Input filtering order** parameter, to indicate whether filtering is being applied to the input signal.

Input Filtering Order	Unconnected Block Icon	Connected Block Icon
First-order filtering		
Second-order filtering		

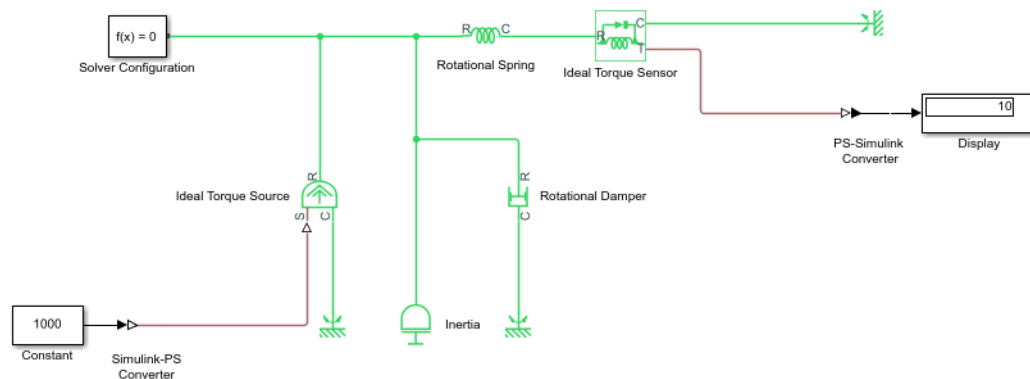
You specify the desired unit as the **Input signal unit** parameter. The parameter value controls the unit of the physical signal at the output port of the block, which serves as the input signal for the Simscape physical network.

Note You can also specify a physical unit on the Simulink signal connected to the input port of the block. In this case, the software checks that the two units match (for more information, see “Working with Simulink Units”). The Simulink signal units do not propagate into the physical network.

If you leave the block unitless, with the **Input signal unit** parameter set to 1, then the physical signal unit is inferred from the destination block. The default destination block units are meter-

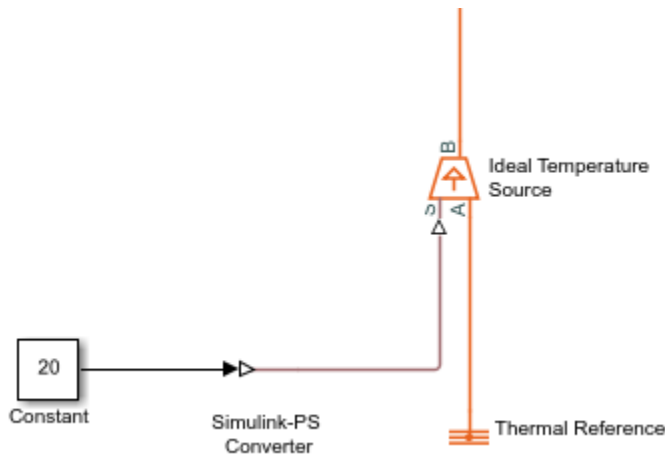
kilogram-second or MKS (SI). If you specify different units, commensurate with the expected default units of the destination block input, then the unit manager attaches these units to the input Simulink signal value and performs the necessary unit conversion when providing the signal to the destination block.

In the diagram below, the Ideal Torque Source block expects a torque signal, in N*m, on its S port. The Constant source block provides the value for this input signal. If you left the Simulink-PS Converter block unitless, the Ideal Torque Source block would generate torque of 1000 N*m. The parameters of other blocks in this example are chosen so that the output value of the Ideal Torque Sensor block is equal to the torque generated by the Ideal Torque Source block, and therefore the Display block would show the value of 1000. If you change the **Input signal unit** parameter value in the Simulink-PS Converter block to N*cm, the unit manager performs the conversion and the Ideal Torque Source block generates torque of 10 N*m; the torque value in the Display block changes to 10, as shown in the diagram.



When the input signal is related to thermodynamic variables and contains units of temperature, you must decide whether affine conversion needs to be applied. For more information, see “When to Apply Affine Conversion”. Usually, if the input signal represents a relative temperature, that is, a change in temperature, you need to apply linear conversion, $\Delta T_{new} = L * \Delta T_{old}$ (the default method). However, if the input signal represents an absolute temperature, you need to apply affine conversion, $T_{new} = L * T_{old} + O$.

For example, in the Simulink-PS Converter block shown in the following diagram, if you type degC in the **Input signal unit** field and select the **Apply affine conversion** check box, the temperature generated by the Ideal Temperature Source block is equal to 293.15 K. However, if you leave the **Apply affine conversion** check box clear, the output of the Ideal Temperature Source block is 20 K.



Note Untyped inputs do not support affine units. If you specify affine units in a Simulink-PS Converter block and then connect it directly to an untyped input port, the signal value is converted to the corresponding fundamental unit and further mathematical operations are performed with that value.

Parameters

- “Units” on page 1-634
- “Input Handling” on page 1-634

Units

Input signal unit

Units to be assigned to the physical signal. These units must be commensurate with the expected default units of the destination block input. You can select a unit from the drop-down list, or type the desired unit name, such as *rpm*, or a valid expression, such as *rad/s*. For more information and a list of unit abbreviations, see “How to Specify Units in Block Dialogs” and “Unit Definitions”. The default value is 1, which means that the units of the physical signal at the block output match the expected default units of the destination block input.

Apply affine conversion

This check box is applicable only for units that can be converted either with or without an affine offset, such as thermal units. For more information, see “Thermal Unit Conversions”.

Input Handling

Filtering and derivatives

This parameter lets you decide whether to filter input and thus provide time derivatives of the input signal, or provide the time derivatives through additional input ports on the Simulink-PS Converter block:

- **Provide signals** — Select whether you want to provide just the input signal, or also provide time derivatives of the input signal as additional input signals, by using the **Provided signals** parameter. The default input handling options are **Provide signals** and **Input only**. If you use an explicit solver, it is recommended that you provide input derivatives by

selecting one of the other options. For more information, see “Filtering Input Signals and Providing Time Derivatives”.

- **Filter input, derivatives calculated** — Filter the input through a low-pass filter, which also provides input derivatives. In this case, the input signal is modified (through filtering) before being converted to a physical signal. The first-order filter provides one derivative, while the second-order filter provides the first and second derivatives. If you use this option, set the appropriate **Input filtering time constant** parameter value, as described below.
- **Zero derivatives (piecewise constant)** — If your input signal is piecewise constant (such as step), this option lets you explicitly set the input derivatives to zero.

Provided signals

This parameter is applicable only if the **Filtering and derivatives** parameter is set to **Provide signals**. It lets you provide time derivatives of the input signal as additional input signals:

- **Input only** — Provide just the input signal. This is the default. If you select this option, the block has one Simulink input port and one physical signal output port.
- **Input and first derivative** — If you select this option, an additional Simulink input port appears on the Simulink-PS Converter block, to let you connect the signal providing input derivative.
- **Input and first two derivatives** — If you select this option, two additional Simulink input ports appear on the Simulink-PS Converter block, to let you connect the signals providing input derivatives.

Input filtering order

This parameter is applicable only if the **Filtering and derivatives** parameter is set to **Filter input, derivatives calculated**. It lets you specify the number of time derivatives computed for the input signal by selecting the filter order:

- **First-order filtering** — Provides only the first derivative.
- **Second-order filtering** — Provides the first and second derivatives.

Input filtering time constant (in seconds)

This parameter is applicable only if the **Filtering and derivatives** parameter is set to **Filter input, derivatives calculated**. It specifies the filter time constant, which 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. The default value is .001 s.

Restricted Parameters

When your model is in Restricted editing mode, you cannot modify any of the block parameters, with the following exception: if the **Filtering and derivatives** parameter is set to **Filter input, derivatives calculated** prior to entering Restricted mode, you can change the value of the **Input filtering time constant** parameter.

Ports

The block has one, two, or three Simulink input ports (depending on the **Provided signals** parameter value), located on its left side, and a physical signal output port, located on its right side (in the block default orientation).

See Also

PS-Simulink Converter

Introduced in R2007a

Slider-Crank

Generic slider-crank mechanism

Library: Simscape / Foundation Library / Mechanical / Mechanisms



Description

The Slider-Crank block represents the slider-crank mechanism as a converter between the continuous rotational motion of the crank and the oscillating translational motion of the slider.

The mechanism has two connections:

- Port C corresponds to the crank and is a mechanical rotational conserving port.
- Port S corresponds to the slider and is a mechanical translational conserving port.

The **Crank radius** and **Rod length** parameters describe the mechanism geometry. The **Crank inertia** parameter helps move the mechanism through the positions at the top and bottom when the rod is exactly aligned with the center of the crank.

The purpose of the **Slider stiffness** and **Slider damping** parameters is to reduce inertia coupling between the ports, which can cause computational issues. The block applies the stiffness and damping between the ideal position of the slider (as calculated by the geometry) and the actual position of the translational conserving port, **S**. This compliance avoids a direct nonlinear coupling between the ports. Higher stiffness causes the motion to be closer to ideal, but increases the system stiffness. The damping helps smooth the ringing that can be caused by the stiffness.

Variables

You can set initialization targets for the crank angle and velocity at the start of simulation, using both positive and negative values.

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

C – Crank

mechanical rotational

Mechanical rotational conserving port associated with the crank.

S – Slider

mechanical translational

Mechanical translational conserving port associated with the slider.

Parameters

Crank radius — Radius of the crank

0.1 m (default)

The radius of the crank. The parameter value must be greater than zero.

Rod length — Length of the rod

0.2 m (default)

Length of the rod connecting the crank and the slider. The parameter value must be greater than **Crank radius**.

Crank inertia — Inertia of the crank

0.01 kg*m² (default)

The inertia of the crank, which helps move the mechanism through the positions at the top and bottom when the rod is exactly aligned with the center of the crank. The parameter value must be greater than zero.

Slider stiffness — Stiffness applied to the slider position

1e6 N/m (default)

Together with **Slider damping**, introduces a compliance that helps reduce inertia coupling between the ports. Higher stiffness causes the motion to be closer to ideal, but increases the system stiffness. The parameter value must be greater than zero.

Slider damping — Damping applied to the slider position

1e3 N/(m/s) (default)

Together with **Slider stiffness**, introduces a compliance that helps reduce inertia coupling between the ports. Higher damping values smooth the ringing that can be caused by the stiffness. The parameter value must be greater than zero.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

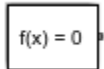
See Also

Gear Box | Lever | Wheel and Axle

Introduced in R2019a

Solver Configuration

Physical Networks environment and solver configuration



Library

Utilities

Description

Each physical network represented by a connected Simscape block diagram requires solver settings information for simulation. The Solver Configuration block specifies the solver parameters that your model needs before you can begin simulation.

Each topologically distinct Simscape block diagram requires exactly one Solver Configuration block to be connected to it.

Ports

The block has one conserving port. You can add this block anywhere on a physical network circuit by creating a branching point and connecting it to the only port of the Solver Configuration block.

Parameters

Start simulation from steady state

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. For more information, see “Initial Conditions Computation”. Simulation then starts from this steady state.

For models compatible with frequency-and-time equation formulation, when you select this check box, the solver attempts to perform sinusoidal steady-state initialization. In other words, initialization is performed using frequency-time equations, and then the simulation proceeds using the actual equation formulation and other options selected in the Solver Configuration block. For more information, see “Frequency and Time Simulation Mode”.

Note Using the **Initial state** option on the **Data Import/Export** pane of the Configuration Parameters dialog box overrides the **Start simulation from steady state** option.

Consistency tolerance

This parameter affects the nonlinear solver used for computing initial conditions and for transient initialization. It determines how accurately the algebraic constraints are to be satisfied at the beginning of simulation and after every discrete event (for example, a discontinuity resulting from

a valve opening, a hard stop, and so on). Decrease the parameter value (that is, tighten tolerance) to obtain a more reliable time simulation. Increase the parameter value (that is, relax the tolerance) if solving for initial conditions failed to converge, or to reduce the computation time.

The default value is $1e-9$, which is applicable to most cases.

Use local solver

Lets you use a sample-based local solver with a sample time specified by the **Sample time** parameter. In sample-based simulation, all the Physical Network states, which are otherwise continuous, become represented to Simulink as discrete states. The solver updates the states once per time step. This option is especially useful for generated code or hardware-in-the-loop (HIL) simulations.

Note If you use a local solver, simultaneous use of Simulink or Simulink Control Design™ linearization tools is not recommended.

Solver type

Select the solver type used for updating the states: Backward Euler, Trapezoidal Rule, or Partitioning. The **Use local solver** check box must be selected. The default is Backward Euler.

Sample time

Specify the sample time for the local solver. The **Use local solver** check box must be selected. The sample time must be positive. The default is $.001$ s.

Partition method

Select whether to prioritize speed or robustness when using Partitioning local solver:

- **Robust simulation** — Increase simulation robustness.
- **Fast simulation** — Improve simulation performance.

The **Use local solver** check box must be selected and the **Solver type** must be set to Partitioning. The default is **Robust simulation**.

Partition storage method

When you use the Partitioning solver, it solves the small switched linear equations consecutively. You can choose to store the matrix inverses, to improve the simulation performance. Then, if the same configuration is detected in a subsequent time step, the partitioning solver uses the stored matrix inverses, instead of recomputing them. Select the method used for storing partitioning data:

- **As needed** — Compute matrix inverses during simulation, as needed. This method does not require as much memory but can result in performance spikes.
- **Exhaustive** — Compute and store matrix inverses before simulation. This method improves the simulation performance but requires more memory. Use the parameter **Partition memory budget [kB]** to specify the maximum allowed memory budget for storing the data.

The **Use local solver** check box must be selected and the **Solver type** must be set to Partitioning. The default is **As needed**.

Partition memory budget [kB]

Specify the maximum memory budget, in kB, allowed for storing cached partition data. The **Partition storage method** method must be set to **Exhaustive**. The default is 1024 kB. If this

budget is exceeded, simulation errors out. You can adjust this value based on your available memory resources and on the **Total memory estimate** data in the Statistics Viewer. For more information, see “Model Statistics Available when Using the Partitioning Solver”.

Use fixed-cost runtime consistency iterations

Lets you perform transient initialization at a fixed computational cost.

If you select this check box, you can specify the number of nonlinear and mode iterations for transient initialization. If the system does not converge once it performs the specified number of iterations, it ignores the failure and goes to the next step.

If you clear the check box, the system uses a more robust and time-consuming algorithm, performing as many iterations as necessary to reach convergence, and errors out if it fails to reach convergence at the time of transient initialization.

Nonlinear iterations

Specify the number of Newton iterations to be performed at the time of transient initialization. The **Use fixed-cost runtime consistency iterations** check box must be selected. The default number is 3.

Mode iterations

Specify the number of mode iterations to be performed at the time of transient initialization. The **Use fixed-cost runtime consistency iterations** check box must be selected. The default number is 2.

Only one major mode update per step is performed when using local solvers, therefore this parameter is not available if the **Use local solver** check box is selected.

Compute impulses

Lets you manage computational cost of impulse detection during transient initialization, both for global and local solvers. This check box becomes available when the **Use fixed-cost runtime consistency iterations** check box is selected.

Event-based methods of state reinitialization and impulse handling let you model physical phenomena, such as collisions and bouncing balls, and provide a significant boost in simulation speed for such models. However, impulse detection can add cost to transient initialization. This cost is proportional to the number of impulse iterations performed to reach convergence.

If you select the **Compute impulses** check box, you can specify the number of impulse iterations to perform during transient initialization. If the system does not converge upon reaching these numbers, it ignores the failure and goes to the next step.

If you clear the check box, the system computes impulses as many times as necessary to reach convergence.

Impulse iterations

Specify the number of impulse iterations at the time of transient initialization. The **Compute impulses** check box must be selected. The default number is 2.

Linear Algebra

Specifies how the solver treats matrices. The parameter can have one of three values: **auto**, **Sparse**, or **Full**. The default value of the parameter is **auto**. With this setting, the solver automatically selects the appropriate option, either sparse or full, for treating the matrices.

Your choice of this parameter value is implemented in both model simulation and code generated from your model.

Equation formulation

Specifies how the solver treats sinusoidal variables. The parameter can have one of two values: `Time` or `Frequency` and `time`. The default value of the parameter is `Time`.

Use the `Frequency` and `time` value to speed up simulation of systems with a single nominal frequency. For more information, see “Frequency and Time Simulation Mode”.

Delay memory budget [kB]

Specify the maximum memory budget, in kB, allowed for processing delays when simulating models that contain either blocks from the Delays library, or custom blocks using the `delay` Simscape language construct. The purpose of this parameter is to protect against excessive memory swapping. If this budget is exceeded, simulation errors out. You can adjust this value based on your available memory resources. The default number is 1024 kB.

Apply filtering at 1-D/3-D connections when needed

This option is applicable only for models that connect blocks from Simscape Multibody Second Generation library to Simscape blocks, or blocks from other add-on products. Use the Statistics Viewer to determine whether your model has 1-D/3-D connections. For more information, see “1-D/3-D Interface Statistics”.

When a Simscape Multibody block is connected directly to a Simscape network, an additional derivative may be required for the network to be solved. When you select this check box, the solver automatically applies input filtering to the signal entering the Simulink-PS Converter block to obtain this additional derivative. The **Filtering time constant** parameter provides the time constant for the delay.

Note This check box is selected by default. If you clear it, and the 1-D/3-D connection requires the additional derivative, the solver issues an error message.

Filtering time constant

This parameter specifies the filtering time constant for when the **Apply filtering at 1-D/3-D connections when needed** check box is selected. The parameter value applies globally to all connections belonging to the network that includes this Solver Configuration block. The default value is 0.001 s.

See Also

- “How Simscape Simulation Works”
- “Setting Up Solvers for Physical Models”
- “Making Optimal Solver Choices for Physical Simulation”
- “Finding an Operating Point”
- “Linearizing at an Operating Point”

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

Introduced in R2007a

Spectrum Analyzer

Display frequency spectrum

Library: Simscape / Utilities



Description

Note The Spectrum Analyzer block in the Simscape product contains a subset of functionality of the DSP System Toolbox™ block with the same name. This page describes the block configuration and functionality available with a Simscape license. If you also have a DSP System Toolbox license, then the Spectrum Analyzer block in the **Simscape > Utilities** library is identical to the block in the **DSP System Toolbox > Sinks** library. For more information, see Spectrum Analyzer in the DSP System Toolbox documentation.

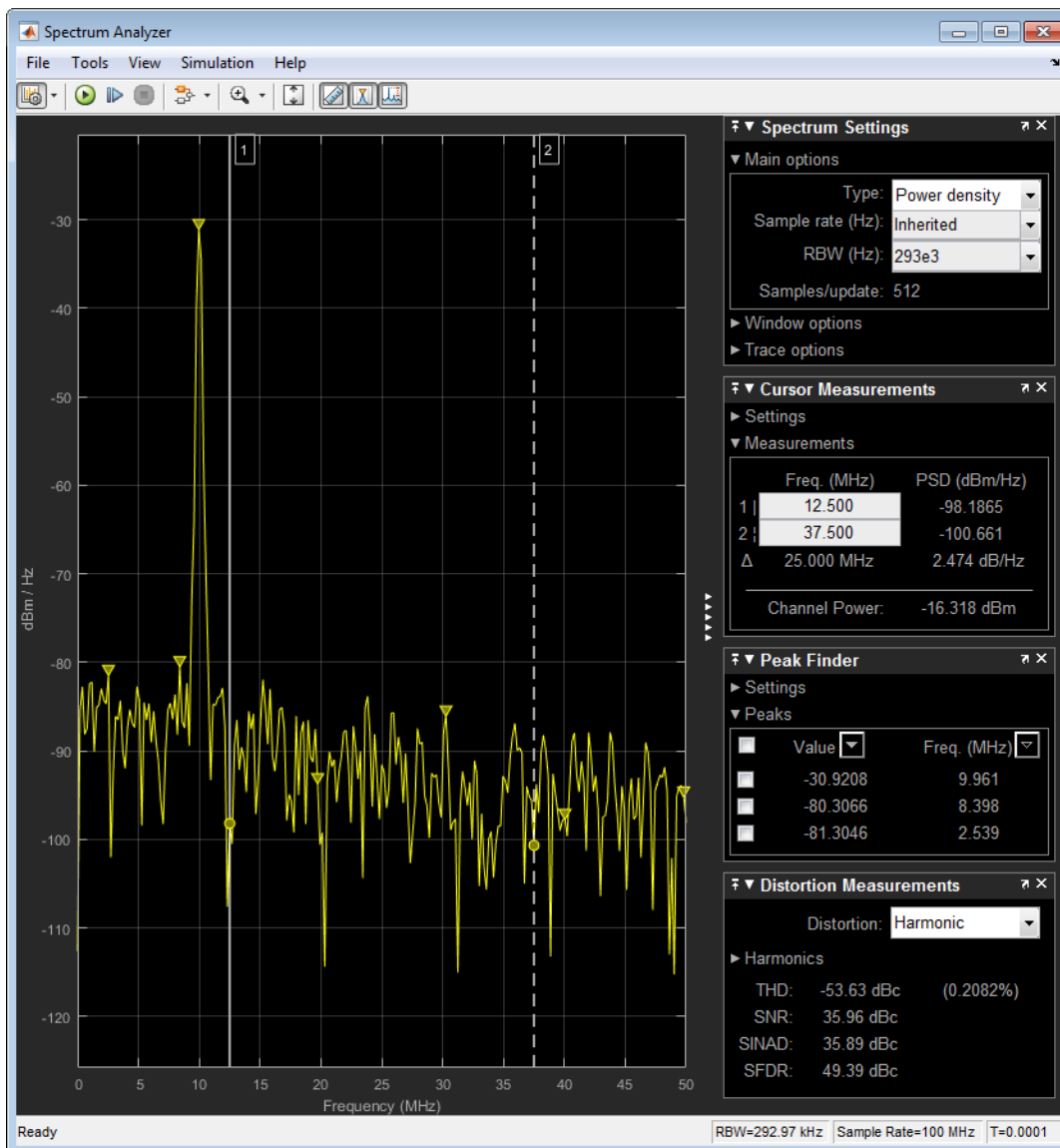
The Spectrum Analyzer block accepts input signals with discrete sample times and displays frequency spectra of these signals.

To use a Spectrum Analyzer block, instead of a regular scope, in a Simscape model:

- 1 Add a Spectrum Analyzer block to your block diagram.
- 2 If your model uses a variable-step solver, also add a Rate Transition block and connect it to the input of the Spectrum Analyzer, setting the **Output port sample time** to the sample time you wish the Spectrum Analyzer to use.

If your model uses a local solver, then it produces output physical signals with discrete sample times and you do not need to add a Rate Transition block. However, if you need to down-sample from the solver fixed step size, you can also use a Rate Transition block. For more information on using local solvers, see “Making Optimal Solver Choices for Physical Simulation”.

- 3 Use a PS-Simulink Converter block to connect the output physical signal of interest to the input of the Spectrum Analyzer block (or to the input of the Rate Transition block, if using one). For more information, see “Connecting Simscape Diagrams to Simulink Sources and Scopes”. You can also use additional signal processing blocks between the PS-Simulink Converter and the Spectrum Analyzer to enhance signal quality.
- 4 Run the simulation. The Spectrum Analyzer, referred to here as the scope, opens and displays the frequency spectrum of the signal.



Note To prevent the scope from opening when you run your model, right-click the scope icon and select **Comment Out**. If the scope is already open, and you comment it out in the model, the scope displays, "No data can be shown because this scope is commented out." Select **Uncomment** to turn the scope back on.

Reduce Plot Rate to Improve Performance

By default, Spectrum Analyzer updates the display at fixed intervals of time at a rate not exceeding 20 hertz. If you want Spectrum Analyzer to plot a spectrum on every simulation time step, you can disable the **Reduce Plot Rate to Improve Performance** option. In the Spectrum Analyzer menu, select **Simulation > Reduce Plot Rate to Improve Performance** to clear the check box. Tunable.

Note When this check box is selected, Spectrum Analyzer may display a misleading spectrum in some situations. For example, if the input signal is wide-band with non-stationary behavior, such as a chirp signal, Spectrum Analyzer might display a stationary spectrum. The reason for this behavior is that Spectrum Analyzer buffers the input signal data and only updates the display periodically at approximately 20 times per second. Therefore, Spectrum Analyzer does not render changes to the spectrum that occur and elapse between updates, which gives the impression of an incorrect spectrum. To ensure that spectral estimates are as accurate as possible, clear the **Reduce Plot Rate to Improve Performance** check box. When you clear this box, Spectrum Analyzer calculates spectra whenever there is enough data, rendering results correctly.

Limitations

This reference page describes the Spectrum Analyzer block available with Simscape or RF Blockset™. If you have DSP System Toolbox, more parameters and measurements are available. For information about the full Spectrum Analyzer, see Spectrum Analyzer.

Ports

Input

Port_1 — Signals to visualize

scalar | vector | matrix | array

Connect the signals you want to visualize. You can have up to 96 input ports. Input signals can have these characteristics:

- **Signal Domain** — Frequency or time signals
- **Type** — Discrete (sample-based and frame-based).
- **Data type** — Any data type that Simulink supports. See “Data Types Supported by Simulink”.
- **Dimension** — One dimensional (vector), two dimensional (matrix), or multidimensional (array). Input must have fixed number of channels. See “Signal Dimensions” and “Determine Signal Dimensions”.

Data Types: single | double | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64 | fixed point

Parameters

This section lists the parameters available in the Spectrum Analyzer when you do not have DSP System Toolbox. For the full parameter list, see Spectrum Analyzer.

Spectrum Settings

The **Spectrum Settings** pane appears at the right side of the Spectrum Analyzer window. This pane controls how the spectrum is calculated. To show the Spectrum Settings, in the Spectrum Analyzer

menu, select **View > Spectrum Settings** or use the  button in the toolbar.

Main

Type — Type of spectrum to display

Power (default) | Power density | RMS

Power — Spectrum Analyzer shows the power spectrum.

Power density — Spectrum Analyzer shows the power spectral density. The power spectral density is the magnitude of the spectrum normalized to a bandwidth of 1 hertz.

RMS — Spectrum Analyzer shows the root mean squared spectrum.

Tunable: Yes

Programmatic Use

See `SpectrumType`.

Sample rate — Sample rate of the input signal in hertz

Inherited (default) | positive scalar

Sample rate of the input signal in hertz, specified as either

- `Inherited` to use the same sample rate as the input signal.
- Positive scalar. The specified sample rate must be at least twice the input signal sample rate. Otherwise, you might see unexpected behavior in your signal visualization due to aliasing.

Programmatic Use

See `SampleRate`.

RBW (Hz) — Resolution bandwidth

Auto (default) | positive scalar

The resolution bandwidth in hertz. This parameter defines the smallest positive frequency that can be resolved. By default, this parameter is set to `Auto`. In this case, the Spectrum Analyzer determines the appropriate value to ensure that there are 1024 RBW intervals over the specified frequency span.

If you set this parameter to a numeric value, the value must allow at least two RBW intervals over the specified frequency span. In other words, the ratio of the overall frequency span to RBW must be greater than two:

$$\frac{span}{RBW} > 2$$

Tunable: Yes

Programmatic Use

See `RBW`.

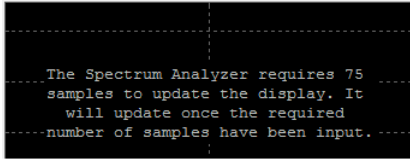
Samples/update — Required number of input samples

positive scalar

This property is read-only.

The number of input samples required to compute one spectral update. You cannot modify this parameter; it is shown in the spectrum analyzer for informational purposes only. This parameter is directly related to **RBW (Hz)/Window length/Number of frequency bands**. For more details, see “Algorithms” (DSP System Toolbox).

If the input does not have enough samples to achieve the resolution bandwidth that you specify, Spectrum Analyzer produces a message on the display.



Window

Overlap (%) — Segment overlap percentage

0 (default) | scalar between 0 and 100

This parameter defines the amount of overlap between the previous and current buffered data segments. The overlap creates a window segment that is used to compute a spectral estimate. The value must be greater than or equal to zero and less than 100.

Tunable: Yes

Programmatic Use

See `OverlapPercent`.

Window — Windowing method

Hann (default) | Rectangular

The windowing method to apply to the spectrum. Windowing is used to control the effect of sidelobes in spectral estimation. The window you specify affects the window length required to achieve a resolution bandwidth and the required number of samples per update. For more information about windowing, see “Windows” (Signal Processing Toolbox).

Tunable: Yes

Programmatic Use

See `Window`.

NENBW — Normalized effective noise bandwidth

scalar

This property is read-only.

The normalized effective noise bandwidth of the window. You cannot modify this parameter; it is shown for informational purposes only. This parameter is a measure of the noise performance of the window. The value is the width of a rectangular filter that accumulates the same noise power with the same peak power gain.

The rectangular window has the smallest NENBW, with a value of 1. All other windows have a larger NENBW value. For example, the Hann window has an NENBW value of approximately 1.5.

Trace

Units — Spectrum units

dBm (default)

This property is read-only.

The units of the spectrum. To change units, you must have the DSP System Toolbox.

Tunable: Yes

Programmatic Use

See `SpectrumUnits`.

Averaging method — Smoothing method

`Exponential` (default) | `Running`

Specify the smoothing method as:

- `Exponential` — Weighted average of samples. Use the `Forgetting factor` property to specify the weighted forgetting factor.
- `Running` — Running average of the last n samples. Use the `Averages` property to specify n .

For more information about the averaging methods, see “Averaging Method” on page 1-660.

Programmatic Use

See `AveragingMethod`.

Averages — Number of spectral averages

1 (default) | positive integer

Specify the number of spectral averages as a positive integer. The spectrum analyzer computes the current power spectrum estimate by computing a running average of the last N power spectrum estimates. This parameter defines the number of spectral averages, N .

Dependency

This parameter applies only when **Averaging method** is `Running`.

Programmatic Use

See `SpectralAverages`.

Forgetting factor — Weighting forgetting factor

0.9 (default) | scalar in the range (0,1]

Specify the exponential weighting as a scalar value greater than 0 and less than or equal to 1.

Dependency

This parameter applies only when the **Averaging method** is `Exponential`.

Programmatic Use

See `ForgettingFactor`.

Reference load — Reference load

1 (default) | positive real scalar

The reference load in ohms that the Spectrum Analyzer uses as a reference to compute power values.

Programmatic Use

See `ReferenceLoad`.

Scale — Scale of frequency axis

Linear (default) | Logarithmic

Choose a linear or logarithm scale for the frequency axis. When the frequency span contains negative frequency values, you cannot choose the logarithmic option.

Programmatic Use

See `FrequencyScale`.

Offset — Constant frequency offset

0 (default) | scalar

The constant frequency offset to apply to the entire spectrum, or a vector of frequencies to apply to each spectrum for multiple inputs. The offset parameter is added to the values on the Frequency axis in the Spectrum Analyzer window. This parameter is not used in any spectral computations. You must take the parameter into consideration when you set the **Span (Hz)** and **CF (Hz)** parameters to ensure that the frequency span is within the “Nyquist frequency interval” (DSP System Toolbox).

Dependency

To use this parameter, set “Input domain” (DSP System Toolbox) to Time.

Programmatic Use

See `FrequencyOffset`.

Two-sided spectrum — Enable two-sided spectrum view

off (default) | on


Select this check box to enable a two-sided spectrum view. In this view, both negative and positive frequencies are shown. If you clear this check box, Spectrum Analyzer shows a one-sided spectrum with only positive frequencies. Spectrum Analyzer requires that this parameter is selected when the input signal is complex-valued.

Programmatic Use

See `PlotAsTwoSidedSpectrum`.

Configuration Properties

The **Configuration Properties** dialog box controls visual aspects of the Spectrum Analyzer. To open the Configuration Properties, in the Spectrum Analyzer menu, select **View > Configuration**

Properties or select the  button in the toolbar dropdown.

Title — Display title

character vector | string

Specify the display title. Enter `%<SignalLabel>` to use the signal labels in the Simulink model as the axes titles.

Tunable: Yes

Programmatic Use

See `Title`.

Show Legend — Display signal legend

off (default) | on

Show signal legend. The names listed in the legend are the signal names from the model. For signals with multiple channels, a channel index is appended after the signal name. Continuous signals have straight lines before their names and discrete signals have step-shaped lines.

From the legend, you can control which signals are visible. This control is equivalent to changing the visibility in the **Style** parameters. In the scope legend, click a signal name to hide the signal in the scope. To show the signal, click the signal name again. To show only one signal, right-click the signal name, which hides all other signals. To show all signals, press **ESC**.

Note The legend only shows the first 20 signals. Any additional signals cannot be viewed or controlled from the legend.

Dependency

To enable this parameter, set “View” (DSP System Toolbox) to Spectrum or Spectrum and spectrogram.

Programmatic Use

See ShowLegend.

Show grid — Show internal grid lines

on (default) | off

Show internal grid lines on the Spectrum Analyzer

Programmatic Use

See ShowGrid.

Y-limits (minimum) — Y-axis minimum

-80 (default) | scalar

Specify the minimum value of the y-axis.

Programmatic Use

See YLimits.

Y-limits (maximum) — Y-axis maximum

20 (default) | scalar

Specify the maximum value of the y-axis.

Programmatic Use

See YLimits.

Y-label — Y-axis label

character vector | string

To display signal units, add (%<SignalUnits>) to the label. At the beginning of a simulation, Simulink replaces (%SignalUnits) with the units associated with the signals. For example, if you have a signal for velocity with units of m/s enter

Velocity (%<SignalUnits>)

Programmatic Use

See YLabel.

Style


The **Style** dialog box controls how the Spectrum Analyzer appears. To open the Style properties, in the Spectrum Analyzer menu, select **View > Style** or select the  button in the toolbar drop-down.

Figure color – Window background

gray (default) | color picker

Specify the color that you want to apply to the background of the scope figure.

Plot type – Plot type

Line (default) | Stem

Specify whether to display a Line or Stem plot.

Programmatic Use

See PlotType.

Axes colors – Axes background color

black (default) | color picker

Specify the color that you want to apply to the background of the axes.

Properties for line – Channel for visual property settings

channel names

Specify the channel for which you want to modify the visibility, line properties, and marker properties.

Visible – Channel visibility

on (default) | off

Specify whether the selected channel is visible. If you clear this check box, the line disappears. You can also change signal visibility using the scope legend.

Line – Line style

line, 0.5, yellow (default)

Specify the line style, line width, and line color for the selected channel.

Marker – Data point markers

none (default)

Specify marks for the selected channel to show at its data points. This parameter is similar to the 'Marker' property for plots. You can choose any of the marker symbols from the drop-down.

Axes Scaling

The **Axes Scaling** dialog box controls the axes limits of the Spectrum Analyzer. To open the Axes Scaling properties, in the Spectrum Analyzer menu, select **Tools > Axes Scaling > Axes Scaling Properties**.

Axes scaling — Automatic axes scaling

Auto (default) | Manual | After N Updates

Specify when the scope automatically scales the y-axis. By default, this parameter is set to **Auto**, and the scope does not shrink the y-axis limits when scaling the axes. You can select one of the following options:

- **Auto** — The scope scales the axes as needed, both during and after simulation. Selecting this option shows the **Do not allow Y-axis limits to shrink**.
- **Manual** — When you select this option, the scope does not automatically scale the axes. You can manually scale the axes in any of the following ways:
 - Select **Tools > Scaling Properties**.
 - Press one of the **Scale Axis Limits** toolbar buttons.
 - When the scope figure is the active window, press **Ctrl+A**.
- **After N Updates** — Selecting this option causes the scope to scale the axes after a specified number of updates. This option is useful, and most efficient, when your frequency signal values quickly reach steady-state after a short period. Selecting this option shows the **Number of updates** edit box where you can modify the number of updates to wait before scaling.

Tunable: Yes

Programmatic Use

See AxesScaling.

Do not allow Y-axis limits to shrink — Axes scaling limits

on (default) | off

When you select this parameter, the y-axis is allowed to grow during axes scaling operations. If you clear this check box, the y-axis limits can shrink during axes scaling operations.

Dependency

This parameter appears only when you select **Auto** for the **Axis scaling** parameter. When you set the **Axes scaling** parameter to **Manual** or **After N Updates**, the y-axis limits can shrink.

Number of updates — Number of updates before scaling

10 (default) | positive number

The number of updates after which the axes scale, specified as a positive integer. If the spectrogram is displayed, this parameter specifies the number of updates after which the color axes scales.

Tunable: Yes

Dependency

This parameter appears only when you set “Axes scaling/Color scaling” (DSP System Toolbox) to **After N Updates**.

Programmatic Use

See `AxesScalingNumUpdates`.

Scale limits at stop — Scale axes at stop

off (default) | on

Select this check box to scale the axes when the simulation stops. If the spectrogram is displayed, select this check box to scale the color when the simulation stops. The y-axis is always scaled. The x-axis limits are only scaled if you also select the **Scale X-axis limits** check box.

Data range (%) — Percent of axes

100 (default) | number in the range [1,100]

Set the percentage of the axis that the scope uses to display the data when scaling the axes. If the spectrogram is displayed, set the percentage of the power values range within the colormap. Valid values are from 1 through 100. For example, if you set this parameter to 100, the scope scales the axis limits such that your data uses the entire axis range. If you then set this parameter to 30, the scope increases the y-axis or color range such that your data uses only 30% of the axis range.

Tunable: Yes

Align — Alignment along axes

Center (default) | Bottom | Top | Left | Right

Specify where the scope aligns your data along the axis when it scales the axes. If the spectrogram is displayed, specify where the scope aligns your data along the axis when it scales the color. If you are using CCDF Measurements (DSP System Toolbox), the x axis is also configurable.

Tunable: Yes

Algorithms**Spectrum Estimation — Welch's Method**

When you choose the Welch method, the power spectrum estimate is averaged modified periodograms.

Given the signal input, x , the Spectrum Analyzer does the following:

- 1 Multiplies x by the given window and scales the result by the window power. The Spectrum Analyzer uses the RBW or the Window Length setting in the **Spectrum Settings** pane to determine the data window length.
- 2 Computes the FFT of the signal, Y , and takes the square magnitude using $Z = Y \cdot \text{conj}(Y)$.
- 3 Computes the current power spectrum estimate by taking the moving average of the last N number of Z 's, and scales the answer by the sample rate. For details on the moving average methods, see "Averaging Method" on page 1-660.

Spectrum Analyzer requires that a minimum number of samples to compute a spectral estimate. This number of input samples required to compute one spectral update is shown as **Samples/update** in the **Main options** pane. This value is directly related to resolution bandwidth, RBW , by the following equation, or to the window length, by the equation shown in step 2.

$$N_{samples} = \frac{\left(1 - \frac{O_p}{100}\right) \times NENBW \times F_s}{RBW}$$

The normalized effective noise bandwidth, $NENBW$, is a factor that depends on the windowing method. Spectrum Analyzer shows the value of $NENBW$ in the **Window Options** pane of the **Spectrum Settings** pane. Overlap percentage, O_p , is the value of the **Overlap %** parameter in the **Window Options** pane of the **Spectrum Settings** pane. F_s is the sample rate of the input signal. Spectrum Analyzer shows sample rate in the **Main Options** pane of the **Spectrum Settings** pane.

- 1 When in **RBW (Hz)** mode, the window length required to compute one spectral update, N_{window} , is directly related to the resolution bandwidth and normalized effective noise bandwidth:

$$N_{window} = \frac{NENBW \times F_s}{RBW}$$

When in **Window Length** mode, the window length is used as specified.

- 2 The number of input samples required to compute one spectral update, $N_{samples}$, is directly related to the window length and the amount of overlap by the following equation.

$$N_{samples} = \left(1 - \frac{O_p}{100}\right) N_{window}$$

When you increase the overlap percentage, fewer new input samples are needed to compute a new spectral update. For example, if the window length is 100, then the number of input samples required to compute one spectral update is given as shown in the following table.

O_p	$N_{samples}$
0%	100
50%	50
80%	20

- 3 The normalized effective noise bandwidth, $NENBW$, is a window parameter determined by the window length, N_{window} , and the type of window used. If $w(n)$ denotes the vector of N_{window} window coefficients, then $NENBW$ is given by the following equation.

$$NENBW = N_{window} \times \frac{\sum_{n=1}^{N_{window}} w^2(n)}{\left[\sum_{n=1}^{N_{window}} w(n)\right]^2}$$

- 4 When in **RBW (Hz)** mode, you can set the resolution bandwidth using the value of the **RBW (Hz)** parameter on the **Main options** pane of the **Spectrum Settings** pane. You must specify a value to ensure that there are at least two RBW intervals over the specified frequency span. The ratio of the overall span to RBW must be greater than two:

$$\frac{span}{RBW} > 2$$

By default, the **RBW (Hz)** parameter on the **Main options** pane is set to **Auto**. In this case, the Spectrum Analyzer determines the appropriate value to ensure that there are 1024 RBW

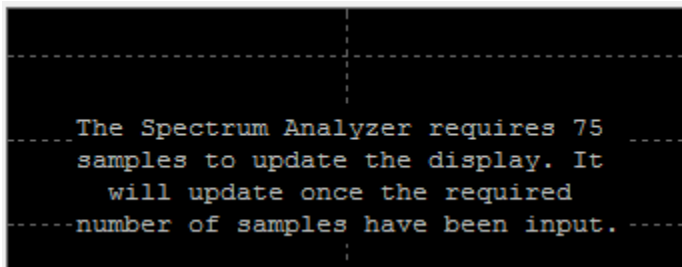
intervals over the specified frequency span. When you set **RBW (Hz)** to Auto, *RBW* is calculated as:

$$RBW_{auto} = \frac{span}{1024}$$

- 5 When in **Window Length** mode, you specify N_{window} and the resulting *RBW* is:

$$\frac{NENBW \times F_s}{N_{window}}$$

Sometimes, the number of input samples provided are not sufficient to achieve the resolution bandwidth that you specify. When this situation occurs, Spectrum Analyzer displays a message:



Spectrum Analyzer removes this message and displays a spectral estimate when enough data has been input.

Note The number of FFT points (N_{fft}) is independent of the window length (N_{window}). You can set them to different values if N_{fft} is greater than or equal to N_{window} .

Nyquist frequency interval

When the `PlotAsTwoSidedSpectrum` property is set to `true`, the interval is

$$\left[-\frac{SampleRate}{2}, \frac{SampleRate}{2} \right] + FrequencyOffset \text{ hertz.}$$

When the `PlotAsTwoSidedSpectrum` property is set to `false`, the interval is

$$\left[0, \frac{SampleRate}{2} \right] + FrequencyOffset \text{ hertz.}$$

Periodogram and Spectrogram

Spectrum Analyzer calculates and plots the power spectrum, power spectrum density, and RMS computed by the modified Periodogram estimator. For more information about the Periodogram method, see `periodogram`.

Power Spectral Density — The power spectral density (PSD) is given by the following equation.

$$PSD(f) = \frac{1}{P} \sum_{p=1}^P \frac{\left| \sum_{n=1}^{N_{FFT}} x^p[n] e^{-j2\pi f(n-1)T} \right|^2}{F_s \times \sum_{n=1}^{N_{window}} w^2[n]}$$

In this equation, $x[n]$ is the discrete input signal. On every input signal frame, Spectrum Analyzer generates as many overlapping windows as possible, with each window denoted as $x^{(p)}[n]$, and computes their periodograms. Spectrum Analyzer displays a running average of the P most current periodograms.

Power Spectrum — The power spectrum is the product of the power spectral density and the resolution bandwidth, as given by the following equation.

$$P_{\text{spectrum}}(f) = \text{PSD}(f) \times \text{RBW} = \text{PSD}(f) \times \frac{F_s \times N_{\text{ENBW}}}{N_{\text{window}}} = \frac{1}{P} \sum_{p=1}^P \frac{\left| \sum_{n=1}^{N_{\text{FFT}}} x^{(p)}[n] e^{-j2\pi f(n-1)T} \right|^2}{\left[\sum_{n=1}^{N_{\text{window}}} w[n] \right]^2}$$

Frequency Vector

When set to **Auto**, the frequency vector for frequency-domain input is calculated by the software.

When the `PlotAsTwoSidedSpectrum` property is set to true, the frequency vector is:

$$\left[-\frac{\text{SampleRate}}{2}, \frac{\text{SampleRate}}{2} \right]$$

When the `PlotAsTwoSidedSpectrum` property is set to false, the frequency vector is:

$$\left[0, \frac{\text{SampleRate}}{2} \right]$$

Occupied BW

The *Occupied BW* is calculated as follows.

- 1 Calculate the total power in the measured frequency range.
- 2 Determine the lower frequency value. Starting at the lowest frequency in the range and moving upward, the power distributed in each frequency is summed until this result is

$$\frac{100 - \text{OccupiedBW}\%}{2}$$

of the total power.

- 3 Determine the upper frequency value. Starting at the highest frequency in the range and moving downward, the power distributed in each frequency is summed until the result reaches

$$\frac{100 - \text{OccupiedBW}\%}{2}$$

of the total power.

- 4 The bandwidth between the lower and upper power frequency values is the occupied bandwidth.
- 5 The frequency halfway between the lower and upper frequency values is the center frequency.

Distortion Measurements

The *Distortion Measurements* are computed as follows.

- 1 Spectral content is estimated by finding peaks in the spectrum. When the algorithm detects a peak, it records the width of the peak and clears all monotonically decreasing values. That is, the algorithm treats all these values as if they belong to the peak. Using this method, all spectral content centered at DC (0 Hz) is removed from the spectrum and the amount of bandwidth cleared (W_0) is recorded.
- 2 The fundamental power (P_1) is determined from the remaining maximum value of the displayed spectrum. A local estimate (Fe_1) of the fundamental frequency is made by computing the central moment of the power near the peak. The bandwidth of the fundamental power content (W_1) is recorded. Then, the power from the fundamental is removed as in step 1.
- 3 The power and width of the higher-order harmonics (P_2, W_2, P_3, W_3 , etc.) are determined in succession by examining the frequencies closest to the appropriate multiple of the local estimate (Fe_1). Any spectral content that decreases monotonically about the harmonic frequency is removed from the spectrum first before proceeding to the next harmonic.
- 4 Once the DC, fundamental, and harmonic content is removed from the spectrum, the power of the remaining spectrum is examined for its sum ($P_{remaining}$), peak value ($P_{maxspur}$), and median value ($P_{estnoise}$).
- 5 The sum of all the removed bandwidth is computed as $W_{sum} = W_0 + W_1 + W_2 + \dots + W_n$.

The sum of powers of the second and higher-order harmonics are computed as $P_{harmonic} = P_2 + P_3 + P_4 + \dots + P_n$.

- 6 The sum of the noise power is estimated as:

$$P_{noise} = (P_{remaining} \cdot dF + P_{est.noise} \cdot W_{sum})/RBW$$

Where dF is the absolute difference between frequency bins, and RBW is the resolution bandwidth of the window.

- 7 The metrics for SNR, THD, SINAD, and SFDR are then computed from the estimates.

$$THD = 10 \cdot \log_{10} \left(\frac{P_{harmonic}}{P_1} \right)$$

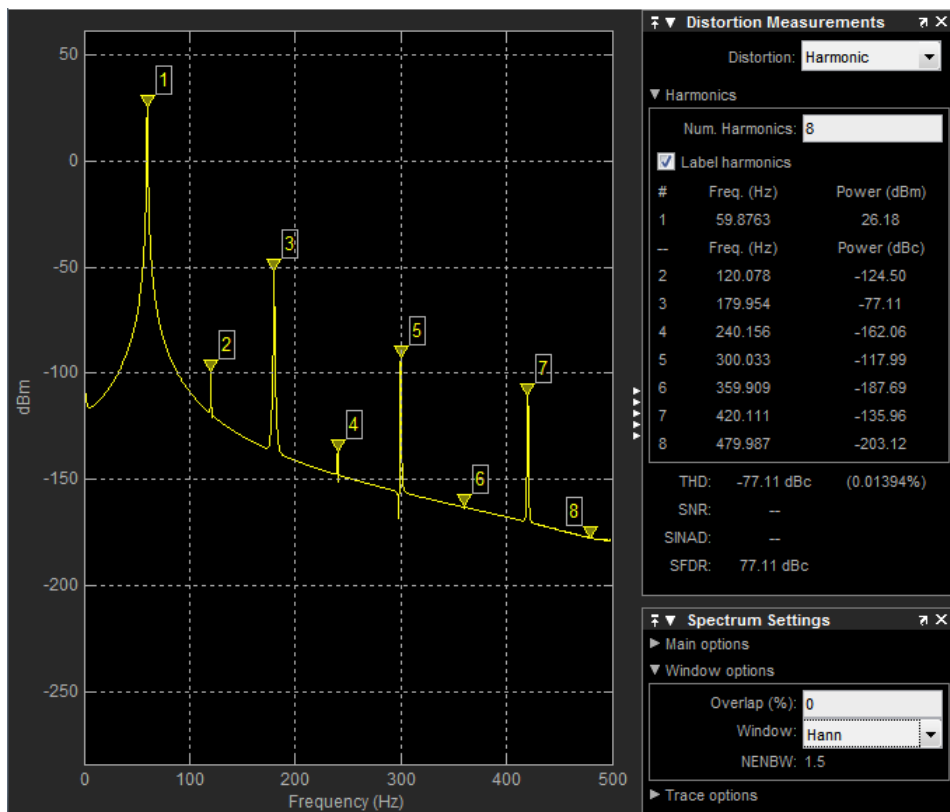
$$SINAD = 10 \cdot \log_{10} \left(\frac{P_1}{P_{harmonic} + P_{noise}} \right)$$

$$SNR = 10 \cdot \log_{10} \left(\frac{P_1}{P_{noise}} \right)$$

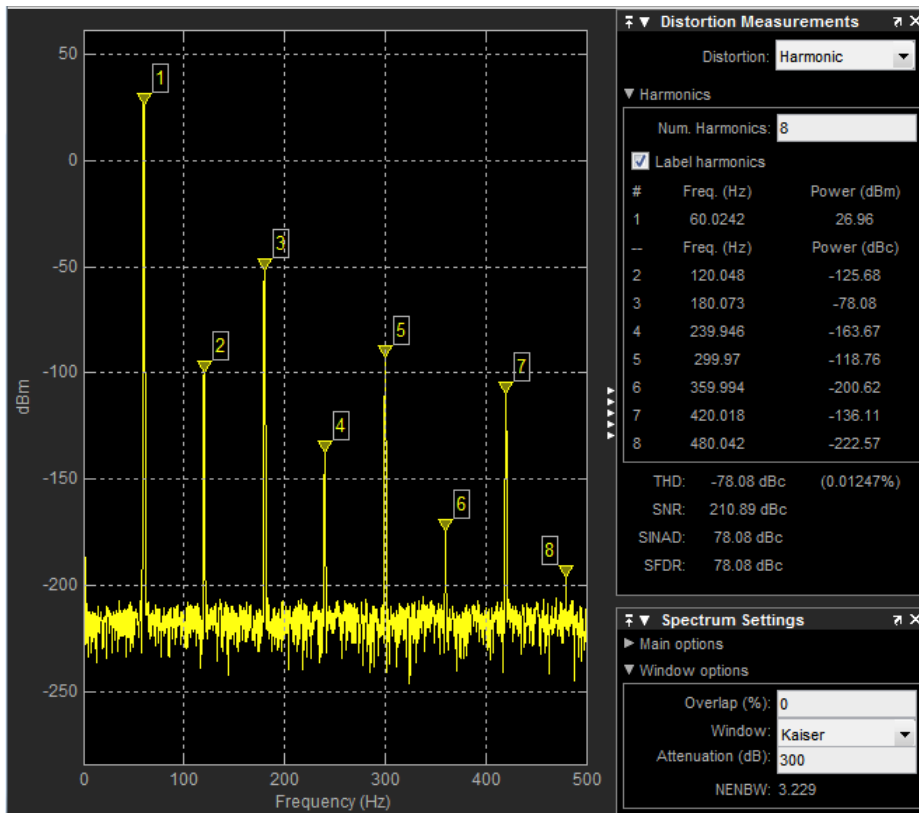
$$SFDR = 10 \cdot \log_{10} \left(\frac{P_1}{\max(P_{maxspur}, \max(P_2, P_3, \dots, P_n))} \right)$$

Harmonic Measurements

- 1 The harmonic distortion measurements use the spectrum trace shown in the display as the input to the measurements. The default Hann window setting of the Spectrum Analyzer may exhibit leakage that can completely mask the noise floor of the measured signal.



The harmonic measurements attempt to correct for leakage by ignoring all frequency content that decreases monotonically away from the maximum of harmonic peaks. If the window leakage covers more than 70% of the frequency bandwidth in your spectrum, you may see a blank reading (-) reported for **SNR** and **SINAD**. If your application can tolerate the increased equivalent noise bandwidth (ENBW), consider using a Kaiser window with a high attenuation (up to 330 dB) to minimize spectral leakage.



- 2 The DC component is ignored.
- 3 After windowing, the width of each harmonic component masks the noise power in the neighborhood of the fundamental frequency and harmonics. To estimate the noise power in each region, Spectrum Analyzer computes the median noise level in the nonharmonic areas of the spectrum. It then extrapolates that value into each region.
- 4 N^{th} order intermodulation products occur at $A*F1 + B*F2$,

where $F1$ and $F2$ are the sinusoid input frequencies and $|A| + |B| = N$. A and B are integer values.

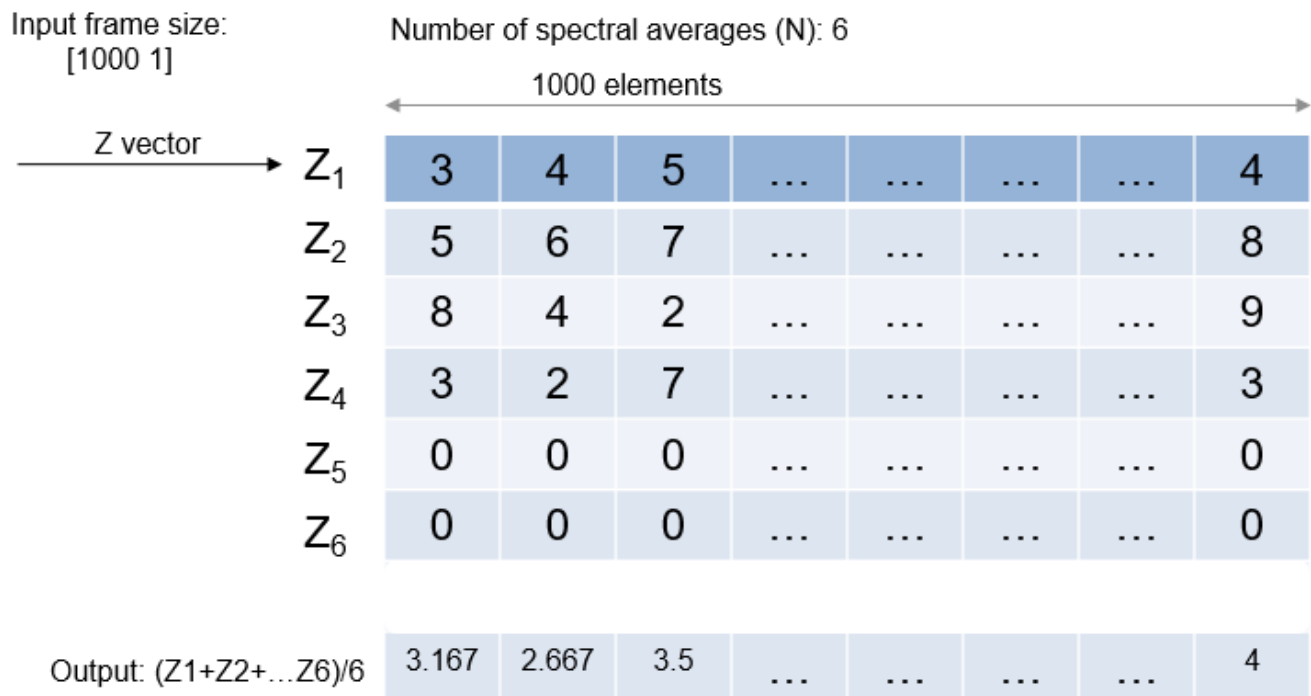
- 5 For intermodulation measurements, the third-order intercept (TOI) point is computed as follows, where P is power in decibels of the measured power referenced to 1 milliwatt (dBm):

- $TOI_{lower} = P_{F1} + (P_{F2} - P_{(2F1-F2)})/2$
- $TOI_{upper} = P_{F2} + (P_{F1} - P_{(2F2-F1)})/2$
- $TOI = (TOI_{lower} + TOI_{upper})/2$

Averaging Method

The moving average is calculated using one of the two methods:

- Running — For each frame of input, average the last N scaled Z vectors, which are computed by the algorithm. The variable N is the value you specify for the number of spectral averages. If the algorithm does not have enough Z vectors, the algorithm uses zeros to fill the empty elements.



- **Exponential** — The moving average algorithm using the exponential weighting method updates the weights and computes the moving average recursively for each Z vector that comes in by using the following recursive equations:

$$w_N = \lambda w_{N-1} + 1$$

$$\bar{z}_N = \left(1 - \frac{1}{w_N}\right)\bar{z}_{N-1} + \left(\frac{1}{w_N}\right)z_N$$

- λ — Forgetting factor.
- w_N — Weighting factor applied to the current Z vector.
- z_N — Current Z vector.
- \bar{z}_{N-1} — Moving average until the previous Z vector.
- $\left(1 - \frac{1}{w_N}\right)\bar{z}_{N-1}$ — Effect of the previous Z vectors on the average.
- \bar{z}_N — Moving average including the current Z vector.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Spectrum Analyzer | SpectrumAnalyzerConfiguration | dsp.SpectrumAnalyzer

Introduced in R2016b

Switch

Switch controlled by external physical signal



Library

Electrical Elements

Description

The Switch block models a switch controlled by an external physical signal. If the external physical signal PS is greater than the value specified in the **Threshold** parameter, then the switch is closed, otherwise the switch is open.

Electrical switches add discontinuities to your model, and therefore your choice of the solver can influence the model behavior. For detailed information about the solver settings, see “Making Optimal Solver Choices for Physical Simulation”.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Closed resistance R_{closed}

The resistance of the switch when it is closed. The parameter value must be greater than zero. The default value is 0.01Ω .

Open conductance G_{open}

The conductance of the switch when it is open. The parameter value must be greater than zero. The default value is $1e-8 \text{ 1}/\Omega$.

Threshold

The threshold value for opening and closing the switch. If the external physical signal PS is greater than this value, then the switch is closed, otherwise the switch is open. The default value is 0.5 .

Ports

The block has two electrical conserving ports and one physical signal port PS.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS Switch

Introduced in R2007a

Temperature Sensor

Ideal temperature sensor



Library

Thermal Sensors

Description

The Temperature Sensor block represents an ideal temperature sensor, that is, a device that determines the temperature differential measured between two points without drawing any heat.

Connections A and B are thermal conserving ports that connect to the two points where temperature is being monitored. Port T is a physical signal port that outputs the temperature differential value.

The block positive direction is from port A to port B. The measured temperature is determined as $T = T_A - T_B$.

Ports

The block has the following ports:

A

Thermal conserving port associated with the sensor positive probe.

B

Thermal conserving port associated with the sensor negative probe.

T

Physical signal output port for temperature.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

PS-Simulink Converter

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007b

Temperature Source

Constant source of thermal energy, characterized by temperature

Library: Simscape / Foundation Library / Thermal / Thermal Sources



Description

The Temperature Source block represents an ideal source of thermal energy that is powerful enough to maintain specified temperature at its outlet regardless of the heat flow consumed by the system.

The source generates constant absolute temperature, defined by the **Temperature** parameter value.

Ports

Conserving

A — Source outlet

thermal

Thermal conserving port associated with the source outlet.

Parameters

Temperature — Absolute temperature at source outlet

293.15 K (default)

Constant absolute temperature at the source outlet port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Temperature Source

Introduced in R2017b

Thermal Liquid Settings (TL)

Physical properties of a thermal liquid

Library: Simscape / Foundation Library / Thermal Liquid / Utilities



Description

The Thermal Liquid Settings (TL) block provides the physical properties of a fluid to a thermal liquid network. The properties are global: they apply not to one component but to all those that comprise the network. Every thermal liquid network in a model must connect to exactly one instance of this block. Among the properties specified in this block are:

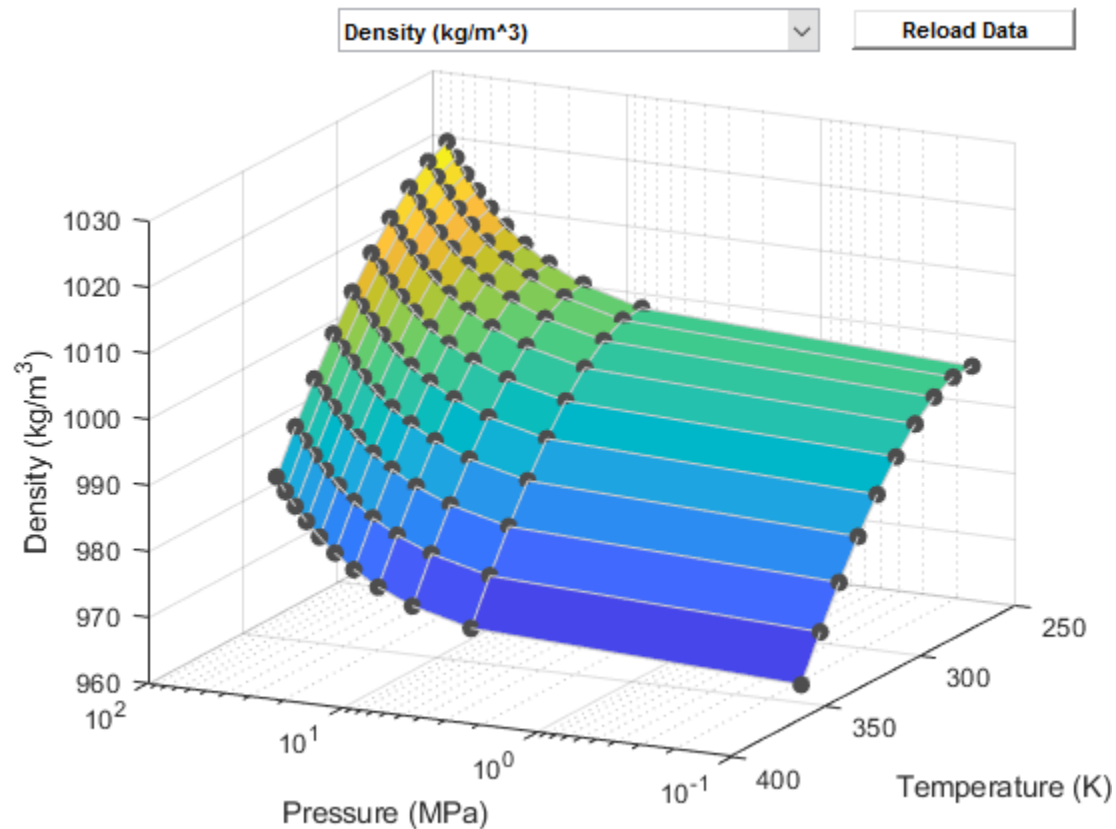
- Thermodynamic properties — Density, specific internal energy, and specific heat
- Derivative properties — Bulk modulus and thermal expansion coefficient
- Transport properties — Kinematic viscosity and thermal conductivity

Each fluid property is specified in tabulated form against both temperature and pressure or against temperature alone. Use the option to ignore pressure-induced changes if those changes are known to be negligible, if pressure is expected to be nearly constant, or if fluid property data is accessible in terms of temperature only.

Data Visualization

The block provides the option to plot the specified fluid properties over their temperature and pressure domains. To create the plots, right-click the block and select **Foundation Library > Plot Fluid Properties**. Use the drop-down list located at the top of the plot to select the fluid property to visualize. Click the **Reload** button to regenerate a plot following a block parameter update.

Use the plots to visualize the dependences of the fluid properties on pressure and temperature—for example, to more easily catch anomalies in the specified data. Most fluid properties are shown as variable over pressure only if they are specified as functions of pressure. Exceptions include density, which derives a pressure dependence from the bulk modulus and thermal expansion coefficient, and any properties calculated from density, such as specific heat.



Thermal Liquid Properties Plot

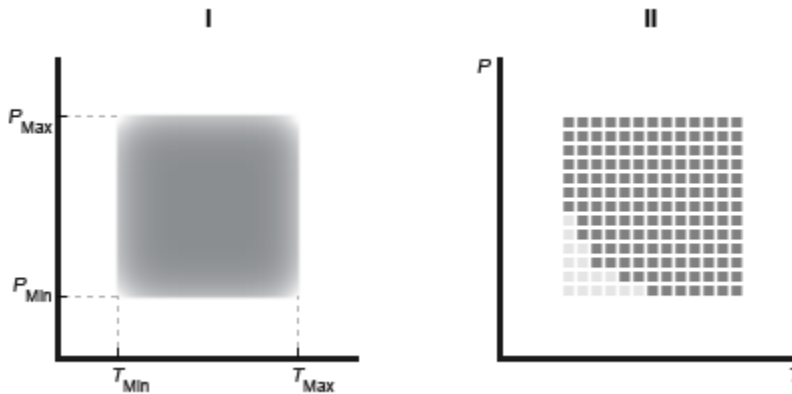
Parameterizations

The block provides several parameterizations for the independent state variables, density, and specific internal energy. The parameterization that you select for each determines the data that you must obtain and the state variables with respect to which you must specify it. The setting of the **Table dimensions** parameter affects which parameters you see in other tabs.

Pressure and Temperature

Pressure and temperature are the *across* variables of the thermal liquid domain. As such, they are the natural choice of independent state variables against which to specify all other fluid properties. The block provides two parameterizations based on these state variables. They are available through the **Table dimensions** parameter:

- 2D tables based on temperature and pressure — Provide tabulated data against both temperature and pressure. The region of valid temperatures and pressures is specified in terms of minimum and maximum values (**I** in the figure) or in terms of a validity matrix (**II**).



Validity Region Types

- **1D vectors based on temperature** — Provide tabulated data against temperature and ignore any dependences on pressure. The region of valid temperatures and pressures is specified in terms of minimum and maximum values only. The bulk modulus provides a pressure dependence to density and to any fluid properties calculated from density.

Density and Derivatives

The block provides three types of parameterizations for density and for the two derivative parameters that the density calculation often depends on—the isothermal bulk modulus and the isobaric thermal expansion coefficient. Options include:

- **Density, bulk modulus, and thermal expansion coefficient tables** — Provide tabulated data for density, the bulk modulus, and the thermal expansion coefficient. The data must be specified against both temperature and pressure. The **Table dimensions** parameter must be set to 2D tables based on temperature and pressure.
- **Density table/vector** — Provide tabulated data for density against temperature (and, in the 2D case, pressure). The block computes the isothermal bulk modulus and isobaric thermal expansion coefficient from the tabulated data using the finite-difference method. The isothermal bulk modulus β is defined as:

$$\frac{1}{\beta} = \frac{1}{\rho} \left(\frac{\partial \rho}{\partial p} \right)_T,$$

where ρ is density, T is temperature, and p is pressure. The isobaric thermal expansion coefficient α is defined as:

$$\alpha = - \frac{1}{\rho} \left(\frac{\partial \rho}{\partial T} \right)_p.$$

- **Reference Density** — Provide the density, bulk modulus, and thermal expansion coefficient at a known temperature (and, in the 2D case, pressure). The block uses an analytical expression to calculate the density at other temperatures and pressures. The calculation is based on the expression:

$$\ln \left(\frac{\rho}{\rho_R} \right) = - \alpha (T - T_R) + \frac{p - p_R}{\beta},$$

where the subscript R denotes a reference quantity (specified in the block dialog box).

Specific Internal Energy

As in the case of density, the block provides three types of parameterization for the specific internal energy and for a related quantity from which it can be computed—the specific heat. Options include:

- **Specific internal energy and specific heat tables** — Provide tabulated data for the specific internal energy and the specific heat coefficient. The data must be specified against both temperature and pressure. The **Table dimensions** parameter must be set to **2D tables based on temperature and pressure**.
- **Specific internal energy table/vector** — Provide tabulated data for the specific internal energy against temperature (and, in the 2D case, pressure). The block computes the specific heat coefficient from the specific internal energy data. The calculation is based on the expression:

$$c_p(T, p) = \left(\frac{\partial u}{\partial T} \right)_p + \frac{p\alpha}{\rho},$$

where c_p is the specific heat coefficient, and u is the specific internal energy.

- **Specific heat coefficient table/vector** — Provide tabulated data for the specific heat coefficient against temperature (and, in the 2D case, pressure). The block computes the specific internal energy from the specific heat coefficient data. The calculation is based on the expression:

$$u(T, p) = \int_R^T \left(c_p(T, p) - \frac{p\alpha(T, p)}{\rho(T, p)} \right) dT + u_R,$$

where the subscript R denotes a reference quantity. The value of u_R is set to zero, a suitable choice because it is the difference in specific internal energy, rather than its value, that is relevant for simulation. The value of u can differ from that provided in other sources, such as the REFPROP fluids database.

Ports

Conserving

A — Thermal liquid network connector

thermal liquid

Port identifying the thermal liquid network to which the specified fluid properties apply.

Parameters

Temperature and Pressure

Table dimensions — Dimensions of the fluid property tables

2D tables based on temperature and pressure (T,p) (default) | 1D vectors based on temperature (T)

Dimensions of the fluid property tables. This parameter determines whether the fluid properties are variable or constant with pressure. Select **1D vectors based on temperature (T)** to ignore the pressure dependences of the fluid properties.

Temperature vector — Reference temperatures at which to specify the fluid properties

[273.1600 : 10 : 373.16] ' K (default) | M -element column vector with units of temperature

Vector of temperatures at which to specify the values of the fluid properties. Each temperature corresponds to a row of a fluid property table.

Pressure vector — Reference pressures at which to specify the fluid properties

[0.01, 0.1, 5 : 5 : 50] MPa (default) | *N*-element row vector with units of pressure

Vector of pressures at which to specify the values of the fluid properties. Each pressure corresponds to a column of a fluid property table.

Dependencies

This parameter is active only when the **Table dimensions** parameter is set to 2D vectors based on temperature and pressure (T,p).

Atmospheric pressure — Absolute pressure of the thermal liquid network environment

0.101325 MPa (default) | scalar with units of pressure

Absolute pressure of the outside environment of the attached thermal liquid network. The default value corresponds to the earth's standard atmospheric pressure.

Valid pressure-temperature region parameterization — Specification method for the pressure-temperature validity region

Specified minimum and maximum values (default) | Validity matrix

Specification method for the pressure-temperature validity region. You can specify the limits of a square pressure-temperature region or a matrix of validity values for an arbitrarily shaped pressure-temperature region.

Dependencies

This parameter is active only when the **Table dimensions** parameter is set to 2D vectors based on temperature and pressure (T,p).

Minimum valid temperature — Lowest temperature allowed in a model

273.16 K (default) | scalar with units of temperature

Lowest temperature that the attached thermal liquid network is allowed to reach during simulation.

Maximum valid temperature — Highest temperature allowed in a model

363.16 K (default) | scalar with units of temperature

Highest temperature that the attached thermal liquid network is allowed to reach during simulation.

Minimum valid pressure — Lowest pressure allowed in a model

0.05 MPa (default) | scalar with units of pressure

Lowest pressure that the attached thermal liquid network is allowed to reach during simulation.

Maximum valid pressure — Highest pressure allowed in a model

50 MPa (default) | scalar with units of pressure

Highest pressure that the attached thermal liquid network is allowed to reach during simulation.

Pressure-temperature validity matrix — Matrix of allowed and disallowed pressure-temperature conditions

11-by-12 matrix of zeros and ones (default) | unitless *M*-by-*N* matrix

Matrix of validity values for the various temperature-pressure data points. A value of 1 denotes a valid point and a value of 0 denotes an invalid point.

Dependencies

This parameter is active only when the **Valid pressure-temperature region parameterization** parameter is set to `Validity matrix`.

Density

Density parameterization — Specification method for the fluid density

Density, bulk modulus and thermal expansion tables - `rho(T,p)`, `beta(T,p)`, `alpha(T,p)` (default) | Density table - `rho(T,p)` | Reference density | Density vector - `rho(T)`

Specification method for the density of the thermal liquid. The density can be a function of temperature and pressure or as a function of temperature alone.

Density table, `rho(T,p)` — Matrix of fluid densities at the specified temperatures and pressures

11-by-12 matrix (default) | *M*-by-*N* matrix with units of mass/volume

Tabulated density data specified at the temperatures and pressures provided in the **Temperature and Pressure** tab. Each row corresponds to a different temperature and each column corresponds to a different pressure.

Density vector, `rho(T)` — Vector of fluid densities at the specified temperatures

11-element column vector (default) | *M*-element vector with units of mass/volume

Tabulated density data specified at the temperatures provided in the **Temperature and Pressure** tab. Each element corresponds to a different temperature. The dependence of density on pressure is ignored.

Isothermal bulk modulus table, `beta(T,p)` — Matrix of isothermal bulk moduli at the specified temperatures and pressures

11-by-12 matrix (default) | *M*-by-*N* matrix with units of pressure

Tabulated data for the isothermal bulk modulus at the temperatures and pressures provided in the **Temperature and Pressure** tab. Each row corresponds to a different temperature and each column corresponds to a different pressure.

Isobaric thermal expansion coefficient table, `alpha(T,p)` — Matrix of isobaric thermal expansion coefficients at the specified temperatures and pressures

11-by-12 matrix (default) | *M*-by-*N* matrix with units of 1/temperature

Tabulated data for the isobaric thermal expansion coefficient at the temperatures and pressures provided in the **Temperature and Pressure** tab. Each row corresponds to a different temperature and each column corresponds to a different pressure.

Reference density — Density at a known operating condition

998.21 kg/m³ (default) | scalar with units of mass/volume

Density of the thermal liquid at a known operating point. The block uses this data to arrive at an analytical expression for density as a function pressure and temperature.

Reference temperature — Temperature at a known operating condition

293.16 K (default) | scalar with units of temperature

Temperature of the thermal liquid at a known operating point. The block uses this data to arrive at an analytical expression for density as a function pressure and temperature.

Reference pressure — Pressure at a known operating condition

0.101325 MPa (default) | scalar with units of pressure

Pressure of the thermal liquid at a known operating point. The block uses this data to arrive at an analytical expression for density as a function pressure and temperature.

Constant isothermal bulk modulus — Isothermal bulk modulus at a known operating condition

2.1791 GPa (default) | scalar with units of pressure

Isothermal bulk modulus of the thermal liquid at a known operating point. The block uses this data to arrive at an analytical expression for density as a function pressure and temperature.

Constant isobaric thermal expansion coefficient — Isobaric thermal expansion coefficient at a known operating condition

2.0691e-4 1/K (default) | scalar with units of 1/temperature

Isobaric thermal expansion coefficient of the thermal liquid at a known operating point. The block uses this data to arrive at an analytical expression for density as a function pressure and temperature.

Internal Energy**Internal energy parameterization — Specification method for the fluid internal energy**Specific internal energy and specific heat tables - $u(T,p)$, $cp(T,p)$ (default) | Specific internal energy table - $u(T,p)$ | Specific heat table - $cp(T,p)$

Specification method for the internal energy of the thermal liquid. The internal energy can be a function of temperature and pressure or as a function of temperature alone. It can also be calculated from specific heat data that you specify in the place of internal energy.

Specific internal energy table, $u(T,p)$ — Matrix of internal energies at the specified temperatures and pressures11-by-12 matrix (default) | M -by- N matrix with units of energy/mass

Tabulated data for the specific internal energy at the temperatures and pressures provided in the **Temperature and Pressure** tab. Each row corresponds to a different temperature and each column corresponds to a different pressure.

Specific internal energy vector, $u(T)$ — Vector of internal energies at the specified temperatures11-element vector (default) | M -element vector with units of energy/mass

Tabulated data for the specific internal energy at the temperatures provided in the **Temperature and Pressure** tab. Each element corresponds to a different temperature. The dependence of the specific internal energy on pressure is ignored.

Specific heat at constant pressure table, $cp(T,p)$ — Matrix of specific heats at the specified temperatures and pressures11-by-12 matrix (default) | M -by- N matrix with units of energy/(mass*temperature)

Tabulated data for the specific heat at the temperatures and pressures provided in the **Temperature and Pressure** tab. Each row corresponds to a different temperature and each column corresponds to a different pressure.

Specific heat at constant pressure vector, $u(T)$ — Vector of specific heats at the specified temperatures11-element vector (default) | M -element vector with units of energy/(mass*temperature)

Tabulated data for the specific heat at the temperatures provided in the **Temperature and Pressure** tab. Each row corresponds to a different temperature. The dependence of the specific heat on pressure is ignored.

Viscosity and Conductivity**Kinematic viscosity table, $\nu(T,p)$ — Matrix of kinematic viscosities at the specified temperatures and pressures**11-element vector (default) | M -element vector with units of area/time

Tabulated data for the kinematic viscosity at the temperatures and pressures provided in the **Temperature and Pressure** tab. Each row corresponds to a different temperature and each column corresponds to a different pressure.

Thermal conductivity table, $K(T,p)$ — Matrix of thermal conductivities at the specified temperatures and pressures11-element vector (default) | M -element vector with units of power/(length*temperature)

Tabulated data for the thermal conductivity at the temperatures and pressures provided in the **Temperature and Pressure** tab. Each row corresponds to a different temperature and each column corresponds to a different pressure.

Kinematic viscosity vector, $\nu(T)$ — Vector of kinematic viscosities at the specified temperatures11-by-12 matrix (default) | M -by- N matrix with units of area/time

Tabulated data for the kinematic viscosity at the temperatures provided in the **Temperature and Pressure** tab. Each row corresponds to a different temperature. The dependence of the kinematic viscosity on pressure is ignored.

Thermal conductivity vector, $K(T)$ — Vector of thermal conductivities at the specified temperatures11-by-12 matrix (default) | M -by- N matrix with units of power/(length*temperature)

Tabulated data for the thermal conductivity at the temperatures provided in the **Temperature and Pressure** tab. Each row corresponds to a different temperature. The dependence of the thermal conductivity on pressure is ignored.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2013b

Thermal Mass

Mass in thermal systems



Library

Thermal Elements

Description

The Thermal Mass block represents a thermal mass, which reflects the ability of a material or a combination of materials to store internal energy. The property is characterized by mass of the material and its specific heat. The thermal mass is described with the following equation:

$$Q = c \cdot m \frac{dT}{dt}$$

where

Q	Heat flow
c	Specific heat of mass material
m	Mass
T	Temperature
t	Time

The block has one thermal conserving port. The block positive direction is from its port towards the block. This means that the heat flow is positive if it flows into the block.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Mass

Mass. The default value is 1 kg.

Specific heat

Specific heat of the material. The default value is 447 J/kg/K.

Ports

The block has one thermal conserving port, associated with the mass connection to the system.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mass

Introduced in R2007b

Thermal Reference

Reference connection for thermal ports



Library

Thermal Elements

Description

The Thermal Reference block represents a thermal reference point, that is, a point with an absolute zero temperature, with respect to which all the temperatures in the system are determined.

Ports

The block has one thermal conserving port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Topics

“Grounding Rules”

Introduced in R2007b

Thermal Resistance

Constant resistance in thermal systems

Library: Simscape / Foundation Library / Thermal / Thermal Elements



Description

Thermal resistance is an abstract quantity that relates the thermal conductivity, the heat transfer coefficient, and the radiation coefficient. The Thermal Resistance block lets you model heat transfer in generalized terms, independent of whether it is by conduction, convection, radiation, or a combination thereof.

The heat transfer equation for the Thermal Resistance block is:

$$R \cdot Q = \Delta T$$

where:

- R is the thermal resistance.
- Q is the heat flow rate.
- ΔT is the temperature difference between layers.

Thermal resistance is related to the other heat transfer quantities as follows:

$$R = \frac{D}{k \cdot A} = \frac{1}{h \cdot A} = \frac{1}{r \cdot A(T_A^2 + T_B^2)(T_A + T_B)}$$

where:

- D is material thickness, that is, distance between layers.
- A is area normal to the heat flow direction.
- k is thermal conductivity of the material.
- h is convection heat transfer coefficient.
- r is radiation coefficient.
- T_A and T_B are temperatures at ports A and B, respectively.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

A — Layer A

thermal

Thermal conserving port associated with layer A.

B — Layer B

thermal

Thermal conserving port associated with layer B.

Parameters

Thermal resistance — Thermal resistance value

1 K/W (default)

Thermal resistance value.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Conductive Heat Transfer | Convective Heat Transfer | Radiative Heat Transfer | Variable Thermal Resistance

Introduced in R2017b

Thermal Resistor

Resistor with thermal port



Library

Electrical Elements

Description

The Thermal Resistor block represents a temperature-dependent resistor. When the temperature at the thermal port is T , the resistance is

$$R = R_0(1 + \alpha(T - T_0))$$

where:

- R_0 is the nominal resistance at the reference temperature T_0 .
- α is the temperature coefficient.

The following equation describes the thermal behavior of the block:

$$Q = K_d t_c \frac{dT}{dt} - i^2 R$$

where:

- Q is the net heat flow into port **H**.
- K_d is the **Dissipation factor** parameter value.
- t_c is the **Thermal time constant** parameter value.
- dT/dt is the rate of change of the temperature.
- i is the current through the resistor.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

In particular, the **Temperature** variable lets you set a high-priority target for the temperature of the thermal resistor at the start of the simulation. The default value is 300 K.

Parameters

Nominal resistance

The nominal resistance of the thermistor at the reference temperature. Many datasheets quote the nominal resistance at 25°C (298.15 K) and list it as R25. The default value is 1 Ω .

Reference temperature

The temperature at which the nominal resistance was measured. The default value is 298.15 K.

Temperature coefficient

The coefficient α in the equation that describes resistance as a function of temperature. The default value is 5e-05 1/K.

Thermal time constant

The time it takes the resistor temperature to reach 63% of the final temperature change when a step change in ambient temperature occurs. The default value is 10 s.

Dissipation factor

The thermal power required to raise the thermal resistor temperature by one K. The default value is 0.001 W/K.

Ports

The block has the following ports:

H

Resistor thermal port

+

Positive electrical port

-

Negative electrical port

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Resistor | Variable Resistor

Introduced in R2016a

Thermodynamic Properties Sensor (2P)

Measure temperature, specific enthalpy, specific entropy, and specific volume



Library

Two-Phase Fluid/Sensors

Description

The Thermodynamic Properties Sensor (2P) block measures the absolute temperature, specific enthalpy, specific entropy, and specific volume of a two-phase fluid. The measurements are taken at the two-phase fluid branch connected to port **A**. The sensor is ideal: it exchanges neither mass nor energy with the remainder of the system.

Ports

The block has one two-phase fluid conserving port, **A**, and four physical signal output ports:

- **T** — Temperature
- **h** — Specific enthalpy
- **v** — Specific volume
- **s** — Specific entropy

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Two-Phase Fluid Properties (2P)

Introduced in R2015b

Thermodynamic Properties Sensor (G)

Measure specific enthalpy, density, specific heat at constant pressure, and specific entropy

Library: Simscape / Foundation Library / Gas / Sensors



Description

The Thermodynamic Properties Sensor (G) block represents an ideal sensor that measures specific enthalpy, density, specific heat at constant pressure, and specific entropy in a gas network. There is no mass or energy flow through the sensor.

The block measures these thermodynamic properties at the node connected to port **A**. The first three measurements (specific enthalpy, density, specific heat at constant pressure) are taken with respect to absolute zero. There is no universal definition of what constitutes zero entropy, therefore you can use the block parameters to provide the reference entropy value for sensor calibration.

Ports

Output

h — Specific enthalpy measurement, kJ/kg

physical signal

Physical signal output port for measuring the gas specific enthalpy.

rho — Density measurement, kg/m³

physical signal

Physical signal output port for measuring the gas density.

cp — Specific heat at constant pressure measurement, kJ/(kg*K)

physical signal

Physical signal output port for measuring the gas specific heat at constant pressure.

s — Specific entropy measurement, kJ/(kg*K)

physical signal

Physical signal output port for measuring the gas specific entropy.

Conserving

A — Sensor inlet

gas

Gas conserving port. Connect this port to the node in the gas network where you want to measure the thermodynamic properties. All measurements are taken with respect to absolute zero.

Parameters

Specific entropy at reference pressure and temperature — Reference specific entropy used for sensor calibration

3.8635 kJ/kg/K (default)

The sensor is calibrated so that it outputs this specific entropy value when port **A** is at reference pressure and temperature.

Reference pressure for specific entropy — Reference pressure used for sensor calibration

0.101325 MPa (default)

The sensor is calibrated so that it outputs the value specified by the **Specific entropy at reference pressure and temperature** parameter when port **A** is at reference pressure and temperature.

Reference temperature for specific entropy — Reference temperature used for sensor calibration

293.15 K (default)

The sensor is calibrated so that it outputs the value specified by the **Specific entropy at reference pressure and temperature** parameter when port **A** is at reference pressure and temperature.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Gas Properties (G) | Mass & Energy Flow Rate Sensor (G) | Pressure & Temperature Sensor (G) | Volumetric Flow Rate Sensor (G)

Topics

“Modeling Gas Systems”

Introduced in R2017a

Thermodynamic Properties Sensor (MA)

Measure specific enthalpy, density, specific heat at constant pressure, and specific entropy of air mixture

Library: Simscape / Foundation Library / Moist Air / Sensors



Description

The Thermodynamic Properties Sensor (MA) block represents an ideal sensor that measures specific enthalpy, density, specific heat at constant pressure, and specific entropy of the air mixture in a moist air network. There is no mass or energy flow through the sensor.

The block measures these thermodynamic properties at the node connected to port **A**.

You have a choice of outputting these measurements normalized by the mass of the whole air mixture, or by the mass of dry air and trace gas only. The second option is consistent with the American Society of Heating, Refrigerating and Air Conditioning Engineers (ASHRAE) standards.

The specific entropy reference value, used for sensor calibration, is 0 for each constituent of the moist mixture at atmospheric pressure and atmospheric temperature. Atmospheric pressure and atmospheric temperature are defined by the parameters of the Moist Air Properties (MA) block connected to the circuit.

Ports

Output

h — Specific enthalpy measurement, J/kg

physical signal

Physical signal output port for measuring the specific enthalpy of the air mixture.

rho — Density measurement, kg/m³

physical signal

Physical signal output port for measuring the density of the air mixture.

cp — Specific heat at constant pressure measurement, J/(kg*K)

physical signal

Physical signal output port for measuring the specific heat of the air mixture at constant pressure.

s — Specific entropy measurement, J/(kg*K)

physical signal

Physical signal output port for measuring the specific entropy of the air mixture.

Conserving

A – Sensor inlet

moist air

Moist air conserving port. Connect this port to the node in the moist air network where you want to measure the thermodynamic properties.

Parameters

Measurements based on – Select measurement convention for sensor output

Unit mass of moist air mixture (default) | Unit mass of dry air and trace gas

Select whether measurements are normalized by the mass of the whole air mixture, or by the mass of dry air and trace gas only:

- **Unit mass of moist air mixture** – For each thermodynamic property, the measured mass of that property in a moist air volume is divided by the mass of the moist air mixture.
- **Unit mass of dry air and trace gas** – For each thermodynamic property, the measured mass of that property in a moist air volume is divided by the mass of dry air and trace gas (leaving out the mass of the water vapor). Use this option when you need to compare your measurements with ASHRAE figures and charts.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mass & Energy Flow Rate Sensor (MA) | Moist Air Properties (MA) | Pressure & Temperature Sensor (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Thermodynamic Properties Sensor (TL)

Measure specific internal energy, density, and specific heat at constant pressure

Library: Simscape / Foundation Library / Thermal Liquid / Sensors



Description

The Thermodynamic Properties Sensor (TL) block represents an ideal sensor that measures specific internal energy, density, and specific heat at constant pressure in a thermal liquid network. There is no mass or energy flow through the sensor.

The block measures these thermodynamic properties at the node connected to port **A**. All measurements are taken with respect to absolute zero.

Ports

Output

u — Specific internal energy measurement, kJ/kg

physical signal

Physical signal output port for measuring the specific internal energy of thermal liquid.

rho — Density measurement, kg/m³

physical signal

Physical signal output port for measuring the density of thermal liquid.

cp — Specific heat at constant pressure measurement, kJ/(kg*K)

physical signal

Physical signal output port for measuring the thermal liquid specific heat at constant pressure.

Conserving

A — Sensor inlet

thermal liquid

Thermal liquid conserving port. Connect this port to the node in the thermal liquid network where you want to measure the thermodynamic properties. All measurements are taken with respect to absolute zero.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mass & Energy Flow Rate Sensor (TL) | Pressure & Temperature Sensor (TL) | Thermal Liquid Settings (TL) | Volumetric Flow Rate Sensor (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2017a

Trace Gas Source (MA)

Inject or extract trace gas at a constant rate

Library: Simscape / Foundation Library / Moist Air / Sources /
Moisture & Trace Gas Sources



Description

The Trace Gas Source (MA) block represents a constant source or sink of trace gas for the connected moist air volume. A positive or negative trace gas mass flow rate causes trace gas levels to increase or decrease, respectively.

The energy associated with the added or removed trace gas is

$$\Phi_s = \begin{cases} \dot{m}_{\text{specified}} \cdot h_g(T_{\text{specified}}), & \text{if } \dot{m}_{\text{specified}} \geq 0 \\ \dot{m}_{\text{specified}} \cdot h_g(T_s), & \text{if } \dot{m}_{\text{specified}} < 0 \end{cases}$$

where:

- $\dot{m}_{\text{specified}}$ is the trace gas mass flow rate specified by the **Trace gas mass flow rate** parameter.
- h_g is the trace gas specific enthalpy.
- $T_{\text{specified}}$ is the temperature of added trace gas, as specified by the block parameters. The block uses this value to evaluate the specific enthalpy of the added trace gas only. The specific enthalpy of removed trace gas is based on the temperature of the connected moist air volume.
- T_s is the temperature at port **S**, which is the same as the temperature of the connected moist air volume.

Port **S** is a moist air source conserving port. Connect this port to port **S** of a block with finite moist air volume to add or remove trace gas through that block. For more information, see “Using Moisture and Trace Gas Sources”.

Ports

Conserving

S – Inject or extract trace gas

moist air source

Connect this port to port **S** of a block with finite moist air volume to add or remove trace gas through that block.

Parameters

Rate of added trace gas — Constant mass flow rate through the source

0 kg/s (default)

Trace gas mass flow rate through the source. A positive value adds trace gas to the connected moist air volume. A negative value extracts trace gas from that volume.

Added trace gas temperature specification — Select specification method for the temperature of added trace gas

Atmospheric temperature (default) | Specified temperature

Select a specification method for the trace gas temperature:

- **Atmospheric temperature** — Use the atmospheric temperature, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified temperature** — Specify a value by using the **Temperature of added trace gas** parameter.

Temperature of added trace gas — Trace gas temperature

293.15 K (default)

Enter the desired temperature of added trace gas. This temperature remains constant during simulation. The block uses this value to evaluate the specific enthalpy of the added trace gas only. The specific enthalpy of removed trace gas is based on the temperature of the connected moist air volume.

Dependencies

Enabled when the **Added trace gas temperature specification** parameter is set to Specified temperature.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Trace Gas Source (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Translational Damper

Viscous damper in mechanical translational systems



Library

Mechanical Translational Elements

Description

The Translational Damper block represents an ideal mechanical translational viscous damper, described with the following equations:

$$F = Dv$$

$$v = v_R - v_C$$

where

F	Force transmitted through the damper
D	Damping (viscous friction) coefficient
v	Relative velocity
v_R, v_C	Absolute velocities of terminals R and C, respectively

The block positive direction is from port R to port C. This means that the force is positive if it acts in the direction from R to C.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Damping coefficient

Damping coefficient, defined by viscous friction. The default value is 100 N/(m/s).

Ports

The block has the following ports:

R

Mechanical translational conserving port associated with the damper rod.

C

Mechanical translational conserving port associated with the damper case.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

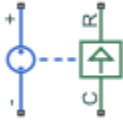
See Also

[Translational Friction](#) | [Translational Hard Stop](#) | [Translational Spring](#)

Introduced in R2007a

Translational Electromechanical Converter

Interface between electrical and mechanical translational domains



Library

Electrical Elements

Description

The Translational Electromechanical Converter block provides an interface between the electrical and mechanical translational domains. It converts electrical energy into mechanical energy in the form of translational motion, and vice versa. The converter is described with the following equations:

$$F = K \cdot I$$

$$V = K \cdot U$$

where

V	Voltage across the electrical ports of the converter
I	Current through the electrical ports of the converter
F	Force
U	Speed
K	Constant of proportionality

The Translational Electromechanical Converter block represents a lossless electromechanical energy conversion, therefore the same constant of proportionality is used in both equations.

Connections + and - are conserving electrical ports corresponding to the positive and negative terminals of the converter, respectively. Connections C and R are conserving mechanical translational ports. If the current flowing from the positive to the negative terminal is positive, then the resulting force is positive acting from port C to port R. This direction can be altered by using a negative value for K.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Constant of proportionality K

Constant of proportionality for electromechanical conversions. The default value is 0.1 V/(m/s).

Ports

The block has the following ports:

- +
Electrical conserving port associated with the converter positive terminal.
- Electrical conserving port associated with the converter negative terminal.
- C
Mechanical translational conserving port.
- R
Mechanical translational conserving port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Rotational Electromechanical Converter

Introduced in R2007a

Translational Free End

Translational port terminator with zero force

Library: Simscape / Foundation Library / Mechanical / Translational Elements



Description

The Translational Free End block represents a mechanical translational port that moves freely, without force.

You can use this block to terminate mechanical translational ports on other blocks that you want to leave unconnected. This is not necessary because if you leave a conserving port unconnected, the physical network sets all the Through variables at that port to 0. However, terminator blocks improve diagram readability.

You can also use this block to set the initial translational velocity at a node.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

V — Zero force

mechanical translational

Mechanical translational conserving port that moves freely, without force.

Compatibility Considerations

Capping unconnected ports is no longer required

Behavior changed in R2019b

This block is no longer required. Starting in R2019b, the restriction that disallowed unconnected conserving ports in Simscape models has been lifted. Now, if you leave a conserving port unconnected, the physical network sets all the Through variables at this port to 0. However, you can still use terminator blocks to improve diagram readability.

There are no plans to remove the terminator blocks. All existing models that use these blocks work the same as in previous releases.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Cap (2P) | Cap (G) | Cap (MA) | Cap (TL) | Hydraulic Cap | Open Circuit | Perfect Insulator | Rotational Free End

Introduced in R2012b

Translational Friction

Friction in contact between moving bodies

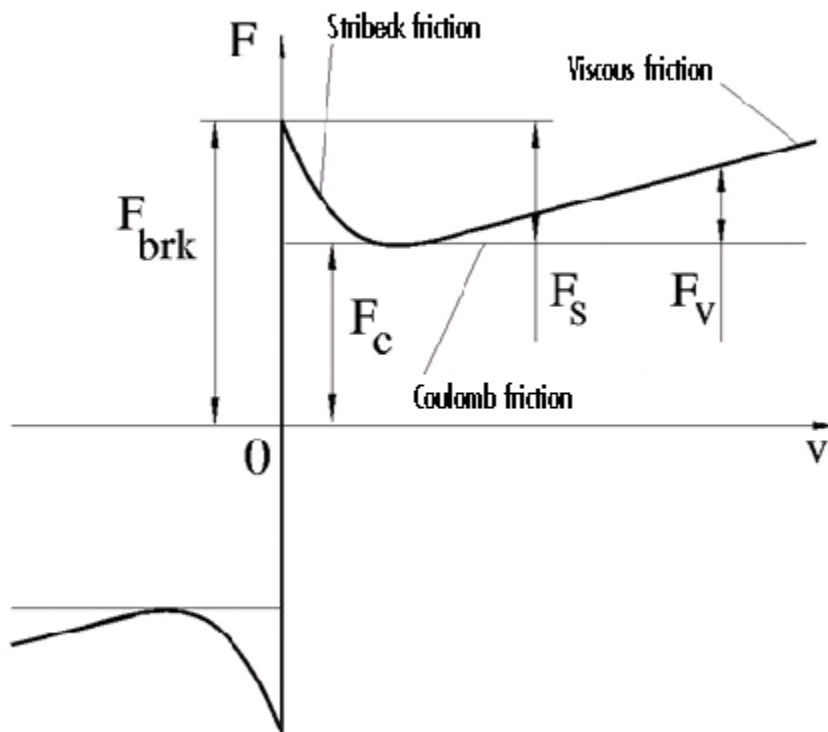


Library

Mechanical Translational Elements

Description

The Translational Friction block represents friction in contact between moving bodies. The friction force is simulated as a function of relative velocity and is assumed to be the sum of Stribeck, Coulomb, and viscous components, as shown in the following figure.



The Stribeck friction, F_s , is the negatively sloped characteristics taking place at low velocities (see [1] on page 1-700). The Coulomb friction, F_c , results in a constant force at any velocity. The viscous friction, F_v , opposes motion with the force directly proportional to the relative velocity. The sum of the Coulomb and Stribeck frictions at the vicinity of zero velocity is often referred to as the breakaway friction, F_{brk} . The friction is approximated with the following equations:

$$F = \sqrt{2e}(F_{brk} - F_C) \cdot \exp\left(-\left(\frac{v}{v_{St}}\right)^2\right) \cdot \frac{v}{v_{St}} + F_C \cdot \tanh\left(\frac{v}{v_{Coul}}\right) + fv$$

$$v_{St} = v_{brk}\sqrt{2}$$

$$v_{Coul} = v_{brk}/10$$

$$v = v_R - v_C$$

where

F	Friction force
F_C	Coulomb friction
F_{brk}	Breakaway friction
v_{brk}	Breakaway friction velocity
v_{St}	Stribeck velocity threshold
v_{Coul}	Coulomb velocity threshold
v_R, v_C	Absolute velocities of terminals R and C, respectively
v	Relative velocity
f	Viscous friction coefficient

The exponential function used in the Stribeck portion of the force equation is continuous and decays at velocity magnitudes greater than the breakaway friction velocity.

The hyperbolic tangent function used in the Coulomb portion of the force equation ensures that the equation is smooth and continuous through $v = 0$, but quickly reaches its full value at nonzero velocities.

The block positive direction is from port R to port C. This means that if the port R velocity is greater than that of port C, the block transmits force from R to C.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Breakaway friction force

The breakaway friction force, which is the sum of the Coulomb and the static frictions. It must be greater than or equal to the Coulomb friction force value. The default value is 25 N.

Breakaway friction velocity

The velocity at which the Stribeck friction is at its peak. At this point, the sum of the Stribeck and Coulomb friction is the **Breakaway friction force**. The default value is 0.1 m/s.

Coulomb friction force

The Coulomb friction force, which is the friction that opposes motion with a constant force at any velocity. The default value is 20 N.

Viscous friction coefficient

Proportionality coefficient between the friction force and the relative velocity. The parameter value must be greater than or equal to zero. The default value is 100 N/(m/s).

Ports

The block has the following ports:

R

Mechanical translational conserving port.

C

Mechanical translational conserving port.

References

[1] B. Armstrong, C.C. de Wit, *Friction Modeling and Compensation*, The Control Handbook, CRC Press, 1995

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

[Translational Damper](#) | [Translational Hard Stop](#) | [Translational Spring](#)

Introduced in R2007a

Translational Hard Stop

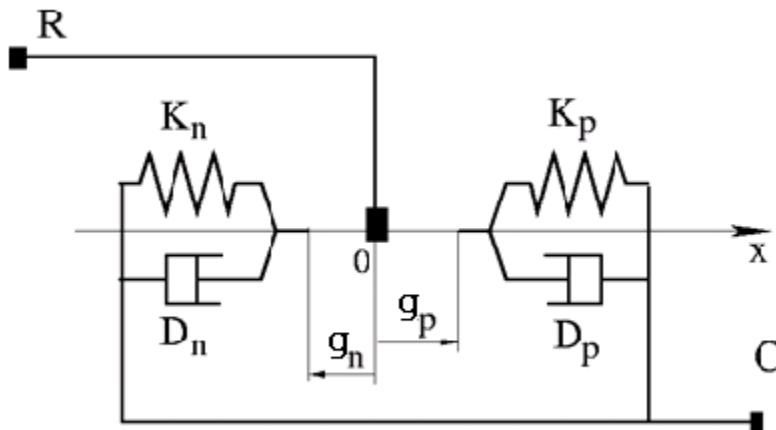
Double-sided translational hard stop

Library: Simscape / Foundation Library / Mechanical / Translational Elements



Description

The Translational Hard Stop block represents a double-sided mechanical translational hard stop that restricts motion of a body between upper and lower bounds. The impact interaction between the slider and the stops is assumed to be elastic. This means that the stop is represented as a spring that comes into contact with the slider as the gap is cleared and opposes slider penetration into the stop with the force linearly proportional to this penetration. To account for energy dissipation and nonelastic effects, damping is introduced as the block's parameter, thus making it possible to account for energy loss. The schematic shows the idealization of the mechanical translational hard stop adopted in the block.



The basic hard stop model, Full stiffness and damping applied at bounds, damped rebound, is described with the following equations:

$$F = \begin{cases} K_p \cdot (x - g_p) + D_p \cdot v & \text{for } x \geq g_p \\ 0 & \text{for } g_n < x < g_p \\ K_n \cdot (x - g_n) + D_n \cdot v & \text{for } x \leq g_n \end{cases}$$

$$v = \frac{dx}{dt}$$

where

- F is interaction force between the slider and the case.

- g_p is the initial gap between the slider and upper bound.
- g_n is the initial gap between the slider and lower bound.
- x is the slider position.
- K_p is contact stiffness at upper bound.
- K_n is contact stiffness at lower bound.
- D_p is damping coefficient at upper bound.
- D_n is damping coefficient at lower bound.
- v is the slider velocity.
- t is time.

In the `Full stiffness and damping applied at bounds, undamped rebound hard stop` model, equations contain additional terms, `ge(v,0)` and `le(v,0)`. These terms ensure that damping is not applied on the rebound.

$$F = \begin{cases} K_p \cdot (x - g_p) + D_p \cdot v \cdot ge(v, 0) & \text{for } x \geq g_p \\ 0 & \text{for } g_n < x < g_p \\ K_n \cdot (x - g_n) + D_n \cdot v \cdot le(v, 0) & \text{for } x \leq g_n \end{cases}$$

Relational functions `ge` (greater or equal) and `le` (less or equal) do not generate zero crossings when velocity changes sign. For more information, see “Enabling and Disabling Zero-Crossing Conditions in Simscape Language”. However, the solver treats `ge` and `le` functions as nonlinear. Therefore, if `simscape.findNonlinearBlocks` indicates that the rest of your network is linear or switched linear, use the `Full stiffness and damping applied at bounds, damped rebound` model to improve performance.

The default hard stop model, `Stiffness and damping applied smoothly through transition region, damped rebound`, adds two transitional regions to the equations, one at each bound. While the slider travels through a transition region, the block smoothly ramps up the force from zero to the full value. At the end of the transition region, the full stiffness and damping are applied. On the rebound, both stiffness and damping forces are smoothly decreased back to zero. These equations also use the `ge` and `le` relational functions, which do not produce zero crossings.

The block is oriented from R to C. This means that the block transmits force from port R to port C when the gap is closed in the positive direction.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

R — Rod (slider)

mechanical translational

Mechanical translational conserving port associated with the slider that travels between stops installed on the case.

C – Case

mechanical translational

Mechanical translational conserving port associated with the case.

Parameters**Upper bound – Gap between slider and upper bound**

0.1 m (default)

Gap between the slider and the upper bound. The direction is specified with respect to the local coordinate system, with the slider located in the origin. A positive value of the parameter specifies the gap between the slider and the upper bound. A negative value sets the slider as penetrating into the upper bound.

Lower bound – Gap between slider and lower bound

-0.1 m (default)

Gap between the slider and the lower bound. The direction is specified with respect to the local coordinate system, with the slider located in the origin. A negative value of the parameter specifies the gap between the slider and the lower bound. A positive value sets the slider as penetrating into the lower bound.

Contact stiffness at upper bound – Elasticity of collision at upper bound

1e6 N/m (default)

This parameter specifies the elastic property of colliding bodies when the slider hits the upper bound. The greater the value of the parameter, the less the bodies penetrate into each other, the more rigid the impact becomes. Lesser value of the parameter makes contact softer, but generally improves convergence and computational efficiency.

Contact stiffness at lower bound – Elasticity of collision at lower bound

1e6 N/m (default)

This parameter specifies the elastic property of colliding bodies when the slider hits the upper bound. The greater the value of the parameter, the less the bodies penetrate into each other, the more rigid the impact becomes. Lesser value of the parameter makes contact softer, but generally improves convergence and computational efficiency.

Contact damping at upper bound – Dissipating property at upper bound

150 N/(m/s) (default)

This parameter specifies dissipating property of colliding bodies when the slider hits the upper bound. The greater the value of the parameter, the more energy dissipates during an interaction.

Contact damping at lower bound – Dissipating property at lower bound

150 N/(m/s) (default)

This parameter specifies dissipating property of colliding bodies when the slider hits the lower bound. The greater the value of the parameter, the more energy dissipates during an interaction.

Hard stop model — Select the hard stop model

Stiffness and damping applied smoothly through transition region, damped rebound (default) | Full stiffness and damping applied at bounds, undamped rebound | Full stiffness and damping applied at bounds, damped rebound

Select the hard stop model:

- Stiffness and damping applied smoothly through transition region, damped rebound — Specify a transition region, in which the force is scaled from zero. At the end of the transition region, the full stiffness and damping are applied. This model has damping applied on the rebound, but it is limited to the value of the stiffness force. In this sense, damping can reduce or eliminate the force provided by the stiffness, but never exceed it. All equations are smooth and produce no zero crossings.
- Full stiffness and damping applied at bounds, undamped rebound — This model has full stiffness and damping applied with impact at upper and lower bounds, with no damping on the rebound. Equations produce no zero crossings when velocity changes sign, but there is a position-based zero crossing at the bounds. Having no damping on rebound helps to push the slider past this position quickly. This model has nonlinear equations.
- Full stiffness and damping applied at bounds, damped rebound — This model has full stiffness and damping applied with impact at upper and lower bounds, with damping applied on the rebound as well. Equations are switched linear, but produce position-based zero crossings. Use this hard stop model if `simscape.findNonlinearBlocks` indicates that this is the block that prevents the whole network from being switched linear.

Transition region — Region where force is ramped up

0.1 mm (default)

Region where the force is ramped up from zero to the full value. At the end of the transition region, the full stiffness and damping are applied.

Dependencies

Enabled when the **Hard stop model** parameter is set to Stiffness and damping applied smoothly through transition region, damped rebound.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Translational Damper | Translational Friction | Translational Spring

Introduced in R2007a

Translational Hydro-Mechanical Converter

Interface between hydraulic and mechanical translational domains



Library

Hydraulic Elements

Description

The Translational Hydro-Mechanical Converter block models an ideal transducer that converts hydraulic energy into mechanical energy, in the form of translational motion of the converter output member, and vice versa. The compressibility option makes the converter account for dynamic variations of the fluid density.

Using this block as a basic element, you can build a large variety of hydraulic cylinder models by adding application-specific effects, such as leakage, friction, hard stops, and so on.

The converter is simulated according to the following equations:

$$q = \frac{d\left(\frac{\rho}{\rho_l^0} V\right)}{dt} = \frac{d\left(\frac{\rho}{\rho_l^0}\right)}{dt} V + \frac{\rho}{\rho_l^0} \cdot \varepsilon \cdot (v_R - v_C) \cdot A$$

$$F = \varepsilon \cdot p \cdot A$$

$$\rho = \begin{cases} \frac{\left(\frac{\alpha}{1-\alpha}\right)\rho_g^0 + \rho_l^0}{\left(\frac{\alpha}{1-\alpha}\right)\left(\frac{p_0}{p}\right)^{\frac{1}{\gamma}} + e^{-\frac{p-p_0}{\beta_l}}} & \text{if compressibility is on} \\ \rho_l^0 & \text{if compressibility is off} \end{cases}$$

where

q	Flow rate to the converter chamber
A	Effective piston area
v_R	Converter rod velocity
v_C	Converter case velocity
F	Force developed by the converter
p	Gauge pressure of fluid in the converter chamber
V	Piston volume
α	Relative amount of trapped air

ρ_l^0	Fluid density at atmospheric conditions
ρ_g^0	Gas density at atmospheric conditions
p_0	Atmospheric pressure
γ	Specific heat ratio
β_l	Bulk modulus at atmospheric conditions and no gas
ε	Converter orientation with respect to the globally assigned positive direction. If pressure applied at port A exerts force in positive direction, ε equals 1. If pressure applied at port A exerts force in negative direction, ε equals -1.

The piston volume is computed according to

$$V = V_{dead} + A \cdot (x_0 + x)$$

$$\frac{dx}{dt} = \varepsilon \cdot (v_R - v_C)$$

where

V_{dead}	Chamber dead volume
x_0	Piston initial position
x	Piston displacement from initial position

Port A is a hydraulic conserving port associated with the converter inlet. Ports R and C are translational mechanical conserving ports associated with the rod and the case of the converter, respectively.

The block dialog box does not have a **Source code** link. To view the underlying component source, open the following files in the MATLAB editor:

- For incompressible converter implementation — `translational_converter_incompressible.ssc`
- For compressible converter implementation — `translational_converter_compressible.ssc`

Basic Assumptions and Limitations

The block simulates an ideal converter, with an option to account for fluid compressibility. Other effects, such as hard stops, inertia, or leakage, are modeled outside of the converter.

Parameters

Piston area

Effective piston area. The default value is $5e-4 \text{ m}^2$.

Converter orientation

Specifies converter orientation with respect to the globally assigned positive direction. The converter can be installed in two different ways, depending upon whether it exerts force in the positive or in the negative direction when pressure is applied at its inlet. If pressure applied at port A exerts force in negative direction, set the parameter to `Acts in negative direction`. The default value is `Acts in positive direction`.

Compressibility

Specifies whether fluid density is taken as constant or varying with pressure. The default value is **Off**, in which case the block models an ideal transducer. If you select **On**, the block dialog box displays additional parameters that let you model dynamic variations of the fluid density without adding any extra blocks.

Piston initial position

Initial offset of the piston from the cylinder cap. The default value is 0 .

Dead volume

Volume of fluid in the chamber at zero piston position. The default value is $1e-4 \text{ m}^3$.

Specific heat ratio

Gas-specific heat ratio. The default value is 1.4 .

Initial pressure

Initial pressure in the chamber. This parameter specifies the initial condition for use in computing the block's initial state at the beginning of a simulation run. The default value is 0 .

Restricted Parameters

When your model is in Restricted editing mode, you cannot modify the following parameters:

- **Converter orientation**
- **Compressibility**

All other block parameters are available for modification.

Ports

The block has the following ports:

A

Hydraulic conserving port associated with the converter inlet.

R

Mechanical translational conserving port associated with the rod of the converter.

C

Mechanical translational conserving port associated with the case of the converter.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Rotational Hydro-Mechanical Converter

Introduced in R2007a

Translational Inerter

Two-port inertia in mechanical translational systems



Library

Mechanical Translational Elements

Description

The Translational Inerter block represents a device that has force proportional to the rate of change of the relative velocity across the ports. It is essentially a two-port inertia that works on the velocity difference between the ports, not the absolute velocity. An inerter is the mechanical equivalent of a capacitor. An inerter with one port connected to ground essentially behaves as a mass with the mass equal to the inerter's inertance.

Use this block in high performance suspension systems, to decouple weave and roll modes, or in applications where you need to model a passively tuned mass-spring-damper response.

The block is described with the following equations:

$$f = B \frac{dv}{dt}$$

$$v = v_R - v_C$$

where

F	Force transmitted through the inerter
B	Inertance
v	Relative velocity
v_R, v_C	Absolute velocities at ports R and C, respectively

The block positive direction is from port R to port C. This means that the force is positive if it acts in the direction from R to C.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see "Set Priority and Initial Target for Block Variables".

Parameters

Inertance

Proportionality coefficient between the force and the rate of change of the relative velocity across the ports. The default value is 1 kg.

Ports

The block has the following ports:

R

Mechanical translational conserving port associated with the rod.

C

Mechanical translational conserving port associated with the case.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mass

Introduced in R2015b

Translational Mechanical Converter (2P)

Interface between two-phase fluid and mechanical translational networks



Library

Two-Phase Fluid/Elements

Description

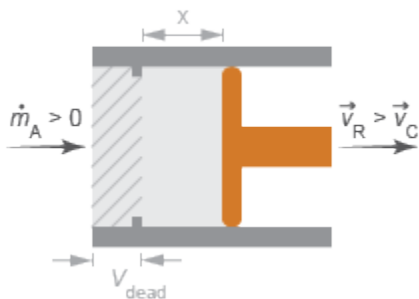
The Translational Mechanical Converter (2P) block models an interface between two-phase fluid and mechanical translational networks. The interface converts pressure in the fluid network into force in the mechanical translational network and vice versa.

This block enables you to model a linear actuator powered by a two-phase fluid system. It does not, however, account for mass, friction, or hard stops, common in linear actuators. You can model these effects separately using Simscape blocks such as Mass, Translational Friction, and Translational Hard Stop.

Port A represents the inlet through which fluid enters and exits the converter. Ports C and R represent the converter casing and moving interface, respectively. Port H represents the wall through which the converter exchanges heat with its surroundings.

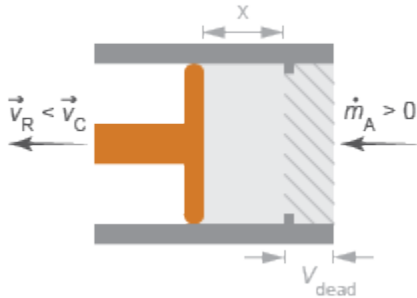
Force Direction

The force direction depends on the mechanical orientation of the converter. If the **Mechanical Orientation** parameter is set to positive, then a positive flow rate through the inlet tends to translate the moving interface in the positive direction relative to the converter casing.



Positive Mechanical Orientation

If the **Mechanical Orientation** parameter is set to negative, then a positive mass flow rate through the inlet tends to translate the moving interface in the negative direction relative to the converter casing.



Negative Mechanical Orientation

The flow resistance between port A and the converter interior is assumed negligible. Pressure losses between the two is approximately zero. The pressure at port A is therefore equal to that in the converter:

$$p_A = p_I,$$

where:

- p_A is the pressure at port A.
- p_I is the pressure in the converter.

Similarly, the thermal resistance between port H and the converter interior is assumed negligible. The temperature gradient between the two is approximately zero. The temperature at port H is therefore equal to that in the converter:

$$T_H = T_I,$$

where:

- T_H is the temperature at port H.
- T_I is the temperature in the converter.

Fluid Volume

The volume of fluid in the converter is the sum of the dead and displaced fluid volumes. The dead volume is the amount of fluid left in the converter at a zero interface displacement. This volume enables you to model the effects of dynamic compressibility and thermal capacity even when the interface is in its zero position.

The displacement volume is the amount of fluid added to the converter due to translation of the moving interface. This volume increases with the interface displacement. The total volume in the converter as a function of the interface displacement is

$$V = V_{dead} + S_{int} x_{int},$$

where:

- V is the total volume of fluid in the converter.
- V_{dead} is the dead volume of the converter.
- S_{int} is the cross-sectional area of the interface, assumed equal to that of the inlet.
- x_{int} is the displacement of the moving interface.
- ϵ_{or} is an integer encoding the mechanical orientation of the converter:

$$\epsilon_{\text{or}} = \begin{cases} +1, & \text{if the mechanical orientation is positive} \\ -1, & \text{if the mechanical orientation is negative} \end{cases}$$

Force Balance

At equilibrium, the internal pressure in the converter counteracts the external pressure of its surroundings and the force exerted by the mechanical network on the moving interface. This force is the reverse of that applied by the fluid network. The force balance in the converter is therefore

$$pI S_{\text{int}} = p_{\text{atm}} S_{\text{int}} - F_{\text{int}} \epsilon_{\text{or}},$$

where:

- p_{atm} is the environmental pressure outside the converter.
- F_{int} is the magnitude of the force exerted by the fluid network on the moving interface.

Energy Balance

The total energy in the converter can change due to energy flow through the inlet, heat flow through the converter wall, and work done by the fluid network on the mechanical network. The energy flow rate, given by the energy conservation equation, is therefore

$$\dot{E} = \phi_A + \phi_H - pI S_{\text{int}} \dot{x}_{\text{int}} \epsilon_{\text{or}},$$

where:

- E is the total energy of the fluid in the converter.
- ϕ_A is the energy flow rate into the converter through port A.
- ϕ_H is the heat flow rate into the converter through port H.

Taking the fluid kinetic energy in the converter to be negligible, the total energy of the fluid reduces to:

$$E = M u_I,$$

where:

- M is the fluid mass in the converter.
- u_I is the specific internal energy of the fluid in the converter.

Mass Balance

The fluid mass in the converter can change due to flow through the inlet, represented by port A. The mass flow rate, given by the mass conservation equation, is therefore

$$\dot{M} = \dot{m}_A,$$

where:

- \dot{m}_A is the mass flow rate into the converter through port A.

A change in fluid mass can accompany a change in fluid volume, due to translation of the moving interface. It can also accompany a change in mass density, due to an evolving pressure or specific internal energy in the converter. The mass rate of change in the converter is then

$$\dot{M} = \left[\left(\frac{\partial \rho}{\partial p} \right)_u \dot{p}_I + \left(\frac{\partial \rho}{\partial u} \right)_p \dot{u}_I \right] V + \frac{S_{int} \dot{x}_{int} \epsilon_{or}}{v_I},$$

where:

- $\left(\frac{\partial \rho}{\partial p} \right)_u$ is the partial derivative of density with respect to pressure at constant specific internal energy.
- $\left(\frac{\partial \rho}{\partial u} \right)_p$ is the partial derivative of density with respect to specific internal energy at constant pressure.
- v_I is the specific volume of the fluid in the converter.

The block blends the density partial derivatives of the various domains using a cubic polynomial function. At a vapor quality of 0-0.1, this function blends the derivatives of the subcooled liquid and two-phase mixture domains. At a vapor quality of 0.9-1, it blends those of the two-phase mixture and superheated vapor domains.

The smoothed density partial derivatives introduce into the original mass conservation equation undesirable numerical errors. To correct for these errors, the block adds the correction term

$$\epsilon_M = \frac{M - V/v_I}{\tau},$$

where:

- ϵ_M is the correction term.
- τ is the phase-change time constant—the characteristic duration of a phase change event. This constant ensures that phase changes do not occur instantaneously, effectively introducing a time lag whenever they occur.

The final form of the mass conservation equation is

$$\left[\left(\frac{\partial \rho}{\partial p} \right)_u \dot{p}_I + \left(\frac{\partial \rho}{\partial u} \right)_p \dot{u}_I \right] V + \frac{D_{vol} \dot{\theta}_{int} \epsilon_{or}}{v_I} = \dot{m}_A + \epsilon_M.$$

The block uses this equation to calculate the internal pressure in the converter given the mass flow rate through the inlet.

Assumptions and Limitations

- The converter walls are rigid. They do not deform under pressure.
- The flow resistance between port A and the converter interior is negligible. The pressure is the same at port A and in the converter interior.

- The thermal resistance between port H and the converter interior is negligible. The temperature is the same at port H and in the converter interior.
- The moving interface is perfectly sealed. No fluid leaks across the interface.
- Mechanical effects such as hard stops, inertia, and friction, are ignored.

Parameters

Parameters Tab

Mechanical orientation

Alignment of the moving interface relative to the direction of flow. If the mechanical orientation is positive, a positive flow rate into the converter through port A corresponds to a positive translation of port R relative to port C. If the mechanical orientation is negative, a positive flow rate corresponds to a negative translation instead. The default setting is **Positive**.

Interface cross-sectional area

Area normal to the direction of flow at the converter inlet. This area need not be the same as the inlet area. Pressure losses due to changes in flow area inside the converter are ignored. The default value is 0.01 m^2 .

Interface initial displacement

Translational offset of the moving interface at the start of simulation. A zero displacement corresponds to a total fluid volume in the converter equal to the specified dead volume. The default value is 0 m .

This parameter must be greater than or equal to zero if the **Mechanical orientation** parameter is set to **Positive**. It must be smaller than or equal to zero if the **Mechanical orientation** parameter is set to **Negative**.

Dead volume

Volume of fluid left in the converter when the interface displacement is zero. The dead volume enables the block to account for mass and energy storage in the converter even at a zero interface displacement. The default value is $1\text{e-}5 \text{ m}^3$.

Cross-sectional area at port A

Flow area of the converter inlet, represented by port **A**. This area need not be the same as the interface cross-sectional area. Pressure losses due to changes in flow area inside the converter are ignored. The default value is 0.01 m^2 .

Environment pressure specification

Pressure characteristics of the surrounding environment. Select **Atmospheric pressure** to set the environment pressure to the atmospheric pressure specified in the Two-Phase Fluid Properties (2P) block. Select **Specified pressure** to set the environment pressure to a different value. The default setting is **Atmospheric pressure**.

Environment pressure

Absolute pressure of the surrounding environment. The environment pressure acts against the internal pressure of the converter and affects the motion of the converter shaft. This parameter is active only when the **Environment pressure specification** parameter is set to **Specified pressure**. The default value, 0.101325 MPa , corresponds to atmospheric pressure at mean sea level.

Effects and Initial Conditions Tab

Initial fluid energy specification

Thermodynamic variable in terms of which to define the initial conditions of the component. The default setting is Temperature.

Initial pressure

Pressure in the chamber at the start of simulation, specified against absolute zero. The default value is 0.101325 MPa.

Initial temperature

Temperature in the chamber at the start of simulation, specified against absolute zero. This parameter is active when the **Initial fluid energy specification** option is set to Temperature. The default value is 293.15 K.

Initial vapor quality

Mass fraction of vapor in the chamber at the start of simulation. This parameter is active when the **Initial fluid energy specification** option is set to Vapor quality. The default value is 0.5.

Initial vapor void fraction

Volume fraction of vapor in the chamber at the start of simulation. This parameter is active when the **Initial fluid energy specification** option is set to Vapor void fraction. The default value is 0.5.

Initial specific enthalpy

Specific enthalpy of the fluid in the chamber at the start of simulation. This parameter is active when the **Initial fluid energy specification** option is set to Specific enthalpy. The default value is 1500 kJ/kg.

Initial specific internal energy

Specific internal energy of the fluid in the chamber at the start of simulation. This parameter is active when the **Initial fluid energy specification** option is set to Specific internal energy. The default value is 1500 kJ/kg.

Phase change time constant

Characteristic duration of a phase-change event. This constant introduces a time lag into the transition between phases. The default value is 0.1 s.

Ports

The block has the following ports:

A

Two-phase fluid conserving port associated with the converter inlet.

H

Thermal conserving port representing the converter surface through which heat exchange occurs.

R

Mechanical translational conserving port associated with the converter rod.

C

Mechanical translational conserving port associated with the converter case.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Rotational Mechanical Converter (2P)

Introduced in R2015b

Translational Mechanical Converter (G)

Interface between gas and mechanical translational networks

Library: Simscape / Foundation Library / Gas / Elements



Description

The Translational Mechanical Converter (G) block models an interface between a gas network and a mechanical translational network. The block converts gas pressure into mechanical force and vice versa. It can be used as a building block for linear actuators.

The converter contains a variable volume of gas. The pressure and temperature evolve based on the compressibility and thermal capacity of this gas volume. If **Mechanical orientation** is set to **Positive**, then an increase in the gas volume results in a positive displacement of port **R** relative to port **C**. If **Mechanical orientation** is set to **Negative**, then an increase in the gas volume results in a negative displacement of port **R** relative to port **C**.

Port **A** is the gas conserving port associated with the converter inlet. Port **H** is the thermal conserving port associated with the temperature of the gas inside the converter. Ports **R** and **C** are the mechanical translational conserving ports associated with the moving interface and converter casing, respectively.

Mass Balance

Mass conservation equation is similar to that for the Constant Volume Chamber (G) block, with an additional term related to the change in gas volume:

$$\frac{\partial M}{\partial p} \cdot \frac{dp_I}{dt} + \frac{\partial M}{\partial T} \cdot \frac{dT_I}{dt} + \rho_I \frac{dV}{dt} = \dot{m}_A$$

where:

- $\frac{\partial M}{\partial p}$ is the partial derivative of the mass of the gas volume with respect to pressure at constant temperature and volume.
- $\frac{\partial M}{\partial T}$ is the partial derivative of the mass of the gas volume with respect to temperature at constant pressure and volume.
- p_I is the pressure of the gas volume. Pressure at port **A** is assumed equal to this pressure, $p_A = p_I$.
- T_I is the temperature of the gas volume. Temperature at port **H** is assumed equal to this temperature, $T_H = T_I$.
- ρ_I is the density of the gas volume.
- V is the volume of gas.
- t is time.

- \dot{m}_A is the mass flow rate at port **A**. Flow rate associated with a port is positive when it flows into the block.

Energy Balance

Energy conservation equation is also similar to that for the Constant Volume Chamber (G) block. The additional term accounts for the change in gas volume, as well as the pressure-volume work done by the gas on the moving interface:

$$\frac{\partial U}{\partial p} \cdot \frac{dp_I}{dt} + \frac{\partial U}{\partial T} \cdot \frac{dT_I}{dt} + \rho_I h_I \frac{dV}{dt} = \Phi_A + Q_H$$

where:

- $\frac{\partial U}{\partial p}$ is the partial derivative of the internal energy of the gas volume with respect to pressure at constant temperature and volume.
- $\frac{\partial U}{\partial T}$ is the partial derivative of the internal energy of the gas volume with respect to temperature at constant pressure and volume.
- Φ_A is the energy flow rate at port **A**.
- Q_H is the heat flow rate at port **H**.
- h_I is the specific enthalpy of the gas volume.

Partial Derivatives for Perfect and Semiperfect Gas Models

The partial derivatives of the mass M and the internal energy U of the gas volume, with respect to pressure and temperature at constant volume, depend on the gas property model. For perfect and semiperfect gas models, the equations are:

$$\frac{\partial M}{\partial p} = V \frac{\rho_I}{p_I}$$

$$\frac{\partial M}{\partial T} = -V \frac{\rho_I}{T_I}$$

$$\frac{\partial U}{\partial p} = V \left(\frac{h_I}{ZRT_I} - 1 \right)$$

$$\frac{\partial U}{\partial T} = V \rho_I \left(c_{pI} - \frac{h_I}{T_I} \right)$$

where:

- ρ_I is the density of the gas volume.
- V is the volume of gas.
- h_I is the specific enthalpy of the gas volume.
- Z is the compressibility factor.
- R is the specific gas constant.
- c_{pI} is the specific heat at constant pressure of the gas volume.

Partial Derivatives for Real Gas Model

For real gas model, the partial derivatives of the mass M and the internal energy U of the gas volume, with respect to pressure and temperature at constant volume, are:

$$\frac{\partial M}{\partial p} = V \frac{\rho_I}{\beta_I}$$

$$\frac{\partial M}{\partial T} = -V \rho_I \alpha_I$$

$$\frac{\partial U}{\partial p} = V \left(\frac{\rho_I h_I}{\beta_I} - T_I \alpha_I \right)$$

$$\frac{\partial U}{\partial T} = V \rho_I (c_{pI} - h_I \alpha_I)$$

where:

- β is the isothermal bulk modulus of the gas volume.
- α is the isobaric thermal expansion coefficient of the gas volume.

Gas Volume

The gas volume depends on the displacement of the moving interface:

$$V = V_{dead} + S_{int} x_{int} \varepsilon_{int}$$

where:

- V_{dead} is the dead volume.
- S_{int} is the interface cross-sectional area.
- x_{int} is the interface displacement.
- ε_{int} is the mechanical orientation coefficient. If **Mechanical orientation** is Positive, $\varepsilon_{int} = 1$. If **Mechanical orientation** is Negative, $\varepsilon_{int} = -1$.

Force Balance

Force balance across the moving interface on the gas volume is

$$F_{int} = (p_{env} - p_I) S_{int} \varepsilon_{int}$$

where:

- F_{int} is the force from port **R** to port **C**.
- p_{env} is the environment pressure.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Gas Volume”.

Assumptions and Limitations

- The converter casing is perfectly rigid.
- There is no flow resistance between port **A** and the converter interior.
- There is no thermal resistance between port **H** and the converter interior.
- The moving interface is perfectly sealed.
- The block does not model mechanical effects of the moving interface, such as hard stop, friction, and inertia.

Ports

Conserving

A — Converter inlet

gas

Gas conserving port associated with the converter inlet.

H — Temperature inside converter

thermal

Thermal conserving port associated with the temperature of the gas inside the converter.

R — Rod

mechanical translational

Mechanical translational conserving port associated with the moving interface.

C — Case

mechanical translational

Mechanical translational conserving port associated with the converter casing.

Parameters

Mechanical orientation — Select the converter orientation

Positive (default) | Negative

Select the relative orientation of the converter with respect to the converter gas volume:

- **Positive** — Increase in the gas volume results in a positive displacement of port **R** relative to port **C**.
- **Negative** — Increase in the gas volume results in a negative displacement of port **R** relative to port **C**.

Interface cross-sectional area — The area on which the gas exerts pressure to generate the translational force

0.01 m² (default)

The area on which the gas exerts pressure to generate the translational force.

Interface initial displacement — Translational offset of port R relative to port C at the start of simulation

0 m (default)

Translational offset of port **R** relative to port **C** at the start of simulation. A value of 0 corresponds to an initial gas volume equal to **Dead volume**.

Dependencies

- If **Mechanical orientation** is **Positive**, the parameter value must be greater than or equal to 0.
- If **Mechanical orientation** is **Negative**, the parameter value must be less than or equal to 0.

Dead volume — Volume of gas when the interface displacement is 01e-5 m³ (default)

Volume of gas when the interface displacement is 0.

Cross-sectional area at port A — Area normal to flow path at the converter inlet0.01 m² (default)

The cross-sectional area of the converter inlet, in the direction normal to gas flow path.

Environment pressure specification — Select a specification method for the environment pressure

Atmospheric pressure (default) | Specified pressure

Select a specification method for the environment pressure:

- **Atmospheric pressure** — Use the atmospheric pressure, specified by the Gas Properties (G) block connected to the circuit.
- **Specified pressure** — Specify a value by using the **Environment pressure** parameter.

Environment pressure — Pressure outside the converter

0.101325 MPa (default)

Pressure outside the converter acting against the pressure of the converter gas volume. A value of 0 indicates that the converter expands into vacuum.

Dependencies

Enabled when the **Environment pressure specification** parameter is set to **Specified pressure**.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Rotational Mechanical Converter (G)

Topics

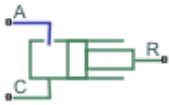
“Modeling Gas Systems”

Introduced in R2016b

Translational Mechanical Converter (IL)

Interface between isothermal liquid and mechanical translational networks

Library: Simscape / Foundation Library / Isothermal Liquid / Elements



Description

The Translational Mechanical Converter (IL) block models an interface between an isothermal liquid network and a mechanical rotational network. The block converts isothermal liquid pressure into mechanical force and vice versa. It can be used as a building block for linear actuators.

The converter contains a variable volume of liquid. If **Model dynamic compressibility** is set to On, then the pressure evolves based on the dynamic compressibility of the liquid volume. The **Mechanical orientation** parameter lets you specify whether an increase in pressure moves port **R** away from or towards port **C**.

Port **A** is the isothermal liquid conserving port associated with the converter inlet. Ports **R** and **C** are the mechanical translational conserving ports associated with the moving interface and converter casing, respectively.

Mass Balance

The mass conservation equations in the mechanical converter volume are

$$\dot{m}_A = \begin{cases} \varepsilon \rho_I S v, & \text{if fluid dynamic compressibility is off} \\ \varepsilon \rho_I S v + \frac{1}{\beta_I} \frac{dp_I}{dt} \rho_I V, & \text{if fluid dynamic compressibility is on} \end{cases}$$

$$v = \frac{dx}{dt}$$

$$V = V_{dead} + \varepsilon S x$$

where:

- \dot{m}_A is the mass flow rate into the converter through port **A**.
- ε is the mechanical orientation of the converter (1 if fluid pressure moves R away from C, -1 if fluid pressure moves R towards C).
- ρ_I is the fluid density inside the converter.
- β_I is the fluid bulk modulus inside the converter.
- S is the cross-sectional area of the converter interface.
- v is the translational velocity of the converter interface.

- x is the displacement of the converter interface.
- V is the liquid volume inside the converter.
- V_{dead} is the dead volume, that is, volume of liquid when the interface displacement is 0.
- p_I is the pressure inside the converter.

Equations used to compute the fluid mixture density and bulk modulus depend on the selected isothermal liquid model. For detailed information, see “Isothermal Liquid Modeling Options”.

Momentum Balance

The momentum conservation equation in the mechanical converter volume is

$$F = \varepsilon(p_{\text{env}} - p)S,$$

where:

- F is the force the liquid exerts on the converter interface.
- p_{env} is the environment pressure outside the converter.

Assumptions and Limitations

- Converter walls are perfectly rigid.
- The converter contains no mechanical hard stops. To include hard stops, use the Translational Hard Stop block.
- The flow resistance between the inlet and the interior of the converter is negligible.
- The kinetic energy of the fluid in the converter is negligible.

Ports

Conserving

A – Converter inlet

isothermal liquid

Isothermal liquid conserving port associated with the converter inlet.

R – Rod

mechanical translational

Mechanical translational conserving port associated with the moving interface.

C – Case

mechanical translational

Mechanical translational conserving port associated with the converter casing.

Parameters

Mechanical orientation – Select the converter orientation

Fluid pressure moves R away from C (default) | Fluid pressure moves R towards C

Select the alignment of moving interface with respect to fluid pressure:

- Fluid pressure moves R away from C — Increase in the fluid pressure results in a positive displacement of port **R** relative to port **C**.
- Fluid pressure moves R towards C — Increase in the fluid pressure results in a negative displacement of port **R** relative to port **C**.

Interface cross-sectional area — The area on which the liquid exerts pressure to generate the translational force

0.01 m² (default) | positive scalar

The area on which the liquid exerts pressure to generate the translational force.

Interface initial displacement — Translational offset of port R relative to port C at the start of simulation

0 m (default) | scalar

Translational offset of port **R** relative to port **C** at the start of simulation. A value of 0 corresponds to an initial liquid volume equal to **Dead volume**.

Dependencies

- If **Mechanical orientation** is Fluid pressure moves R away from C, the parameter value must be greater than or equal to 0.
- If **Mechanical orientation** is Fluid pressure moves R towards C, the parameter value must be less than or equal to 0.

Dead volume — Volume of liquid when the interface displacement is 0

1e-5 m³ (default) | positive scalar

Volume of liquid when the interface displacement is 0.

Environment pressure specification — Select a specification method for the environment pressure

Atmospheric pressure (default) | Specified pressure

Select a specification method for the pressure outside the converter:

- Atmospheric pressure — Use the atmospheric pressure, specified by the Isothermal Liquid Properties (IL) block connected to the circuit.
- Specified pressure — Specify a value by using the **Environment pressure** parameter.

Environment pressure — Pressure outside the converter

0.101325 MPa (default) | positive scalar

Pressure outside the converter acting against the pressure of the converter liquid volume. A value of 0 indicates that the converter expands into vacuum.

Dependencies

Enabled when the **Environment pressure specification** parameter is set to Specified pressure.

Fluid dynamic compressibility — Select whether to model fluid dynamic compressibility

On (default) | Off

Select whether to account for the dynamic compressibility of the liquid. Dynamic compressibility gives the liquid density a dependence on pressure and temperature, impacting the transient response of the system at small time scales.

Initial liquid pressure – Liquid pressure at time zero

0.101325 MPa (default) | positive scalar

Liquid pressure in the converter at the start of simulation.

Dependencies

Enabled when the **Fluid dynamic compressibility** parameter is set to On.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Rotational Mechanical Converter (IL)

Topics

“Modeling Isothermal Liquid Systems”

“Isothermal Liquid Modeling Options”

Introduced in R2020a

Translational Mechanical Converter (MA)

Interface between moist air and mechanical translational networks

Library: Simscape / Foundation Library / Moist Air / Elements



Description

The Translational Mechanical Converter (MA) block models an interface between a moist air network and a mechanical translational network. The block converts moist air pressure into mechanical force and vice versa. You can use it as a building block for linear actuators.

The converter contains a variable volume of moist air. The pressure and temperature evolve based on the compressibility and thermal capacity of this moist air volume. Liquid water condenses out of the moist air volume when it reaches saturation.

If **Mechanical orientation** is set to **Positive**, an increase in the moist air volume inside the converter results in a positive displacement of port **R** relative to port **C**. If **Mechanical orientation** is set to **Negative**, an increase in the moist air volume results in a negative displacement of port **R** relative to port **C**.

The block equations use these symbols. Subscripts **a**, **w**, and **g** indicate the properties of dry air, water vapor, and trace gas, respectively. Subscript **ws** indicates water vapor at saturation. Subscripts **A**, **H**, and **S** indicate the appropriate port. Subscript **I** indicates the properties of the internal moist air volume.

\dot{m}	Mass flow rate
Φ	Energy flow rate
Q	Heat flow rate
p	Pressure
ρ	Density
R	Specific gas constant
V	Volume of moist air inside the converter
c_v	Specific heat at constant volume
h	Specific enthalpy
u	Specific internal energy
x	Mass fraction (x_w is specific humidity, which is another term for water vapor mass fraction)
y	Mole fraction
φ	Relative humidity
r	Humidity ratio

T	Temperature
t	Time

The net flow rates into the moist air volume inside the converter are

$$\begin{aligned}\dot{m}_{net} &= \dot{m}_A - \dot{m}_{condense} + \dot{m}_{wS} + \dot{m}_{gS} \\ \Phi_{net} &= \Phi_A + Q_H - \Phi_{condense} + \Phi_S \\ \dot{m}_{w,net} &= \dot{m}_{wA} - \dot{m}_{condense} + \dot{m}_{wS} \\ \dot{m}_{g,net} &= \dot{m}_{gA} + \dot{m}_{gS}\end{aligned}$$

where:

- $\dot{m}_{condense}$ is the rate of condensation.
- $\Phi_{condense}$ is the rate of energy loss from the condensed water.
- Φ_S is the rate of energy added by the sources of moisture and trace gas. \dot{m}_{wS} and \dot{m}_{gS} are the mass flow rates of water and gas, respectively, through port **S**. The values of \dot{m}_{wS} , \dot{m}_{gS} , and Φ_S are determined by the moisture and trace gas sources connected to port **S** of the converter.

Water vapor mass conservation relates the water vapor mass flow rate to the dynamics of the moisture level in the internal moist air volume:

$$\frac{dx_{wI}}{dt} \rho_I V + x_{wI} \dot{m}_{net} = \dot{m}_{w,net}$$

Similarly, trace gas mass conservation relates the trace gas mass flow rate to the dynamics of the trace gas level in the internal moist air volume:

$$\frac{dx_{gI}}{dt} \rho_I V + x_{gI} \dot{m}_{net} = \dot{m}_{g,net}$$

Mixture mass conservation relates the mixture mass flow rate to the dynamics of the pressure, temperature, and mass fractions of the internal moist air volume:

$$\left(\frac{1}{p_I} \frac{dp_I}{dt} - \frac{1}{T_I} \frac{dT_I}{dt} \right) \rho_I V + \frac{R_a - R_w}{R_I} (\dot{m}_{w,net} - x_w \dot{m}_{net}) + \frac{R_a - R_g}{R_I} (\dot{m}_{g,net} - x_g \dot{m}_{net}) + \rho_I \dot{V} = \dot{m}_{net}$$

where \dot{V} is the rate of change of the converter volume.

Finally, energy conservation relates the energy flow rate to the dynamics of the pressure, temperature, and mass fractions of the internal moist air volume:

$$\rho_I c_{vI} V \frac{dT_I}{dt} + (u_{wI} - u_{aI}) (\dot{m}_{w,net} - x_w \dot{m}_{net}) + (u_{gI} - u_{aI}) (\dot{m}_{g,net} - x_g \dot{m}_{net}) + u_I \dot{m}_{net} = \Phi_{net} - p_I \dot{V}$$

The equation of state relates the mixture density to the pressure and temperature:

$$p_I = \rho_I R_I T_I$$

The mixture specific gas constant is

$$R_I = x_{aI} R_a + x_{wI} R_w + x_{gI} R_g$$

The converter volume is

$$V = V_{dead} + S_{int}d_{int}\varepsilon_{int}$$

where:

- V_{dead} is the dead volume.
- S_{int} is the interface cross-sectional area.
- d_{int} is the interface displacement.
- ε_{int} is the mechanical orientation coefficient. If **Mechanical orientation is Positive**, $\varepsilon_{int} = 1$. If **Mechanical orientation is Negative**, $\varepsilon_{int} = -1$.

The force balance on the mechanical interface is

$$F_{int} = (p_{env} - p_I)S_{int}\varepsilon_{int}$$

where:

- F_{int} is the force from port **R** to port **C**.
- p_{env} is the environment pressure.

Flow resistance and thermal resistance are not modeled in the converter:

$$\begin{aligned} p_A &= p_I \\ T_H &= T_I \end{aligned}$$

When the moist air volume reaches saturation, condensation may occur. The specific humidity at saturation is

$$x_{wsI} = \varphi_{ws} \frac{R_I p_{wsI}}{R_w p_I}$$

where:

- φ_{ws} is the relative humidity at saturation (typically 1).
- p_{wsI} is the water vapor saturation pressure evaluated at T_I .

The rate of condensation is

$$\dot{m}_{condense} = \begin{cases} 0, & \text{if } x_{wI} \leq x_{wsI} \\ \frac{x_{wI} - x_{wsI}}{\tau_{condense}} \rho_I V, & \text{if } x_{wI} > x_{wsI} \end{cases}$$

where $\tau_{condense}$ is the value of the **Condensation time constant** parameter.

The condensed water is subtracted from the moist air volume, as shown in the conservation equations. The energy associated with the condensed water is

$$\Phi_{condense} = \dot{m}_{condense}(h_{wI} - \Delta h_{vapI})$$

where Δh_{vapI} is the specific enthalpy of vaporization evaluated at T_I .

Other moisture and trace gas quantities are related to each other as follows:

$$\phi_{wI} = \frac{y_{wI} p_I}{p_{wsI}}$$

$$y_{wI} = \frac{x_{wI} R_w}{R_I}$$

$$r_{wI} = \frac{x_{wI}}{1 - x_{wI}}$$

$$y_{gI} = \frac{x_{gI} R_g}{R_I}$$

$$x_{aI} + x_{wI} + x_{gI} = 1$$

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables” and “Initial Conditions for Blocks with Finite Moist Air Volume”.

Assumptions and Limitations

- The converter casing is perfectly rigid.
- Flow resistance between the converter inlet and the moist air volume is not modeled. Connect a Local Restriction (MA) block or a Flow Resistance (MA) block to port **A** to model pressure losses associated with the inlet.
- Thermal resistance between port **H** and the moist air volume is not modeled. Use Thermal library blocks to model thermal resistances between the moist air mixture and the environment, including any thermal effects of a chamber wall.
- The moving interface is perfectly sealed.
- The block does not model the mechanical effects of the moving interface, such as hard stops, friction, and inertia.

Ports

Output

W — Rate of condensation measurement, kg/s

physical signal

Physical signal output port that measures the rate of condensation in the converter.

F — Vector physical signal containing pressure, temperature, humidity, and trace gas levels data

physical signal vector

Physical signal output port that outputs a vector signal. The vector contains the pressure (in Pa), temperature (in K), moisture level, and trace gas level measurements inside the component. Use the Measurement Selector (MA) block to unpack this vector signal.

Conserving

A — Converter inlet

moist air

Moist air conserving port associated with the converter inlet.

H — Temperature inside converter

thermal

Thermal conserving port associated with the temperature of the moist air mixture inside the converter.

R — Rod

mechanical translational

Mechanical translational conserving port associated with the moving interface.

C — Case

mechanical translational

Mechanical translational conserving port associated with the converter casing.

S — Inject or extract moisture and trace gas

moist air source

Connect this port to port **S** of a block from the Moisture & Trace Gas Sources library to add or remove moisture and trace gas. For more information, see “Using Moisture and Trace Gas Sources”.

Dependencies

This port is visible only if you set the **Moisture and trace gas source** parameter to **Controlled**.

Parameters

Main

Mechanical orientation — Select the converter orientation

Positive (default) | Negative

Select the relative orientation of the converter with respect to the volume of moist air inside the converter:

- **Positive** — Increase in the moist air volume results in a positive displacement of port **R** relative to port **C**.
- **Negative** — Increase in the moist air volume results in a negative displacement of port **R** relative to port **C**.

Interface cross-sectional area — The area on which the moist air exerts pressure to generate the translational force

0.01 m² (default)

The area on which the moist air exerts pressure to generate the translational force.

Interface initial displacement — Translational offset of port R relative to port C at the start of simulation

0 m (default)

Translational offset of port **R** relative to port **C** at the start of simulation. A value of 0 corresponds to an initial moist air volume equal to **Dead volume**.

Dependencies

- If **Mechanical orientation** is Positive, the parameter value must be greater than or equal to 0.
- If **Mechanical orientation** is Negative, the parameter value must be less than or equal to 0.

Dead volume — Volume of moist air when the interface displacement is 0

1e-5 m³ (default)

Volume of moist air when the interface displacement is 0.

Cross-sectional area at port A — Area normal to flow path at the converter inlet

0.01 m² (default)

Cross-sectional area of the converter inlet, in the direction normal to the air flow path.

Environment pressure specification — Select a specification method for the environment pressure

Atmospheric pressure (default) | Specified pressure

Select a specification method for the environment pressure:

- **Atmospheric pressure** — Use the atmospheric pressure, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified pressure** — Specify a value by using the **Environment pressure** parameter.

Environment pressure — Pressure outside the converter

0.101325 MPa (default)

Pressure outside the converter acting against the pressure of the converter moist air volume. A value of 0 indicates that the converter expands into vacuum.

Dependencies

Enabled when the **Environment pressure specification** parameter is set to Specified pressure.

Moisture and Trace Gas**Relative humidity at saturation — Relative humidity above which condensation occurs**

1 (default)

Relative humidity above which condensation occurs.

Condensation time constant — Time scale for condensation

1e-3 s (default)

Characteristic time scale at which an oversaturated moist air volume returns to saturation by condensing out excess moisture.

Moisture and trace gas source — Model moisture and trace gas levels

None (default) | Constant | Controlled

This parameter controls visibility of port **S** and provides these options for modeling moisture and trace gas levels inside the component:

- **None** — No moisture or trace gas is injected into or extracted from the block. Port **S** is hidden. This is the default.
- **Constant** — Moisture and trace gas are injected into or extracted from the block at a constant rate. The same parameters as in the Moisture Source (MA) and Trace Gas Source (MA) blocks become available in the **Moisture and Trace Gas** section of the block interface. Port **S** is hidden.
- **Controlled** — Moisture and trace gas are injected into or extracted from the block at a time-varying rate. Port **S** is exposed. Connect the Controlled Moisture Source (MA) and Controlled Trace Gas Source (MA) blocks to this port.

Moisture added or removed — Select whether the block adds or removes moisture as water vapor or liquid water

Vapor (default) | Liquid

Select whether the block adds or removes moisture as water vapor or liquid water:

- **Vapor** — The enthalpy of the added or removed moisture corresponds to the enthalpy of water vapor, which is greater than that of liquid water.
- **Liquid** — The enthalpy of the added or removed moisture corresponds to the enthalpy of liquid water, which is less than that of water vapor.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Rate of added moisture — Constant mass flow rate through the block

0 kg/s (default)

Water vapor mass flow rate through the block. A positive value adds moisture to the connected moist air volume. A negative value extracts moisture from that volume.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Added moisture temperature specification — Select specification method for the temperature of added moisture

Atmospheric temperature (default) | Specified temperature

Select a specification method for the moisture temperature:

- **Atmospheric temperature** — Use the atmospheric temperature, specified by the Moist Air Properties (MA) block connected to the circuit.
- **Specified temperature** — Specify a value by using the **Temperature of added moisture** parameter.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Temperature of added moisture – Moisture temperature

293.15 K (default)

Enter the desired temperature of added moisture. This temperature remains constant during simulation. The block uses this value to evaluate the specific enthalpy of the added moisture only. The specific enthalpy of removed moisture is based on the temperature of the connected moist air volume.

Dependencies

Enabled when the **Added moisture temperature specification** parameter is set to Specified temperature.

Rate of added trace gas – Constant mass flow rate through the block

0 kg/s (default)

Trace gas mass flow rate through the block. A positive value adds trace gas to the connected moist air volume. A negative value extracts trace gas from that volume.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Added trace gas temperature specification – Select specification method for the temperature of added trace gas

Atmospheric temperature (default) | Specified temperature

Select a specification method for the trace gas temperature:

- Atmospheric temperature — Use the atmospheric temperature, specified by the Moist Air Properties (MA) block connected to the circuit.
- Specified temperature — Specify a value by using the **Temperature of added trace gas** parameter.

Dependencies

Enabled when the **Moisture and trace gas source** parameter is set to Constant.

Temperature of added trace gas – Trace gas temperature

293.15 K (default)

Enter the desired temperature of added trace gas. This temperature remains constant during simulation. The block uses this value to evaluate the specific enthalpy of the added trace gas only. The specific enthalpy of removed trace gas is based on the temperature of the connected moist air volume.

Dependencies

Enabled when the **Added trace gas temperature specification** parameter is set to Specified temperature.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Rotational Mechanical Converter (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Translational Mechanical Converter (TL)

Interface between thermal liquid and mechanical translational networks

Library: Simscape / Foundation Library / Thermal Liquid / Elements



Description

The Translational Mechanical Converter (TL) block models an interface between a thermal liquid network and a mechanical rotational network. The block converts thermal liquid pressure into mechanical force and vice versa. It can be used as a building block for linear actuators.

The converter contains a variable volume of liquid. The temperature evolves based on the thermal capacity of this volume. If **Model dynamic compressibility** is set to **On**, then the pressure also evolves based on the dynamic compressibility of the liquid volume. If **Mechanical orientation** is set to **Positive**, then an increase in the liquid volume results in a positive displacement of port **R** relative to port **C**. If **Mechanical orientation** is set to **Negative**, then an increase in the liquid volume results in a negative displacement of port **R** relative to port **C**.

Port **A** is the thermal liquid conserving port associated with the converter inlet. Port **H** is the thermal conserving port associated with the temperature of the liquid inside the converter. Ports **R** and **C** are the mechanical translational conserving ports associated with the moving interface and converter casing, respectively.

Mass Balance

The mass conservation equation in the mechanical converter volume is

$$\dot{m}_A = \varepsilon \rho S v + \begin{cases} 0, & \text{if fluid dynamic compressibility is off} \\ V\rho\left(\frac{1}{\beta}\frac{dp}{dt} + \alpha\frac{dT}{dt}\right), & \text{if fluid dynamic compressibility is on} \end{cases}$$

where:

- \dot{m}_A is the liquid mass flow rate into the converter through port A.
- ε is the mechanical orientation of the converter (1 if positive, -1 if negative).
- ρ is the liquid mass density.
- S is the cross-sectional area of the converter interface.
- v is the translational velocity of the converter interface.
- V is the liquid volume inside the converter.
- β is the liquid bulk modulus inside the converter.
- α is the coefficient of thermal expansion of the liquid.
- p is the liquid pressure inside the converter.
- T is the liquid temperature inside the converter.

Momentum Balance

The momentum conservation equation in the mechanical converter volume is

$$F = -\varepsilon(p - p_{\text{Atm}})S$$

where:

- F is the force the liquid exerts on the converter interface.
- p_{Atm} is the atmospheric pressure.

Energy Balance

The energy conservation equation in the mechanical converter volume is

$$\frac{d(\rho u V)}{dt} = \phi_A + Q_H - pS\varepsilon v,$$

where:

- u is the liquid internal energy.
- ϕ_A is the total energy flow rate into the mechanical converter volume through port A.
- Q_H is the heat flow rate into the mechanical converter volume.

Assumptions and Limitations

- Converter walls are not compliant. They cannot deform regardless of internal pressure and temperature.
- The converter contains no mechanical hard stops. To include hard stops, use the Translational Hard Stop block.
- The flow resistance between the inlet and the interior of the converter is negligible.
- The thermal resistance between the thermal port and the interior of the converter is negligible.
- The kinetic energy of the fluid in the converter is negligible.

Ports

Conserving

A – Converter inlet

thermal liquid

Thermal liquid conserving port associated with the converter inlet.

H – Temperature inside converter

thermal

Thermal conserving port associated with the temperature of the liquid inside the converter.

R – Rod

mechanical translational

Mechanical translational conserving port associated with the moving interface.

C – Case

mechanical translational

Mechanical translational conserving port associated with the converter casing.

Parameters**Main****Mechanical orientation – Select the converter orientation**

Positive (default) | Negative

Select the alignment of moving interface with respect to the converter liquid volume:

- **Positive** — Increase in the liquid volume results in a positive displacement of port **R** relative to port **C**.
- **Negative** — Increase in the liquid volume results in a negative displacement of port **R** relative to port **C**.

Interface cross-sectional area – The area on which the liquid exerts pressure to generate the translational force0.01 m² (default)

The area on which the liquid exerts pressure to generate the translational force.

Interface initial displacement – Translational offset of port R relative to port C at the start of simulation

0 m (default)

Translational offset of port **R** relative to port **C** at the start of simulation. A value of 0 corresponds to an initial liquid volume equal to **Dead volume**.

Dependencies

- If **Mechanical orientation** is **Positive**, the parameter value must be greater than or equal to 0.
- If **Mechanical orientation** is **Negative**, the parameter value must be less than or equal to 0.

Dead volume – Volume of liquid when the interface displacement is 01e-5 m³ (default)

Volume of liquid when the interface displacement is 0.

Environment pressure specification – Select a specification method for the environment pressure

Atmospheric pressure (default) | Specified pressure

Select a specification method for the pressure outside the converter:

- **Atmospheric pressure** — Use the atmospheric pressure, specified by the Thermal Liquid Settings (TL) or Thermal Liquid Properties (TL) block connected to the circuit.
- **Specified pressure** — Specify a value by using the **Environment pressure** parameter.

Environment pressure – Pressure outside the converter

0.101325 MPa (default)

Pressure outside the converter acting against the pressure of the converter liquid volume. A value of 0 indicates that the converter expands into vacuum.

Dependencies

Enabled when the **Environment pressure specification** parameter is set to Specified pressure.

Effects and Initial Conditions

Fluid dynamic compressibility – Select whether to model fluid dynamic compressibility
On (default) | Off

Select whether to account for the dynamic compressibility of the liquid. Dynamic compressibility gives the liquid density a dependence on pressure and temperature, impacting the transient response of the system at small time scales.

Initial liquid pressure – Liquid pressure at time zero
0.101325 MPa (default)

Liquid pressure in the converter at the start of simulation.

Dependencies

Enabled when the **Fluid dynamic compressibility** parameter is set to On.

Initial liquid temperature – Liquid temperature at time zero
293.15 K (default)

Liquid temperature in the converter at the start of simulation.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Rotational Mechanical Converter (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2013b

Translational Spring

Ideal spring in mechanical translational systems



Library

Mechanical Translational Elements

Description

The Translational Spring block represents an ideal mechanical linear spring, described with the following equations:

$$F = Kx$$

$$x = x_{init} + x_R - x_C$$

$$v = \frac{dx}{dt}$$

where

F	Force transmitted through the spring
K	Spring rate
x	Relative displacement (spring deformation)
x_{init}	Spring initial displacement (initial deformation); the spring can be initially compressed ($x_{init} > 0$) or stretched ($x_{init} < 0$)
x_R, x_C	Absolute displacements of terminals R and C, respectively
v	Relative velocity
t	Time

The block positive direction is from port R to port C. This means that the force is positive if it acts in the direction from R to C.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Spring rate

Spring rate. The default value is 1000 N/m.

Ports

The block has the following ports:

R

Mechanical translational conserving port.

C

Mechanical translational conserving port.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

[Translational Damper](#) | [Translational Friction](#) | [Translational Hard Stop](#)

Introduced in R2007a

Two-Phase Fluid Properties (2P)

Fluid properties for two-phase fluid network



Library

Two-Phase Fluid/Utilities

Description

The Two-Phase Fluid Properties (2P) block sets the thermophysical properties of a fluid in a two-phase fluid network. These properties, which include density, viscosity, and specific heat, among others, can extend into the supercritical region of the fluid (water, by default, with a supercritical region extending up to 100 MPa in pressure).

The properties apply to the whole of the two-phase fluid network (the group of continuously connected two-phase fluid blocks). A network can have only one working fluid, and therefore only one instance of this block. If the network is modeled without this block, the fluid is set to water and its properties are obtained from the domain definition.

The block parameterizes the fluid properties in terms of pressure and normalized internal energy—a linear transformation of the specific internal energy. In a subcooled liquid, the normalized internal energy definition is

$$\bar{u} = \frac{u - u_{min}}{u_{sat}^L(p) - u_{min}} - 1, \quad u_{min} \leq u < u_{sat}^L(p),$$

where:

- \bar{u} is the normalized internal energy of the fluid.
- u is the specific internal energy of the fluid.
- u_{min} is the lowest specific internal energy allowed in the two-phase fluid network.
- u_{sat}^L is the specific internal energy of the liquid phase at saturation.

In a superheated vapor, the normalized internal energy definition is

$$\bar{u} = \frac{u - u_{max}}{u_{max} - u_{sat}^V(p)} + 2, \quad u_{sat}^V(p) < u \leq u_{max},$$

where:

- u_{max} is the highest specific internal energy allowed in the two-phase fluid network.
- u_{sat}^V is the specific internal energy of the vapor phase at saturation.

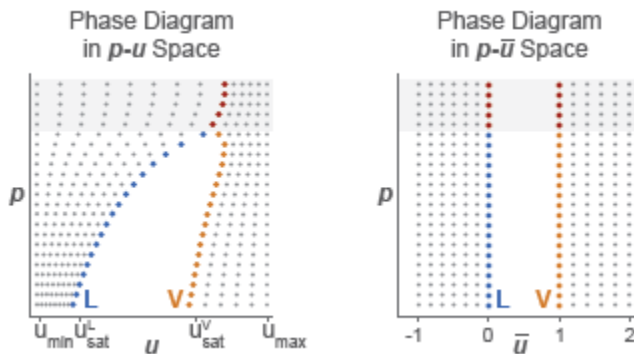
In a two-phase mixture, the normalized internal energy definition is

$$\bar{u} = \frac{u - u_{sat}^L(p)}{u_{sat}^V(p) - u_{sat}^L(p)}, \quad u_{sat}^L(p) \leq u \leq u_{sat}^V(p).$$

These expressions correspond to a normalized internal energy that is at all pressures -1 at the minimum valid specific internal energy, 0 at the liquid saturation boundary, +1 at the vapor saturation boundary, and +2 at the maximum valid specific internal energy.

In a two-phase mixture, the normalized internal energy ranges in value from 0 to 1 and is therefore equivalent to vapor quality—the mass fraction of the vapor phase in a two-phase mixture. In subcooled liquid and superheated vapor, the normalized internal energy behaves as an extension of vapor quality.

The normalized internal energy provides an advantage over the specific internal energy. It transforms the p - u phase diagram so that the subcooled liquid and superheated vapor phases occupy distinct rectangular regions. This transformation, shown in the figure, enables you to specify the fluid properties on separate rectangular p - \bar{u} grids, one for each phase.



A pressure vector, of length N , and two normalized internal energy vectors, of lengths M_L and M_V , provide the (p, \bar{u}) coordinates of the two grids. The pressure vector is common to both grids. The subcooled liquid grid is M_L -by- N in size and the superheated vapor grid M_V -by- N .

Two-way lookup tables provide the fluid property values on the (p, \bar{u}) grids. The table rows correspond to different normalized internal energies and the table columns to pressures. Fluid properties in the p - \bar{u} continuum are computed using linear interpolation between the p - \bar{u} data points.

	p_1	p_2	...	p_N
\bar{u}_1	$f(\bar{u}_1, p_1)$	$f(\bar{u}_1, p_2)$...	$f(\bar{u}_1, p_N)$
\bar{u}_2	$f(\bar{u}_2, p_1)$	$f(\bar{u}_2, p_2)$...	$f(\bar{u}_2, p_N)$
...
\bar{u}_M	$f(\bar{u}_M, p_1)$	$f(\bar{u}_M, p_2)$...	$f(\bar{u}_M, p_N)$

Two-Way Property Lookup Table

Saturated specific internal energy vectors provide the phase boundaries in the (p, \bar{u}) phase diagram. These separate the different regions of the phase diagram—subcooled liquid, two-phase mixture, and superheated vapor.

Along with the minimum and maximum valid specific internal energy values, the saturated specific internal energy vectors enable the Two-Phase Fluid blocks to convert the normalized internal energies specified in this block into the specific internal energies they use for calculation purposes.

Supercritical Fluids

The fluid can become supercritical. Liquid and vapor then cease to exist as separate phases. Their saturation boundaries merge, forming what is known as the pseudocritical line (in the figure, the segment extending upward from the hump of the p - u phase diagram).

The pseudocritical line divides the fluid into liquid-like and vapor-like regions. As the phases are no longer distinct, however, the transition between the regions is gradual rather than abrupt. This means that at the pseudocritical line, the liquid-like and vapor-like fluids must share the same physical properties.

Property Tables

In this block, the liquid-like portion of the supercritical region is conceived as an extension of the liquid phase. Likewise, the vapor-like portion is conceived as an extension of the vapor phase. In other words, the properties of the supercritical fluid are divided into the liquid and vapor property tables of the block.

The distinction between the phases is, however, superfluous in the supercritical region. For breakpoints over the critical pressure, the liquid and vapor property tables must agree at the saturated boundaries (or, more properly, at the pseudocritical line). The supercritical portions of the tables then behave as a continuous whole.

Consider the kinematic viscosity tables as an example. The bottom row of the **Liquid kinematic viscosity table** parameter, as it corresponds to a normalized specific internal energy of 1, gives the data for the saturated liquid. The elements in that row corresponding to pressures at or above the critical point give the data over the pseudocritical line.

Likewise, the upper row of the **Vapor kinematic viscosity table** parameter, as it corresponds to a normalized specific internal energy of 1, gives the data for the saturated vapor. The elements in that row corresponding to pressures at or above the critical point therefore give the data over the pseudocritical line. To be valid, this data must be exactly the same as that given over the pseudocritical line in the liquid table.

Pseudocritical Line

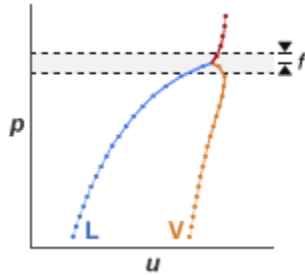
The pseudocritical line is encoded in the saturation boundaries of the liquid and vapor phases. These are specified in the block via the **Saturated liquid specific internal energy vector** and **Saturated vapor internal energy vector** parameters. At pressures below the critical point, the values in one vector will generally differ from those in the other; above the critical point, however, one must always equal the other.

The pseudocritical line can assume any shape. For a simple approximation, it often suffices to extend the specific internal energy of the critical point upward along the pressure axis. This means setting the supercritical portions of the saturated specific internal energy vectors to that critical value. If the exact shape of the pseudocritical line is known, the specific internal energies along that line can be used instead.

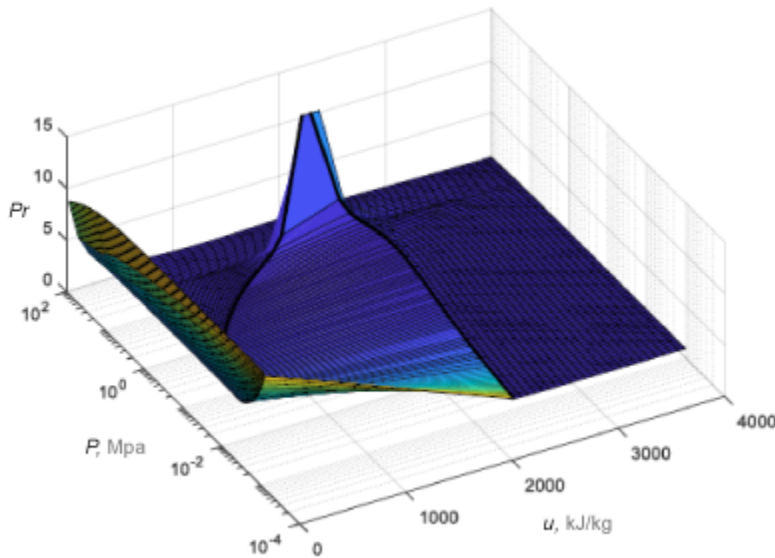
Thermal Transport Properties

The data for the thermal transport properties—the specific heat, thermal conductivity, and Prandtl number—often shows a large peak near the critical point. The scale and size of the peak can, in some

cases, cause numerical problems during simulation. To avoid such problems, the block allows the peak to be clipped. This option is specified in the **Thermal transport properties near critical point** block parameter (by setting it to `Clip peak values`).



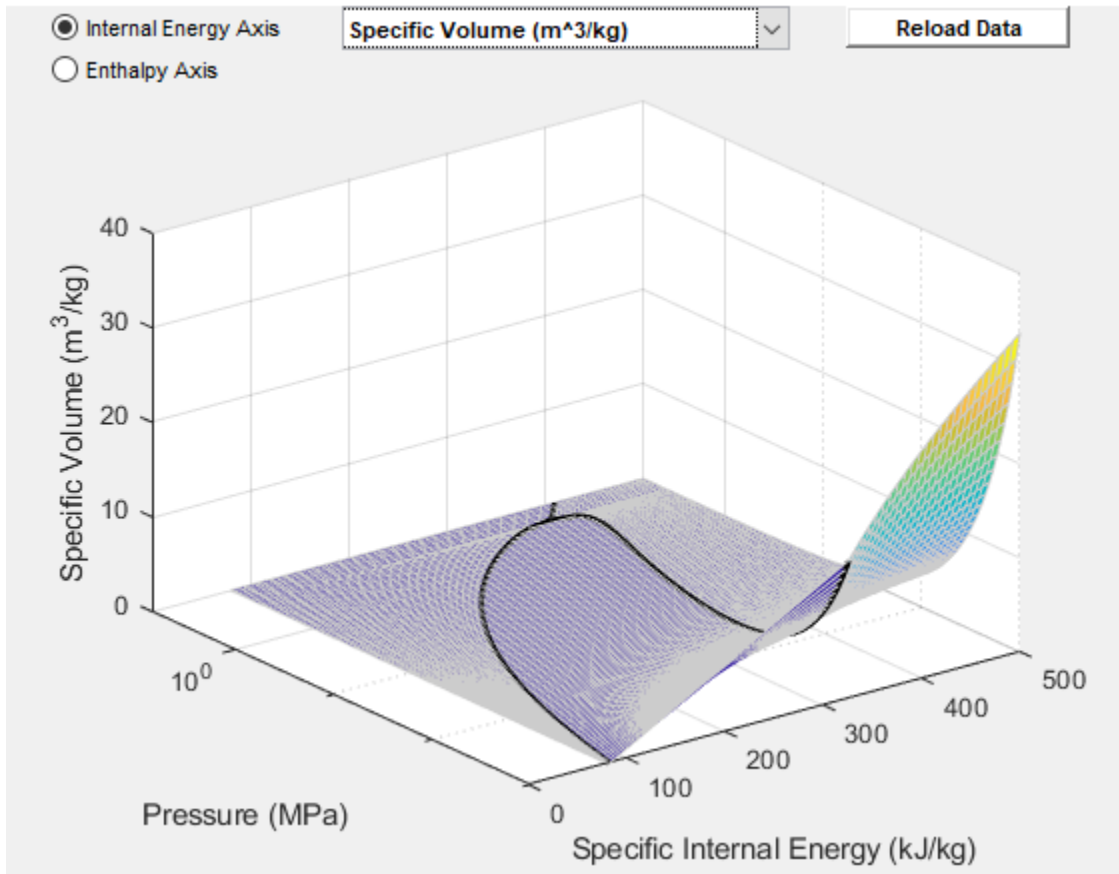
The clipping is limited to a small pressure range around the critical point. That range is specified as a fraction of the critical pressure (f in the figure), and it extends in both directions (by the same amount) from the critical pressure. Within that range, the thermal properties are each limited to a maximum given by the greater of its two end values. The figure shows a plot of the Prandtl number for water with clipping.



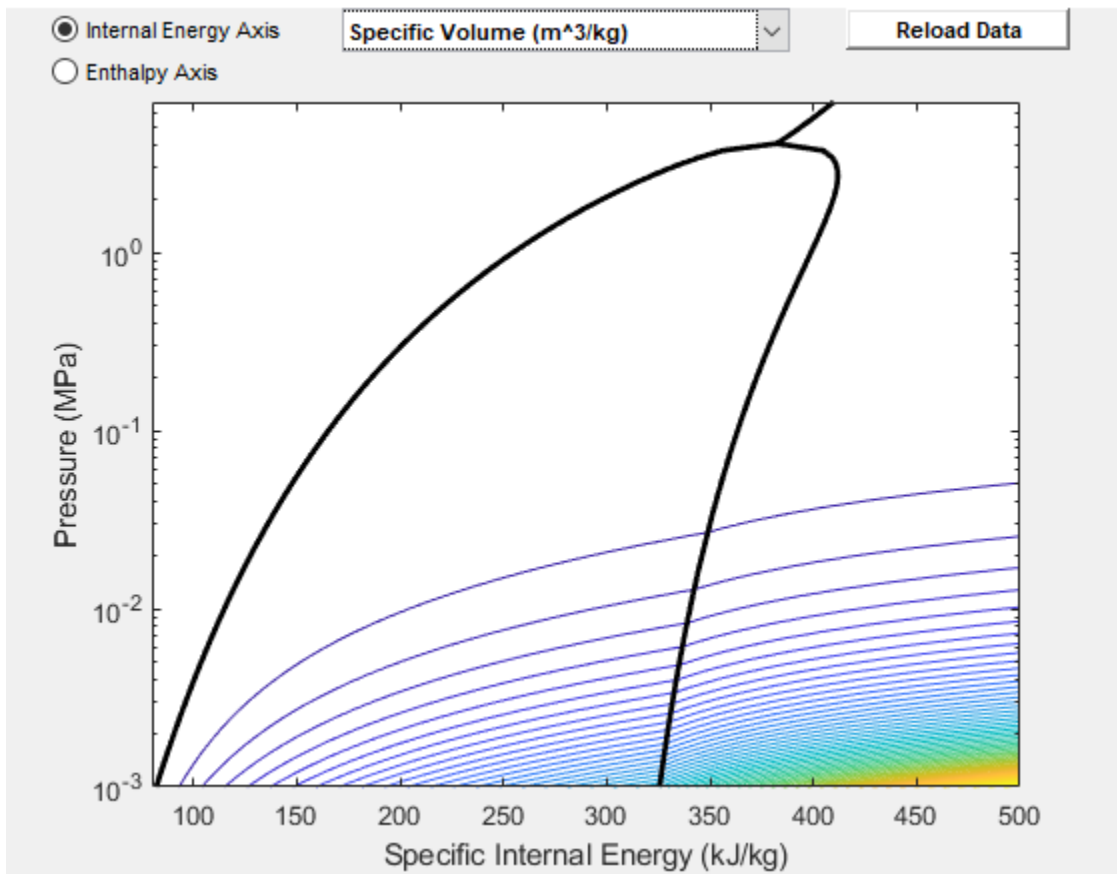
Data Visualization

The block lets you plot the specified two-phase fluid properties as a function of pressure and specific internal energy. Plotting the properties lets you visualize the data before simulating the model.

To plot the data, right-click a Two-Phase Fluid Properties (2P) block in your model and, from the context menu, select **Foundation Library > Plot Fluid Properties (3D)** or **Foundation Library > Plot Fluid Properties (Contours)**. Use the drop-down list located at the top of the plot to select the property to visualize. Click the **Reload** button to regenerate a plot following a block parameter update.



Fluid Properties (3D) Plot



Fluid Properties (Contours) Plot

Parameters

Parameters Tab

Minimum valid specific internal energy

Lowest specific internal energy allowed in the two-phase fluid network. The default value is 0 kJ/kg.

Maximum valid specific internal energy

Highest specific internal energy allowed in the two-phase fluid network. The default value is 4000 kJ/kg.

Pressure vector

Vector of length N containing the pressure values corresponding to the columns of the fluid property tables. The default vector is a logarithmically spaced 100-element vector ranging from $1e-3$ to 100 MPa (spanning both subcritical and supercritical regions of the fluid).

Critical pressure

Pressure at which liquid and vapor cease to exist as distinct phases. Set this parameter to `inf` to model a fluid that is always subcritical.

Thermal transport properties near critical point

Select whether to trim the peaks of the thermal transport properties near the critical point. (The peaks are often sharp and large, which may lead to numerical problems during simulation.) The width of the clipping region is specified in the **Fraction above and below critical pressure for clipping** block parameter (exposed when `Clip peak values` is selected here). Set this parameter to `Do not clip peak values` to leave the thermal transport properties as is.

Fraction above and below critical pressure for clipping

Pressure range above and below the critical pressure, expressed as a fraction of the same, over which to clip the peaks of the thermal transport properties. This parameter is exposed when the **Thermal transport properties near critical point** parameter is set to `Clip peak values`.

Atmospheric pressure

Absolute pressure of the two-phase fluid system surroundings. The default value, 0.101325 Pa, is the atmospheric pressure at mean sea level.

Dynamic pressure threshold for flow reversal

Dynamic pressure at which the flow at a port begins to reverse in direction. Simscape. The dynamic pressure is the difference between the total pressure and the static pressure. Treat this parameter as a means to smooth the flow reversal. Larger values correspond to smoother reversals and smaller values to sharper reversals. The default value is 0.01 Pa.

The Two-Phase Fluid domain uses an upwind scheme that derives the flow rate at a port from its value just upwind of the port. During flow reversals, the source of the flow rate value changes abruptly and the flow rate can become discontinuous. To prevent discontinuities and improve simulation robustness, flow reversals are smoothed out, with the dynamic pressure threshold marking the beginning of the smoothed transitions.

Liquid Properties Tab**Normalized liquid internal energy vector**

Vector of length M_L containing the normalized internal energy values corresponding to the rows of the liquid property tables. The vector must begin at -1 and end at 0 (the saturated liquid state). The default is a uniformly spaced 25-element vector.

Liquid specific volume table

$M_L \times N$ matrix containing the liquid specific volume values corresponding to the normalized liquid internal energy and pressure vectors. Elements at the saturation boundary and at or above the critical pressure must equal their counterparts in the **Vapor specific volume table** parameter. The default matrix is a 25×100 table for water.

Liquid specific entropy table

$M_L \times N$ matrix containing the liquid specific entropy values corresponding to the normalized liquid internal energy and pressure vectors. Elements at the saturation boundary and at or above the critical pressure must equal their counterparts in the **Vapor specific entropy table** parameter. The default matrix is a 25×100 table for water.

Liquid temperature table

$M_L \times N$ matrix containing the liquid temperature values corresponding to the normalized liquid internal energy and pressure vectors. Elements at the saturation boundary and at or above the critical pressure must equal their counterparts in the **Vapor temperature table** parameter. The default matrix is a 25×100 table for water.

Liquid kinematic viscosity table

$M_L \times N$ matrix containing the liquid kinematic viscosity values corresponding to the normalized liquid internal energy and pressure vectors. Elements at the saturation boundary and at or above

the critical pressure must equal their counterparts in the **Vapor kinematic viscosity table** parameter. The default matrix is a 25×100 table for water.

Liquid thermal conductivity table

$M_L \times N$ matrix containing the liquid thermal conductivity values corresponding to the normalized liquid internal energy and pressure vectors. Elements at the saturation boundary and at or above the critical pressure must equal their counterparts in the **Vapor thermal conductivity table** parameter. The default matrix is a 25×100 table for water.

Liquid Prandtl number table

$M_L \times N$ matrix containing the liquid Prandtl number values corresponding to the normalized liquid internal energy and pressure vectors. Elements at the saturation boundary and at or above the critical pressure must equal their counterparts in the **Vapor Prandtl number table** parameter. The default matrix is a 25×100 table for water.

Saturated liquid specific internal energy vector

Vector of length N containing the saturated liquid specific internal energy values corresponding to the pressure vector. Elements at or above the critical pressure must equal their counterparts in the **Saturated vapor specific internal energy vector** block parameter. The default is a 100-element vector for water.

Vapor Properties Tab

Normalized vapor internal energy vector

Vector of length M_V containing the normalized internal energy values corresponding to the rows of the vapor property tables. The vector must begin at 1 (the saturated vapor state) and end at 2. The default is a uniformly spaced 25-element vector.

Vapor specific volume table

$M_V \times N$ matrix containing the vapor specific volume values corresponding to the normalized vapor internal energy and pressure vectors. Elements at the saturation boundary and at or above the critical pressure must equal their counterparts in the **Liquid specific volume table** parameter. The default matrix is a 25×100 table for water.

Vapor specific entropy table

$M_V \times N$ matrix containing the vapor specific entropy values corresponding to the normalized vapor internal energy and pressure vectors. Elements at the saturation boundary and at or above the critical pressure must equal their counterparts in the **Liquid specific entropy table** parameter. The default matrix is a 25×100 table for water.

Vapor temperature table

$M_V \times N$ matrix containing the vapor temperature values corresponding to the normalized vapor internal energy and pressure vectors. Elements at the saturation boundary and at or above the critical pressure must equal their counterparts in the **Liquid temperature table** parameter. The default matrix is a 25×100 table for water.

Vapor kinematic viscosity table

$M_V \times N$ matrix containing the vapor kinematic viscosity values corresponding to the normalized vapor internal energy and pressure vectors. Elements at the saturation boundary and at or above the critical pressure must equal their counterparts in the **Liquid kinematic viscosity table** parameter. The default matrix is a 25×100 table for water.

Vapor thermal conductivity table

$M_V \times N$ matrix containing the vapor thermal conductivity values corresponding to the normalized vapor internal energy and pressure vectors. Elements at the saturation boundary and at or above

the critical pressure must equal their counterparts in the **Liquid thermal conductivity table** parameter. The default matrix is a 25×100 table for water.

Vapor Prandtl number table

$M_V \times N$ matrix containing the vapor Prandtl number values corresponding to the normalized vapor internal energy and pressure vectors. Elements at the saturation boundary and at or above the critical pressure must equal their counterparts in the **Liquid Prandtl number table** parameter. The default matrix is a 25×100 table for water.

Saturated vapor specific internal energy vector

Vector of length N containing the saturated vapor specific internal energy values corresponding to the pressure vector. Elements at or above the critical pressure must equal their counterparts in the **Saturated liquid specific internal energy vector** block parameter. The default is a 100-element vector for water.

Ports

The block has a two-phase fluid conserving port. This port identifies the two-phase fluid network whose fluid properties the block provides.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

twoPhaseFluidTables

Topics

“Manually Generate Fluid Property Tables”

Introduced in R2015b

Vapor Quality Sensor (2P)

Measure vapor fraction in two-phase fluids



Library

Two-Phase Fluid/Sensors

Description

The Vapor Quality Sensor (2P) block measures the fraction of vapor in a two-phase fluid. That fraction can be expressed on a mass basis (the vapor quality of the fluid) or on a volume basis (the vapor void fraction). The choice of basis is given in the **Vapor fraction specification** block parameter.

The measurement is made at port **A** and reported, as a physical signal, at port **X**. Its value can range from 0 in pure liquid to 1 in pure vapor. Intermediate values correspond to two-phase mixtures. The sensor is ideal—neither mass nor energy exchanges take place, and the measurement is therefore undisturbed by them.

If pressure should rise above the critical point for the fluid, vapor and liquid cease to exist, and the measurement becomes undefined. The block then outputs -1, a dummy value that serves merely to indicate a critical or supercritical state.

Parameters

Vapor fraction specification

Type of measurement to report at port **X**. Select **Vapor quality** for the mass fraction of vapor in a two-phase fluid. Select **Vapor void fraction** for the volume fraction instead. The measurement can range from 0 to 1 in both cases (save for critical and supercritical fluids, for which the output is always -1).

Ports

The block has two ports:

- **A** — Two-phase fluid node at which to measure the proportion of vapor in a two-phase fluid mixture.
- **X** — Proportion of vapor expressed as a mass fraction or as a volume fraction. The type of fraction depends on the **Vapor fraction specification** setting.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Two-Phase Fluid Properties (2P)

Introduced in R2018b

Variable Area Hydraulic Orifice

Hydraulic variable orifice created by cylindrical spool and sleeve



Library

Hydraulic Elements

Description

The Variable Area Hydraulic Orifice block models a variable orifice created by a cylindrical sharp-edged spool and a variable-area slot in a sleeve. The area of the orifice is expected to be computed outside the block and imported via the AR physical signal connection. The **Minimum area** parameter specifies the minimum orifice area value. If the input signal falls below this level (for example, turns negative), the area is saturated to this value. The flow rate through the orifice is proportional to the orifice area and the pressure differential across the orifice.

The flow rate is determined according to the following equations:

$$q = C_D \cdot A \sqrt{\frac{2}{\rho}} \cdot \frac{p}{(p^2 + p_{cr}^2)^{1/4}}$$

$$p = p_A - p_B$$

where

q	Flow rate
p	Pressure differential
p_A, p_B	Gauge pressures at the block terminals
C_D	Flow discharge coefficient
A	Orifice passage area
ρ	Fluid density
p_{cr}	Minimum pressure for turbulent flow, when the block transitions from laminar to turbulent regime

The minimum pressure for turbulent flow, p_{cr} , is calculated according to the laminar transition specification method:

- By pressure ratio — The transition from laminar to turbulent regime is defined by the following equations:

$$p_{cr} = (p_{avg} + p_{atm})(1 - B_{lam})$$

$$p_{avg} = (p_A + p_B)/2$$

where

p_{avg}	Average pressure between the block terminals
p_{atm}	Atmospheric pressure, 101325 Pa
B_{lam}	Pressure ratio at the transition between laminar and turbulent regimes (Laminar flow pressure ratio parameter value)

- By Reynolds number — The transition from laminar to turbulent regime is defined by the following equations:

$$p_{cr} = \frac{\rho}{2} \left(\frac{Re_{cr} \cdot \nu}{C_D \cdot D_H} \right)^2$$

$$D_H = \sqrt{\frac{4A}{\pi}}$$

where

D_H	Orifice hydraulic diameter
ν	Fluid kinematic viscosity
Re_{cr}	Critical Reynolds number (Critical Reynolds number parameter value)

The block positive direction is from port A to port B. This means that the flow rate is positive if it flows from A to B and the pressure differential is determined as $p = p_A - p_B$.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Basic Assumptions and Limitations

- Fluid inertia is not taken into account.

Parameters

Flow discharge coefficient

Semi-empirical parameter for orifice capacity characterization. Its value depends on the geometrical properties of the orifice, and usually is provided in textbooks or manufacturer data sheets. The default value is 0.7.

Minimum area

Leakage area of the completely closed orifice. If the input signal falls below this level (for example, turns negative), the area is saturated to this value. The parameter value must be greater than zero. The default value is $1e-12 \text{ m}^2$.

Laminar transition specification

Select how the block transitions between the laminar and turbulent regimes:

- **Pressure ratio** — The transition from laminar to turbulent regime is smooth and depends on the value of the **Laminar flow pressure ratio** parameter. This method provides better simulation robustness.
- **Reynolds number** — The transition from laminar to turbulent regime is assumed to take place when the Reynolds number reaches the value specified by the **Critical Reynolds number** parameter.

Laminar flow pressure ratio

Pressure ratio at which the flow transitions between laminar and turbulent regimes. The default value is 0.999. This parameter is visible only if the **Laminar transition specification** parameter is set to **Pressure ratio**.

Critical Reynolds number

The maximum Reynolds number for laminar flow. The value of the parameter depends on the orifice geometrical profile. You can find recommendations on the parameter value in hydraulics textbooks. The default value is 12, which corresponds to a round orifice in thin material with sharp edges. This parameter is visible only if the **Laminar transition specification** parameter is set to **Reynolds number**.

Global Parameters

Parameters determined by the type of working fluid:

- **Fluid density**
- **Fluid kinematic viscosity**

Use the Hydraulic Fluid block or the Custom Hydraulic Fluid block to specify the fluid properties.

Ports

The block has the following ports:

A

Hydraulic conserving port associated with the orifice inlet.

B

Hydraulic conserving port associated with the orifice outlet.

AR

Physical signal port that provides the value of the orifice area.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Constant Area Hydraulic Orifice

Introduced in R2009b

Variable Area Pneumatic Orifice

Sharp-edged variable-area orifice in pneumatic systems



Library

None (example custom library)

Description

Note As of Release R2016b, the Gas block library replaces the Pneumatic library as the recommended way of modeling pneumatic systems. The former Pneumatic library is now included in the product installation as an example custom library. The pneumatic domain definition is still provided with the software, and all the pneumatic blocks in your legacy models continue to work as before. However, these blocks no longer receive full production support and can be removed in a future release. For more information, see the R2016b Release Notes.

The Variable Area Pneumatic Orifice block models the flow rate of an ideal gas through a sharp-edged variable-area orifice. The area of the orifice is expected to be computed outside the block and imported via the AR physical signal connection. The **Minimum area** parameter specifies the minimum orifice area value. If the input signal falls below this level (for example, turns negative), the area is saturated to this value.

The flow rate through the orifice is proportional to the orifice area and the pressure differential across the orifice.

$$G = C_d \cdot A \cdot p_i \sqrt{\frac{2\gamma}{\gamma-1} \cdot \frac{1}{RT_i} \left[\left(\frac{p_o}{p_i} \right)^{\frac{2}{\gamma}} - \left(\frac{p_o}{p_i} \right)^{\frac{\gamma+1}{\gamma}} \right]}$$

where

G	Mass flow rate
C_d	Discharge coefficient, to account for effective loss of area due to orifice shape
A	Orifice cross-sectional area
p_i, p_o	Absolute pressures at the orifice inlet and outlet, respectively. The inlet and outlet change depending on flow direction. For positive flow ($G > 0$), $p_i = p_A$, otherwise $p_i = p_B$.
γ	The ratio of specific heats at constant pressure and constant volume, c_p / c_v
R	Specific gas constant
T	Absolute gas temperature

The choked flow occurs at the critical pressure ratio defined by

$$\beta_{cr} = \frac{p_o}{p_i} = \left(\frac{2}{\gamma + 1} \right)^{\frac{\gamma}{\gamma - 1}}$$

after which the flow rate depends on the inlet pressure only and is computed with the expression

$$G = C_d \cdot A \cdot p_i \sqrt{\frac{\gamma}{RT_i} \cdot \beta_{cr}^{\frac{\gamma + 1}{\gamma}}}$$

The square root relationship has infinite gradient at zero flow, which can present numerical solver difficulties. Therefore, for very small pressure differences, defined by $p_o / p_i > 0.999$, the flow equation is replaced by a linear flow-pressure relationship

$$G = k C_d \cdot A \cdot T_i^{-0.5} (p_i - p_o)$$

where k is a constant such that the flow predicted for p_o / p_i is the same as that predicted by the original flow equation for $p_o / p_i = 0.999$.

The heat flow out of the orifice is assumed equal to the heat flow into the orifice, based on the following considerations:

- The orifice is square-edged or sharp-edged, and as such is characterized by an abrupt change of the downstream area. This means that practically all the dynamic pressure is lost in the expansion.
- The lost energy appears in the form of internal energy that rises the output temperature and makes it very close to the inlet temperature.

Therefore, $q_i = q_o$, where q_i and q_o are the input and output heat flows, respectively.

The block positive direction is from port A to port B. This means that the flow rate is positive if it flows from A to B.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see "Set Priority and Initial Target for Block Variables".

Basic Assumptions and Limitations

- The gas is ideal.
- Specific heats at constant pressure and constant volume, c_p and c_v , are constant.
- The process is adiabatic, that is, there is no heat transfer with the environment.
- Gravitational effects can be neglected.
- The orifice adds no net heat to the flow.

Parameters

Discharge coefficient, Cd

Semi-empirical parameter for orifice capacity characterization. Its value depends on the geometrical properties of the orifice, and usually is provided in textbooks or manufacturer data sheets. The default value is 0.82.

Minimum area

Specifies the minimum orifice area value. If the input signal falls below this level (for example, turns negative), the area is saturated to this value. The default value is $1e-12 \text{ m}^2$.

Ports

The block has the following ports:

A

Pneumatic conserving port associated with the orifice inlet for positive flow.

B

Pneumatic conserving port associated with the orifice outlet for positive flow.

AR

Physical signal port that provides the value of the orifice area.

See Also

Constant Area Pneumatic Orifice | Constant Area Pneumatic Orifice (ISO 6358)

Introduced in R2009b

Variable Hydraulic Chamber

Hydraulic capacity of variable volume with compressible fluid



Library

Hydraulic Elements

Description

The Variable Hydraulic Chamber block models fluid compressibility in variable volume chambers. The fluid is considered to be a mixture of liquid and a small amount of entrained, nondissolved gas. Use this block together with the Translational Hydro-Mechanical Converter block.

Note The Variable Hydraulic Chamber block takes into account only the flow rate caused by fluid compressibility. The fluid volume consumed to create piston velocity is accounted for in the Translational Hydro-Mechanical Converter block.

The chamber is simulated according to the following equations (see [1 on page 1-761, 2 on page 1-761]):

$$q = \frac{V_0 + V}{E} \cdot \frac{dp}{dt}$$

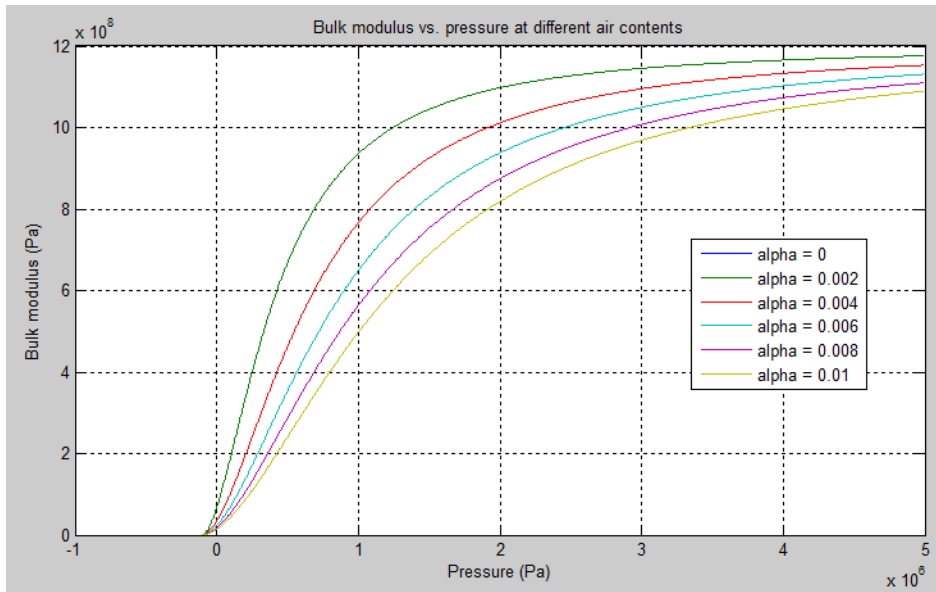
$$E = E_l \frac{1 + \alpha \left(\frac{p_a}{p_a + p} \right)^{1/n}}{1 + \alpha \frac{p_a^{1/n}}{n \cdot (p_a + p)^{\frac{n+1}{n}}} E_l}$$

where

q	Flow rate due to fluid compressibility
V_0	Initial volume of fluid in the chamber
V	Chamber volume change, provided through port V
E	Fluid bulk modulus
E_l	Pure liquid bulk modulus
p	Gauge pressure of fluid in the chamber
p_α	Atmospheric pressure
α	Relative gas content at atmospheric pressure, $\alpha = V_G/V_L$

V_G	Gas volume at atmospheric pressure
V_L	Volume of liquid
n	Gas-specific heat ratio

The main objective of representing fluid as a mixture of liquid and gas is to introduce an approximate model of cavitation, which takes place in a chamber if pressure drops below fluid vapor saturation level. As it is seen in the graph below, the bulk modulus of a mixture decreases as the gauge pressure approaches zero, thus considerably slowing down further pressure change. At gauge pressures far above zero, a small amount of undissolved gas has practically no effect on the system behavior.



For information on how to reproduce this graph, see Constant Volume Hydraulic Chamber.

Cavitation is an inherently thermodynamic process, requiring consideration of multiple-phase fluids, heat transfers, etc., and as such cannot be accurately simulated with Simscape software. But the simplified version implemented in the block is good enough to signal if pressure falls below dangerous level, and to prevent computation failure that normally occurs at negative pressures.

If pressure falls below absolute vacuum (-101325 Pa), the simulation stops and an error message is displayed.

Port A is a hydraulic conserving port associated with the chamber inlet. Port V is a physical signal port that provides the chamber volume variation.

The block positive direction is from port A to the reference point. This means that the flow rate is positive if it flows into the chamber.

Variables

Use the **Variables** tab to set the priority and initial target values for the block variables prior to simulation. For more information, see “Set Priority and Initial Target for Block Variables”.

Basic Assumptions and Limitations

- Fluid density remains constant.
- Chamber volume cannot be less than the dead volume.
- Fluid fills the entire chamber volume.

Parameters

Chamber dead volume

Minimal volume of fluid in the chamber. The default value is $1e-4 \text{ m}^3$.

Specific heat ratio

Gas-specific heat ratio. The default value is 1.4.

Restricted Parameters

When your model is in Restricted editing mode, you cannot modify the following parameter:

- **Chamber orientation**

All other block parameters are available for modification.

Global Parameters

Parameters determined by the type of working fluid:

- **Fluid density**
- **Fluid kinematic viscosity**

Use the Hydraulic Fluid block or the Custom Hydraulic Fluid block to specify the fluid properties.

Ports

The block has the following ports:

A

Hydraulic conserving port associated with the chamber inlet.

V

Physical signal port that provides the chamber volume variation.

References

- [1] Manring, N.D., *Hydraulic Control Systems*, John Wiley & Sons, New York, 2005
- [2] Meritt, H.E., *Hydraulic Control Systems*, John Wiley & Sons, New York, 1967

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Constant Volume Hydraulic Chamber

Introduced in R2009b

Variable Local Restriction (2P)

Time-varying flow resistance



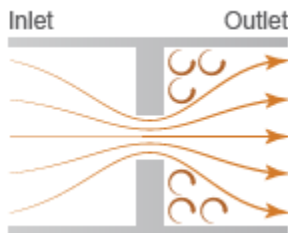
Library

Two-Phase Fluid/Elements

Description

The Variable Local Restriction (2P) block models the pressure drop due to a time-varying flow resistance such as a valve. Ports A and B represent the restriction inlet and outlet. Port AR sets the time-varying restriction area, specified as a physical signal.

The restriction consists of a contraction followed by a sudden expansion in flow area. The contraction causes the fluid to accelerate and its pressure to drop. The expansion recovers the lost pressure though only in part, as the flow separates from the wall, losing momentum in the process.



Local Restriction Schematic

Mass Balance

The mass balance equation is

$$\dot{m}_A + \dot{m}_B = 0,$$

where:

- \dot{m}_A and \dot{m}_B are the mass flow rates into the restriction through port A and port B.

Energy Balance

The energy balance equation is

$$\phi_A + \phi_B = 0,$$

where:

- ϕ_A and ϕ_B are the energy flow rates into the restriction through port A and port B.

The local restriction is assumed to be adiabatic and the change in specific total enthalpy is therefore zero. At port A,

$$u_A + p_A \nu_A + \frac{w_A^2}{2} = u_R + p_R \nu_R + \frac{w_R^2}{2},$$

while at port B,

$$u_B + p_B \nu_B + \frac{w_B^2}{2} = u_R + p_R \nu_R + \frac{w_R^2}{2},$$

where:

- u_A , u_B , and u_R are the specific internal energies at port A, at port B, and the restriction aperture.
- p_A , p_B , and p_R are the pressures at port A, port B, and the restriction aperture.
- ν_A , ν_B , and ν_R are the specific volumes at port A, port B, and the restriction aperture.
- w_A , w_B , and w_R are the ideal flow velocities at port A, port B, and the restriction aperture.

The ideal flow velocity is computed as

$$w_A = \frac{\dot{m}_{ideal} \nu_A}{S}$$

at port A, as

$$w_B = \frac{\dot{m}_{ideal} \nu_B}{S}$$

at port B, and as

$$w_R = \frac{\dot{m}_{ideal} \nu_R}{S_R},$$

inside the restriction, where:

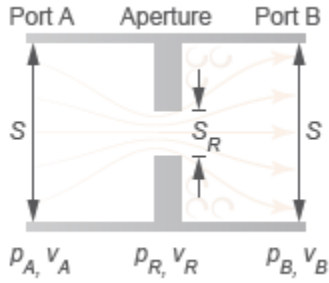
- \dot{m}_{ideal} is the ideal mass flow rate through the restriction.
- S is the flow area at port A and port B.
- S_R is the flow area of the restriction aperture.

The ideal mass flow rate through the restriction is computed as:

$$\dot{m}_{ideal} = \frac{\dot{m}_A}{C_D},$$

where:

- C_D is the flow discharge coefficient for the local restriction.



Local Restriction Variables

Momentum Balance

The change in momentum between the ports reflects in the pressure loss across the restriction. That loss depends on the mass flow rate through the restriction, though the exact dependence varies with flow regime. When the flow is turbulent:

$$\dot{m} = S_R(p_A - p_B) \sqrt{\frac{2}{|\rho_A - \rho_B| \nu_R K_T}},$$

where K_T is defined as:

$$K_T = \left(1 + \frac{S_R}{S}\right) \left(1 - \frac{\nu_{in} S_R}{\nu_{out} S}\right) - 2 \frac{S_R}{S} \left(1 - \frac{\nu_{out} S_R}{\nu_R S}\right),$$

in which the subscript *in* denotes the inlet port and the subscript *out* the outlet port. Which port serves as the inlet and which serves as the outlet depends on the pressure differential across the restriction. If pressure is greater at port **A** than at port **B**, then port **A** is the inlet; if pressure is greater at port **B**, then port **B** is the inlet.

When the flow is laminar:

$$\dot{m} = S_R(p_A - p_B) \sqrt{\frac{2}{\Delta p_{Th} \nu_R \left(1 - \frac{S_R}{S}\right)^2}},$$

where Δp_{Th} denotes the threshold pressure drop at which the flow begins to smoothly transition between laminar and turbulent:

$$\Delta p_{Th} = \left(\frac{p_A + p_B}{2}\right) (1 - B_L),$$

in which B_{Lam} is the **Laminar flow pressure ratio** block parameter. The flow is laminar if the pressure drop from port **A** to port **B** is below the threshold value; otherwise, the flow is turbulent.

The pressure at the restriction area, p_R likewise depends on the flow regime. When the flow is turbulent:

$$p_{R,L} = p_{in} - \frac{\nu_R}{2} \left(\frac{\dot{m}}{S_R}\right)^2 \left(1 + \frac{S_R}{S}\right) \left(1 - \frac{\nu_{in} S_R}{\nu_R S}\right).$$

When the flow is laminar:

$$p_{R,L} = \frac{p_A + p_B}{2}.$$

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Minimum restriction area

Area normal to the flow path at the restriction aperture when the restriction is in the fully closed state. The area obtained from physical signal AR saturates at this value. Input values smaller than the minimum restriction area are ignored and replaced by the value specified here. The default value of $1e-10 \text{ m}^2$.

Maximum restriction area

Area normal to the flow path at the restriction aperture when the restriction is in the fully open state. The area obtained from physical signal AR saturates at this value. Input values greater than the maximum restriction area are ignored and replaced by the value specified here. The default value is 0.005 m^2 .

Cross-sectional area at ports A and B

Area normal to the flow path at the restriction ports. The ports are assumed to be identical in cross-section. The default value, 0.01 m^2 , is the same as the restriction aperture area.

Flow discharge coefficient

Ratio of the actual to the theoretical mass flow rate through the restriction. The discharge coefficient is an empirical parameter used to account for non-ideal effects such as those due to restriction geometry. The default value is 0.64 .

Laminar flow pressure ratio

Ratio of the outlet to the inlet port pressure at which the flow regime is assumed to switch from laminar to turbulent. The prevailing flow regime determines the equations used in simulation. The pressure drop across the restriction is linear with respect to the mass flow rate if the flow is laminar and quadratic (with respect to the mass flow rate) if the flow is turbulent. The default value is 0.999 .

Ports

A pair of two-phase fluid conserving ports labeled A and B represent the restriction inlet and outlet. A physical signal input port labeled AR controls the cross-sectional area of the restriction aperture, located between the restriction inlet and outlet.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Local Restriction (2P)

Introduced in R2015b

Variable Reluctance

Variable reluctance in electromagnetic systems



Library

Magnetic Elements

Description

The Variable Reluctance block models a variable reluctance, that is, a component that resists flux flow. The ratio of the magnetomotive force (mmf) across the component to the resulting flux that flows through the component is defined as the reluctance, and is dependent on the value of the input physical signal.

The block is based on the following equations:

$$MMF = \Phi \cdot \mathfrak{R}$$

$$\mathfrak{R} = \frac{X}{\mu_0 \cdot \mu_r \cdot A}$$

where

MMF	Magnetomotive force (mmf) across the component
Φ	Flux through the component
\mathfrak{R}	Reluctance
X	Value presented at the physical signal port
μ_0	Permeability constant
μ_r	Relative permeability of the material
A	Cross-sectional area of the section being modeled

Connections N and S are magnetic conserving ports. The mmf across the reluctance is given by $MMF(N) - MMF(S)$, and the sign of the flux is positive when flowing through the device from N to S.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Minimum length or thickness $X \geq 0$

The minimum value of length of air gap or thickness of section. If the input signal falls below this level (for example, turns negative), this minimum value is used. The parameter value must be nonnegative. The default value is 0.

Cross-sectional area

Area of the section being modeled. The default value is 0.01 m^2 .

Relative permeability of material

Relative permeability of the section material. The default value is 1.

Ports

The block has the following ports:

N

Magnetic conserving port associated with the block North terminal.

S

Magnetic conserving port associated with the block South terminal.

The block also has one physical signal input port that provides the value of the length of air gap or thickness of section.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

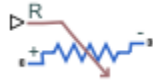
See Also

Fundamental Reluctance | Reluctance

Introduced in R2010a

Variable Resistor

Linear variable resistor in electrical systems



Library

Electrical Elements

Description

The Variable Resistor block models a linear variable resistor, described with the following equation:

$$V = I \cdot R$$

where

V	Voltage
I	Current
R	Resistance, that is, the value presented at the control port

Connections **+** and **-** are conserving electrical ports corresponding to the positive and negative terminals of the resistor, respectively. **R** is a physical signal input port that controls the resistance value. The current is positive if it flows from positive to negative, and the voltage across the resistor is equal to the difference between the voltage at the positive and the negative terminal, $V(+)$ - $V(-)$.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Minimum resistance $R \geq 0$

The minimum resistance value. If the input signal falls below this level (for example, turns negative), this minimum resistance value is used. The parameter value must be nonnegative. The default value is 0.

Ports

The block has the following ports:

+

Electrical conserving port associated with the resistor positive terminal.

- Electrical conserving port associated with the resistor negative terminal.

R

Physical signal input port that provides the resistance value.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Resistor | Thermal Resistor

Introduced in R2007a

Variable Thermal Resistance

Variable resistance in thermal systems

Library: Simscape / Foundation Library / Thermal / Thermal Elements



Description

The Variable Thermal Resistance block lets you model a variable heat transfer process in generalized terms, independent of whether it is by conduction, convection, radiation, or a combination thereof. Thermal resistance is an abstract quantity that relates the thermal conductivity, the heat transfer coefficient, and the radiation coefficient. Thermal resistance of this block can vary with time, according to the input physical signal at port R.

The heat transfer equation for the Thermal Resistance block is:

$$R \cdot Q = \Delta T$$

where:

- R is the thermal resistance.
- Q is the heat flow rate.
- ΔT is the temperature difference between layers.

Thermal resistance is related to the other heat transfer quantities as follows:

$$R = \frac{D}{k \cdot A} = \frac{1}{h \cdot A} = \frac{1}{r \cdot A(T_A^2 + T_B^2)(T_A + T_B)}$$

where:

- D is material thickness, that is, distance between layers.
- A is area normal to the heat flow direction.
- k is thermal conductivity of the material.
- h is convection heat transfer coefficient.
- r is radiation coefficient.
- T_A and T_B are temperatures at ports A and B, respectively.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Input

R — Thermal resistance control signal, K/W

physical signal

Input physical signal that specifies the thermal resistance value.

Conserving

A — Layer A

thermal

Thermal conserving port associated with layer A.

B — Layer B

thermal

Thermal conserving port associated with layer B.

Parameters

Minimum thermal resistance — Minimum thermal resistance value allowed

0 K/W (default)

If the input physical signal falls below this value, the resistance remains at this value.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Conductive Heat Transfer | Convective Heat Transfer | Radiative Heat Transfer | Thermal Resistance

Introduced in R2017b

Variable Volume Chamber

Hydraulic capacity of variable volume with compressible fluid



Library

None (kept for compatibility purposes only)

Description

The Variable Volume Chamber block has been deprecated and removed from the library as of Version 3.0 (R2008b). Documentation is kept for compatibility reasons. If you use this block in your older models, it will still work. However, support may be discontinued in a future version. Replace this block with the Hydraulic Piston Chamber block.

See Also

Constant Volume Hydraulic Chamber

Hydraulic Piston Chamber

Translational Hydro-Mechanical Converter

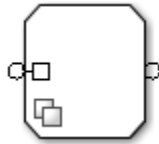
Variable Hydraulic Chamber

Introduced in R2007a

Variant Connector

Remove or disconnect physical components from network

Library: Simscape / Utilities



Description

The Variant Connector block lets you define variant choices in a physical network. Variant choices allow you to exclude some components from simulation without physically removing the components from the physical network. The components to be disconnected must be connected to the port associated with condition data. During simulation, the variant condition is propagated to all the connected components within the physical network. If the variant control associated with the block evaluates to `true`, all the components that are connected to the block become active. If the variant control evaluates to `false`, all the components connected to the block become inactive.

Note The Variant Connector block does not propagate the variant condition across the boundary between the Simscape physical network and the Simulink blocks connected to it. In other words, a block that has both conserving ports and Simulink signal ports, such as a Simulink-PS Converter or a PS-Simulink Converter block, does not propagate the variant condition from a Variant Connector block between the two parts of the model.

Ports

Conserving

Port_1 – Condition port

untyped conserving port

This port is located on the left side of the block and is labeled with a rectangle. During simulation, if the variant choice associated with the block evaluates to `true`, all the components that are connected to this port become active. If the variant choice associated with the block evaluates to `false`, all the components connected to this port become inactive.

By default, this port is untyped. You define the type of this port by connecting it to a conserving port of another block, components in the network, or a Simscape Bus port.

Port_2 – Connection port

untyped conserving port

Conserving connection port. By default, this port is untyped.

You define the type of the port by connecting it to a conserving port of another block, components in the network, or a Simscape Bus port.

Parameters

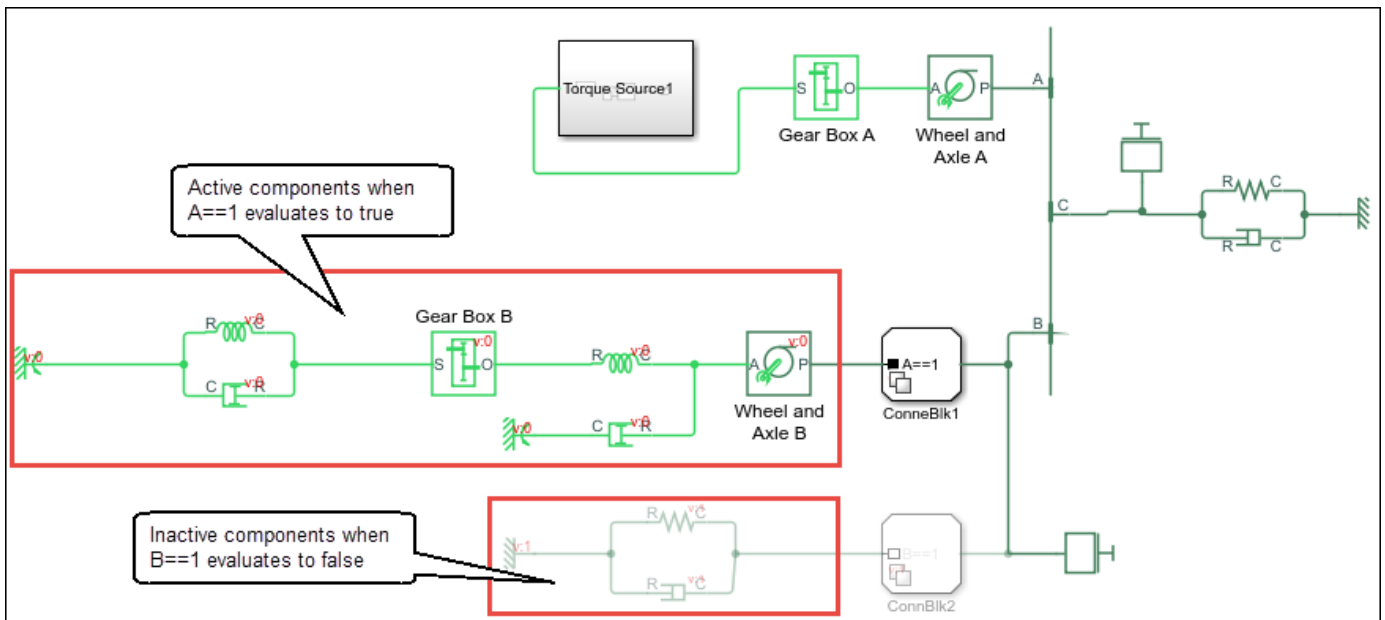
Connector Type — Type of connector block

Leaf (default) | Primary | Nonprimary

Select a connector type based on the components to deactivate.

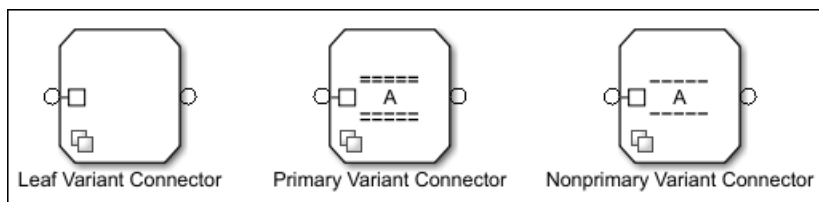
- **Leaf:** If you want to deactivate all the components connected to the block, set **Connection Type** to Leaf.

For example, in this model, the variant condition associated with the Variant Connector block, ConneBlk1, is $A == 1$. The variant condition associated with the Variant Connector block, ConneBlk2, is $B == 1$. During simulation, when $A == 1$ evaluates to true and $B == 1$ evaluates to false, all the components connected to ConneBlk1 block become active, and all the components connected to ConneBlk2 block become inactive.



- **Primary and Nonprimary:** If you want to restrict the propagation of the variant condition to a set of components, create a “Bounded region” on page 1-777 by using primary and nonprimary connector blocks. To form a bounded region, the primary and nonprimary connector blocks must have the same **Connector Tag** property.

When you set **Connection Type** to Primary or Nonprimary, the connector tag of the block is displayed on the block icon. The connector tag of the primary connector block is displayed with double lines and the connector tag of the nonprimary connector block is displayed with single lines.



Connector Tag — Unique identifier of bounded region

A (default) | Valid MATLAB identifier that starts with a letter, contains no spaces or special characters and is at most 63 characters long

Specify an identifier in the **Connector Tag** parameter. The primary and nonprimary connector blocks with the same tag form a “Bounded region” on page 1-777.

Dependencies

To enable this parameter, set **Connector Type** to Primary or Nonprimary.

Variant control expression — Variant controls available in global workspace

'Choice_1' (default) | Boolean condition expression

Displays the variant controls available in the global workspace. The variant control can be a Boolean condition expression containing a regular MATLAB variable or a `Simulink.Variant` object representing a Boolean condition expression.

To edit a variant name, double-click a **Variant control expression** cell and type in the variant control expression. Click **Apply** after you edit a variant control name.

When the variant control associated with the block evaluates to `true`, all the components connected to the block become active.

Programmatic Use

Block Parameter: `VariantControls`

Type: cell array of character vectors

Value: Variant control that is associated with the Variant choice

Default: 'Variant'

Condition (read-only) — Condition for Variant controls

no default

This read-only field is based on the condition for the associated Variant control in the global workspace. Create or change a Variant condition in the `Simulink.Variant` parameter dialog box or in the global workspace.

For more information, see “Create Variant Controls Programmatically” and `Simulink.Variant`.

Show variant condition on block — Display variant condition on block icon

off (default) | on

Select **Show variant condition on block** to display the variant condition associated with the block on the block icon.

Open block in Variant Manager — Open Variant Manager window

no default

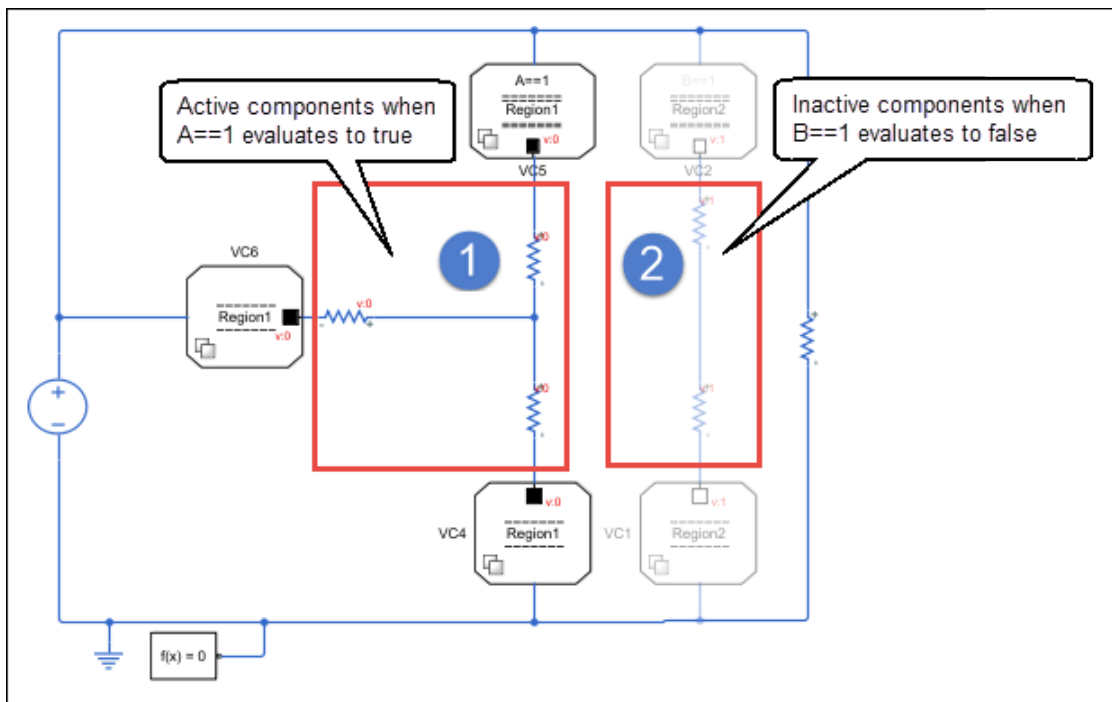
Click **Open block in Variant Manager** to open the Variant Manager.

More About**Bounded region**

A bounded region is a part of a model located between one Variant Connector block where **Connector type** is set to Primary and any number of Variant Connector blocks where **Connector**

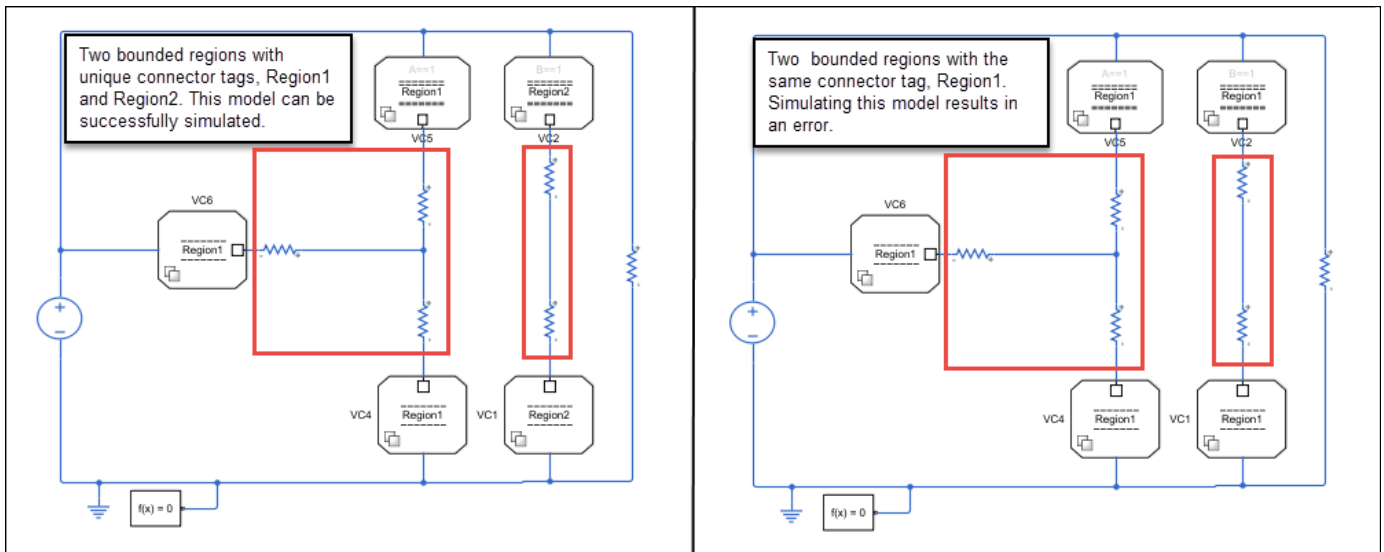
type is set to Nonprimary. All of the blocks in a bounded region must use the same **Connector tag** parameter.

For example, this model has two bounded regions. The first region has one primary and two nonprimary Variant Connector blocks. These blocks are connected to each other by the connector tag **Region1** and the associated variant condition $A == 1$. The second region has one primary and one nonprimary Variant Connector block. These blocks are connected to each other by the connector tag **Region2** and the associated variant condition $B == 1$. During simulation, when $A == 1$ evaluates to true, the components in the first region become active. When $B == 1$ evaluates to false, the components in the second region become inactive.

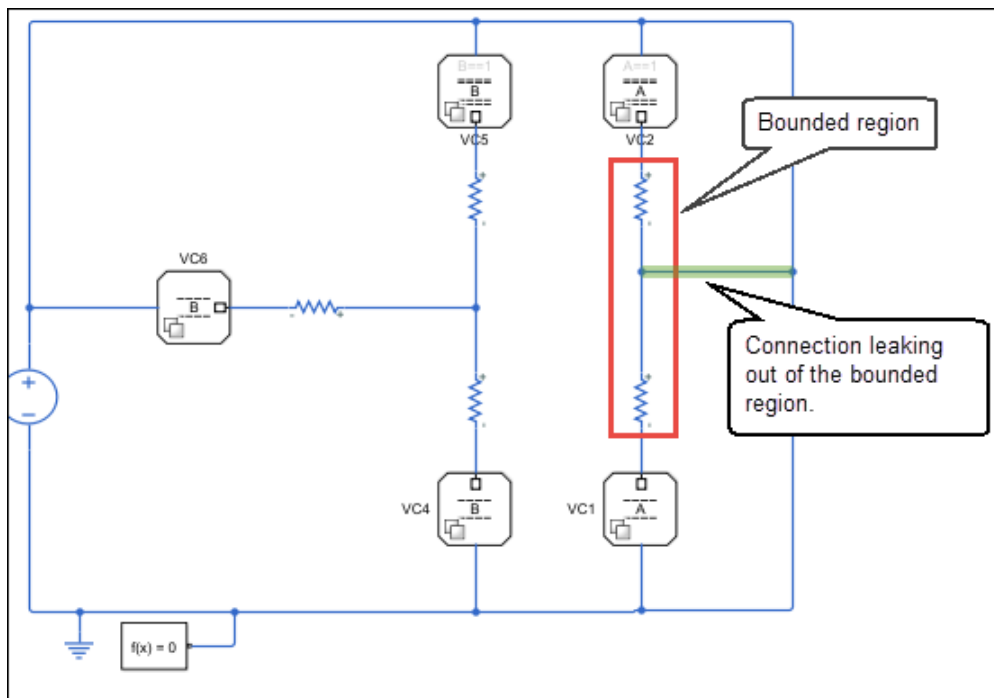


When you create a bounded region, ensure that:

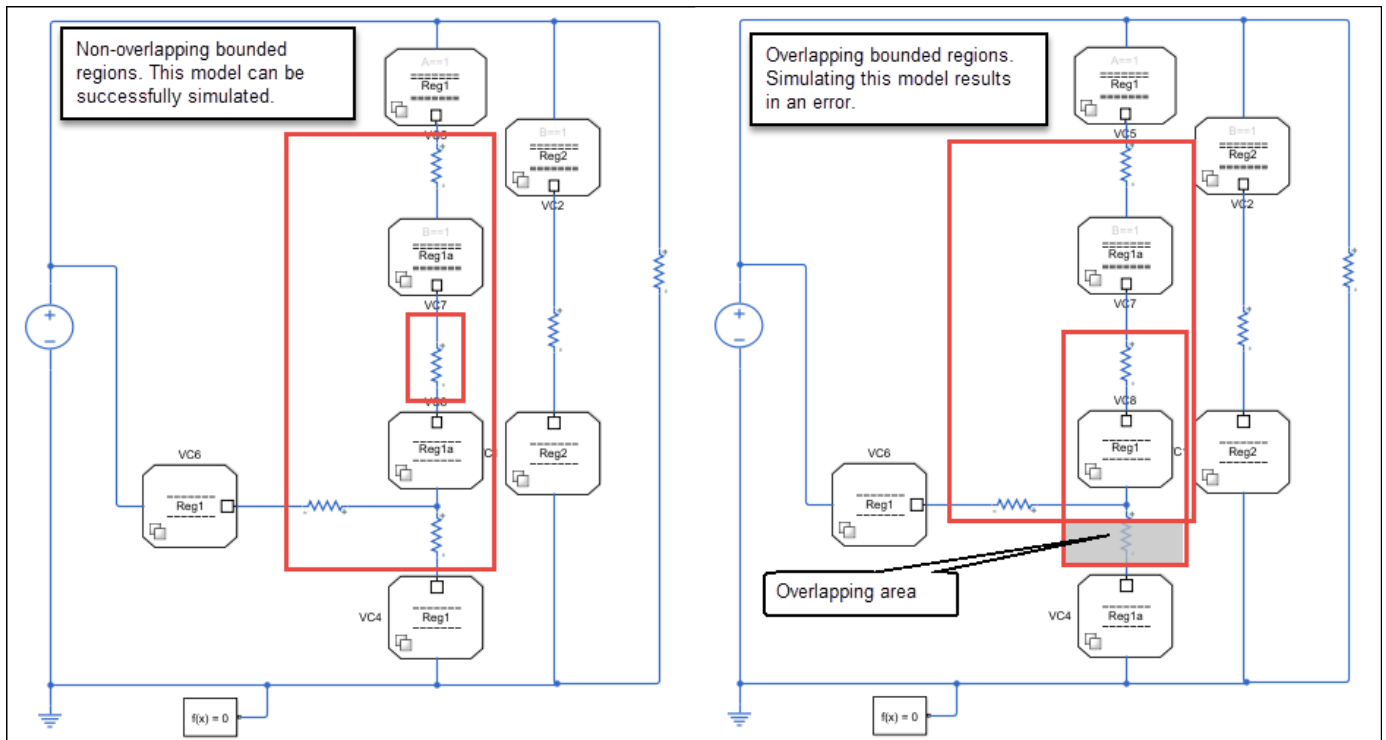
- Each region has only one primary Variant Connector block. Each primary Variant Connector block is connected to at least one nonprimary Variant Connector block. All Variant Connector blocks must be connected using the port associated with condition information.
- All the primary and nonprimary blocks are at the same level of model hierarchy.
- Each region has a unique connector tag.



- The region is bounded and has no external connections.



- There are no Variant Connector blocks where the **Condition type** is set to Leaf inside the bounded region.
- No bounded regions overlap each other.



Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Topics

“Model Variants in an Electrical Circuit Using Variant Connector Blocks”

“Model Variants in a Mechanical System Using Variant Connector Blocks”

Introduced in R2020b

Voltage-Controlled Current Source

Linear voltage-controlled current source



Library

Electrical Sources

Description

The Voltage-Controlled Current Source block models a linear voltage-controlled current source, described with the following equation:

$$I = K \cdot (V(+)) - V(-))$$

where

I	Current
K	Transconductance
$V(+), V(-)$	Voltages presented at the + and - control ports

To use the block, connect the + and - ports on the left side of the block (the control ports) to the control voltage source. The two ports on the right side of the block (the output ports) generate the output current. The arrow indicates the positive direction of the current flow.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Transconductance K

Transconductance, or the change in output current divided by the change in input voltage that causes it. The default value is 1 1/Ω.

Ports

The block has four electrical conserving ports. Connections + and - on the left side of the block are the control ports. The other two ports are the electrical terminals that provide the output current. The arrow indicates the positive direction of the current flow.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

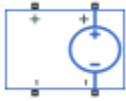
See Also

Current-Controlled Current Source | Current-Controlled Voltage Source | Voltage-Controlled Voltage Source

Introduced in R2007a

Voltage-Controlled Voltage Source

Linear voltage-controlled voltage source



Library

Electrical Sources

Description

The Voltage-Controlled Voltage Source block models a linear voltage-controlled voltage source, described with the following equation:

$$V = K \cdot (V(+)) - V(-)$$

where

V	Output voltage
K	Voltage gain
$V(+), V(-)$	Voltages presented at the + and - control ports

To use the block, connect the + and - ports on the left side of the block (the control ports) to the control voltage source. The two ports on the right side of the block (the output ports) generate the output voltage. Polarity is indicated by the + and - signs.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Parameters

Voltage gain K

The change in the output voltage divided by the change in the control voltage that causes it. The default value is 1.

Ports

The block has four electrical conserving ports. Connections + and - on the left side of the block are the control ports. The other two ports are the electrical terminals that provide the output voltage. Polarity is indicated by the + and - signs.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Current-Controlled Current Source | Current-Controlled Voltage Source | Voltage-Controlled Current Source

Introduced in R2007a

Voltage Sensor

Voltage sensor in electrical systems



Library

Electrical Sensors

Description

The Voltage Sensor block represents an ideal voltage sensor, that is, a device that converts voltage measured between two points of an electrical circuit into a physical signal proportional to the voltage.

Connections + and - are electrical conserving ports through which the sensor is connected to the circuit. Connection V is a physical signal port that outputs the measurement result.

Ports

The block has the following ports:

+

Electrical conserving port associated with the sensor positive terminal.

-

Electrical conserving port associated with the sensor negative terminal.

V

Physical signal output port for voltage.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Current Sensor | PS-Simulink Converter

Topics

“Connecting Simscape Diagrams to Simulink Sources and Scopes”

Introduced in R2007a

Volumetric Flow Rate Sensor (2P)

Measure volumetric flow rate



Library

Two-Phase Fluid/Sensors

Description

The Volumetric Flow Rate Sensor (2P) block measures the volumetric flow rate through the two-phase fluid branch defined by ports A and B. The flow rate is positive if fluid flows from port A to port B.

Ports

The block has two two-phase fluid conserving ports, A and B. Physical signal port V outputs the volumetric flow rate value.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Volumetric Flow Rate Source (2P)

Introduced in R2015b

Volumetric Flow Rate Sensor (G)

Measure volumetric flow rate

Library: Simscape / Foundation Library / Gas / Sensors



Description

The Volumetric Flow Rate Sensor (G) block represents an ideal sensor that measures volumetric flow rate in a gas network. There is no change in pressure or temperature across the sensor.

Because the flow rate is a Through variable, the block must connect in series with the component being measured.

The relative orientation of ports **A** and **B** establishes the measurement sign. The sign is positive if flow occurs from port **A** to port **B**. Switching port connections reverses the measurement sign.

Physical signal port **V** outputs the volumetric flow rate through the sensor. Connect this port to a PS-Simulink Converter block to transform the output physical signal into a Simulink signal, for example, for plotting or additional data processing.

Ports

Output

V — Volumetric flow rate, m^3/s

physical signal

Physical signal output port for the volumetric flow rate measurement.

Conserving

A — Sensor inlet

gas

Gas conserving port. Flow rate is positive if gas flows from port **A** to port **B**.

B — Sensor outlet

gas

Gas conserving port. Flow rate is positive if gas flows from port **A** to port **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mass & Energy Flow Rate Sensor (G) | Pressure & Temperature Sensor (G) | Thermodynamic Properties Sensor (G)

Topics

“Modeling Gas Systems”

Introduced in R2019a

Volumetric Flow Rate Sensor (MA)

Measure volumetric flow rate in a moist air network

Library: Simscape / Foundation Library / Moist Air / Sensors



Description

The Volumetric Flow Rate Sensor (MA) block represents an ideal sensor that measures volumetric flow rate in a moist air network. There is no change in pressure, temperature, specific humidity, or trace gas mass fraction across the sensor.

Because the flow rate is a Through variable, the block must connect in series with the component being measured.

The relative orientation of ports **A** and **B** establishes the measurement sign. The sign is positive if flow occurs from port **A** to port **B**. Switching port connections reverses the measurement sign.

Physical signal port **V** outputs the volumetric flow rate through the sensor. Connect this port to a PS-Simulink Converter block to transform the output physical signal into a Simulink signal, for example, for plotting or additional data processing.

Ports

Output

V — Mixture volumetric flow rate, m^3/s

physical signal

Physical signal output port for the mixture volumetric flow rate measurement.

Conserving

A — Sensor inlet

moist air

Moist air conserving port. Volumetric flow rate is positive if air flows from port **A** to port **B**.

B — Sensor outlet

moist air

Moist air conserving port. Volumetric flow rate is positive if air flows from port **A** to port **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mass & Energy Flow Rate Sensor (MA) | Pressure & Temperature Sensor (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Volumetric Flow Rate Sensor (TL)

Measure volumetric flow rate

Library: Simscape / Foundation Library / Thermal Liquid / Sensors



Description

The Volumetric Flow Rate Sensor (TL) block represents an ideal sensor that measures volumetric flow rate in a thermal liquid network.

Because the flow rate is a Through variable, the block must connect in series with the component being measured.

The relative orientation of ports **A** and **B** establishes the measurement sign. The sign is positive if flow occurs from port **A** to port **B**. Switching port connections reverses the measurement sign.

Physical signal port **V** outputs the volumetric flow rate through the sensor. Connect this port to a PS-Simulink Converter block to transform the output physical signal into a Simulink signal, for example, for plotting or additional data processing.

Ports

Output

V — Volumetric flow rate, m³/s

physical signal

Physical signal output port for the volumetric flow rate measurement.

Conserving

A — Sensor inlet

thermal liquid

Thermal liquid conserving port. Flow rate is positive if fluid flows from port **A** to port **B**.

B — Sensor outlet

thermal liquid

Thermal liquid conserving port. Flow rate is positive if fluid flows from port **A** to port **B**.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Mass & Energy Flow Rate Sensor (TL) | Pressure & Temperature Sensor (TL) | Thermodynamic Properties Sensor (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2016a

Volumetric Flow Rate Source (2P)

Generate constant volumetric flow rate



Library

Two-Phase Fluid/Sources

Description

The Volumetric Flow Rate Source (2P) block generates a constant volumetric flow rate in a two-phase fluid network branch. The source has two inlets, labeled **A** and **B**, with independently specified cross-sectional areas. By default, the source does isentropic work on the fluid, though the block provides the option to ignore this work.

The source is ideal. In other words, it maintains the specified flow rate regardless of the pressure differential produced between its ports. In addition, because the source is isentropic, there is no viscous friction between the ports and no heat exchange with the environment. Use this block to model an idealized pump or compressor or to set a boundary condition in a model.

Mass Balance

The volume of fluid in the source is considered negligible and is ignored in a model. There is no fluid accumulation between the ports and the sum of all mass flow rates into the source must therefore equal zero:

$$\dot{m}_A + \dot{m}_B = 0,$$

where \dot{m} denotes the mass flow rate into the source through a port. Its value at port **A** is calculated from the specified volumetric flow rate:

$$\dot{m}_A = \begin{cases} \frac{\dot{V}}{v_B}, & \text{if } \dot{V} \geq 0 \\ \frac{\dot{V}}{v_A}, & \text{otherwise} \end{cases},$$

where \dot{V} is volumetric flow rate and v is specific volume.

Energy Balance

By default, the source maintains the specified flow rate by performing isentropic work on the incoming fluid, though the block provides the option to ignore this term. The rate at which the source does work, if considered in the model, must equal the sum of the energy flow rates through the ports:

$$\phi_A + \phi_B + \phi_{\text{Work}} = 0,$$

where ϕ denotes the energy flow rate into the source through a port or by means of work. The energy flow rate due to work is equal to the power generated by the source. Its value is calculated from the specific total enthalpies at the ports:

$$\phi_{\text{Work}} = \dot{m}_A(h_A - h_B).$$

The specific total enthalpy h is defined as:

$$h_* = u_* + p_*v_* + \frac{1}{2} \left(\frac{\dot{m}_*v_*}{S} \right)^2,$$

where the asterisk denotes a port (**A** or **B**) and:

- u is specific internal energy.
- p is pressure.
- S is flow area.

The specific internal energy in the equation is obtained from the tabulated data of the Two-Phase Fluid Properties (2P) block. Its value is uniquely determined from the constraint that the work done by the source is isentropic. The specific entropy, a function of specific internal energy, must then have the same value at ports **A** and **B**:

$$s_A(p_A, u_A) = s_B(p_B, u_B),$$

where s is specific entropy. If the **Power added** parameter is set to **None**, the specific total enthalpies at the ports have the same value ($h_A = h_B$) and the work done by the source reduces to zero ($\phi_{\text{Work}} = 0$).

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

A – Inlet

two-phase fluid

Opening through fluid can enter and exit the source.

B – Inlet

two-phase fluid

Opening through fluid can enter and exit the source.

Parameters

Power added – Parameterization for the calculation of power

Isentropic power (default) | None

Parameterization for the calculation of power. Work is isentropic and its calculation is based on the assumptions of zero friction losses and zero heat exchange with the environment. Change to None to prevent the source from impacting the temperature of the fluid—for example, when using this block as a boundary condition in a model.

Volumetric flow rate — Volume pumped per unit time from port A to port B

0 m³/s (default) | scalar with units of volume/time

Volume of fluid pumped through the ports per unit time. A positive rate corresponds to a flow directed from port **A** to port **B**. The specified rate is maintained no matter the pressure differential produced between the ports.

Cross-sectional area at port A — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of volume/time

Area of the fluid opening normal to the direction of flow.

Cross-sectional area at port B — Inlet area normal to the direction of flow

0.01 m² (default) | scalar with units of volume/time

Area of the fluid opening normal to the direction of flow.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Mass Flow Rate Source (2P) | Controlled Volumetric Flow Rate Source (2P) | Mass Flow Rate Source (2P)

Introduced in R2015b

Volumetric Flow Rate Source (G)

Generate constant volumetric flow rate

Library: Simscape / Foundation Library / Gas / Sources



Description

The Volumetric Flow Rate Source (G) block represents an ideal mechanical energy source in a gas network. The source can maintain a constant volumetric flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. A positive volumetric flow rate causes gas to flow from port **A** to port **B**.

The volumetric flow rate and mass flow rate are related through the expression

$$\dot{m} = \begin{cases} \rho_B \dot{V} & \text{for } \dot{V} \geq 0 \\ \rho_A \dot{V} & \text{for } \dot{V} < 0 \end{cases}$$

where:

- \dot{m} is the mass flow rate from port **A** to port **B**.
- ρ_A and ρ_B are densities at ports **A** and **B**, respectively.
- \dot{V} is the volumetric flow rate.

You can choose whether the source performs work on the gas flow:

- If the source is isentropic (**Power added** parameter is set to **Isentropic power**), then the isentropic relation depends on the gas property model.

Gas Model	Equations
Perfect gas	$\frac{(p_A)^{Z \cdot R/c_p}}{T_A} = \frac{(p_B)^{Z \cdot R/c_p}}{T_B}$
Semiperfect gas	$\int_0^{T_A} \frac{c_p(T)}{T} dT - Z \cdot R \cdot \ln(p_A) = \int_0^{T_B} \frac{c_p(T)}{T} dT - Z \cdot R \cdot \ln(p_B)$
Real gas	$s(T_A, p_A) = s(T_B, p_B)$

The power delivered to the gas flow is based on the specific total enthalpy associated with the isentropic process.

$$\Phi_{work} = -\dot{m}_A \left(h_A + \frac{w_A^2}{2} \right) - \dot{m}_B \left(h_B + \frac{w_B^2}{2} \right)$$

- If the source performs no work (**Power added** parameter is set to **None**), then the defining equation states that the specific total enthalpy is equal on both sides of the source. It is the same for all three gas property models.

$$h_A + \frac{w_A^2}{2} = h_B + \frac{w_B^2}{2}$$

The power delivered to the gas flow $\Phi_{\text{work}} = 0$.

The equations use these symbols:

c_p	Specific heat at constant pressure
h	Specific enthalpy
\dot{m}	Mass flow rate (flow rate associated with a port is positive when it flows into the block)
p	Pressure
R	Specific gas constant
s	Specific entropy
T	Temperature
w	Flow velocity
Z	Compressibility factor
Φ_{work}	Power delivered to the gas flow through the source

Subscripts A and B indicate the appropriate port.

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Conserving

A – Source inlet

gas

Gas conserving port. A positive mass flow rate causes gas to flow from port **A** to port **B**.

B – Source outlet

gas

Gas conserving port. A positive mass flow rate causes gas to flow from port **A** to port **B**.

Parameters

Power added – Select whether the source performs work

Isentropic power (default) | None

Select whether the source performs work on the gas flow:

- **Isentropic power** — The source performs isentropic work on the gas to maintain the specified mass flow rate, regardless of the pressure differential. Use this option to represent an idealized pump or compressor and properly account for the energy input and output, especially in closed-loop systems.
- **None** — The source performs no work on the flow, neither adding nor removing power, regardless of the mass flow rate produced by the source. Use this option to set up the desired flow condition upstream of the system, without affecting the temperature of the flow.

Volumetric flow rate — Constant volumetric flow rate through the source

0 m³/s (default)

Desired volumetric flow rate of gas through the source.

Cross-sectional area at port A — Area normal to flow path at port A

0.01 m² (default)

Area normal to flow path at port A.

Cross-sectional area at port B — Area normal to flow path at port B

0.01m² (default)

Area normal to flow path at port B.

Extended Capabilities**C/C++ Code Generation**

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Volumetric Flow Rate Source (G)

Topics

“Modeling Gas Systems”

Introduced in R2019a

Volumetric Flow Rate Source (MA)

Generate constant volumetric flow rate

Library: Simscape / Foundation Library / Moist Air / Sources



Description

The Volumetric Flow Rate Source (MA) block represents an ideal mechanical energy source in a moist air network. The source can maintain a constant volumetric flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. A positive volumetric flow rate causes moist air to flow from port **A** to port **B**.

The volumetric flow rate and mass flow rate are related through the expression

$$\dot{m} = \begin{cases} \rho_B \dot{V} & \text{for } \dot{V} \geq 0 \\ \rho_A \dot{V} & \text{for } \dot{V} < 0 \end{cases}$$

where:

- \dot{m} is the mass flow rate from port **A** to port **B**.
- ρ_A and ρ_B are densities at ports **A** and **B**, respectively.
- \dot{V} is the volumetric flow rate.

The equations describing the source use these symbols.

c_p	Specific heat at constant pressure
h	Specific enthalpy
h_t	Specific total enthalpy
\dot{m}	Mass flow rate (flow rate associated with a port is positive when it flows into the block)
p	Pressure
ρ	Density
R	Specific gas constant
s	Specific entropy
T	Temperature
Φ_{work}	Power delivered to the moist air flow through the source

Subscripts A and B indicate the appropriate port.

Mass balance:

$$\dot{m}_A + \dot{m}_B = 0$$

$$\dot{m}_{wA} + \dot{m}_{wB} = 0$$

$$\dot{m}_{gA} + \dot{m}_{gB} = 0$$

Energy balance:

$$\Phi_A + \Phi_B + \Phi_{work} = 0$$

If the source performs no work (**Power added** parameter is set to None), then $\Phi_{work} = 0$.

If the source is isentropic (**Power added** parameter is set to Isentropic power), then

$$\Phi_{work} = \dot{m}_A(h_{tB} - h_{tA})$$

where

$$h_{tA} = h_A + \frac{1}{2} \left(\frac{\dot{m}_A}{\rho_A S_A} \right)^2$$

$$h_{tB} = h_B + \frac{1}{2} \left(\frac{\dot{m}_B}{\rho_B S_B} \right)^2$$

The mixture-specific enthalpies, $h_A = h(T_A)$ and $h_B = h(T_B)$, are constrained by the isentropic relation, that is, there is no change in entropy:

$$\int_{T_A}^{T_B} \frac{1}{T} dh(T) = R \ln \left(\frac{p_B}{p_A} \right)$$

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Conserving

A — Source inlet

moist air

Moist air conserving port. A positive volumetric flow rate causes moist air to flow from port **A** to port **B**.

B — Source outlet

moist air

Moist air conserving port. A positive volumetric flow rate causes moist air to flow from port **A** to port **B**.

Parameters

Power added — Select whether the source performs work

Isentropic power (default) | None

Select whether the source performs work on the moist air flow:

- **Isentropic power** — The source performs isentropic work on the moist air to maintain the specified mass flow rate, regardless of the pressure differential. Use this option to represent an idealized pump or compressor and properly account for the energy input and output, especially in closed-loop systems.
- **None** — The source performs no work on the flow, neither adding nor removing power, regardless of the mass flow rate produced by the source. Use this option to set up the desired flow condition upstream of the system, without affecting the temperature of the flow.

Mixture volumetric flow rate — Constant volumetric flow rate through the source

0 m³/s (default)

Desired volumetric flow rate of the moist air through the source.

Cross-sectional area at port A — Area normal to flow path at port A

0.01 m² (default)

Area normal to flow path at port A.

Cross-sectional area at port B — Area normal to flow path at port B

0.01 m² (default)

Area normal to flow path at port B.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Volumetric Flow Rate Source (MA)

Topics

“Modeling Moist Air Systems”

“Modeling Moisture and Trace Gas Levels”

Introduced in R2018a

Volumetric Flow Rate Source (TL)

Generate constant volumetric flow rate

Library: Simscape / Foundation Library / Thermal Liquid / Sources



Description

The Volumetric Flow Rate Source (TL) block represents an ideal mechanical energy source in a thermal liquid network. The source can maintain a constant volumetric flow rate regardless of the pressure differential. There is no flow resistance and no heat exchange with the environment. A positive volumetric flow rate causes the fluid to flow from port **A** to port **B**.

The volumetric flow rate and mass flow rate are related through the expression

$$\dot{m} = \begin{cases} \rho_B \dot{V} & \text{for } \dot{V} \geq 0 \\ \rho_A \dot{V} & \text{for } \dot{V} < 0 \end{cases}$$

where:

- \dot{m} is the mass flow rate from port **A** to port **B**.
- ρ_A and ρ_B are densities at ports **A** and **B**, respectively.
- \dot{V} is the volumetric flow rate.

The energy balance at the source is a function of the energy flow rates through ports **A** and **B** and the work done on the fluid:

$$\phi_A + \phi_B + \phi_{work} = 0,$$

where:

- ϕ_A is the energy flow rate into the source through port **A**.
- ϕ_B is the energy flow rate into the source through port **B**.
- ϕ_{work} is the isentropic work done on the fluid.

The isentropic work term is

$$\phi_{work} = \frac{\dot{m}(p_B - p_A)}{\rho_{avg}},$$

where:

- ϕ_{work} is the isentropic work done on the thermal liquid.
- p_A is the pressure at port **A**.
- p_B is the pressure at port **B**.

- ρ_{avg} is the average liquid density,

$$\rho_{avg} = \frac{\rho_A + \rho_B}{2}.$$

Assumptions and Limitations

- There are no irreversible losses.
- There is no heat exchange with the environment.

Ports

Conserving

A — Source inlet

thermal liquid

Thermal liquid conserving port. A positive flow rate causes the fluid to flow from port **A** to port **B**.

B — Source outlet

thermal liquid

Thermal liquid conserving port. A positive flow rate causes the fluid to flow from port **A** to port **B**.

Parameters

Volumetric flow rate — Constant volumetric flow rate through the source

0 m³/s (default)

Desired volumetric flow rate of the fluid through the source.

Cross-sectional area at ports A and B — Area normal to flow path at inlet and outlet

0.01 m² (default)

Area normal to the direction of flow at the source inlet and outlet. The two cross-sectional areas are assumed identical.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Controlled Mass Flow Rate Source (TL) | Controlled Pressure Source (TL) | Controlled Volumetric Flow Rate Source (TL) | Mass Flow Rate Source (TL) | Pressure Source (TL)

Topics

“Modeling Thermal Liquid Systems”

Introduced in R2016a

Wheel and Axle

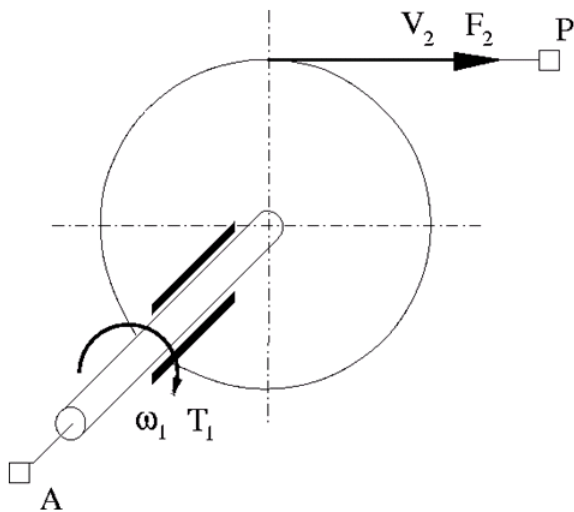
Wheel and axle mechanism in mechanical systems

Library: Simscape / Foundation Library / Mechanical / Mechanisms



Description

The Wheel and Axle block represents a wheel and axle mechanism shown in the following schematic.



The wheel and the axle have the same axis, and the axis is assumed to be rigidly connected to the frame, thus making this mechanism an ideal converter of mechanical rotational into mechanical translational motion. The mechanism has two connections: a mechanical rotational port A, which corresponds to the axle, and a mechanical translational port P, which corresponds to the wheel periphery. The mechanism is described with the following equations:

$$T = r \cdot F \cdot \text{or}$$

$$v = r \cdot \omega \cdot \text{or}$$

where

- T is torque on the axle.
- F is force on the wheel periphery.
- ω is angular velocity.
- v is linear velocity on the wheel periphery.
- r is wheel radius.

- *or* is mechanism orientation indicator. The value is +1 if axle rotation in the globally assigned positive direction is converted into translational motion in positive direction. The value is -1 if positive rotation results in translational motion in negative direction.

Use the block in simulation of rack-pinions, steering wheels, hoisting devices, windlasses, and so on.

The block positive directions are from A to the reference point and from the reference point to P.

Variables

To set the priority and initial target values for the block variables prior to simulation, use the **Variables** tab in the block dialog box (or the **Variables** section in the block Property Inspector). For more information, see “Set Priority and Initial Target for Block Variables”.

Ports

Conserving

A – Axle

mechanical rotational

Mechanical rotational conserving port associated with the axle.

P – Wheel periphery

mechanical translational

Mechanical translational conserving port associated with the wheel periphery.

Parameters

Wheel radius – Radius of the wheel

0.05 m (default)

Radius of the wheel.

Mechanism orientation – Relationship between axle rotation and direction of translational motion

Drives in positive direction (default) | Drives in negative direction

The value *Drives in positive direction* specifies a mechanism where axle rotation in the globally assigned positive direction is converted into translational motion in positive direction. The value *Drives in negative direction* specifies a mechanism where axle rotation in the globally assigned positive direction is converted into translational motion in negative direction.

Extended Capabilities

C/C++ Code Generation

Generate C and C++ code using Simulink® Coder™.

See Also

Gear Box | Lever | Slider-Crank

Introduced in R2007a

Asynchronous Sample & Hold

(Legacy block) Output sample-and-hold signal with external trigger



Description

This block has been renamed in R2019a. For more information, see PS Asynchronous Sample & Hold.

For upgrade information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Introduced in R2011b

Counter

(Legacy block) Increment output signal by 1 with every time step



Description

This block has been renamed in R2019a. For more information, see PS Counter.

For upgrade information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Introduced in R2012b

Random Number

(Legacy block) Generate normally distributed random numbers for physical modeling



Description

This block has been renamed in R2019a. For more information, see PS Random Number.

For upgrade information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Introduced in R2013a

Repeating Sequence

(Legacy block) Output periodic piecewise linear signal



Description

This block has been renamed in R2019a. For more information, see PS Repeating Sequence.

For upgrade information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Introduced in R2012b

Uniform Random Number

(Legacy block) Generate uniformly distributed random numbers for physical modeling



Description

This block has been renamed in R2019a. For more information, see PS Uniform Random Number.

For upgrade information, see “Upgrading Models with Legacy Physical Signal Blocks”.

Introduced in R2013a

Functions

hydraulicToIsothermalLiquid

Upgrade hydraulic block diagram system to use isothermal liquid blocks

Syntax

```
newfile = hydraulicToIsothermalLiquid(oldfile,newpath)
```

Description

`newfile = hydraulicToIsothermalLiquid(oldfile,newpath)` replaces blocks from the Foundation > Hydraulic library in the specified block diagram system, `oldfile`, with equivalent Isothermal Liquid library blocks, while trying to preserve the parameter values and connections between the blocks, where possible. If you have a Simscape Fluids license, this tool also replaces blocks from the Fluids > Hydraulics (Isothermal) library with equivalent blocks from the Fluids > Isothermal Liquid and Foundation > Isothermal Liquid libraries. `oldfile` can be a model, subsystem, or library.

Isothermal Liquid block libraries are structured similar to other fluid domains, such as Thermal Liquid, and often there is no one-to-one correspondence between the Isothermal Liquid and Hydraulic library blocks. The conversion tool lists all the issues encountered during the conversion in an HTML report, saves both the report and the converted block diagram system in the location specified by `newpath`, and returns the name of the converted system, `newfile`.

Both the `newfile` and the `newpath` arguments are optional. If you omit the `newpath` argument, you must have write permissions for your current working folder, because the tool then saves the converted system and the report in the current folder.

The new file is based on the last saved version of the old file. That is, if you modify the original block diagram system and do not save it before using the conversion tool, the modifications are not reflected in the new system. It is also good practice to make sure that the original block diagram system compiles without issues before using the conversion tool.

For more information on upgrade considerations and process, see “Upgrading Hydraulic Models To Use Isothermal Liquid Blocks”.

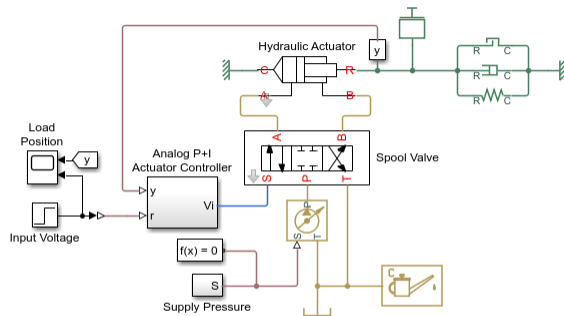
Examples

Convert Model to Use Isothermal Liquid Blocks

This example shows how to use the conversion tool on a model containing Hydraulic blocks. The tool generated a converted model and an HTML report. The next example, “Clean Up Model After Conversion” on page 2-4, shows how you can use the HTML report to review and fix the issues encountered by the conversion tool.

Open the Hydraulic Actuator with Analog Position Controller example model:

```
ssc_hydraulic_actuator_analog_control
```



Hydraulic Actuator with Analog Position Controller

1. [Plot pressures](#) in cylinder ([see code](#))
2. [Explore simulation results](#) using [sscexplore](#)
3. [Learn more](#) about this example

This example model contains blocks from Hydraulic libraries.

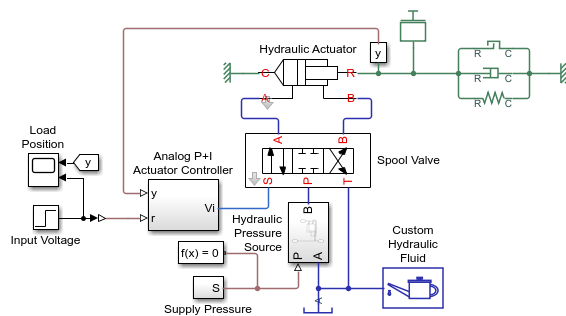
Convert the model to replace Hydraulic library blocks with the equivalent blocks from the Isothermal Liquid library:

```
hydraulicToIsothermalLiquid(bdroot)
```

ans =

```
'ssc_hydraulic_actuator_analog_control_converted'
```

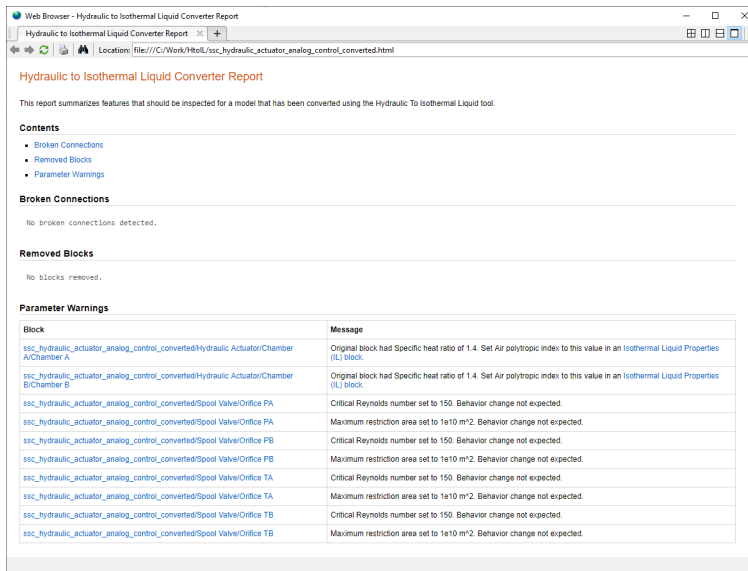
The conversion tool forms the new model name by appending `_converted` to the name of the original model. Because you did not specify the `newpath` argument, the conversion tool saves the `ssc_hydraulic_actuator_analog_control_converted` model in the current folder.



Hydraulic Actuator with Analog Position Controller

1. [Plot pressures](#) in cylinder ([see code](#))
2. [Explore simulation results](#) using [sscexplore](#)
3. [Learn more](#) about this example

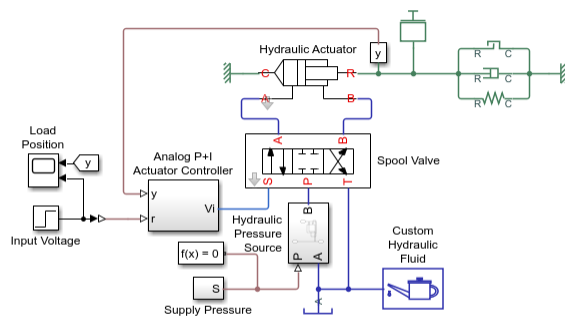
The tool also generates an HTML report and saves it in the current folder. The report lists any issues encountered during the conversion process.



Review the HTML report and manually fix the remaining issues. In this example, the converted model does not contain broken connections or removed blocks, but the conversion tool generated several parameter warnings that require your attention. For more information, see “Clean Up Model After Conversion” on page 2-4.

Clean Up Model After Conversion

In the previous example, “Convert Model to Use Isothermal Liquid Blocks” on page 2-2, you used the conversion tool on the `ssc_hydraulic_actuator_analog_control` model. The tool generated a converted model, `ssc_hydraulic_actuator_analog_control_converted`, and an HTML report. In this example, you will use the HTML report to review and fix the issues encountered by the conversion tool.



Hydraulic Actuator with Analog Position Controller

1. [Plot pressures](#) in cylinder (see code)
2. [Explore simulation results](#) using `sscexplore`
3. [Learn more](#) about this example

Web Browser - Hydraulic to Isothermal Liquid Converter Report

Hydraulic to Isothermal Liquid Converter Report

This report summarizes features that should be inspected for a model that has been converted using the Hydraulic To Isothermal Liquid tool.

Contents

- Broken Connections
- Removed Blocks
- Parameter Warnings

Broken Connections

No broken connections detected.

Removed Blocks

No blocks removed.

Parameter Warnings

Block	Message
ssc_hydraulic_actuator_analog_control_convertedHydraulic Actuator/Chamber A/Chamber A	Original block had Specific heat ratio of 1.4. Set Air polytropic index to this value in an Isothermal Liquid Properties (IL) block.
ssc_hydraulic_actuator_analog_control_convertedHydraulic Actuator/Chamber B/Chamber B	Original block had Specific heat ratio of 1.4. Set Air polytropic index to this value in an Isothermal Liquid Properties (IL) block.
ssc_hydraulic_actuator_analog_control_convertedSpool Valve/Orifice PA	Critical Reynolds number set to 150. Behavior change not expected.
ssc_hydraulic_actuator_analog_control_convertedSpool Valve/Orifice PA	Maximum restriction area set to 1e10 m ² . Behavior change not expected.
ssc_hydraulic_actuator_analog_control_convertedSpool Valve/Orifice PB	Critical Reynolds number set to 150. Behavior change not expected.
ssc_hydraulic_actuator_analog_control_convertedSpool Valve/Orifice PB	Maximum restriction area set to 1e10 m ² . Behavior change not expected.
ssc_hydraulic_actuator_analog_control_convertedSpool Valve/Orifice TA	Critical Reynolds number set to 150. Behavior change not expected.
ssc_hydraulic_actuator_analog_control_convertedSpool Valve/Orifice TA	Maximum restriction area set to 1e10 m ² . Behavior change not expected.
ssc_hydraulic_actuator_analog_control_convertedSpool Valve/Orifice TB	Critical Reynolds number set to 150. Behavior change not expected.
ssc_hydraulic_actuator_analog_control_convertedSpool Valve/Orifice TB	Maximum restriction area set to 1e10 m ² . Behavior change not expected.

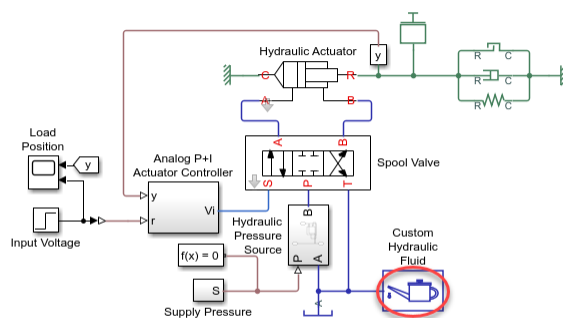
For a listing of conversion messages and suggested actions to address them, see “Conversion Messages”. For each message, clicking a link in the **Block** column of the HTML report opens the appropriate subsystem, as necessary, and highlights the block that generated the message.

The first two messages in the HTML report refer to the two chambers of the Hydraulic Actuator subsystem. In the original model, each of these chambers was implemented by a Translational Hydro-Mechanical Converter block. The conversion tool replaced each of these blocks with a Translational Mechanical Converter (IL) block.

The Translational Hydro-Mechanical Converter blocks in the original model had a **Specific heat ratio** parameter, but in the isothermal liquid domain, all of the fluid properties are defined in the Isothermal Liquid Properties (IL) block.

The conversion tool prints the value of the **Specific heat ratio** parameter in the original block for your convenience: Original block had Specific heat ratio of 1.4. Set Air polytropic index to this value in an Isothermal Liquid Properties (IL) block.

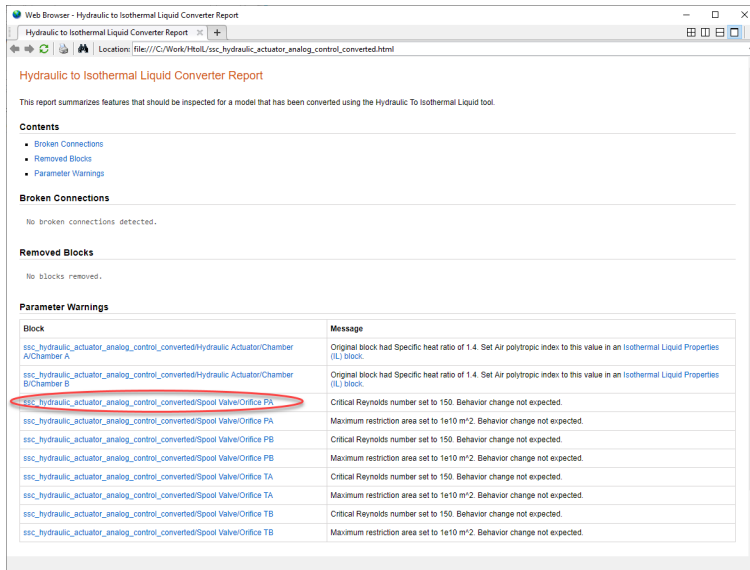
Open the Isothermal Liquid Properties (IL) block in the converted model and set its **Air polytropic index** parameter to 1.4.



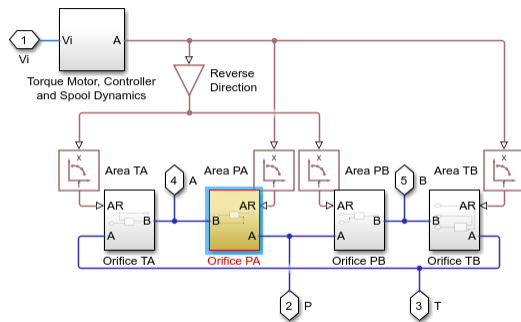
Hydraulic Actuator with Analog Position Controller

1. [Plot pressures](#) in cylinder ([see code](#))
2. [Explore simulation results](#) using [ssexplore](#)
3. [Learn more](#) about this example

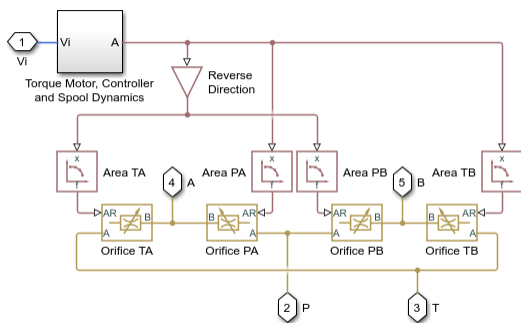
After thus addressing the first two conversion messages, click the third link in the **Block** column of the HTML report.



The tool looks under the mask of the Spool Valve block and highlights the Orifice PA subsystem.

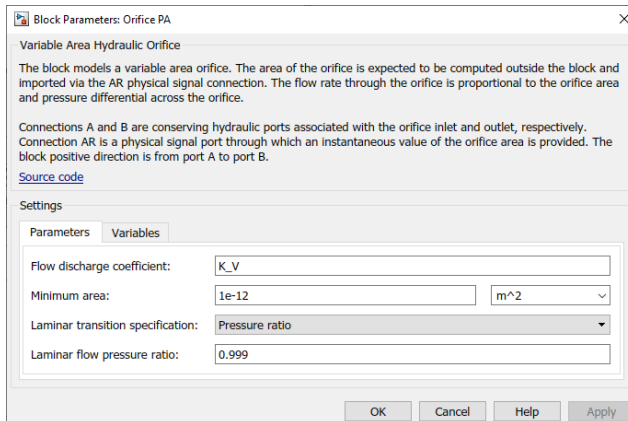


In the original model, `ssc_hydraulic_actuator_analog_control`, look under the mask of the Spool Valve block.



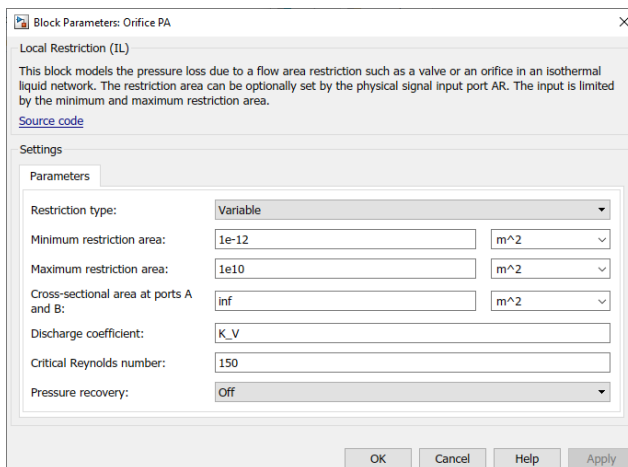
Orifice PA is one of four Variable Area Hydraulic Orifice blocks that comprise the valve, and each of these blocks has similar conversion messages. The manual cleanup process for each of these blocks is also similar.

Double-click the Orifice PA block in the original model to see its parameters.



The conversion tool replaced each Variable Area Hydraulic Orifice block with a Local Restriction (IL) block. (For more information, see “Upgrading Your Models”.) Because the port locations for these blocks are different, the conversion tool placed each of the Local Restriction (IL) blocks in a subsystem to preserve the diagram layout.

Double-click the Orifice PA subsystem in the converted model, and then the Orifice PA block inside it, to see its parameters.



The first conversion message for the Orifice PA block, **Critical Reynolds number** set to 150, stems from the fact that the original Variable Area Hydraulic Orifice block had the **Laminar transition specification** parameter set to **Pressure ratio**. The replacement Local Restriction (IL) block can specify the transition between the laminar and turbulent regime only by the critical Reynolds number, and the conversion tool sets the **Critical Reynolds number** parameter to its default value of 150. Because the Variable Area Hydraulic Orifice block in the original model used the default **Laminar flow pressure ratio** parameter value of 0.999, no action is necessary.

The second conversion message for the Orifice PA block, **Maximum restriction area** set to **1e10 m²**, stems from the fact that the original Variable Area Hydraulic Orifice block assumed an

infinitely large maximum opening area, while the replacement Local Restriction (IL) block, with **Restriction type** set to **Variable**, requires a **Maximum restriction area** parameter value less than `inf`.

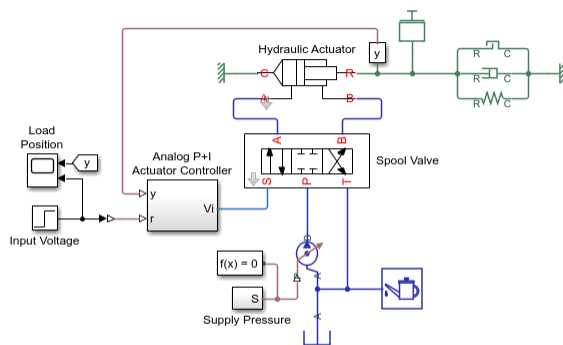
The conversion tool sets the **Maximum restriction area** parameter in the replacement Local Restriction (IL) block to an arbitrary large value, $1e10 \text{ m}^2$. No action is necessary, but you can adjust this value to match the data sheet for your model, if desired.

The rest of the conversion messages refer to the other three orifices and also say that the behavior change is not expected. You can inspect these three blocks in the original model in a similar way, to verify that no action is necessary.

At this point, after you addressed all the conversion messages, the required model cleanup is complete. You can compare the simulation results of the original and the converted model to make sure they are the same. You can also perform further cosmetic cleanup, if desired. For example, you can move the converted blocks that had changes to their port placement out of subsystems and manually reroute the connection lines.

To see an example of this model after conversion and cleanup, open the Hydraulic Actuator with Analog Position Controller example model in the Isothermal Liquid domain:

`ssc_il_actuator_analog_control`



Hydraulic Actuator with Analog Position Controller

1. [Plot pressures](#) in cylinder (see code)
2. [Explore simulation results](#) using `sscexplore`
3. [Learn more](#) about this example

Compare this model with the original model, `ssc_hydraulic_actuator_analog_control`, and the converted model generated by the tool, `ssc_hydraulic_actuator_analog_control_converted`.

Input Arguments

oldfile — Block diagram system name or handle

character vector | string scalar | handle

The name of the block diagram system to convert, specified as a character vector, string scalar, or a handle. `oldfile` can be a model, subsystem, or library. It must be either on the MATLAB path or loaded prior to using the conversion tool.

Example: `'HydraulicActuator'`

newpath — Location of converted block diagram system and report

character vector | string scalar

Location where the tool saves the converted block diagram system and the report, specified as an absolute or relative path name. For more information, see “Absolute and Relative Path Names”.

This argument is optional. If you omit the `newpath` argument, the tool saves the converted system and report in your current working folder.

Example: 'C:\Work'

Output Arguments**newfile — Name of converted block diagram system**

character vector

Name of the converted model, subsystem, or library, returned as a character vector. The conversion tool forms the new file name by appending `_converted` to the name of the original file, `oldfile`.

See Also

Interface (H-IL)

Topics

“Upgrading Hydraulic Models To Use Isothermal Liquid Blocks”

“Upgrading Simscape Fluids Models Containing Hydraulics (Isothermal) Blocks” (Simscape Fluids)

Introduced in R2020a

pm_adddimension

Adds new dimension to unit registry

Syntax

```
pm_adddimension(dimension,unitname)
```

Description

`pm_adddimension(dimension,unitname)` adds a new unit dimension with a fundamental unit, `unitname`.

Examples

Add Unit Dimension

Add a new unit dimension.

```
pm_adddimension('length','m')
```

The unit registry contains a new dimension, length, with a fundamental unit of meter, m.

Input Arguments

dimension — Name of dimension to add to the unit registry

character vector | string scalar

Name of dimension to add to the unit registry, specified as a character vector or string scalar. You can specify any name.

Data Types: char | string

unitname — Fundamental unit for new dimension

character vector | string scalar

Fundamental unit used for the new dimension, specified as a character vector or string scalar. The unit name must begin with a letter and contain only letters and numbers.

Data Types: char | string

See Also

`pm_addunit` | `pm_getdimensions` | `pm_getunits`

Topics

“Unit Definitions”

Introduced in R2007a

pm_addunit

Add new unit to unit registry

Syntax

```
pm_addunit(unitname, conversion, unitexpression)
```

Description

`pm_addunit(unitname, conversion, unitexpression)` introduces a new unit, `unitname`, defined as `conversion * unitexpression`.

The first argument, `unitname`, must be a valid unit name, that is, it must begin with a letter and contain only letters and numbers.

The second argument, `conversion`, may 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. For more information, see “Thermal Unit Conversions”.

The third argument, `unitexpression`, must be a valid unit expression in terms of units already defined in the unit registry.

The following operators are supported in the unit mathematical expressions:

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

Examples

Add a new unit centimeter, `cm`, in terms of meter, `m`:

```
pm_addunit('cm', 0.01, 'm');
```

Add a new unit newton, `N`, in terms of kilograms, meters, and seconds:

```
pm_addunit('N', 1, 'kg*m/s^2');
```

Add a new unit Fahrenheit, `degF`, in terms of Celsius:

```
pm_addunit('degF', [5/9 -32*5/9], 'degC');
```

See Also

`pm_adddimension` | `pm_getdimensions` | `pm_getunits`

Introduced in R2007a

pm_getdimensions

Get information about all dimensions in unit registry

Syntax

```
[dimensions, units] = pm_getdimensions
```

Description

[dimensions, units] = pm_getdimensions returns all dimensions registered in the unit registry in a cell array, dimensions. Their corresponding units are returned in the units cell array.

Examples

List all dimensions currently defined in the registry:

```
pm_getdimensions
```

```
ans =
```

```
    'charge'  
    'length'  
    'mass'  
    'mole'  
    'temperature'  
    'time'
```

See Also

pm_adddimension | pm_addunit | pm_getunits

Introduced in R2009a

pm_getunits

Get information about all units in unit registry

Syntax

```
[units, conversions, expressions] = pm_getunits
```

Description

`[units, conversions, expressions] = pm_getunits` returns all units in the registry in a cell array, `units`. Their corresponding conversions and base expressions are returned in `conversions` and `expressions`, respectively. For fundamental units, the conversion is 1.0 and the base expression is the unit itself.

Examples

List all units currently defined in the registry:

```
pm_getunits
```

```
ans =
```

```
110×1 cell array
```

```
{'m'      }
{'kg'     }
{'s'      }
{'C'      }
{'K'      }
{'mol'    }
{'cm'     }
{'mm'     }
{'km'     }
{'um'     }
{'degC'   }
{'degF'   }
{'degR'   }
{'deltaK' }
{'deltadegC'}
{'deltadegF'}
{'deltadegR'}
{'in'     }
{'ft'     }
{'mi'     }
{'yd'     }
{'l'      }
{'gal'    }
{'igal'   }
{'g'      }
{'mg'     }
{'t'      }
{'lbm'    }
```



```
{'oz' }
{'slug' }
{'N' }
{'lbf' }
{'dyn' }
{'lb' }
{'kN' }
{'mN' }
{'min' }
{'hr' }
{'ms' }
{'us' }
{'ns' }
{'rad' }
{'deg' }
{'rev' }
{'mph' }
{'fpm' }
{'fps' }
{'rpm' }
{'Hz' }
{'kHz' }
{'MHz' }
{'GHz' }
{'gee' }
{'J' }
{'kJ' }
{'Btu_IT' }
{'eV' }
{'W' }
{'HP_DIN' }
{'V' }
{'A' }
{'F' }
{'H' }
{'Ohm' }
{'S' }
{'Wb' }
{'T' }
{'G' }
{'mV' }
{'kV' }
{'pA' }
{'nA' }
{'uA' }
{'mA' }
{'kA' }
{'pF' }
{'nF' }
{'uF' }
{'mF' }
{'nH' }
{'uH' }
{'mH' }
{'kOhm' }
{'MOhm' }
{'GOhm' }
{'nS' }
```

```
{ 'uS'      }  
{ 'mS'      }  
{ 'uW'      }  
{ 'mW'      }  
{ 'kW'      }  
{ 'MW'      }  
{ 'lpm'     }  
{ 'gpm'     }  
{ 'P'       }  
{ 'cP'      }  
{ 'reyn'    }  
{ 'St'      }  
{ 'cSt'     }  
{ 'newt'    }  
{ 'Pa'      }  
{ 'bar'     }  
{ 'psi'     }  
{ 'ksi'     }  
{ 'atm'     }  
{ 'uPa'     }  
{ 'kPa'     }  
{ 'MPa'     }  
{ 'GPa'     }  
{ 'kbar'    }
```

See Also

`pm_adddimension` | `pm_addunit` | `pm_getdimensions`

Introduced in R2007a

simscape.dependency.file

Package: `simscape.dependency`

Check dependencies for single file

Syntax

```
[fn_list, missing] = simscape.dependency.file('fileName')
[fn_list, missing] = simscape.dependency.file('fileName', dependencyType)
[fn_list, missing] = simscape.dependency.file('fileName', dependencyType,
isRecursive)
[fn_list, missing] = simscape.dependency.file('fileName', dependencyType,
isRecursive, doTMWFile)
```

Description

`[fn_list, missing] = simscape.dependency.file('fileName')` returns two cell arrays of character vectors: full path names of existing dependency files, `fn_list`, and missing files, `missing`. These cell arrays list the existing and missing files that are needed for the specified Simscape file to build successfully, or to correctly visualize and execute in MATLAB.

`[fn_list, missing] = simscape.dependency.file('fileName', dependencyType)` returns dependency files of the specified type.

`[fn_list, missing] = simscape.dependency.file('fileName', dependencyType, isRecursive)` lets you specify whether analysis is recursive on the generated dependency files. By default, returns only the top-level dependency files.

`[fn_list, missing] = simscape.dependency.file('fileName', dependencyType, isRecursive, doTMWFile)` lets you specify whether to include files inside the MATLAB root folder (installation directory) in the analysis.

Input Arguments

dependencyType

Enumerated value of type `simscape.DependencyType`, which specifies the type of returned files:

All (default)	All the dependency files
Auxiliary	Files that are not necessary to convert the file and use it in block diagrams, but are needed to visualize it correctly, for example, block icon images
Core	Files necessary to convert the file and use it in block diagrams, for example, a domain file referenced by the component file being analyzed

Derived	Internally generated files that are not necessary for sharing the component file being analyzed, but including them will avoid rebuilding the library on the same platform.
Simulink	Additional files that help visualize the block generated from the component file being analyzed. These files are not necessary for simulation.

These enumerated values have the following order: `Core`, `Derived`, `Auxiliary`, `Simulink`, `All`. The return is accumulative. This means that for a requested file type, all earlier file types are also returned. For example, if you specify *dependencyType* as `simscape.DependencyType.Derived`, the analysis returns both `Core` and `Derived` files.

doTMWFile

Logical value that indicates whether the file analysis includes files inside the MATLAB root folder (installation directory):

`true` (default)
`false`

fileName

The name of the Simscape file (with path), or class method, for which the dependencies are checked. In case of multiple files with the same name, only the first file of the specified name on the MATLAB path is analyzed.

isRecursive

Logical value that indicates whether the analysis is recursive on the generated dependency files:

`true`
`false` (default)

See Also

`simscape.dependency.lib` | `simscape.dependency.model`

Topics

“Checking File and Model Dependencies”

Introduced in R2009b

simscape.dependency.lib

Package: `simscape.dependency`

Check dependencies for library package

Syntax

```
[fn_list, missing] = simscape.dependency.lib('libName')
[fn_list, missing] = simscape.dependency.lib('libName', dependencyType)
[fn_list, missing] = simscape.dependency.lib('libName', dependencyType,
'mdlFileName')
[fn_list, missing] = simscape.dependency.lib('libName', dependencyType,
'mdlFileName', isRecursive)
[fn_list, missing] = simscape.dependency.lib('libName', dependencyType,
'mdlFileName', isRecursive, doTMWFile)
```

Description

`[fn_list, missing] = simscape.dependency.lib('libName')` returns two cell arrays of character vectors: full path names of existing dependency files, `fn_list`, and missing files, `missing`. These cell arrays list the existing and missing files that are needed for the specified Simscape library package to build successfully, or to correctly visualize and execute in MATLAB.

`[fn_list, missing] = simscape.dependency.lib('libName', dependencyType)` returns dependency files of the specified type.

`[fn_list, missing] = simscape.dependency.lib('libName', dependencyType, 'mdlFileName')` lets you specify the name of the library model. When not specified, or specified as an empty character vector (''), `libName_lib` is used.

`[fn_list, missing] = simscape.dependency.lib('libName', dependencyType, 'mdlFileName', isRecursive)` lets you specify whether analysis is recursive on the generated dependency files. By default, returns only the top-level dependency files.

`[fn_list, missing] = simscape.dependency.lib('libName', dependencyType, 'mdlFileName', isRecursive, doTMWFile)` lets you specify whether to include files inside the MATLAB root folder (installation directory) in the analysis.

If the package contains Simscape protected files, with the corresponding Simscape source files in the same folder, the analysis returns the names of protected files and then analyzes the source files for further dependencies. If the package contains Simscape protected files without the corresponding source files, the protected file names are returned without further analysis.

Input Arguments

dependencyType

Enumerated value of type `simscape.DependencyType`, which specifies the type of returned files:

All (default)	All the dependency files
Auxiliary	Files that are not necessary to build the library, or run the models built from its blocks, but are needed to visualize it correctly, for example, block icon images or <code>lib.m</code> files.
Core	Files necessary to build the library or run the models built from its blocks, such as Simscape files or MATLAB files.
Derived	Internally generated files that are not necessary for sharing the library, but including them will avoid rebuilding the library on the same platform.
Simulink	Additional files that help visualize the blocks generated from the library components. These files are not necessary for simulation.

These enumerated values have the following order: `Core`, `Derived`, `Auxiliary`, `Simulink`, `All`. The return is accumulative. This means that for a requested file type, all earlier file types are also returned. For example, if you specify `dependencyType` as `simscape.DependencyType.Derived`, the analysis returns both `Core` and `Derived` files.

doTMWFile

Logical value that indicates whether the file analysis includes files inside the MATLAB root folder (installation directory):
`true` (default)
`false`

isRecursive

Logical value that indicates whether the analysis is recursive on the generated dependency files:
`true`
`false` (default)

libName

The name of a Simscape library package. The package folder name begins with a leading `+` character, whereas the argument to `simscape.dependency.lib` must omit the `+` character. You must run the command from the folder containing the top-level package, or from inside the package folder. In the latter case, you can omit the name of the library package if it is the only argument.

mdlFileName

The name of the library model (either without path, or with relative path, or with absolute path). The model file extension (`.slx` or `.mdl`) is optional.

Default: `libName_lib`

See Also

`simscape.dependency.file` | `simscape.dependency.model`

Topics

“Checking File and Model Dependencies”

Introduced in R2009b

simscape.dependency.model

Package: `simscape.dependency`

Check dependencies for model

Syntax

```
[fn_list, missing, reference2fnList, reference2missing] =  
simscape.dependency.model('modelName')
```

Description

[*fn_list*, *missing*, *reference2fnList*, *reference2missing*] = `simscape.dependency.model('modelName')` checks dependencies for a model containing Simscape and Simulink blocks. *modelName* specifies the name of the model (either without path, or with relative path, or with absolute path). The model file extension (`.slx` or `.mdl`) is optional.

You must open the model first.

This command returns dependency information regarding Simscape files and blocks only. To perform a complete dependencies check for a model, use the Dependency Analyzer. For more information, see “Analyze Model Dependencies”.

If during the analysis this command encounters a Simscape file located inside the MATLAB root folder, it returns the file name without performing any further analysis on this file, because all the dependent files in this case are part of standard MathWorks installation.

Output Arguments

fn_list

A cell array of character vectors containing the full paths of all existing files referenced by the model *modelName*.

missing

A cell array of character vectors containing the names of all files that are referenced by the model *modelName* but cannot be found.

reference2fnList

A list of structures, each of which includes a field 'names' as a list of file names causing the reference, and a field 'type' as the reference type for each file. Two reference types are used: 'Simscape component' indicates reference from a model block. 'Simscape' indicates reference from a file.

reference2missing

A list of structures, each of which includes a field 'names' as a list of missing file names, and a field 'type' as the reference type for each file. Two reference types are used: 'Simscape component' indicates reference from a model block. 'Simscape' indicates reference from a file.

See Also

`simscape.dependency.file` | `simscape.dependency.lib`

Topics

“Checking File and Model Dependencies”

Introduced in R2009b

simscape.findNonlinearBlocks

Check model for blocks with nonlinear equations

Syntax

```
blockList = simscape.findNonlinearBlocks(modelName)
```

Description

`blockList = simscape.findNonlinearBlocks(modelName)` checks a model and reports which blocks, if any, contain nonlinear equations that keep the physical networks in the model from being linear or switched linear. `modelName` is the name of the model, in single quotes.

You do not have to open the model before using the function.

The function reports how many physical networks in the model are linear or switched linear, how many networks are nonlinear because of the nonlinear blocks, and which blocks have nonlinear equations. It returns a cell array of the block names. If none of the blocks have nonlinear equations, the cell array is empty.

Examples

Check Model for Nonlinear Blocks

Check the Nonlinear Bipolar Transistor example model for blocks with nonlinear equations. You do not have to open the model before running the diagnostic. This model is on the MATLAB® path, therefore you do not have to include the full path into the model name.

```
nonlinBlocks = simscape.findNonlinearBlocks('ssc_bipolar_nonlinear')
```

Found network that contains nonlinear equations in the following blocks:

```
{'ssc_bipolar_nonlinear/AC Voltage 1kHz/10mV,1kHz'      }
{'ssc_bipolar_nonlinear/Nonlinear NPN Transistor/D1/exp(x)'}
{'ssc_bipolar_nonlinear/Nonlinear NPN Transistor/D2/exp(x)'}
```

The number of linear or switched linear networks in the model is 0.
The number of nonlinear networks in the model is 1.

```
nonlinBlocks = 3x1 cell
{'ssc_bipolar_nonlinear/AC Voltage 1kHz/10mV,1kHz'      }
{'ssc_bipolar_nonlinear/Nonlinear NPN Transistor/D1/exp(x)'}
{'ssc_bipolar_nonlinear/Nonlinear NPN Transistor/D2/exp(x)'}
```

The diagnostic function found no networks that are linear or switched linear. It found one network that contains nonlinear equations in three blocks. The function displays the names of the blocks in the diagnostic message.

The function returns a cell array, `nonlinBlocks`, containing the names of the blocks with nonlinear equations. The number of rows in the array corresponds to the number of blocks found. Each cell contains the name of a block, including the full path to the block from the root of the model.

Input Arguments

modelName — Model name

character vector | string scalar

Model name, specified as a character vector or string scalar. If the model is not on the MATLAB path, the model name must include the full path.

Data Types: char | string

Output Arguments

blockList — Names of blocks with nonlinear equations

cell array

Names of blocks with nonlinear equations, returned as an n -by-1 cell array. n is the number of blocks in the model that have nonlinear equations.

Each cell contains the name of a block, including the full path to the block from the root of the model.

Limitations

- This function does not work with Simscape Multibody networks.

Introduced in R2017a

simscape.logging.export

Save logged simulation data in MLDATX file

Syntax

```
simscape.logging.export(simlog, fileName)
```

Description

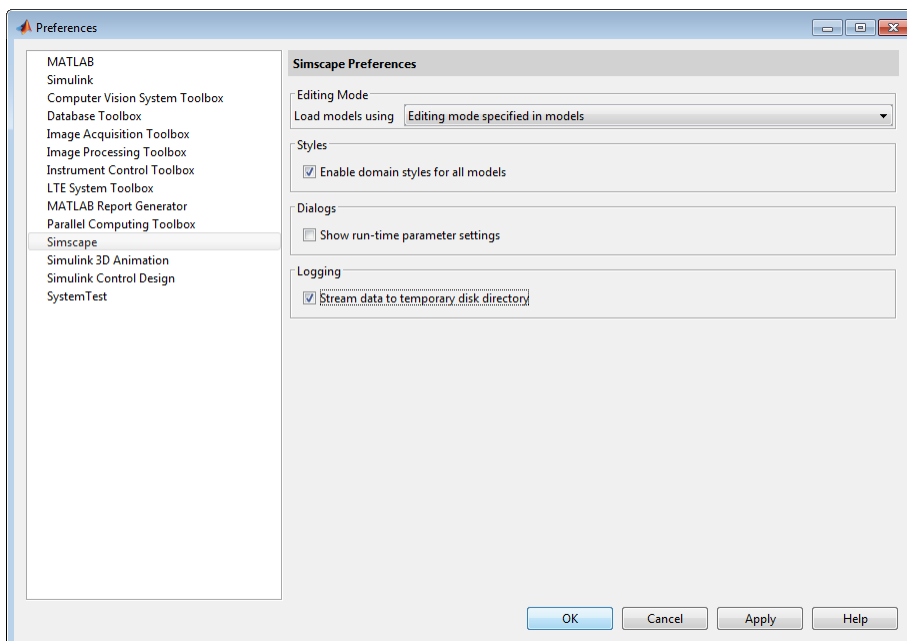
`simscape.logging.export(simlog, fileName)` saves the `simlog` object, containing logged simulation data, for future use. You can use this function only for data logged with the **Stream data to temporary disk directory** preference turned on.

When you stream simulation data to disk, the data is stored as a `simlog` object in a temporary file, and the workspace logging variable references the `simlog` object. The temporary file persists as long as there is a logging variable in the workspace that references the file. This function lets you save the `simlog` object to a different file, specified by the `fileName` argument, in MLDATX format.

Examples

Save Logged Simulation Data Streamed to Disk

To enable streaming data to disk, on the MATLAB Toolstrip, click **Preferences**. In the left pane of the Preferences dialog box, select **Simscape**, then select the **Stream data to temporary disk directory** check box.



Open the Permanent Magnet DC Motor example model, which already has data logging enabled, and run the simulation:

```
ssc_dcmotor
sim('ssc_dcmotor');
```

During the simulation, logged data is streamed to disk, to a temporary MLDATX file. After the simulation, you see the simulation log variable `simlog_ssc_dcmotor` (as specified by the **Workspace variable name** model configuration parameter) in your current workspace:

```
simlog_ssc_dcmotor
simlog_ssc_dcmotor =
    Node with properties:
        id: 'ssc_dcmotor'
        savable: 0
        exportable: 1
        MRRef_Torque: [1x1 simscape.logging.Node]
        Load_Torque: [1x1 simscape.logging.Node]
        DC_Voltage: [1x1 simscape.logging.Node]
        DC_Motor: [1x1 simscape.logging.Node]
        ERef: [1x1 simscape.logging.Node]
        Sensing: [1x1 simscape.logging.Node]
        MRRef_Motor: [1x1 simscape.logging.Node]
```

The `exportable: 1` property of the `simlog_ssc_dcmotor` variable indicates that this variable points to the temporary file on disk, which contains the simulation data. The temporary file exists as long as the variable exists in your workspace, then it is deleted.

To save the logged simulation data for future use, type:

```
simscape.logging.export(simlog_ssc_dcmotor, 'C:\Work\motor_run1');
```

This command creates a file under `C:\Work`, named `motor_run1.mldatx`, and stores the logged simulation data in this file, in MLDATX format.

To retrieve the logged simulation data at a later time and associate it with a workspace variable, use the `simscape.logging.import` function.

Input Arguments

simlog — Logged simulation data

Node object

Logged simulation data, specified as a `Node` object, with the `exportable` property set to 1. You refer to the `simlog` object by the name of the corresponding simulation log workspace variable. You specify the name of the simulation log variable by using the **Workspace variable name** parameter in the **Simscape** pane of the Configuration Parameters dialog box.

fileName — File name and path

character vector | string scalar

File name and path, specified as a character vector or string scalar. The function stores the `simlog` object in the specified file, in MLDATX or HDF5 format. If you omit the file extension, the data is

stored in an MLDATX file. To save the `simlog` object in HDF5 format, the `fileName` character vector must include the `.h5` extension. If you specify a file extension other than `.h5` or `.mldatx`, you get an error. If you do not include the path, the file resides in the current working directory.

If the file already exists, the function overwrites it without a warning. However, if you import a node from a file, and then try to export it to the same file, a message informs you that in this case the file cannot be overwritten.

Example: `'C:\Work\motor_run1.mldatx'`

Data Types: `char` | `string`

Compatibility Considerations

File format when streaming logged simulation data to disk has changed from HDF5 to MLDATX

Behavior changed in R2020b

In R2020b, the file format for streaming logged simulation data to disk has changed from HDF5 to MLDATX. As a result, the default format for `simscape.logging.export` and `simscape.logging.import` has also changed to MLDATX. However, you can still export and import HDF5 files:

- When you use `simscape.logging.export` without specifying the file extension, the data is stored in an MLDATX file in the temporary directory.
- Currently, you can still export data in HDF5 format, to share it with colleagues who are using older software releases. To do this, specify `.h5` file extension when using `simscape.logging.export`. The function stores the data in HDF5 format and issues a warning that this functionality will be removed in a future release. If you specify a file extension other than `.h5` or `.mldatx`, you get an error.
- When you import an HDF5 file from a previous release using `simscape.logging.import`, the data is automatically converted to the new format, loaded into the Node object, and copied to a binary MLDATX file in the temporary directory. If there are two files with the same name and different extensions (`.h5` and `.mldatx`), then the MLDATX file is loaded.

See Also

`simscape.logging.import`

Topics

“About Simulation Data Logging”

“Stream Logging Data to Disk”

Introduced in R2016a

simscape.logging.import

Create simulation log variable to access data in MLDATX file

Syntax

```
var = simscape.logging.import(fileName)
```

Description

`var = simscape.logging.import(fileName)` creates a workspace variable `var`, of type `simscape.logging.Node`, which references the `simlog` object in the specified MLDATX file.

You can use this function to view and analyze simulation data that was logged with the **Stream data to temporary disk directory** preference turned on.

When you stream simulation data to disk, you can save the `simlog` object as an MLDATX file by using the `simscape.logging.export` function. The `simscape.logging.import` function lets you retrieve that data at a later time, for example, when the model is not in memory. The function associates data in the file with the workspace variable `var`, which you can use to access the logged simulation data. If you do not assign a variable name when calling the function, then the workspace variable name is `ans`.

Examples

View Simulation Data Stored in File

This example shows how you can view and analyze logged simulation data, previously saved in an MLDATX file. It builds up on the `simscape.logging.export` example, which shows how to save logged simulation data, streamed to disk, in a file named `C:\Work\motor_run1.mldatx`.

To retrieve that data at a later time, even when the model is not in memory, type:

```
run1 = simscape.logging.import('C:\Work\motor_run1')
```

```
run1 =
```

```
Node with properties:
```

```

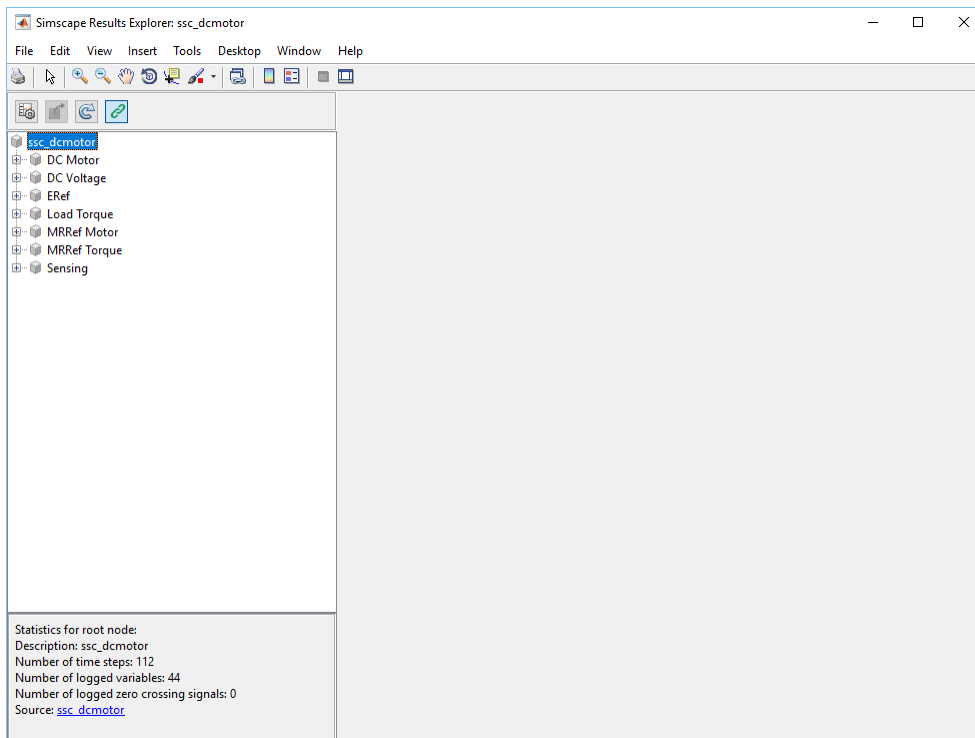
        id: 'ssc_dcmotor'
    savable: 0
    exportable: 1
    MRRef_Torque: [1x1 simscape.logging.Node]
    Load_Torque: [1x1 simscape.logging.Node]
    DC_Voltage: [1x1 simscape.logging.Node]
    DC_Motor: [1x1 simscape.logging.Node]
        ERef: [1x1 simscape.logging.Node]
    Sensing: [1x1 simscape.logging.Node]
    MRRef_Motor: [1x1 simscape.logging.Node]
```

Variable `run1`, of type `Node`, appears in your current workspace. Its properties are identical to the properties of the simulation log variable `simlog_ssc_dcmotor`, which was created as a result of simulating the `ssc_dcmotor` model and logging data to disk.

Explore the simulation data:

```
sscexplore(run1)
```

A new Simscape Results Explorer window opens. It contains logged simulation data, previously saved to disk. The root node, `ssc_dcmotor`, is selected in the left pane by default. As you expand and select nodes in the left pane, the corresponding plots appear in the right pane.



Input Arguments

fileName — File name and path

character vector | string scalar

File name and path, specified as a character vector or string scalar. The file must be in MLDATX or HDF5 format and contain logged simulation data, specified as a `Node` object. If you omit the extension, and there are two files with the same name and different extensions (`.h5` and `.mldatx`), then the MLDATX file is loaded. If you specify an extension other than `.mldatx` or `.h5`, or try to import an MLDATX or HDF5 file that contains some other type of data, you get an error message.

Example: `'C:\Work\motor_run1.mldatx'`

Data Types: `char` | `string`

Compatibility Considerations

File format when streaming logged simulation data to disk has changed from HDF5 to MLDATX

Behavior changed in R2020b

In R2020b, the file format for streaming logged simulation data to disk has changed from HDF5 to MLDATX. As a result, the default format for `simscape.logging.import` and `simscape.logging.export` has also changed to MLDATX. However, you can still import and export HDF5 files:

- When you import an HDF5 file from a previous release using `simscape.logging.import`, the data is automatically converted to the new format, loaded into the `Node` object, and copied to a binary MLDATX file in the temporary directory. If there are two files with the same name and different extensions (`.h5` and `.mldatx`), then the MLDATX file is loaded.
- When you use `simscape.logging.export` without specifying the file extension, the data is stored in an MLDATX file in the temporary directory.
- Currently, you can still export data in HDF5 format, to share it with colleagues who are using older software releases. To do this, specify `.h5` file extension when using `simscape.logging.export`. The function stores the data in HDF5 format and issues a warning that this functionality will be removed in a future release. If you specify a file extension other than `.h5` or `.mldatx`, you get an error.

See Also

`simscape.logging.export` | `sscexplore`

Topics

“About Simulation Data Logging”

“Stream Logging Data to Disk”

“About the Simscape Results Explorer”

Introduced in R2016a

simscape.logging.plot

Package: `simscape.logging`

Plot logged simulation data for Node or Series

Syntax

```
fh = simscape.logging.plot(logobject)
fh = simscape.logging.plot(logobject,Name,Value)
```

Description

`fh = simscape.logging.plot(logobject)` plots the simulation series values along the *y*-axis, with time along the *x*-axis. `logobject` is a `simscape.logging.Node` or `simscape.logging.Series` object, or a homogeneous cell array of such objects. If `logobject` is a node, the function plots all nonempty series associated with the specified node and its children. Depending on the type of `logobj`, `fh` is a structure (for a node) or a cell array (for a series) of handles to the resulting figures.

`fh = simscape.logging.plot(logobject,Name,Value)` lets you filter the data being plotted by using one or more `Name,Value` pair arguments. For example, specify 'units' followed by a unit name to plot only nodes and series that are commensurate with the specified unit.

Examples

Plot Simulation Data

Plot velocity of port **R** of a Translational Spring block.

Open the Mass-Spring-Damper with Controller example model:

```
ssc_mass_spring_damper_control
```

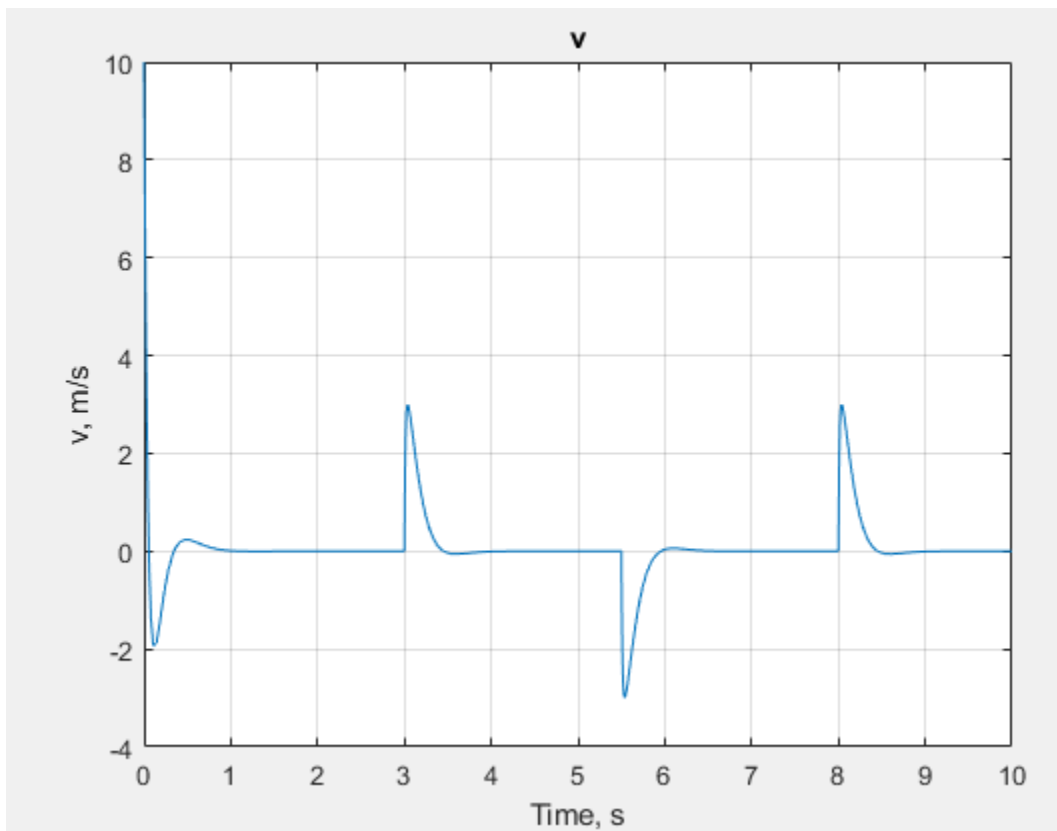
This example model has data logging enabled for the whole model, with the **Workspace variable name** parameter set to `simlog_ssc_mass_spring_damper_control`.

Simulate the model to log the simulation data:

```
sim('ssc_mass_spring_damper_control');
```

Plot velocity of port **R** of the Translational Spring block `Spring`.

```
simscape.logging.plot(simlog_ssc_mass_spring_damper_control.Spring.R);
```



Filter Data to Plot

Use the name-value pair arguments to filter simulation data being plotted.

For a model with the default workspace variable name, `simlog`, plot only linear positions and velocities (series that are commensurate with units of `mm` and `mm/s`), in those units, for the top-level model node, its children and their children, within the time range between 1 and 3 seconds:

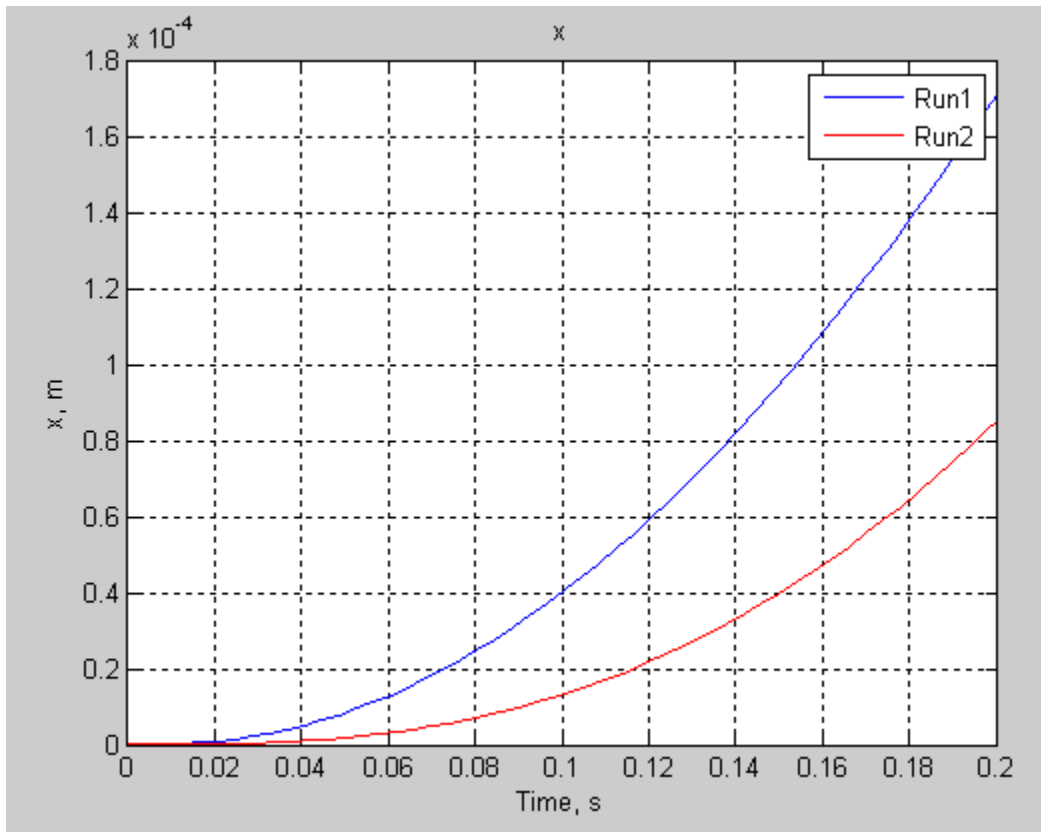
```
fh = simscape.logging.plot(simlog, 'units', {'mm', 'mm/s'}, 'time', [1 3], 'depth', 2);
```

Compare Data from Two Simulation Runs

Use the workspace variable name `simlog1` to log the data from the first run, and the workspace variable name `simlog2` to log the data from the second run.

Plot deformation of the Translational Spring block `TS` from both runs on the same axis, with the corresponding legend.

```
simscape.logging.plot({simlog1.TS.x simlog2.TS.x}, 'names', {'Run1' 'Run2'});
```



Input Arguments

Logobject — Simulation data to plot

scalar Series object | nonscalar Series object | cell array of Series objects | Node object | cell array of Node objects

Simulation data to plot, specified as a `simscape.logging.Node` or `simscape.logging.Series` object, or a homogeneous cell array of such objects. `logobject` must include a full identifier path to the node or series, starting with the workspace log variable name.

The table describes the resulting plots based on the type of the `logobject` argument:

Scalar Series object	Plots the simulation series values along the <i>y</i> -axis, with time along the <i>x</i> -axis.
Nonscalar Series object	Plots each dimension of the series values on a different axis in the same figure window.
Cell array of Series objects	Plots all series objects with commensurate units on the same axis (superimposed), and each dimension for a nonscalar series on a different axis in the same figure window.

The input arguments are binned based on commensurate units. For each bin, all Series objects with the same dimension as the first Series object in that bin are plotted and others are ignored.

Node object Plots all nonempty series associated with the **Node** and its children (up to the level defined by the **depth**). If the **Node** has multiple simulation variable nodes as children at level 1, these children are plotted in the same figure window but on a different axis. Descendants at other levels are plotted in different figure windows. All dimensions of a nonscalar series are plotted on the same axis.

Cell array of **Node** objects Plots commensurate series superimposed on the same axis.

Intended for use to compare simulation data from different runs. All entries of the cell array are required to be equivalent to each other, meaning that the **Node** objects must have same hierarchy, and the series for each node must have the same dimensions and commensurate units.

Name-Value Pair Arguments

Specify optional comma-separated pairs of **Name**, **Value** arguments. **Name** is the argument name and **Value** is the corresponding value. **Name** must appear inside quotes. You can specify several name and value pair arguments in any order as **Name1**, **Value1**, . . . , **NameN**, **ValueN**.

Example: `fh = simscape.logging.plot(simlog, 'units', 'mm', 'time', [1 3])` plots all of the linear position variables in the model (series that are commensurate with units of mm), in those units, within the time range between 1 and 3 seconds.

depth — Number of children levels

`intmax` (default) | nonnegative integer

Number of children levels to plot for a **Node** object, specified as the comma-separated pair consisting of **'depth'** and a nonnegative integer. By default, the function plots all descendants of the **Node** object that have nonempty series. Specifying **depth** lets you limit the number of levels to plot, for example:

'depth' , 0	No children; plot the nonempty series of the specified node only.
'depth' , 1	Plot the nonempty series of the specified node and its children.
'depth' , 2	Plot the nonempty series of the specified node, its children, and their children.

If **logobject** is a **Series** object, this argument is ignored.

Example: `simscape.logging.plot(simlog.Translational_Spring, 'depth', 1)` plots all of the variables associated with the **Translational Spring** block, but not with its ports.

names — Plot legend

cell array of character vectors or string scalars

Plot legend, specified as the comma-separated pair consisting of **'names'** and a cell array of character vectors or string scalars. The number of elements in the cell array must be same as the number of elements of **logobject**.

By default, plots have no legend.

time — Time range for plotting data

`[]` (default) | 1x2 vector, [*start_time* *end_time*] in seconds

Time range for plotting the data, specified as the comma-separated pair consisting of **'time'** and a 1x2 vector, [*start_time* *end_time*], in seconds.

[] plots all data.

units — Units for plotting data

character vector | string scalar | cell array of character vectors or string scalars

Units for plotting the data, specified as the comma-separated pair consisting of 'units' and a unit name, or a cell array of unit names. Unit names must appear inside single quotes (' ') or double quotes (" ").

This argument plots the series values in the specified units, and also filters the data to plot only nodes and series that are commensurate with the specified unit.

Example: `fh = Simscape.Logging.plot(simlog, 'units', {'mm', 'mm/s'})` plots all of the linear position and velocity variables in the model (series that are commensurate with units of mm and mm/s), in those units.

viewer — Plot data in the Simulation Data Inspector

'datainspector' | "datainspector"

Alternative destination to plot the data for a Node object, specified as the comma-separated pair consisting of 'viewer' and datainspector inside single quotes (' ') or double quotes (" ").

If you specify this name-value pair argument, the function plots the data in the Simulation Data Inspector. By default, the function plots the data in a plot window.

If logobject is a Series object, this argument is ignored.

Output Arguments

fh — Handles to the resulting plot figure windows

structure | cell array

Handles to the resulting plot figure windows, returned as a structure or a cell array, depending on the type of logobject:

- If logobject is a Node, fh is a structure with the same hierarchy as the object being plotted. For example, if a specific child is not plotted then that field in the output structure is empty.
- If logobject is a Series, fh is a cell array.

See Also

`Simscape.Logging.Node` | `Simscape.Logging.Series` | `Simscape.Logging.plotxy`

Topics

"Log and Plot Simulation Data"

Introduced in R2010b

simscape.logging.plotxy

Package: `simscape.logging`

Plot logged simulation data for one node or series against another

Syntax

```
fh = simscape.logging.plotxy(x,y)
fh = simscape.logging.plotxy(x,y,Name,Value)
```

Description

`fh = simscape.logging.plotxy(x,y)` plots the simulation series values of object `y` along the `y`-axis, with series values of object `x` along the `x`-axis. `x` and `y` can be `simscape.logging.Series` objects, `simscape.logging.Node` objects, or homogeneous cell arrays of such objects. If `x` or `y` is a node, it must be a simulation variable node (one that has a direct child series). The values of this child series are then plotted along the respective axis.

If `x` and `y` are cell arrays, they must be of the same size, or one of them can be a scalar. `x` and `y` must have the same time vectors.

`fh` is a cell array of figure handles, one for each `y` versus `x` plot generated.

`fh = simscape.logging.plotxy(x,y,Name,Value)` lets you customize the plot by using one or more `Name,Value` pair arguments. For example, specify `'time'` followed by a 1x2 vector, `[start_time end_time]`, to plot only the data within this time range.

Examples

Plot Motor Torque Against Its Angular Velocity

Plot the motor torque against its angular velocity, in default units.

Open the Permanent Magnet DC Motor example model:

```
ssc_dcmotor
```

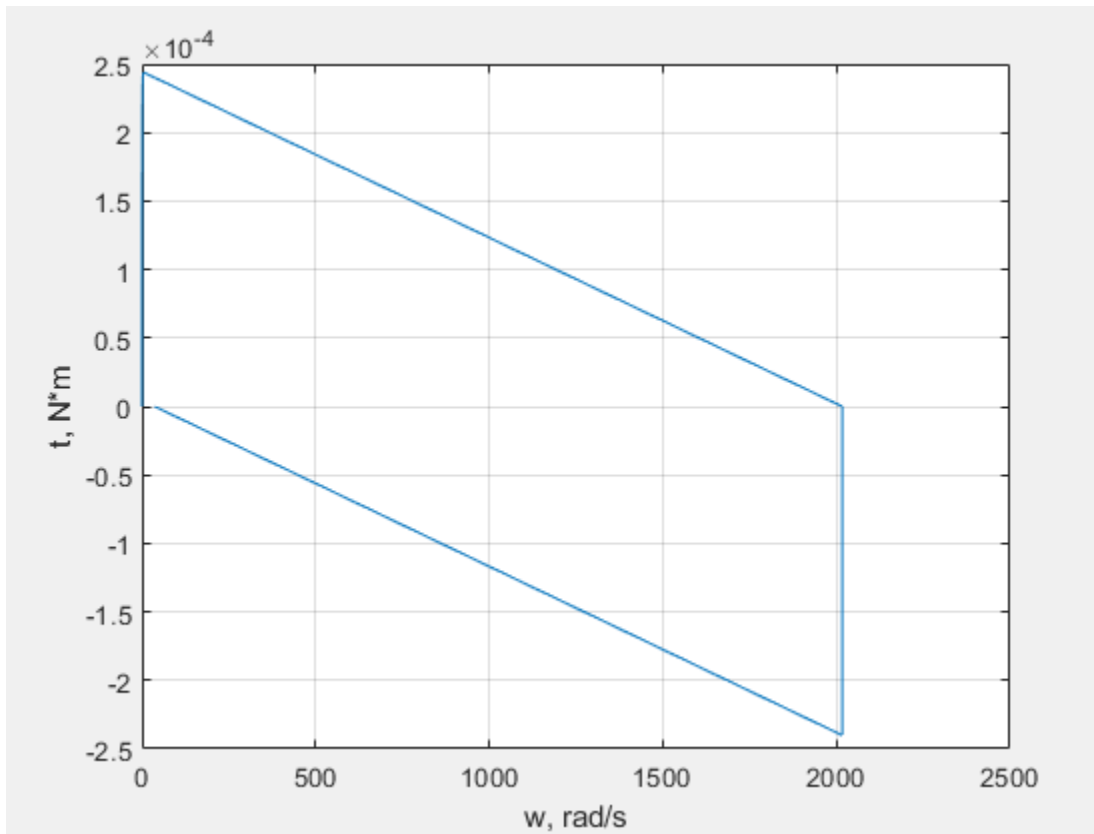
This example model has data logging enabled for the whole model, with the **Workspace variable name** parameter set to `simlog_ssc_dcmotor`.

Simulate the model to log the simulation data:

```
sim('ssc_dcmotor');
```

Plot the motor torque against its angular velocity:

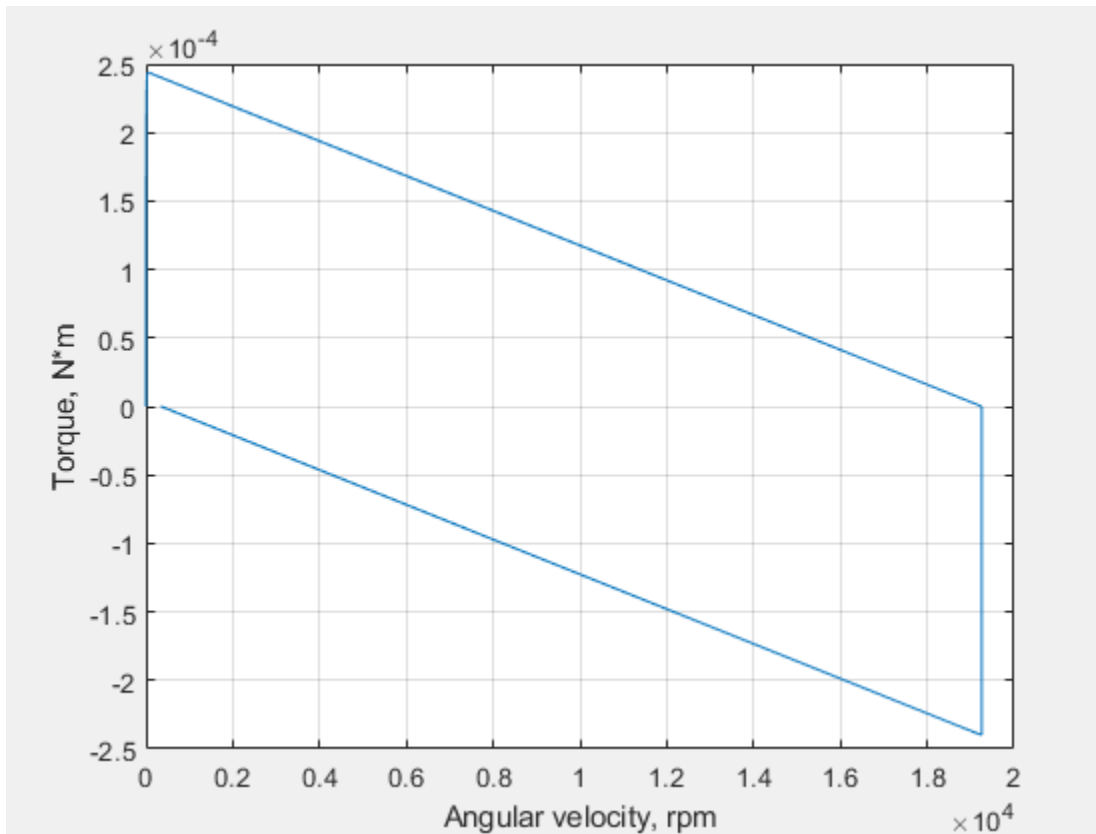
```
simscape.logging.plotxy(simlog_ssc_dcmotor.DC_Motor.Rotational_Electromechanical_Converter.R.w,
    simlog_ssc_dcmotor.DC_Motor.Inertia.t)
```



Customize the Motor Torque-Velocity Plot

Plot the motor torque, in default units, against its angular velocity, in rpm, and add axis names.

```
simscape.logging.plotxy(simlog_ssc_dcmotor.DC_Motor.Rotational_Electromechanical_Converter.R.w, .  
    simlog_ssc_dcmotor.DC_Motor.Inertia.t, 'xunit', 'rpm', 'xname', 'Angular velocity', 'yname', 'Torque
```

Input Arguments

x — Simulation data to plot along x-axis

scalar `Series` object | nonscalar `Series` object | cell array of `Series` objects | `Node` object | cell array of `Node` objects

Simulation data to plot along the x-axis, specified as a `simscape.logging.Series` object, `simscape.logging.Node` object, or a homogeneous cell array of such objects. x must include a full identifier path to the node or series, starting with the workspace log variable name.

If x is a node, it must be a simulation variable node (one that has a direct child series). The values of this child series are then plotted along the x-axis.

If x and y are cell arrays, they must be of the same size, or one of them can be a scalar. x and y must have the same time vectors.

y — Simulation data to plot along y-axis

scalar `Series` object | nonscalar `Series` object | cell array of `Series` objects | `Node` object | cell array of `Node` objects

Simulation data to plot along the y-axis, specified as a `simscape.logging.Series` object, `simscape.logging.Node` object, or a homogeneous cell array of such objects. y must include a full identifier path to the node or series, starting with the workspace log variable name.

If y is a node, it must be a simulation variable node (one that has a direct child series). The values of this child series are then plotted along the y-axis.

If `x` and `y` are cell arrays, they must be of the same size, or one of them can be a scalar. `x` and `y` must have the same time vectors.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1, Value1, ..., NameN, ValueN`.

Example: `fh = simscape.logging.plotxy(simlog.TS.C.v.series, simlog.TS.R.v.series, 'xunit', 'mm/s', 'yunit', 'mm/s')` plots velocities of ports **C** and **R** of the Translational Spring block **TS** against each other, in mm/s.

time — Time range for plotting data

`[]` (default) | `1x2` vector, `[start_time end_time]` in seconds

Time range for plotting the data, specified as the comma-separated pair consisting of `'time'` and a `1x2` vector, `[start_time end_time]`, in seconds.

`[]` plots all data.

xname — Name of x-axis

character vector | string scalar | cell array of character vectors or string scalars

Name of the plot x-axis, specified as the comma-separated pair consisting of `'xname'` and a character vector, string scalar, or a cell array of character vectors or string scalars. The number of elements in the cell array must be same as the number of elements of `x`.

yname — Name of y-axis

character vector | string scalar | cell array of character vectors or string scalars

Name of the plot y-axis, specified as the comma-separated pair consisting of `'yname'` and a character vector, string scalar, or a cell array of character vectors or string scalars. The number of elements in the cell array must be same as the number of elements of `y`.

xunit — Unit for plotting data along x-axis

character vector | string scalar

Unit for plotting the data along the x-axis, specified as the comma-separated pair consisting of `'xunit'` and a unit name, or a cell array of unit names. Unit names must appear inside single quotes (`' '`) or double quotes (`" "`). Specified units must be commensurate with the units of the series values.

yunit — Unit for plotting data along y-axis

character vector | string scalar

Unit for plotting the data along the y-axis, specified as the comma-separated pair consisting of `'xunit'` and a unit name, or a cell array of unit names. Unit names must appear inside single quotes (`' '`) or double quotes (`" "`). Specified units must be commensurate with the units of the series values.

Output Arguments

fh — Handles to the resulting plot figure windows

cell array

Handles to the resulting plot figure windows, one for each y versus x plot generated, returned as a cell array.

See Also

`simscape.logging.Node` | `simscape.logging.Series` | `simscape.logging.plot`

Topics

“Log and Plot Simulation Data”

Introduced in R2010b

simscape.logging.Node

Hierarchy tree for simulation data

Description

`simscape.logging.Node` represents the hierarchy of nodes for logging simulation data in a model. The tree starts with the workspace variable, which represents simulation data for the whole model, and recursively creates nodes for each of the children.

The children depend on the type of the parent node:

- For the top-level simulation log workspace variable, the children are all the Simscape blocks (and subsystems containing Simscape blocks) in the top-level model diagram.
- For a subsystem or a structural block, the children are all the constituent Simscape blocks and subsystems.
- For a block, the children are all its physical ports, Through and Across variables, and all internal variables defined in the block's Simscape file.
- For a physical port, the children are all its Across variables.

Final nodes in this recursion correspond to all the variables logged for the model. Final nodes do not have children nodes, and contain the series data logged during simulation.

Creation

This object is created automatically during simulation, as part of the simulation log workspace variable, if you enable data logging for the model.

Properties

id — Object identifier

character vector | string scalar

The name identifying the `Node` object. For the simulation log workspace variable, this is the name of the top-level block diagram. For blocks and subsystems, the `id` is constructed automatically as a valid MATLAB identifier based on the name of the block or subsystem. For other types, the `id` is the name of the corresponding port or variable.

savable — Logical value that specifies object reuse rules

1 | 0

Logical value that indicates how you can reuse logged simulation data in a future session. If `savable` is 1, use the regular MATLAB interface to save the workspace variable as a MAT-file and load a MAT-file into a variable. This property depends on the logging method of the `Node` object. For more information, see “Saving and Retrieving Logged Simulation Data”.

exportable — Logical value that specifies object reuse rules

1 | 0

Logical value that indicates how you can reuse logged simulation data in a future session. If `exportable` is 1, use `simscape.logging.export` and `simscape.logging.import`. This property depends on the logging method of the Node object. For more information, see “Saving and Retrieving Logged Simulation Data”.

series — Simulation series data

`simscape.logging.Series` object

For `Node` objects that do not have children nodes, and therefore correspond to the logged variables, the `series` property returns a `simscape.logging.Series` object that contains the simulation series data for this variable. For nodes that do not represent variables, the `series` property is hidden. If you access the hidden `series` property for such node, the property returns a `simscape.logging.Series` object representing an empty series (with zero points).

The other properties are dynamic, and represent all the children of the `Node` object.

Object Functions

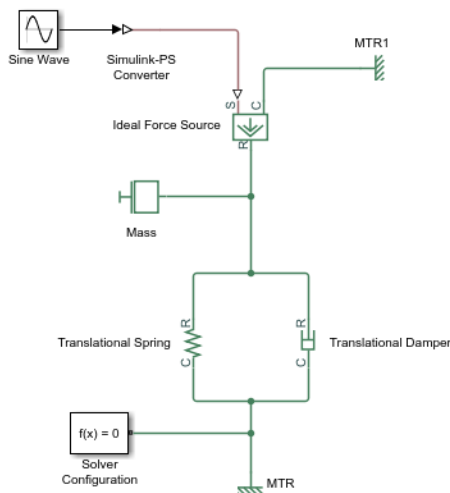
<code>get</code>	Access node of simulation logging data tree using slash-delimited path
<code>getSource</code>	Navigate from node object to block that generated it
<code>plot</code>	Plot all series associated with <code>Node</code> object
<code>plotxy</code>	Plot series associated with two node objects against each other
<code>print</code>	Print complete logging tree of <code>Node</code> object

Examples

Plot Block Velocities

Plot complete logging tree for a model, and then plot velocities of all the blocks in the model.

Consider the following model. The model name is `simple_mech2`, and data logging is enabled with the default workspace variable name, `simlog`.



Print the complete logging tree for the model:

```

print(simlog)

simple_mech2
+-Ideal_Force_Source
| +-C
| | +-v
| +-R
| | +-v
| +-S
| +-f
| +-v
+-MTR
| +-V
| | +-v
| +-f
+-MTR1
| +-V
| | +-v
| +-f
+-Mass
| +-M
| | +-v
| +-f
+-Simulink_PS_Converter
+-Translational_Damper
| +-C
| | +-v
| +-R
| | +-v
| +-f
| +-v
+-Translational_Spring
+-C
| +-v
+-R
| +-v
+-f
+-v
+-x

```

Plot velocities of all the blocks in the model:

```
plot(simlog, 'units', 'm/s', 'depth', 2)
```

This command filters simulation data in two ways. It plots only series that are commensurate with units `m/s` (that is, velocities), based on the `units` argument. And because of the `depth` argument, it plots only those velocity variables that are associated with the block itself. If you refer to the logging tree, only the Ideal Force Source, Translational Damper, and Translational Spring blocks have a velocity (`v`) variable at the second level. Because of the `depth` argument, velocities of the block ports (one level down) do not get plotted.

See Also

`simscape.logging.Series` | `simscape.logging.plot` | `simscape.logging.plotxy`

Topics

“Log and Plot Simulation Data”

“Enable Data Logging for the Whole Model”

“Log Data for Selected Blocks Only”

“Stream Logging Data to Disk”

Introduced in R2010b

get

Package: `simscape.logging`

Access node of simulation logging data tree using slash-delimited path

Syntax

```
node = get(simlog, simlogPath)
```

Description

`node = get(simlog, simlogPath)` returns a `simscape.logging.Node` object associated with the specified path, `simlogPath`, in the simulation logging data tree, `simlog`. Before you call this object function, you must have the simulation log variable in your current workspace. Create the simulation log variable by simulating the model with data logging turned on, or load a previously saved variable from a file.

This object function provides an alternative method of traversing a simulation logging data tree, by using a slash-delimited path of block names instead of a dot-delimited path of node IDs. It is especially convenient if your model contains blocks with non-ASCII characters in their names, because these names are not easily translatable into node IDs. The added benefit is tab completion, which lets you specify `simlogPath` with little or no typing.

Examples

Get Node Using Full Block Path Name and Tab Completion

Open the Permanent Magnet DC Motor example model, which already has data logging enabled, and run the simulation to create the simulation log variable `simlog_ssc_dcmotor` (as specified by the **Workspace variable name** model configuration parameter) in your current workspace:

```
ssc_dcmotor
sim('ssc_dcmotor');
```

Get the node corresponding to the Inertia block in the DC Motor subsystem:

```
n = get(simlog_ssc_dcmotor, 'DC Motor/Inertia')
n =
```

Node with properties:

```
id: 'Inertia'
w: [1x1 simscape.logging.Node]
t: [1x1 simscape.logging.Node]
I: [1x1 simscape.logging.Node]
```

`n` is the `Node` object corresponding to the Inertia block in the DC Motor subsystem.

Note that the path starts with a block name (DC Motor), not with the name of the model (`ssc_dcmotor`).

Instead of typing the path, you can use tab completion.

Start by typing the partial command that includes the name of the simulation log variable:

```
m = get(simlog_ssc_dcmotor,
```

and press the **Tab** key.

The function adds a double quote to the command line and opens a drop-down list of all the top-level blocks in the model diagram. From this drop-down list, select `DC Motor/`. The function adds this name to the path:

```
m = get(simlog_ssc_dcmotor,"DC Motor/
```

Press the **Tab** key again and select `Inertia/` from the drop-down list. Close the double quotes and the parentheses, then press **Return**:

```
m = get(simlog_ssc_dcmotor,"DC Motor/Inertia/")
```

```
m =
```

```
Node with properties:
```

```
id: 'Inertia'
w: [1x1 simscape.logging.Node]
t: [1x1 simscape.logging.Node]
I: [1x1 simscape.logging.Node]
```

`m` is also the `Node` object corresponding to the `Inertia` block in the `DC Motor` subsystem, equivalent to `n`.

Input Arguments

simlog — Simulation log variable

Node object

Simulation log workspace variable that contains the logged model simulation data, specified as a `Node` object. You specify the name of the simulation log variable by using the **Workspace variable name** parameter in the **Simscape** pane of the Configuration Parameters dialog box.

simlogPath — Path to node

character vector | string scalar

Path to a node in a simulation logging data tree, specified as a slash-delimited path of block, port, and variable names. The path must start with a block name, not with the name of the model.

Data Types: `char` | `string`

Output Arguments

node — Node in the simulation data log tree corresponding to the specified path

Node object

Node in the simulation data log tree corresponding to the specified path, returned as a `Node` object. The `Node` object, which is of class `simscape.logging.Node`, contains logged simulation data for the specified block, port, or variable. Generates an error if the node is not found.

See Also

plot | plotxy | simscape.logging.Node

Topics

“Log and Plot Simulation Data”

Introduced in R2020b

getSource

Package: `simscape.logging`

Navigate from node object to block that generated it

Syntax

```
block_id = getSource(node)
```

Description

`block_id = getSource(node)` returns the Simulink Identifier (SID) of the block that generated data in the specified `simscape.logging.Node` object. If the node object corresponds to a variable, returns the parent block for that variable. Before you call this object function, you must load the model. You must also have the simulation log variable in your current workspace. Create the simulation log variable by simulating the model with data logging turned on, or load a previously saved variable from a file.

Examples

Get Full Block Path Name for Node in Simulation Log Tree

Open the Full-Wave Bridge Rectifier example model and run the simulation:

```
ssc_bridge_rectifier
sim('ssc_bridge_rectifier');
```

The model has data logging enabled for all blocks, with **Workspace variable name** parameter set to `simlog_ssc_bridge_rectifier`. Therefore, running the simulation creates the simulation log variable in your current workspace.

Print the complete logging tree for the model:

```
print(simlog_ssc_bridge_rectifier)
```

```
ssc_bridge_rectifier
+-AC_Voltage_Source
| +-i
| +-n
| | +-v
| +-p
| | +-v
| +-v
+-C
| +-i
| +-n
| | +-v
| +-p
| | +-v
| +-v
| +-VC
```

```
+--Diode_1
| +-SimulationStatistics
| | +-zc_1
| | +-crossings
| | +-values
| +-i
| +-n
| | +-v
| +-p
| | +-v
| +-v
+--Diode_2
| +-SimulationStatistics
| | +-zc_1
| | +-crossings
| | +-values
| +-i
| +-n
| | +-v
| +-p
| | +-v
| +-v
+--Diode_3
| +-SimulationStatistics
| | +-zc_1
| | +-crossings
| | +-values
| +-i
| +-n
| | +-v
| +-p
| | +-v
| +-v
+--Diode_4
| +-SimulationStatistics
| | +-zc_1
| | +-crossings
| | +-values
| +-i
| +-n
| | +-v
| +-p
| | +-v
| +-v
+--ERef_T1
| +-V
| +-v
+--ERef_T2
| +-V
| +-v
+--Ideal_Transformer
| +-i1
| +-i2
| +-n1
| | +-v
| +-n2
| | +-v
| +-p1
```

```

| | +-v
| +-p2
| | +-v
| +-v1
| +-v2
+-Load
| +-i
| +-n
| | +-v
| +-p
| | +-v
| +-v
+-Voltage_Sensor
+-V
+-i1
+-n
| +-v
+-p
| +-v
+-v1

```

Find Simulink Identifier for the block corresponding to the Diode_1 node:

```
id = getSource(simlog_ssc_bridge_rectifier.Diode_1)
```

```
id =
```

```
ssc_bridge_rectifier:3
```

`ssc_bridge_rectifier:3` is the Simulink Identifier of the block corresponding to the specified node.

Based on the Simulink Identifier, get the full block path name:

```
blockName = getfullname(id)
```

```
blockName =
```

```
ssc_bridge_rectifier/Diode 1
```

`ssc_bridge_rectifier/Diode 1` is the full path and name of the block.

Input Arguments

node — Node in the simulation data log tree

Node object

Node in the simulation data log tree, specified as a `simscape.logging.Node` object. You specify the name of the simulation log variable by using the **Workspace variable name** parameter on the **Simscape** pane of the Configuration Parameters dialog box. To specify a node within the simulation log variable, provide the complete path to that node through the simulation data tree, starting with the top-level variable name.

Example: `simlog.DC_Motor.Inertia`

Output Arguments

block_id — Simulink Identifier of the block corresponding to the specified node

SID

Simulink Identifier (SID) of the block that generated data in the specified node object. If the node object corresponds to a variable, returns the parent block for that variable.

See Also

`print` | `simscape.logging.Node`

Topics

“Log and Plot Simulation Data”

Introduced in R2015b

plot

Package: `simscape.logging`

Plot all series associated with Node object

Syntax

```
fh = plot(node)
fh = plot(node,Name,Value)
```

Description

`fh = plot(node)` plots all nonempty series associated with the specified node and its children. `fh` is a structure of handles to the resulting figures. `node` is a `simscape.logging.Node` object. `node` must include a full identifier path to the node, starting with the workspace log variable name.

`fh = plot(node,Name,Value)` lets you filter the data being plotted by using one or more `Name,Value` pair arguments. For example, specify 'units' followed by a unit name to plot only nodes that are commensurate with the specified unit.

Examples

Plot Simulation Data for a Node

Plot velocity of port **R** of a Translational Spring block.

Open the Mass-Spring-Damper with Controller example model:

```
ssc_mass_spring_damper_control
```

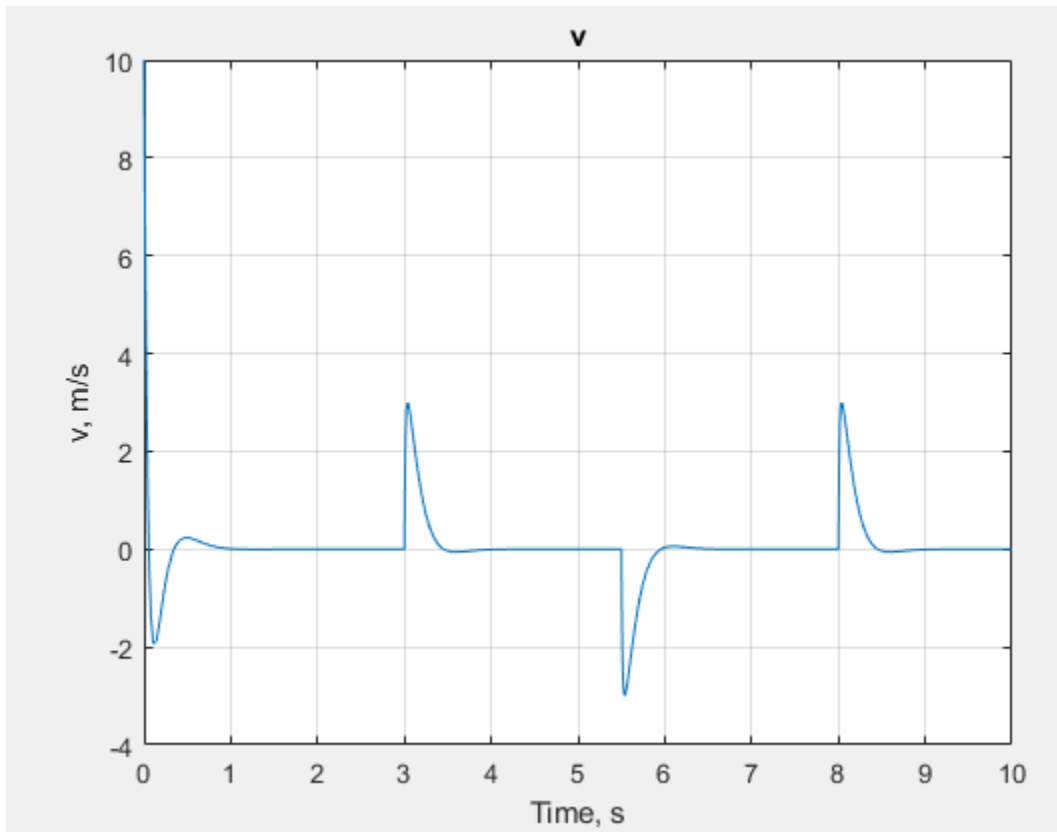
This example model has data logging enabled for the whole model, with the **Workspace variable name** parameter set to `simlog_ssc_mass_spring_damper_control`.

Simulate the model to log the simulation data:

```
sim('ssc_mass_spring_damper_control');
```

Plot velocity of port **R** of the Translational Spring block `Spring`.

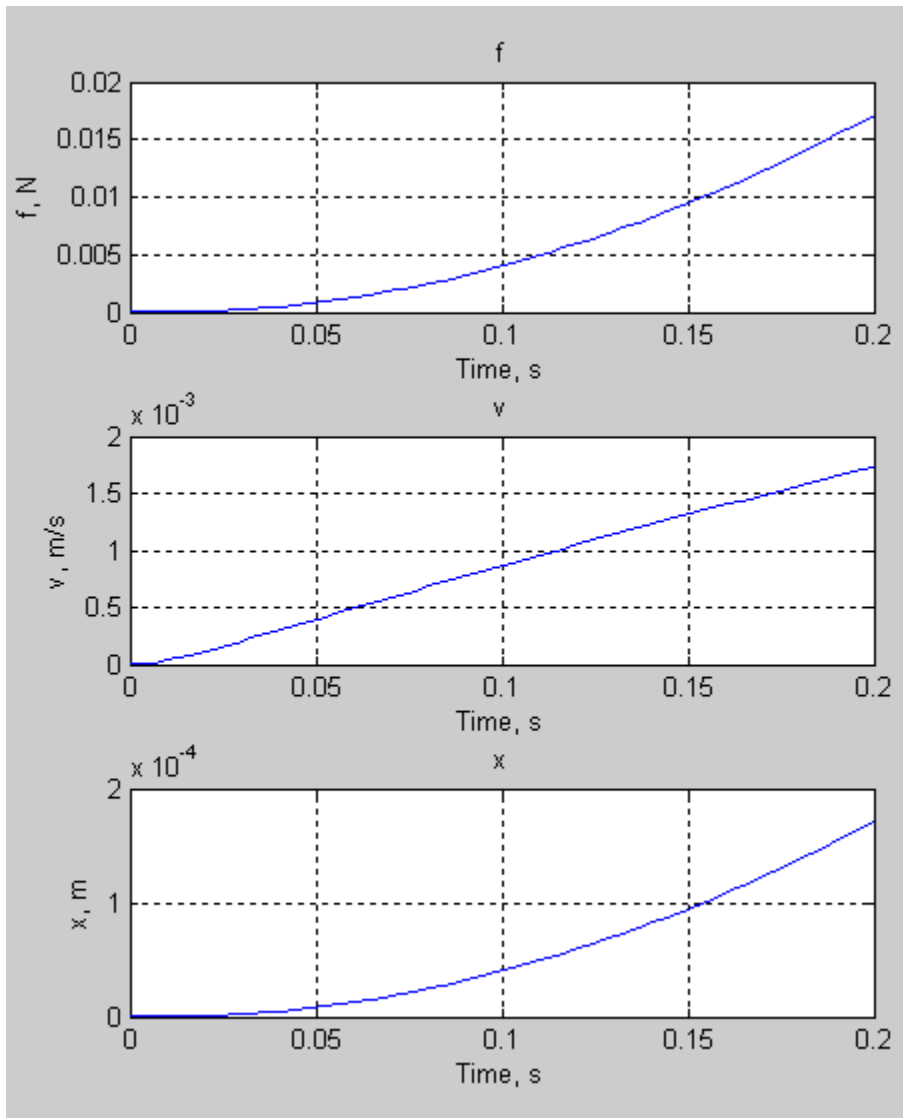
```
plot(simlog_ssc_mass_spring_damper_control.Spring.R);
```



Plot All Variables for a Block

This command plots all the variables associated with a Translational Spring block, but not with its ports. The command assumes that the model has the default workspace variable name, `simlog`, and that the Translational Spring block is located at the top level of the model diagram.

```
plot(simlog.Translational_Spring, 'depth', 1)
```

Input Arguments

node — Simulation data to plot

Node object | cell array of Node objects

Simulation data to plot, specified as a `simscape.logging.Node` object or a homogeneous cell array of such objects. `node` must include a full identifier path to the node, starting with the workspace log variable name.

The table describes the resulting plots based on the type of the node argument:

Node object	Plots all nonempty series associated with the <code>Node</code> and its children (up to the level defined by the <code>depth</code>). If the <code>Node</code> has multiple simulation variable nodes as children at level 1, these children are plotted in the same figure window but on a different axis. Descendants at other levels are plotted in different figure windows. All dimensions of a nonscalar series are plotted on the same axis.
Cell array of Node objects	Plots commensurate series superimposed on the same axis. Intended for use to compare simulation data from different runs. All entries of the cell array are required to be equivalent to each other, meaning that the <code>Node</code> objects must have same hierarchy, and the series for each node must have the same dimensions and commensurate units.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1, Value1, ..., NameN, ValueN`.

Example: `fh = plot(simlog, 'units', 'mm', 'time', [1 3])` plots all of the linear position variables in the model (series that are commensurate with units of mm), in those units, within the time range between 1 and 3 seconds.

depth — Number of children levels

`intmax` (default) | nonnegative integer

Number of children levels to plot, specified as the comma-separated pair consisting of `'depth'` and a nonnegative integer. By default, the function plots all descendants of node that have nonempty series. Specifying `depth` lets you limit the number of levels to plot, for example:

<code>'depth', 0</code>	No children; plot the nonempty series of the specified node only.
<code>'depth', 1</code>	Plot the nonempty series of the specified node and its children.
<code>'depth', 2</code>	Plot the nonempty series of the specified node, its children, and their children.

names — Plot legend

cell array of character vectors or string scalars

Plot legend, specified as the comma-separated pair consisting of `'names'` and a cell array of character vectors or string scalars. The number of elements in the cell array must be same as the number of elements of `node`.

By default, plots have no legend.

time — Time range for plotting data

`[]` (default) | 1x2 vector, `[start_time end_time]` in seconds

Time range for plotting the data, specified as the comma-separated pair consisting of `'time'` and a 1x2 vector, `[start_time end_time]`, in seconds.

`[]` plots all data.

units — Units for plotting data

character vector | string scalar | cell array of character vectors or string scalars

Units for plotting the data, specified as the comma-separated pair consisting of 'units' and a unit name, or a cell array of unit names. Unit names must appear inside single quotes (' ') or double quotes (" ").

This argument plots the series values in the specified units, and also filters the data to plot only nodes that have series commensurate with the specified unit.

Example: `fh = plot(simlog,'units',{'mm','mm/s'})` plots all of the linear position and velocity variables in the model (series that are commensurate with units of mm and mm/s), in those units.

viewer — Plot data in the Simulation Data Inspector

`'datainspector' | "datainspector"`

Alternative destination to plot the data, specified as the comma-separated pair consisting of 'viewer' and `datainspector` inside single quotes (' ') or double quotes (" ").

If you specify this name-value pair argument, the function plots the data in the Simulation Data Inspector. By default, the function plots the data in a plot window.

Output Arguments

fh — Handles to the resulting plot figure windows

structure

Handles to the resulting plot figure windows, returned as a structure with the same hierarchy as the node being plotted. For example, if a specific child is not plotted then that field in the output structure is empty.

See Also

`simscape.logging.Node` | `simscape.logging.plot`

Topics

"Log and Plot Simulation Data"

Introduced in R2010b

plotxy

Package: `simscape.logging`

Plot series associated with two node objects against each other

Syntax

```
fh = plotxy(x,y)
fh = plotxy(x,y,Name,Value)
```

Description

`fh = plotxy(x,y)` plots the simulation series values of node `y` along the `y`-axis, with series values of node `x` along the `x`-axis. `fh` is a cell array of handles to the resulting figures. `x` and `y` are `simscape.logging.Node` objects or homogeneous cell arrays of such objects. Each object must be a simulation variable node (one that has a direct child series). The values of this child series are plotted along the respective axis. All series must have the same time vectors. If `x` and `y` are cell arrays, they must be of the same size, or one of them can be a scalar.

`fh = plotxy(x,y,Name,Value)` lets you customize the plot by using one or more `Name,Value` pair arguments. For example, specify `'time'` followed by a 1x2 vector, [`start_time end_time`], to plot only the data within this time range.

Examples

Plot Motor Torque Node Against Its Angular Velocity Node

Plot the motor torque variable node against its angular velocity variable node.

Open the Permanent Magnet DC Motor example model:

```
ssc_dcmotor
```

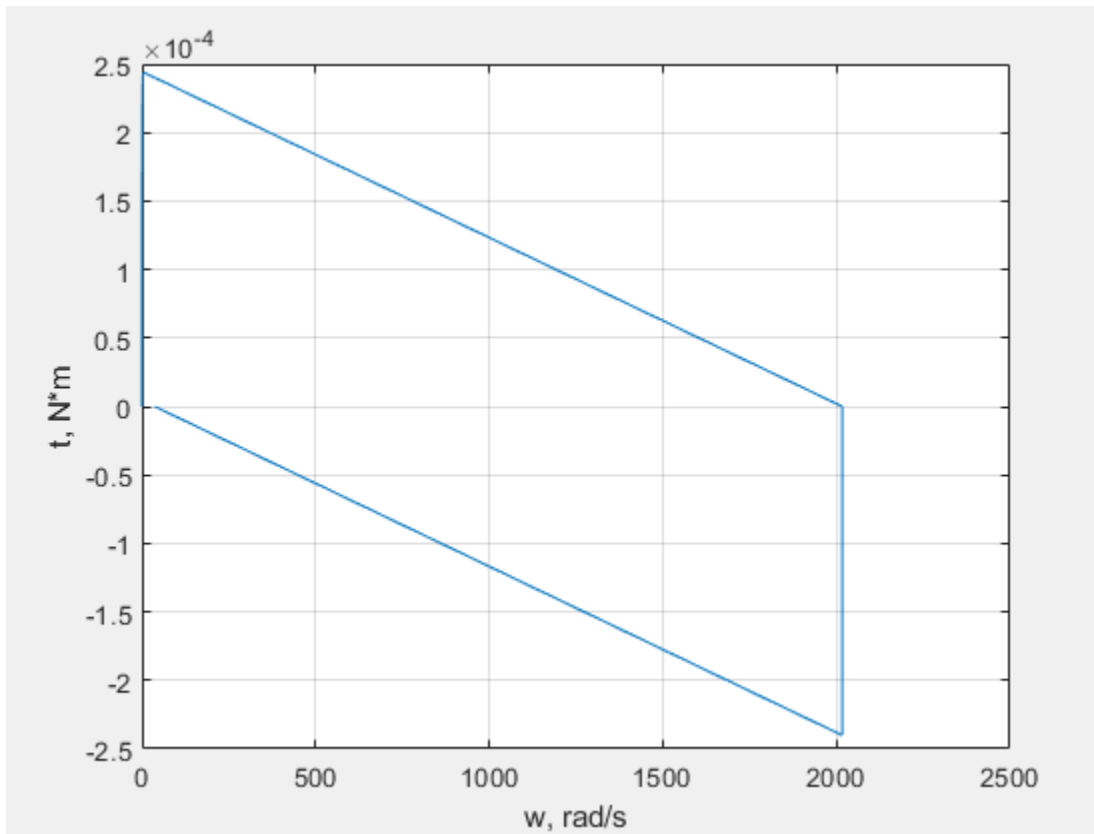
This example model has data logging enabled for the whole model, with the **Workspace variable name** parameter set to `simlog_ssc_dcmotor`.

Simulate the model to log the simulation data:

```
sim('ssc_dcmotor');
```

Plot the motor torque against its angular velocity:

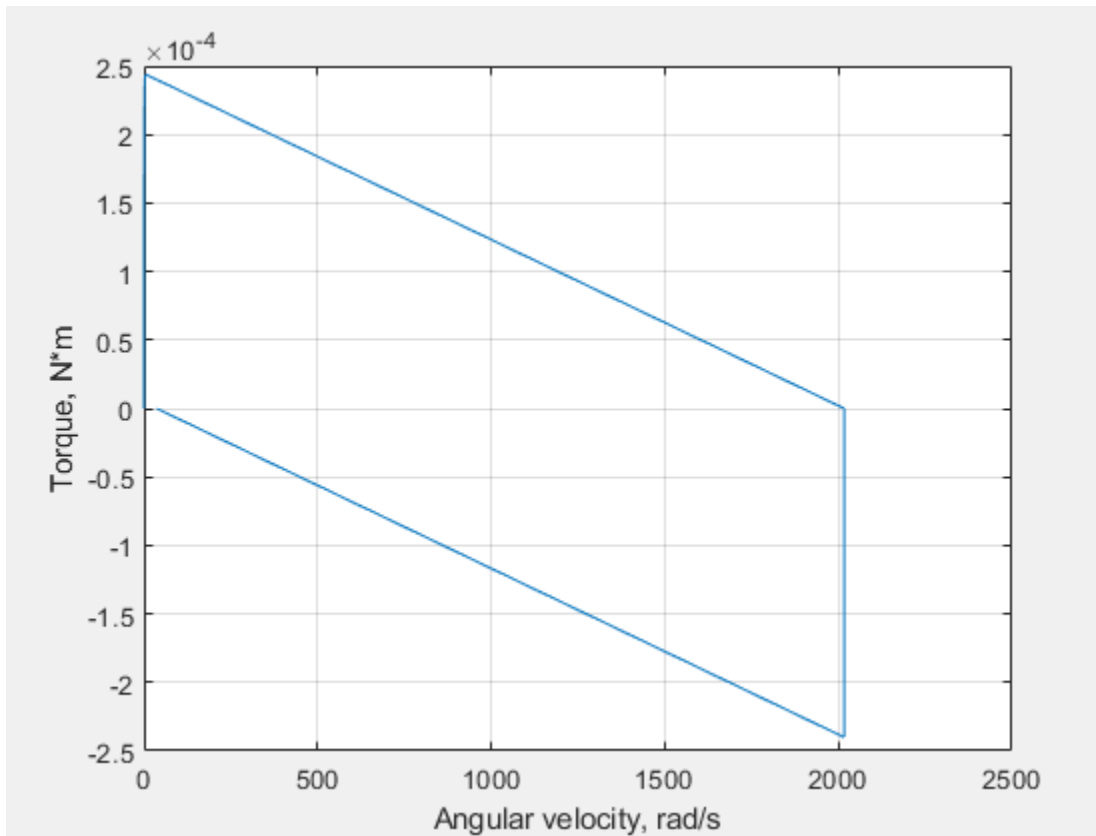
```
simscape.logging.plotxy(simlog_ssc_dcmotor.DC_Motor.Rotational_Electromechanical_Converter.R.w, .
    simlog_ssc_dcmotor.DC_Motor.Inertia.t)
```



Customize the Plot Axis Names

When you plot the variable nodes against each other, the default plot displays the name of the variable and the unit name along each axis. To customize the axis names, use name-value pair arguments.

```
simscape.logging.plotxy(simlog_ssc_dcmotor.DC_Motor.Rotational_Electromechanical_Converter.R.w, .  
    simlog_ssc_dcmotor.DC_Motor.Inertia.t, 'xname', 'Angular velocity', 'yname', 'Torque')
```



Input Arguments

x — Simulation data to plot along x-axis

Node object | cell array of Node objects

Simulation data to plot along the x-axis, specified as a `simscape.logging.Node` object or a homogeneous cell array of such objects. Each object must be a simulation variable node (one that has a direct child series). The values of this child series are plotted along the respective axis. If `x` and `y` are cell arrays, they must be of the same size, or one of them can be a scalar. All series must have the same time vectors. `x` must include a full identifier path to the node, starting with the workspace log variable name.

y — Simulation data to plot along y-axis

Node object | cell array of Node objects

Simulation data to plot along the y-axis, specified as a `simscape.logging.Node` object or a homogeneous cell array of such objects. Each object must be a simulation variable node (one that has a direct child series). The values of this child series are plotted along the respective axis. If `x` and `y` are cell arrays, they must be of the same size, or one of them can be a scalar. All series must have the same time vectors. `y` must include a full identifier path to the node, starting with the workspace log variable name.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1, Value1, ..., NameN, ValueN`.

Example: `fh = plotxy(simlog.TS.C.v, simlog.TS.R.v, 'xunit', 'mm/s', 'yunit', 'mm/s')` plots velocities of ports **C** and **R** of the Translational Spring block **TS** against each other, in mm/s.

xname — Name of x-axis

character vector | string scalar | cell array of character vectors or string scalars

Name of the plot x-axis, specified as the comma-separated pair consisting of 'xname' and a character vector, string scalar, or a cell array of character vectors or string scalars. The number of elements in the cell array must be same as the number of elements of `x`.

yname — Name of y-axis

character vector | string scalar | cell array of character vectors or string scalars

Name of the plot y-axis, specified as the comma-separated pair consisting of 'yname' and a character vector, string scalar, or a cell array of character vectors or string scalars. The number of elements in the cell array must be same as the number of elements of `y`.

xunit — Unit for plotting data along x-axis

character vector | string scalar

Unit for plotting the data along the x-axis, specified as the comma-separated pair consisting of 'xunit' and a unit name, or a cell array of unit names. Unit names must appear inside single quotes (' ') or double quotes (" "). Specified units must be commensurate with the units of the series values.

yunit — Unit for plotting data along y-axis

character vector | string scalar

Unit for plotting the data along the y-axis, specified as the comma-separated pair consisting of 'yunit' and a unit name, or a cell array of unit names. Unit names must appear inside single quotes (' ') or double quotes (" "). Specified units must be commensurate with the units of the series values.

Output Arguments

fh — Handles to the resulting plot figure windows

cell array

Handles to the resulting plot figure windows, one for each y versus x plot generated, returned as a cell array.

See Also

`simscape.logging.Node` | `simscape.logging.plotxy`

Topics

"Log and Plot Simulation Data"

Introduced in R2010b

print

Package: `simscape.logging`

Print complete logging tree of Node object

Syntax

```
print(node)
```

Description

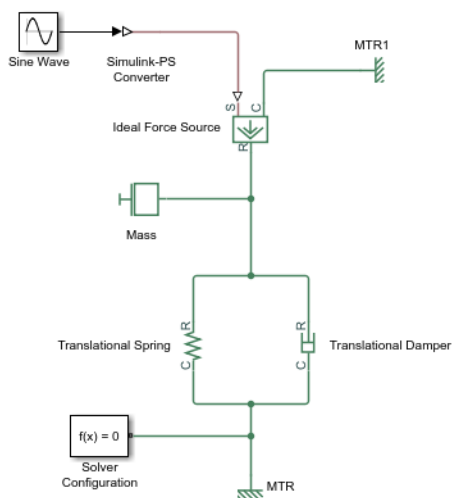
`print(node)` prints the complete logging tree starting with the specified node. `node` is a `simscape.logging.Node` object. `node` must include a full identifier path to the node, starting with the workspace log variable name. Before you call this object function, you must load the model. You must also have the simulation log variable in your current workspace. Create the simulation log variable by simulating the model with data logging turned on, or load a previously saved variable from a file.

Examples

Print Logging Tree

Plot complete logging tree for a model, and then a logging tree for a node in the model.

Consider the following model. The model name is `simple_mech2`, and data logging is enabled with the default workspace variable name, `simlog`.



Print the complete logging tree for the whole model:

```
print(simlog)
```



```

simple_mech2
+-Ideal_Force_Source
| +-C
| | +-v
| +-R
| | +-v
| +-S
| +-f
| +-v
+-MTR
| +-V
| | +-v
| +-f
+-MTR1
| +-V
| | +-v
| +-f
+-Mass
| +-M
| | +-v
| +-f
+-Simulink_PS_Converter
+-Translational_Damper
| +-C
| | +-v
| +-R
| | +-v
| +-f
| +-v
+-Translational_Spring
  +-C
  | +-v
  +-R
  | +-v
  +-f
  +-v
  +-x

```

Print the logging tree just for the Mass block:

```
print(simlog.Mass)
```

```

Mass
+-M
| +-v
+-f

```

Input Arguments

node — Initial node for printing the logging tree

Node object

Initial node for printing the logging tree, specified as a `simscape.logging.Node` object. `node` must include a full identifier path to the node, starting with the workspace log variable name.

Example: `simlog.DC_Motor.Inertia`

See Also

`simscape.logging.Node`

Topics

“Log and Plot Simulation Data”

Introduced in R2010b

simscape.logging.Series

Time-value series for simulation data

Description

`simscape.logging.Series` represents simulation data for a variable in a model. The series is a representation containing time-value pairs for each simulation step. The size of the series is determined by the number of simulation steps. You can also limit the size by specifying the maximum number of logged steps when you set your data logging preferences.

Final nodes in the data logging tree correspond to all the variables logged for the model. Final nodes do not have children nodes, and contain the series data logged during simulation. To specify a `Series` object, start with a full identifier path from the workspace log variable name to the name of the variable node associated with the series, and conclude with `.series`. For example, `simlog.Translational_Spring.v.series` is the series containing the simulation data for the variable `v` (velocity) of a Translational Spring block at the top level of a block diagram, given the default workspace variable name, `simlog`.

Creation

This object is created automatically during simulation, as part of the simulation log workspace variable, if you enable data logging for the model.

Properties

points — Series size

nonnegative integer

Size, or number of steps, in the simulation series.

dimension — Dimension of variable

two-element vector

Dimension of variable represented by the series.

unit — Logical value that specifies object reuse rules

character vector

The default unit associated with the values in the series.

conversion — Thermal unit conversion type

'absolute' | 'relative'

Thermal unit conversion type (absolute or relative). For more information, see “Thermal Unit Conversions”.

Object Functions

plot Plot logged simulation series values against time
plotxy Plot two series against each other
time Extract time vector from simulation series
values Extract values vector from simulation series

Examples

Plot Simulation Data for a Series

Plot velocity of port **R** of a Translational Spring block.

Open the Mass-Spring-Damper with Controller example model:

```
ssc_mass_spring_damper_control
```

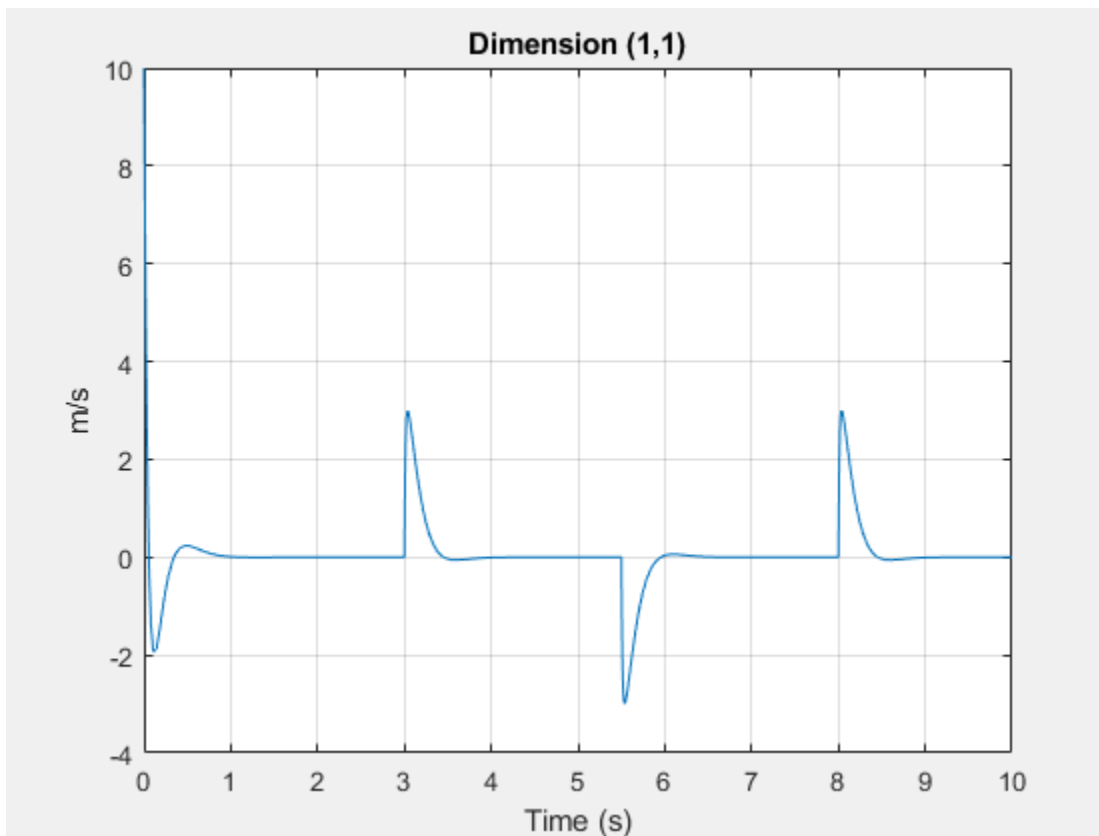
This example model has data logging enabled for the whole model, with the **Workspace variable name** parameter set to `simlog_ssc_mass_spring_damper_control`.

Simulate the model to log the simulation data:

```
sim('ssc_mass_spring_damper_control');
```

Plot velocity of port **R** of the Translational Spring block Spring.

```
plot(simlog_ssc_mass_spring_damper_control.Spring.R.v.series);
```



See Also

[simscape.logging.Node](#) | [simscape.logging.plot](#) | [simscape.logging.plotxy](#)

Topics

“Log and Plot Simulation Data”
“Enable Data Logging for the Whole Model”
“Log Data for Selected Blocks Only”
“Stream Logging Data to Disk”

Introduced in R2010b

plot

Package: `simscape.logging`

Plot logged simulation series values against time

Syntax

```
fh = plot(series)
fh = plot(series,Name,Value)
```

Description

`fh = plot(series)` plots the simulation series values along the *y*-axis, with time along the *x*-axis. `fh` is a structure of handles to the resulting figures. `series` is a `simscape.logging.Series` object. `series` must include a full identifier path to the series, starting with the workspace log variable name.

`fh = plot(series,Name,Value)` lets you customize the plot by using one or more `Name,Value` pair arguments. For example, specify 'time' followed by a 1x2 vector, [*start_time end_time*], to plot only the data within this time range.

Examples

Plot Simulation Data for a Series

Plot velocity of port **R** of a Translational Spring block.

Open the Mass-Spring-Damper with Controller example model:

```
ssc_mass_spring_damper_control
```

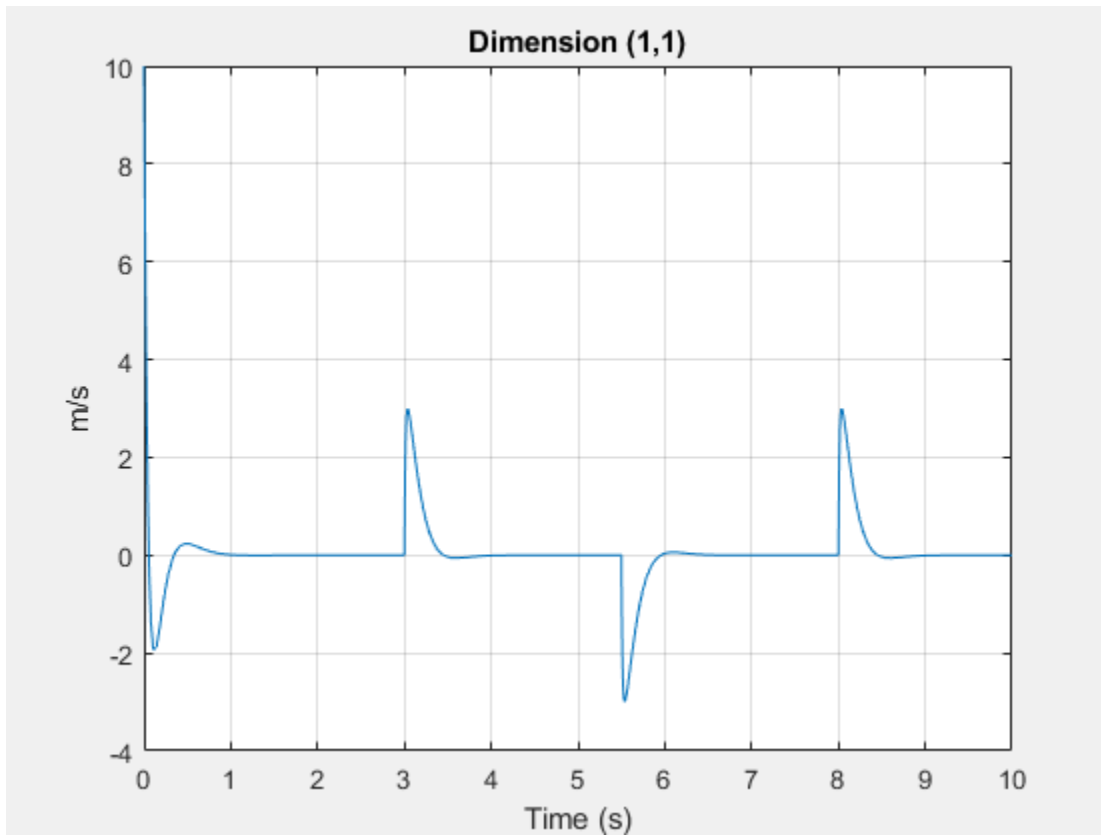
This example model has data logging enabled for the whole model, with the **Workspace variable name** parameter set to `simlog_ssc_mass_spring_damper_control`.

Simulate the model to log the simulation data:

```
sim('ssc_mass_spring_damper_control');
```

Plot velocity of port **R** of the Translational Spring block Spring.

```
plot(simlog_ssc_mass_spring_damper_control.Spring.R.v.series);
```

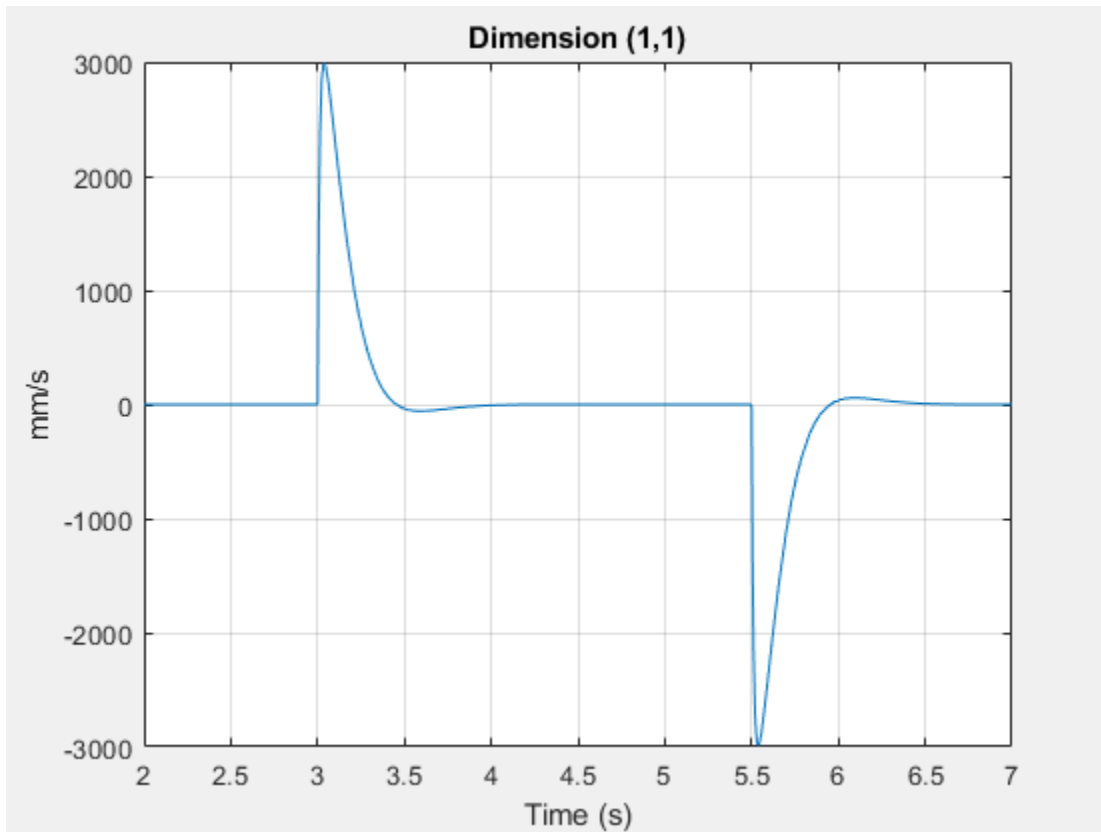


Customize the Plot

Customize the previous plot using the name-value pair arguments.

This command plots the velocity of port **R** of the Translational Spring block in mm/s and only within the time range between 2 and 7 seconds.

```
plot(simlog_ssc_mass_spring_damper_control.Spring.R.v.series,'units','mm/s','time',[2 7]);
```



Input Arguments

series — Simulation data to plot

scalar `Series` object | nonscalar `Series` object | cell array of `Series` objects

Simulation data to plot, specified as a `simscape.logging.Series` object or a homogeneous cell array of such objects. `series` must include a full identifier path to the series, starting with the workspace log variable name.

The table describes the resulting plots based on the type of the `series` argument:

Scalar <code>Series</code> object	Plots the simulation series values along the y -axis, with time along the x -axis.
Nonscalar <code>Series</code> object	Plots each dimension of the series values on a different axis in the same figure window.
Cell array of <code>Series</code> objects	Plots all series objects with commensurate units on the same axis (superimposed), and each dimension for a nonscalar series on a different axis in the same figure window.

The input arguments are binned based on commensurate units. For each bin, all `Series` objects with the same dimension as the first `Series` object in that bin are plotted and others are ignored.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1, Value1, ..., NameN, ValueN`.

Example: `fh = plot(simlog.Translational_Spring.R.v.series, 'units', 'mm/s')` plots velocity of port **R** of the Translational Spring block in mm/s.

names — Plot legend

cell array of character vectors or string scalars

Plot legend, specified as the comma-separated pair consisting of 'names' and a cell array of character vectors or string scalars. The number of elements in the cell array must be same as the number of elements of `series`.

By default, plots have no legend.

time — Time range for plotting data

[] (default) | 1x2 vector, [`start_time` `end_time`] in seconds

Time range for plotting the data, specified as the comma-separated pair consisting of 'time' and a 1x2 vector, [`start_time` `end_time`], in seconds.

[] plots all data.

units — Units for plotting data

character vector | string scalar | cell array of character vectors or string scalars

Units for plotting the data, specified as the comma-separated pair consisting of 'units' and a unit name, or a cell array of unit names. Unit names must appear inside single quotes (' ') or double quotes (" "). Specified units must be commensurate with the units of the series values.

Output Arguments

fh — Handles to the resulting plot figure windows

cell array

Handles to the resulting plot figure windows, returned as a cell array.

See Also

`simscape.logging.Series` | `simscape.logging.plot`

Topics

"Log and Plot Simulation Data"

Introduced in R2010b

plotxy

Package: `simscape.logging`

Plot two series against each other

Syntax

```
fh = plotxy(x,y)
fh = plotxy(x,y,Name,Value)
```

Description

`fh = plotxy(x,y)` plots values of the simulation series `y` along the `y`-axis, with values of the simulation series `x` along the `x`-axis. `fh` is a cell array of handles to the resulting figures. `x` and `y` are `simscape.logging.Series` objects or homogeneous cell arrays of such objects. All series must have the same time vectors. If `x` and `y` are cell arrays, they must be of the same size, or one of them can be a scalar.

`fh = plotxy(x,y,Name,Value)` lets you customize the plot by using one or more `Name,Value` pair arguments. For example, specify `'time'` followed by a 1x2 vector, `[start_time end_time]`, to plot only the data within this time range.

Examples

Plot Motor Torque Series Against Its Angular Velocity Series

Plot the motor torque series against its angular velocity series.

Open the Permanent Magnet DC Motor example model:

```
ssc_dcmotor
```

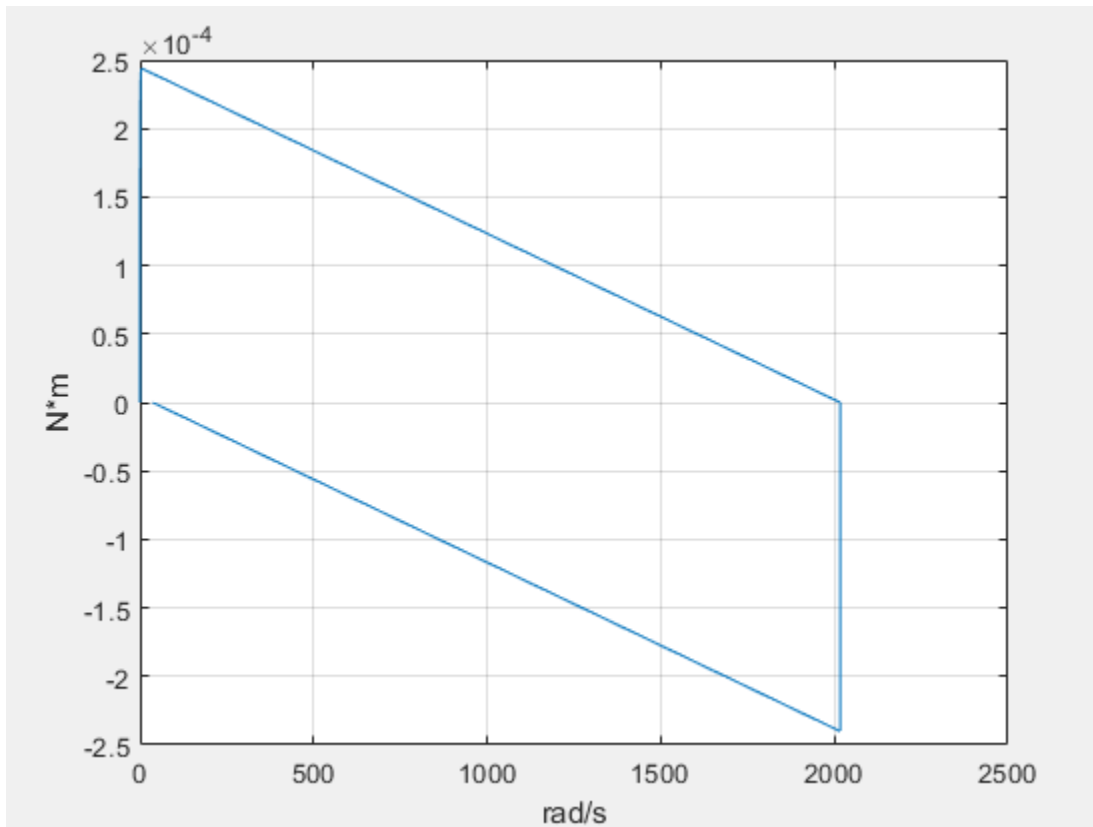
This example model has data logging enabled for the whole model, with the **Workspace variable name** parameter set to `simlog_ssc_dcmotor`.

Simulate the model to log the simulation data:

```
sim('ssc_dcmotor');
```

Plot the motor torque against its angular velocity:

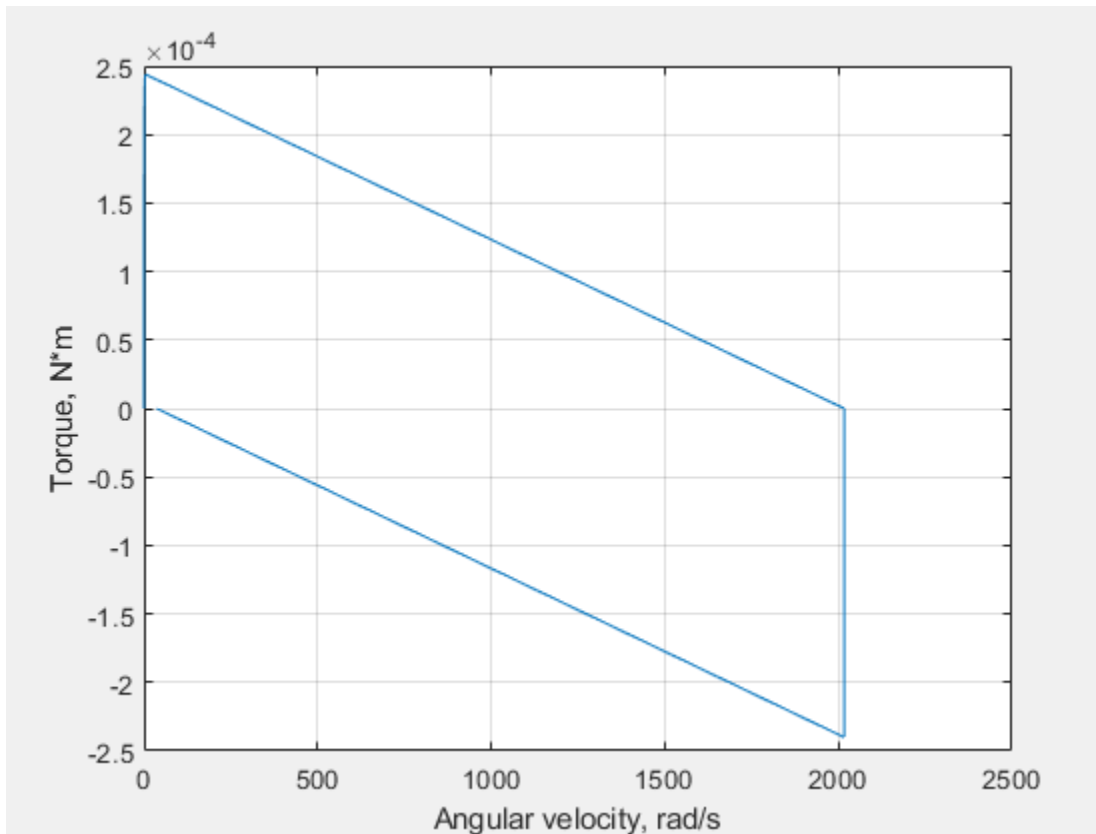
```
simscape.logging.plotxy(simlog_ssc_dcmotor.DC_Motor.Rotational_Electromechanical_Converter.R.w.s
    simlog_ssc_dcmotor.DC_Motor.Inertia.t.series)
```



Add Axis Names to the Plot

When you plot the series against each other, the default plot displays only the unit names along each axis. To add the axis names, use name-value pair arguments.

```
simscape.logging.plotxy(simlog_ssc_dcmotor.DC_Motor.Rotational_Electromechanical_Converter.R.w.s  
simlog_ssc_dcmotor.DC_Motor.Inertia.t.series,'xname','Angular velocity','yname','Torque')
```



Input Arguments

x — Simulation data to plot along x-axis

scalar `Series` object | nonscalar `Series` object | cell array of `Series` objects

Simulation data to plot along the x-axis, specified as a `simscape.logging.Series` object or a homogeneous cell array of such objects. `x` must include a full identifier path to the series, starting with the workspace log variable name. If `x` and `y` are cell arrays, they must be of the same size, or one of them can be a scalar. All series must have the same time vectors.

y — Simulation data to plot along y-axis

scalar `Series` object | nonscalar `Series` object | cell array of `Series` objects

Simulation data to plot along the y-axis, specified as a `simscape.logging.Series` object or a homogeneous cell array of such objects. `y` must include a full identifier path to the series, starting with the workspace log variable name. If `x` and `y` are cell arrays, they must be of the same size, or one of them can be a scalar. All series must have the same time vectors.

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1`, `Value1`, ..., `NameN`, `ValueN`.

Example: `fh = plotxy(simlog.TS.C.v.series,simlog.TS.R.v.series,'xunit','mm/s','yunit','mm/s')` plots velocities of ports **C** and **R** of the Translational Spring block **TS** against each other, in mm/s.

time — Time range for plotting data

`[]` (default) | 1x2 vector, `[start_time end_time]` in seconds

Time range for plotting the data, specified as the comma-separated pair consisting of 'time' and a 1x2 vector, `[start_time end_time]`, in seconds.

`[]` plots all data.

xname — Name of x-axis

character vector | string scalar | cell array of character vectors or string scalars

Name of the plot x-axis, specified as the comma-separated pair consisting of 'xname' and a character vector, string scalar, or a cell array of character vectors or string scalars. The number of elements in the cell array must be same as the number of elements of `x`.

yname — Name of y-axis

character vector | string scalar | cell array of character vectors or string scalars

Name of the plot y-axis, specified as the comma-separated pair consisting of 'yname' and a character vector, string scalar, or a cell array of character vectors or string scalars. The number of elements in the cell array must be same as the number of elements of `y`.

xunit — Unit for plotting data along x-axis

character vector | string scalar

Unit for plotting the data along the x-axis, specified as the comma-separated pair consisting of 'xunit' and a unit name, or a cell array of unit names. Unit names must appear inside single quotes (' ') or double quotes (" "). Specified units must be commensurate with the units of the series values.

yunit — Unit for plotting data along y-axis

character vector | string scalar

Unit for plotting the data along the y-axis, specified as the comma-separated pair consisting of 'yunit' and a unit name, or a cell array of unit names. Unit names must appear inside single quotes (' ') or double quotes (" "). Specified units must be commensurate with the units of the series values.

Output Arguments

fh — Handles to the resulting plot figure windows

cell array

Handles to the resulting plot figure windows, one for each y versus x plot generated, returned as a cell array.

See Also

`simscape.logging.Series` | `simscape.logging.plotxy`

Topics

“Log and Plot Simulation Data”

Introduced in R2010b

time

Package: `simscape.logging`

Extract time vector from simulation series

Syntax

```
tv = time(series)
```

Description

`tv = time(series)` returns a row vector of simulation times contained in the series. `series` is a `simscape.logging.Series` object. `series` must include a full identifier path to the series, starting with the workspace log variable name.

Examples

Extract Simulation Time Data for Spring Deformation

Return simulation time data for the deformation of a Translational Spring block.

Open the Mass-Spring-Damper with Controller example model:

```
ssc_mass_spring_damper_control
```

This example model has data logging enabled for the whole model, with the **Workspace variable name** parameter set to `simlog_ssc_mass_spring_damper_control`.

Simulate the model for 1 second, to log the simulation data:

```
paramNameValStruct.StopTime = '1.0';
sim('ssc_mass_spring_damper_control', paramNameValStruct);
```

Return the simulation time vector for the deformation variable of the Translational Spring block, `Spring`. `x` is the deformation variable name, and `series` is the `Series` object containing the simulation data for this variable.

```
t1 = time(simlog_ssc_mass_spring_damper_control.Spring.x.series)
```

```
t1 =
```

```

    0
    0.0072
    0.0143
    0.0223
    0.0323
    0.0447
    0.0602
    0.0799
    0.1064
    0.1447
```

```
0.1833
0.2114
0.2395
0.2776
0.3248
0.3531
0.3814
0.4194
0.4650
0.4940
0.5230
0.5608
0.6056
0.6356
0.6657
0.7033
0.7467
0.7772
0.8077
0.8451
0.8877
0.9263
0.9572
0.9858
1.0000
```

The `t1` vector has 35 values, between 0 and 1, because the simulation series has 35 time steps and the simulation stop time is 1 second.

Input Arguments

series — Simulation series

Series object

Simulation series, specified as a `simscape.logging.Series` object. `series` must include a full identifier path to the series, starting with the workspace log variable name.

Output Arguments

tv — Times, in seconds, corresponding to time steps in simulation series

row vector

Times, in seconds, corresponding to the time steps in the simulation series, returned as a row vector.

See Also

`simscape.logging.Series` | `values`

Topics

“Log and Plot Simulation Data”

Introduced in R2010b

values

Package: `simscape.logging`

Extract values vector from simulation series

Syntax

```
vv = values(series)
vv = values(series,units)
```

Description

`vv = values(series)` returns a row vector of variable values contained in the series, in default units. `series` is a `simscape.logging.Series` object. `series` must include a full identifier path to the series, starting with the workspace log variable name.

`vv = values(series,units)` returns a row vector of variable values contained in the series, in specified units. `units` must be commensurate with the default units of the variable values contained in the series.

Examples

Extract Spring Deformation Values

Return the deformation values of a Translational Spring block, in default units.

Open the Mass-Spring-Damper with Controller example model:

```
ssc_mass_spring_damper_control
```

This example model has data logging enabled for the whole model, with the **Workspace variable name** parameter set to `simlog_ssc_mass_spring_damper_control`.

Simulate the model for 1 second, to log the simulation data:

```
paramNameValStruct.StopTime = '1.0';
sim('ssc_mass_spring_damper_control',paramNameValStruct);
```

Return the deformation values of the Translational Spring block, `Spring.x` is the deformation variable name, and `series` is the `Series` object containing the simulation data for this variable.

```
v1 = values(simlog_ssc_mass_spring_damper_control.Spring.x.series)
```

```
v1 =
```

```

    0
    0.0652
    0.1153
    0.1587
    0.1947
    0.2179
```

```
0.2228
0.2046
0.1591
0.0840
0.0193
-0.0164
-0.0426
-0.0648
-0.0762
-0.0770
-0.0747
-0.0686
-0.0587
-0.0519
-0.0452
-0.0369
-0.0282
-0.0231
-0.0187
-0.0140
-0.0097
-0.0074
-0.0054
-0.0036
-0.0021
-0.0011
-0.0005
-0.0002
-0.0000
```

The `v1` vector has 35 values because the simulation series has 35 time steps. The deformation values are in meters (the default unit of the series).

Extract Spring Deformation Values in Different Unit

The previous example returns the deformation values of a Translational Spring block in default units, meters. In this example, specify a different unit for extracting the series values. The unit you specify must be commensurate with the default units of the variable values contained in the series.

Return the deformation values of the Translational Spring block, `Spring`, in millimeters.

```
v2 = values(simlog_ssc_mass_spring_damper_control.Spring.x.series, 'mm')
```

```
v2 =
    0
  65.1626
 115.2876
 158.6666
 194.6648
 217.8816
 222.7924
 204.5943
 159.1137
  83.9882
  19.3122
```

```

-16.4485
-42.6109
-64.7991
-76.1640
-76.9521
-74.6976
-68.5525
-58.6695
-51.8953
-45.1768
-36.8985
-28.1789
-23.1087
-18.6816
-14.0093
-9.7345
-7.3590
-5.4308
-3.5878
-2.0621
-1.0952
-0.5459
-0.1758
-0.0334

```

The `v2` vector also has 35 values, but these values are in millimeters, so each value is 1000 times bigger than the respective value in `v1`.

Input Arguments

series — Simulation series

Series object

Simulation series, specified as a `simscape.logging.Series` object. `series` must include a full identifier path to the series, starting with the workspace log variable name.

units — Units for plotting data

character vector | string scalar | cell array of character vectors or string scalars

Units for plotting the data, specified as the comma-separated pair consisting of 'units' and a unit name, or a cell array of unit names. Unit names must appear inside single quotes (' ') or double quotes (" "). Specified units must be commensurate with the units of the series values.

Output Arguments

vv — Variable values corresponding to time steps in simulation series

row vector

Variable values corresponding to the time steps in the simulation series, returned as a row vector.

For nonscalar variables of size m -by- n , this method returns a row vector of $m*n*steps$ size, where `steps` is the number of steps in the series, and each $m*n$ block represents the logged value for the variable in a column major form. For example, if a variable size is 2-by-2, then the first four elements in the row vector are the a_{11} , a_{21} , a_{12} , and a_{22} elements at the first time step.

See Also

`simscape.logging.Series` | `time`

Topics

“Log and Plot Simulation Data”

Introduced in R2010b

simscape.logging.findNode

Find Node object corresponding to block or subsystem

Syntax

```
node = simscape.logging.findNode(simlog,block)
```

Description

`node = simscape.logging.findNode(simlog,block)` returns a `simscape.logging.Node` object that contains the logged simulation data for the specified block or subsystem in a model. Before you call this function, you must load the model. You must also have the simulation log variable in your current workspace. Create the simulation log variable by simulating the model with data logging turned on, or load a previously saved variable from a file.

Examples

Find Node for the Current Block

Open the Permanent Magnet DC Motor example model, which already has data logging enabled, and run the simulation to create the simulation log variable `simlog_ssc_dcmotor` (as specified by the **Workspace variable name** model configuration parameter) in your current workspace:

```
ssc_dcmotor
sim('ssc_dcmotor');
```

Open the DC Motor subsystem and select the Inertia block.

Find node corresponding to the selected block:

```
n = simscape.logging.findNode(simlog_ssc_dcmotor,gcbh)
```

`n =`

Node with properties:

```
id: 'Inertia'
w: [1x1 simscape.logging.Node]
t: [1x1 simscape.logging.Node]
I: [1x1 simscape.logging.Node]
```

`n` is the Node object corresponding to the selected block.

Find Node Using Full Block Path Name

Open the Permanent Magnet DC Motor example model, which already has data logging enabled, and run the simulation to create the simulation log variable `simlog_ssc_dcmotor` (as specified by the **Workspace variable name** model configuration parameter) in your current workspace:

```
ssc_dcmotor
sim('ssc_dcmotor');
```

Find node corresponding to the Inertia block in the DC Motor subsystem:

```
n = simscape.logging.findNode(simlog_ssc_dcmotor, 'ssc_dcmotor/DC Motor/Inertia')
n =
Node with properties:
    id: 'Inertia'
    w: [1x1 simscape.logging.Node]
    t: [1x1 simscape.logging.Node]
    I: [1x1 simscape.logging.Node]
```

`n` is the Node object corresponding to the Inertia block in the DC Motor subsystem.

Find node corresponding to the DC Motor subsystem:

```
m = simscape.logging.findNode(simlog_ssc_dcmotor, 'ssc_dcmotor/DC Motor')
m =
Node with properties:
    id: 'DC_Motor'
    Rotor_Resistance: [1x1 simscape.logging.Node]
    Rotor_Inductance: [1x1 simscape.logging.Node]
    Rotational_Electromechanical_Converter: [1x1 simscape.logging.Node]
    Inertia: [1x1 simscape.logging.Node]
    Friction: [1x1 simscape.logging.Node]
```

`m` is the Node object corresponding to the whole DC Motor subsystem.

Input Arguments

simlog — Simulation log variable

Node object

Simulation log workspace variable that contains the logged model simulation data, specified as a Node object. You specify the name of the simulation log variable by using the **Workspace variable name** parameter in the **Simscape** pane of the Configuration Parameters dialog box.

block — Block name or identifier

handle | character vector | string scalar | Simulink.Block object | SID

Block or subsystem name or identifier, specified as a handle, full path to a block or subsystem in the model, Simulink.Block object, or a valid Simulink identifier (SID).

Data Types: double | char | string

Output Arguments

node — Node in the simulation data log tree corresponding to the specified block

Node object

Node in the simulation data log tree corresponding to the specified block, returned as a Node object. The Node object, which is of class `simscape.logging.Node`, contains logged simulation data for the specified block. Returns empty [] if the node is not found.

See Also

`simscape.logging.findPath`

Topics

“About Simulation Data Logging”

“Data Logging Options”

Introduced in R2020a

simscape.logging.findPath

Find path to node in logged simulation data tree

Syntax

```
[isvalid nodepath] = simscape.logging.findPath(simlog,block)
```

Description

`[isvalid nodepath] = simscape.logging.findPath(simlog,block)` returns a logical value and a path to the node in the simulation data tree `simlog`. The node contains logged simulation data for the specified block or subsystem in a model. Before you call this function, you must load the model. You must also have the simulation log variable in your current workspace. Create the simulation log variable by simulating the model with data logging turned on, or load a previously saved variable from a file.

Examples

Find Path for the Current Block Node

Open the Permanent Magnet DC Motor example model, which already has data logging enabled, and run the simulation to create the simulation log variable `simlog_ssc_dcmotor` (as specified by the **Workspace variable name** model configuration parameter) in your current workspace:

```
ssc_dcmotor
sim('ssc_dcmotor');
```

Open the DC Motor subsystem and select the Inertia block.

Find path to the node corresponding to the selected block:

```
[a, b] = simscape.logging.findPath(simlog_ssc_dcmotor,gcbh)
```

```
a =
```

```
1
```

```
b =
```

```
DC_Motor.Inertia
```

`a` returns 1, indicating that the valid path to the node was found. `b` is a character vector containing the path in the simulation log variable to the Node object corresponding to the selected block.

Find Path to the Node Using Full Block Path Name

Open the Permanent Magnet DC Motor example model, which already has data logging enabled, and run the simulation to create the simulation log variable `simlog_ssc_dcmotor` (as specified by the **Workspace variable name** model configuration parameter) in your current workspace:

```
ssc_dcmotor
sim('ssc_dcmotor');
```

Find path to the node corresponding to the Inertia block in the DC Motor subsystem:

```
[a, b] = simscape.logging.findPath(simlog_ssc_dcmotor, 'ssc_dcmotor/DC Motor/Inertia')
```

```
a =
```

```
1
```

```
b =
```

```
DC_Motor.Inertia
```

`a` returns `1`, indicating that the valid path to the node was found. `b` is a character vector containing the path in the simulation log variable to the `Node` object corresponding to the selected block.

Find path to the node corresponding to the top-level model:

```
[a1, b1] = simscape.logging.findPath(simlog_ssc_dcmotor, 'ssc_dcmotor')
```

```
a1 =
```

```
1
```

```
b1 =
```

```
''
```

`a1` returns `1`, indicating that the valid path to the node was found. `b1` is an empty character vector, because `ssc_dcmotor` is the name of the top-level model.

Input Arguments

simlog — Simulation log variable

Node object

Simulation log workspace variable that contains the logged model simulation data, specified as a `Node` object. You specify the name of the simulation log variable by using the **Workspace variable name** parameter on the **Simscape** pane of the Configuration Parameters dialog box.

block — Block name or identifier

handle | character vector | string scalar | `Simulink.Block` object | SID

Block or subsystem name or identifier, specified as a handle, full path to a block or subsystem in the model, `Simulink.Block` object, or a valid Simulink Identifier (SID).

Data Types: double | char | string

Output Arguments

isValid — Logical value indicating whether the match between block and node is found

0 | 1

Logical value indicating whether the match between block and node is found, returned as true (1) or false (0). Returns true (1) if the simulation data log tree contains a node corresponding to the specified block. Returns false (0) if a matching node was not found. The function can return false if the model is configured to log data only for selected blocks (rather than for the whole model) and the specified block was not selected for logging. The function can also return false if the specified block does not produce logged simulation data (for example, a Solver Configuration block or a scope).

nodepath — Path to the corresponding node in the simulation data log tree

character vector

Path to the node containing logged simulation data for the specified block, returned as a character vector. If `isValid` returns false (0), then `nodepath` is an empty character vector. If `block` is the top-level model in the block diagram, then `nodepath` is also an empty character vector, but `isValid` returns true (1).

See Also

`simscape.logging.findNode`

Topics

“About Simulation Data Logging”

“Data Logging Options”

Introduced in R2020a

simscape.logging.sli.findNode

(To be removed) Find Node object corresponding to block or subsystem

Note `simscape.logging.sli.findNode` will be removed in a future release. Use `simscape.logging.findNode` instead. Syntax and arguments of the two functions are identical.

Syntax

```
node = simscape.logging.sli.findNode(simlog,block)
```

Description

`node = simscape.logging.sli.findNode(simlog,block)` returns a `simscape.logging.Node` object that contains the logged simulation data for the specified block or subsystem in a model. Before you call this function, you must load the model. You must also have the simulation log variable in your current workspace. Create the simulation log variable by simulating the model with data logging turned on, or load a previously saved variable from a file.

Examples

Find Node for the Current Block

Open the Permanent Magnet DC Motor example model, which already has data logging enabled, and run the simulation to create the simulation log variable `simlog_ssc_dcmotor` (as specified by the **Workspace variable name** model configuration parameter) in your current workspace:

```
ssc_dcmotor
sim('ssc_dcmotor');
```

Open the DC Motor subsystem and select the Inertia block.

Find node corresponding to the selected block:

```
n = simscape.logging.sli.findNode(simlog_ssc_dcmotor,gcbh)
```

```
n =
```

```
Node with properties:
```

```
id: 'Inertia'
w: [1x1 simscape.logging.Node]
t: [1x1 simscape.logging.Node]
I: [1x1 simscape.logging.Node]
```

`n` is the `Node` object corresponding to the selected block.

Find Node Using Full Block Path Name

Open the Permanent Magnet DC Motor example model, which already has data logging enabled, and run the simulation to create the simulation log variable `simlog_ssc_dcmotor` (as specified by the **Workspace variable name** model configuration parameter) in your current workspace:

```
ssc_dcmotor
sim('ssc_dcmotor');
```

Find node corresponding to the Inertia block in the DC Motor subsystem:

```
n = Simscape.Logging.SLI.findNode(simlog_ssc_dcmotor, 'ssc_dcmotor/DC Motor/Inertia')
```

```
n =
```

```
Node with properties:
```

```
id: 'Inertia'
w: [1x1 Simscape.Logging.Node]
t: [1x1 Simscape.Logging.Node]
I: [1x1 Simscape.Logging.Node]
```

`n` is the `Node` object corresponding to the Inertia block in the DC Motor subsystem.

Find node corresponding to the DC Motor subsystem:

```
m = Simscape.Logging.SLI.findNode(simlog_ssc_dcmotor, 'ssc_dcmotor/DC Motor')
```

```
m =
```

```
Node with properties:
```

```
id: 'DC_Motor'
Rotor_Resistance: [1x1 Simscape.Logging.Node]
Rotor_Inductance: [1x1 Simscape.Logging.Node]
Rotational_Electromechanical_Converter: [1x1 Simscape.Logging.Node]
Inertia: [1x1 Simscape.Logging.Node]
Friction: [1x1 Simscape.Logging.Node]
```

`m` is the `Node` object corresponding to the whole DC Motor subsystem.

Input Arguments

`simlog` — Simulation log variable

Node object

Simulation log workspace variable that contains the logged model simulation data, specified as a Node object. You specify the name of the simulation log variable by using the **Workspace variable name** parameter in the **Simscape** pane of the Configuration Parameters dialog box.

`block` — Block name or identifier

handle | character vector | string scalar | `Simulink.Block` object | SID

Block or subsystem name or identifier, specified as a handle, full path to a block or subsystem in the model, `Simulink.Block` object, or a valid Simulink identifier (SID).

Data Types: double | char | string

Output Arguments

node — Node in the simulation data log tree corresponding to the specified block

Node object

Node in the simulation data log tree corresponding to the specified block, returned as a Node object. The Node object, which is of class `simscape.logging.Node`, contains logged simulation data for the specified block. Returns empty [] if the node is not found.

Compatibility Considerations

simscape.logging.sli.findNode will be removed

Warns starting in R2020a

`simscape.logging.sli.findNode` will be removed in a future release. Use `simscape.logging.findNode` instead. Syntax and arguments of the two functions are identical, but the new function is more efficient.

To update your code, just remove ".sli" from function calls.

See Also

`simscape.logging.sli.findPath`

Topics

"About Simulation Data Logging"

"Data Logging Options"

Introduced in R2015a

simscape.logging.sli.findPath

(To be removed) Find path to node in logged simulation data tree

Note `simscape.logging.sli.findPath` will be removed in a future release. Use `simscape.logging.findPath` instead. Syntax and arguments of the two functions are identical.

Syntax

```
[isvalid nodepath] = simscape.logging.sli.findPath(simlog,block)
```

Description

`[isvalid nodepath] = simscape.logging.sli.findPath(simlog,block)` returns a logical value and a path to the node in the simulation data tree `simlog`. The node contains logged simulation data for the specified block or subsystem in a model. Before you call this function, you must load the model. You must also have the simulation log variable in your current workspace. Create the simulation log variable by simulating the model with data logging turned on, or load a previously saved variable from a file.

Examples

Find Path for the Current Block Node

Open the Permanent Magnet DC Motor example model, which already has data logging enabled, and run the simulation to create the simulation log variable `simlog_ssc_dcmotor` (as specified by the **Workspace variable name** model configuration parameter) in your current workspace:

```
ssc_dcmotor
sim('ssc_dcmotor');
```

Open the DC Motor subsystem and select the Inertia block.

Find path to the node corresponding to the selected block:

```
[a, b] = simscape.logging.sli.findPath(simlog_ssc_dcmotor,gcbh)
```

```
a =
```

```
    1
```

```
b =
```

```
DC_Motor.Inertia
```

`a` returns `1`, indicating that the valid path to the node was found. `b` is a character vector containing the path in the simulation log variable to the Node object corresponding to the selected block.

Find Path to the Node Using Full Block Path Name

Open the Permanent Magnet DC Motor example model, which already has data logging enabled, and run the simulation to create the simulation log variable `simlog_ssc_dcmotor` (as specified by the **Workspace variable name** model configuration parameter) in your current workspace:

```
ssc_dcmotor
sim('ssc_dcmotor');
```

Find path to the node corresponding to the Inertia block in the DC Motor subsystem:

```
[a, b] = simscape.logging.sli.findPath(simlog_ssc_dcmotor, 'ssc_dcmotor/DC Motor/Inertia')
```

```
a =
```

```
1
```

```
b =
```

```
DC_Motor.Inertia
```

`a` returns `1`, indicating that the valid path to the node was found. `b` is a character vector containing the path in the simulation log variable to the `Node` object corresponding to the selected block.

Find path to the node corresponding to the top-level model:

```
[a1, b1] = simscape.logging.sli.findPath(simlog_ssc_dcmotor, 'ssc_dcmotor')
```

```
a1 =
```

```
1
```

```
b1 =
```

```
''
```

`a1` returns `1`, indicating that the valid path to the node was found. `b1` is an empty character vector, because `ssc_dcmotor` is the name of the top-level model.

Input Arguments

simlog — Simulation log variable

Node object

Simulation log workspace variable that contains the logged model simulation data, specified as a `Node` object. You specify the name of the simulation log variable by using the **Workspace variable name** parameter on the **Simscape** pane of the Configuration Parameters dialog box.

block — Block name or identifier

handle | character vector | string scalar | `Simulink.Block` object | SID

Block or subsystem name or identifier, specified as a handle, full path to a block or subsystem in the model, `Simulink.Block` object, or a valid Simulink identifier (SID).

Data Types: double | char | string

Output Arguments

isValid — Logical value indicating whether the match between block and node is found

0 | 1

Logical value indicating whether the match between block and node is found, returned as true (1) or false (0). Returns true (1) if the simulation data log tree contains a node corresponding to the specified block. Returns false (0) if a matching node was not found. The function can return false if the model is configured to log data only for selected blocks (rather than for the whole model) and the specified block was not selected for logging. The function can also return false if the specified block does not produce logged simulation data (for example, a Solver Configuration block or a scope).

nodepath — Path to the corresponding node in the simulation data log tree

character vector

Path to the node containing logged simulation data for the specified block, returned as a character vector. If `isValid` returns false (0), then `nodepath` is an empty character vector. If `block` is the top-level model in the block diagram, then `nodepath` is also an empty character vector, but `isValid` returns true (1).

Compatibility Considerations

simscape.logging.sli.findPath will be removed

Warns starting in R2020a

`simscape.logging.sli.findPath` will be removed in a future release. Use `simscape.logging.findPath` instead. Syntax and arguments of the two functions are identical, but the new function is more efficient.

To update your code, just remove ".sli" from function calls.

See Also

`simscape.logging.sli.findNode`

Topics

"About Simulation Data Logging"

"Data Logging Options"

Introduced in R2015a

simscape.op.create

Package: `simscape.op`

Create operating point by extracting data from model or from logged simulation data

Syntax

```
op = simscape.op.create(simlog,t)
op = simscape.op.create(block,simPhase)
op = simscape.op.create(block,simPhase, true)
```

Description

`op = simscape.op.create(simlog,t)` creates an `OperatingPoint` object `op` by extracting variable targets from logged simulation data at time `t`. If the set of times recorded in the simulation data log `simlog` contains an exact match for time `t`, then the function extracts these variable target values into the operating point data. If there is no exact match, but `t` is between the minimum and maximum times of `simlog`, then the function uses linear interpolation to determine the target values. If `t` is less than the minimum time, then the function extracts the first value for each variable in `simlog`. Similarly, if `t` is greater than the maximum time, then the function extracts the last value in `simlog`.

When you log simulation data in the Simulation Data Inspector, the simulation log does not contain private Simscape language data. Therefore, if you extract an operating point from data logged using the Simulation Data Inspector, private data is not included. For all other methods of creating an operating point, whether from a model or from simulation data logged to memory or disk, private data is included by default.

`op = simscape.op.create(block,simPhase)` creates an `OperatingPoint` object `op` by extracting variable targets from the whole model, or from a specific block or subsystem, at the specified model simulation phase.

`op = simscape.op.create(block,simPhase, true)` creates an `OperatingPoint` object `op` by extracting cached values of variable targets from a model that has been previously initialized or simulated. This method lets you save time by avoiding repeated initialization of the model if the data that you want to extract has not changed. The function returns an error if the model has not been updated, initialized, or simulated earlier in the session.

Examples

Extract Variable Targets from Simulation Log

Open the Permanent Magnet DC Motor example model, which already has data logging enabled. Run the simulation to create the simulation log variable `simlog_ssc_dcmotor` (as specified by the **Workspace variable name** model configuration parameter) in your current workspace:

```
ssc_dcmotor
sim('ssc_dcmotor');
```

Create an `OperatingPoint` object named `op1` from logged simulation data at 0.1 seconds after the start of simulation:

```
op1 = Simscape.OP.create(simlog_ssc_dcmotor, 0.1)
```

```
op1 =
```

```
OperatingPoint with children:
```

```
-----  
DC Motor  
DC Voltage  
ERef  
Load Torque  
MRRef Motor  
MRRef Torque  
Sensing  
-----
```

Extract Variable Targets from Model

Open the Permanent Magnet DC Motor example model and create an `OperatingPoint` object named `op2` using the `Start` values from the model:

```
ssc_dcmotor  
op2 = Simscape.OP.create(gcs, 'Start')
```

```
op2 =
```

```
OperatingPoint with children:
```

```
-----  
DC Motor  
DC Voltage  
ERef  
Load Torque  
MRRef Motor  
MRRef Torque  
Sensing  
-----
```

Extract Variable Targets from Cached Model Data

Initializing a model takes time. If you work with a large model, you can avoid unnecessary repeated initialization by using cached values of variable targets.

Open the Permanent Magnet DC Motor example model and simulate it:

```
ssc_dcmotor  
sim('ssc_dcmotor');
```

Create an `OperatingPoint` object named `op3` using the `Start` values for the DC Motor subsystem without reinitializing the model:

```
op3 = Simscape.OP.create('ssc_dcmotor/DC Motor', 'Start', true)
```

```

op3 =

  OperatingPoint with children:
  -----
  Friction
  Inertia
  Rotational Electromechanical Converter
  Rotor Inductance
  Rotor Resistance
  -----

```

Extract Variable Targets from Block or Subsystem

When you extract operating point data for a block or subsystem in a model, you cannot immediately use this operating point to initialize that block or subsystem, because of the mismatch in the data structure hierarchy. Use the `relativePath` function to determine the correct location, and then add the necessary layers for inserting this data in the operating point for the current model.

Open the Permanent Magnet DC Motor example model.

```
ssc_dcmotor
```

Open the DC Motor subsystem, select the Inductor block, and create an `OperatingPoint` object named `opRI` using the `Start` values from the model:

```
opRI = simscape.op.create(gcf, 'Start')
```

```

opRI =

  OperatingPoint with children:
  -----
  i      1.5000e-09|'A'  None
  i_L      0|'A'      High
  v      1.5000|'V'  None
  -----
  n
  p
  -----

```

Change the initialization target for the Inductor current variable, `i_L`:

```
t = simscape.op.Target(1.5, 'A', 'High');
opRI = set(opRI, 'i_L', t)
```

```

opRI =

  OperatingPoint with children:
  -----
  i      1.5000e-09|'A'  None
  i_L      1.5000|'A'  High
  v      1.5000|'V'  None
  -----
  n
  p
  -----

```

To use the new target for block initialization, you need to create an operating point for the whole model and insert the block operating point at the correct location.

Create an empty `OperatingPoint` object named `opModel`:

```
opModel = simscape.op.OperatingPoint
opModel =
    OperatingPoint with no children.
```

Set the `Identifier` property of the `OperatingPoint` object to match the name of the model and find the relative path for the Inductor block:

```
opModel.Identifier = bdroot(gcf);
relPath = relativePath(opModel, gcb)
relPath =
    'DC Motor/Rotor Inductance'
```

Add the `OperatingPoint` object `opRI` to the `OperatingPoint` object `opModel`:

```
opModel = set(opModel, relPath, opRI)
opModel =
    OperatingPoint with children:
    -----
    DC Motor
    -----
```

The command inserted the data at the location defined by `relPath`, adding the nodes to the data tree, as necessary.

You can now use the `opModel` operating point to initialize the model and apply the new target to the Inductor block.

Input Arguments

simlog — Simulation log variable or node

`simscape.logging.Node` object

Simulation log workspace variable that contains the logged model simulation data, or a node of this variable, specified as a `simscape.logging.Node` object. You specify the name of the simulation log variable by using the **Workspace variable name** parameter in the **Simscape** pane of the Configuration Parameters dialog box.

t — Simulation time

real number

Simulation time for data extraction, specified as a real number.

Data Types: double

block — Block name or identifier

handle | character vector | string scalar | `Simulink.Block` object | SID

Block, subsystem, or model name or identifier, specified as a handle, model name, full path to a block or subsystem in the model, `Simulink.Block` object, or a valid Simulink identifier (SID).

Data Types: `double` | `char` | `string`

simPhase — Model simulation phase

'Start' | 'Prestart'

Model simulation phase for data extraction, specified as one of:

- 'Start' — The function initializes the root model and extracts the variable targets for the whole model, or for the specified block or subsystem, into the operating point data. These targets correspond to Start values in the Variable Viewer.
- 'Prestart' — The function updates the root model and extracts the target values for the whole model, or for the specified block or subsystem, before initializing the model. These targets correspond to Prestart values in the Variable Viewer.

If the model already uses an operating point for initialization, then the function applies the targets in that `OperatingPoint` to the model during both of these phases, and reflects the results in the extracted `OperatingPoint`, `op`.

Data Types: `char`

Output Arguments

op — Operating point

`OperatingPoint` object

Operating point in the base workspace, returned as an `OperatingPoint` object, with variable initialization data extracted from the model or from logged simulation data.

See Also

`hasPrivateData` | `relativePath` | `set` | `simscape.op.OperatingPoint` | `simscape.op.Target`

Topics

"Initialize Model Using Operating Point from Logged Simulation Data"

"Using Operating Point Data for Model Initialization"

"Data Logging"

"Variable Viewer"

Introduced in R2017b

simscape.op.OperatingPoint

Operating point object containing hierarchical target data for variable initialization

Description

`OperatingPoint` objects let you save sets of data necessary to initialize a model, manipulate this data, and then use it to initialize another model, or the same model before another simulation run. These sets of data contain a hierarchy of operating point targets, each target consisting of a variable value, unit, and initialization priority.

Creation

There are several ways to create an `OperatingPoint` object:

- The `simscape.op.OperatingPoint` function (described here) creates an empty `OperatingPoint` object. You can then create `Target` objects and add them to the `OperatingPoint`.
- Instead of adding targets one by one, you can create an `OperatingPoint` object by extracting data from an existing model or from logged simulation data, by using the `simscape.op.create` function.

Syntax

```
op = simscape.op.OperatingPoint
```

Description

`op = simscape.op.OperatingPoint` creates an empty `OperatingPoint` object.

Properties

Identifier — Operating point Simulink identifier (SID)

character vector | string scalar

Simulink identifier (SID) of the `OperatingPoint` object, specified as a character vector or string scalar.

You do not have to set this property to be able to use an operating point for model initialization. In other words, you can initialize model A by using operating point B (or with an empty `Identifier`), as long as the `OperatingPoint` hierarchy matches the model.

For the `relativePath` function to work, the identifier of the operating point must match the name (SID) of the model. If you create an operating point by extracting data from log or model, the extraction algorithms set this property to match the SID of the model or block.

ChildIds — Names of immediate children

cell array

Names of the immediate children of the `OperatingPoint` object, specified as a cell array. These are the names of variables, blocks, or subsystems that comprise the next layer of the operating point hierarchy.

Children — Cell array of immediate children

cell array

Immediate children of the `OperatingPoint` object, specified as a cell array. These are operating point nodes or targets that correspond to the child IDs.

Attributes — Map of operating point attributes

character vector | string scalar | boolean | numeric

Map of the `OperatingPoint` object attributes, specified as a character vector, string scalar, Boolean, or numeric, with the `KeyType` of `char`. For more information, see “Map Containers”.

You can use these attributes to tag operating points and targets with useful metadata. If you create an operating point by extracting data from log or model, the extraction algorithms set the attributes, for example, a Boolean describing whether the target is differential or algebraic. Use this data for filtering out elements of interest.

Object Functions

<code>set</code>	Add or update element of operating point
<code>get</code>	Access element of operating point data tree
<code>relativePath</code>	Get path to node associated with block or subsystem
<code>hasPath</code>	Determine whether operating point data contains element at specified path
<code>remove</code>	Remove element from operating point
<code>move</code>	Move element from one path to another
<code>merge</code>	Create operating point by merging data from two operating points
<code>hasPrivateData</code>	Determine whether operating point data contains private data elements
<code>removePrivateData</code>	Remove private data elements from operating point

Examples

Create an Operating Point and Add Target

Create an empty `OperatingPoint` object named `op`:

```
op = simscape.op.OperatingPoint
op =
    OperatingPoint with no children.
```

Create a `Target` object named `t`, consisting of a variable value, unit, and initialization priority:

```
t = simscape.op.Target(1.5, 'V', 'High')
t =
    Target with properties:
        Description: ''
```

```
Value: 1.5000
Unit: 'V'
Priority: 'High'
Attributes: [0x1 containers.Map]
```

Add the target `t` to the operating point `op` by assigning this target to the variable named `v0`:

```
op = set(op, 'v0', t)
```

```
op =
```

```
OperatingPoint with children:
-----
v0 1.5000|'V' High
-----
```

You can create other `Target` objects or `OperatingPoint` objects and add them as children to the operating point `op`.

See Also

`simscape.op.Target` | `simscape.op.create`

Topics

“Using Operating Point Data for Model Initialization”

“Initialize Model Using Operating Point from Logged Simulation Data”

Introduced in R2017b

set

Package: `simscape.op`

Add or update element of operating point

Syntax

```
opNew = set(op, opPath, newElement)
```

Description

`opNew = set(op, opPath, newElement)` returns a copy of the `OperatingPoint` object `op`, with element `newElement` added at the specified location in the data tree hierarchy. The new element can be either another `OperatingPoint` or a `Target`. If the element already exists in the operating point, its content is replaced.

Examples

Add Element to an Operating Point

Open the Permanent Magnet DC Motor example model and create an `OperatingPoint` object named `op` using the `Start` values from the model:

```
ssc_dcmotor
op = simscape.op.create(gcs, 'Start')
```

```
op =
```

```
OperatingPoint with children:
-----
DC Motor
DC Voltage
ERef
Load Torque
MRRef Motor
MRRef Torque
Sensing
-----
```

Create a `Target` object named `t`, consisting of a variable value, unit, and initialization priority:

```
t = simscape.op.Target(1.5, 'A', 'High')
```

```
t =
```

```
Target with properties:
Description: ''
Value: 1.5000
Unit: 'A'
Priority: 'High'
Attributes: [0x1 containers.Map]
```

Add the target `t` to the operating point `op` by assigning this target to the Inductor current variable, `i_L`, of the Rotor Inductance block in the DC Motor subsystem:

```
op = set(op, 'DC Motor/Rotor Inductance/i_L', t);
```

Input Arguments

op — Original operating point

`OperatingPoint` object

The original operating point in the workspace, specified as an `OperatingPoint` object, to which you are adding the new element.

opPath — Location where you want to add the new element

slash-delimited character vector or string scalar

Location where you want to add the new element, specified as a slash-delimited character vector or string scalar. Define the location by the path through the data tree hierarchy of the original operating point, `op`. Separate the tree node names with slash symbols (`/`). You can use the `relativePath` function to determine the path.

Example: `'DC Motor/Rotor Resistance'`

Data Types: `char` | `string`

newElement — Element to be added to the original operating point

`OperatingPoint` object | `Target` object

Element to be added to the original `OperatingPoint` object, `op`, specified as an `OperatingPoint` or a `Target` object.

Output Arguments

opNew — New operating point

`OperatingPoint` object

New `OperatingPoint` object, which is a copy of the original `OperatingPoint` object, `op`, with `newElement` added at the `opPath` location. You can add elements recursively, that is, the name of the new `OperatingPoint` object, `opNew`, can be the same as the name of the original `OperatingPoint` object, `op`.

See Also

`get` | `relativePath` | `simscape.op.OperatingPoint` | `simscape.op.Target`

Topics

“Using Operating Point Data for Model Initialization”

“Initialize Model Using Operating Point from Logged Simulation Data”

Introduced in R2017b

get

Package: `simscape.op`

Access element of operating point data tree

Syntax

```
opElement = get(op, opPath)
```

Description

`opElement = get(op, opPath)` returns a copy of a node associated with the specified path `opPath` in the operating point `op` data tree. Depending on the path, the element can be either an `OperatingPoint` object or a `Target` object.

Examples

Copy Element from an Operating Point

Open the Permanent Magnet DC Motor example model and create an `OperatingPoint` object named `op` using the **Start** values from the model:

```
ssc_dcmotor
op = simscape.op.create(gcs, 'Start')
```

```
op =
```

```
OperatingPoint with children:
```

```
-----
DC Motor
DC Voltage
ERef
Load Torque
MRRef Motor
MRRef Torque
Sensing
-----
```

Open the DC Motor subsystem, select the Inductor block, and find the relative path to this block in the operating point data hierarchy:

```
relPath = relativePath(op, gcb)
```

```
relPath =
```

```
    'DC Motor/Rotor Inductance'
```

Copy the block data into a new operating point, `opRI`:

```
opRI = get(op, relPath)
```

```

opRI =

OperatingPoint with children:
-----
 i      1.5000e-09 | 'A'  None
 i_L    0          | 'A'  High
 v      1.5000    | 'V'  None
-----
 n
 p
-----

```

Input Arguments

op — Operating point

OperatingPoint object

Operating point in the workspace, specified as an OperatingPoint object, from which you are copying an element.

opPath — Location associated with the element to copy

slash-delimited character vector or string scalar

Location associated with the element to copy, specified as a slash-delimited character vector or string scalar. Define the location by the relative path through the data tree hierarchy of the operating point, op, starting below the root node. The root node is specified by the Identifier property of the OperatingPoint object. Separate the tree node names with slash symbols (/).

Example: 'DC Motor/Rotor Resistance'

Data Types: char | string

Output Arguments

opElement — Element copied from the operating point

OperatingPoint object | Target object

New OperatingPoint or Target object, which is a copy of the op element at the opPath location.

See Also

relativePath | set | simscape.op.OperatingPoint | simscape.op.Target

Topics

“Using Operating Point Data for Model Initialization”

“Initialize Model Using Operating Point from Logged Simulation Data”

Introduced in R2017b

relativePath

Package: `simscape.op`

Get path to node associated with block or subsystem

Syntax

```
opPath = relativePath(op,block)
```

Description

`opPath = relativePath(op,block)` returns a path from the root node of the operating point data tree `op` to the node associated with a given block or subsystem, `block`. The root node is specified by the `Identifier` property of the `OperatingPoint` object.

Examples

Find Relative Path to Block Node in Operating Point Data Tree

Open the Permanent Magnet DC Motor example model and create an `OperatingPoint` object named `op` using the `Start` values from the model:

```
ssc_dcmotor
op = simscape.op.create(gcs, 'Start')
```

```
op =
```

```
OperatingPoint with children:
-----
DC Motor
DC Voltage
ERef
Load Torque
MRRef Motor
MRRef Torque
Sensing
-----
```

Open the DC Motor subsystem, select the Inductor block, and find the relative path to this block in the operating point data hierarchy:

```
relPath = relativePath(op, gcb)
```

```
relPath =
```

```
'DC Motor/Rotor Inductance'
```

Find Relative Path to Block in an Empty Operating Point

When you determine relative path for a block, the operating point does not need to contain a node corresponding to that block. You can obtain operating point data for a block elsewhere, and then use

the `relativePath` function to determine the correct location for inserting this data in the operating point for the current model.

Open the Permanent Magnet DC Motor example model:

```
ssc_dcmotor
```

Create an empty `OperatingPoint` object named `op`:

```
op = simscape.op.OperatingPoint
```

```
op =
```

```
    OperatingPoint with no children.
```

Set the `Identifier` property of the `OperatingPoint` object to match the name of the model:

```
op.Identifier = 'ssc_dcmotor';
```

In the model, open the DC Motor subsystem, select the Inductor block, and find the relative path to this block in the operating point data hierarchy:

```
relPath = relativePath(op, gcb)
```

```
relPath =
```

```
    'DC Motor/Rotor Inductance'
```

Input Arguments

op — Operating point

`OperatingPoint` object

Operating point in the workspace, specified as an `OperatingPoint` object. The `Identifier` property of the `OperatingPoint` object must match the name of the model.

block — Block name or identifier

handle | character vector | string scalar | SID

Block or subsystem name or identifier, specified as a handle, full path to a block or subsystem in the model, or a valid Simulink identifier (SID).

Data Types: double | char | string

Output Arguments

opPath — Path to the corresponding node in the operating point data tree

character vector

Path to the node corresponding to the specified block, returned as a character vector, relative to the root node of the operating point data tree. The root node is specified by the `Identifier` property of the `OperatingPoint` object.

See Also

get | hasPath | set | `simscape.op.OperatingPoint`

Topics

“Using Operating Point Data for Model Initialization”

“Initialize Model Using Operating Point from Logged Simulation Data”

Introduced in R2017b

hasPath

Package: `simscape.op`

Determine whether operating point data contains element at specified path

Syntax

```
hasPath(op,opPath)
```

Description

`hasPath(op,opPath)` returns true (1) if the operating point data tree contains a node corresponding to the specified relative path, `opPath`. Returns false (0) if a matching node was not found.

Examples

Find Whether Operating Point Contains Data for a Block

When you determine relative path for a block, the operating point does not need to contain a node corresponding to that block. Use the `hasPath` function to determine whether the operating point contains data at the specified location.

Open the Permanent Magnet DC Motor example model and create an `OperatingPoint` object named `op` using the Start values from the model:

```
ssc_dcmotor
op = simscape.op.create(gcs, 'Start')
```

```
op =
```

```
OperatingPoint with children:
```

```
-----
DC Motor
DC Voltage
ERef
Load Torque
MRRef Motor
MRRef Torque
Sensing
-----
```

Open the DC Motor subsystem, select the Inductor block, and find the relative path to this block in the operating point data hierarchy:

```
relPath = relativePath(op, gcb)
```

```
relPath =
```

```
'DC Motor/Rotor Inductance'
```


Now determine whether the operating point contains data for this block:

```
hasPath(op, relPath)

ans =

    logical

    1
```

Input Arguments

op — Operating point

OperatingPoint object

Operating point in the workspace, specified as an OperatingPoint object.

opPath — Relative path to element

slash-delimited character vector or string scalar

Relative path in the operating point data tree, specified as a slash-delimited character vector or string scalar. Use the `relativePath` function to determine the path to an element. The element can be a subsystem, block, or variable target.

Data Types: char | string

See Also

`relativePath` | `simscape.op.OperatingPoint`

Topics

“Using Operating Point Data for Model Initialization”

“Initialize Model Using Operating Point from Logged Simulation Data”

Introduced in R2017b

hasPrivateData

Package: `simscape.op`

Determine whether operating point data contains private data elements

Syntax

```
hasPrivateData(op)
```

Description

`hasPrivateData(op)` returns true (1) if the operating point data tree contains elements corresponding to the Simscape language file members with the attribute `ExternalAccess = none`. Returns false (0) if the operating point does not contain private data.

Simscape language members with the attribute `ExternalAccess = none` are not observable by definition. Therefore, you cannot see them in the operating point data, but their inclusion helps restore the simulation state of the model during initialization.

When you create an operating point by extracting it from a model or from logged simulation data, private data is included by default.

Examples

Find and Remove Private Data from Operating Point

In general, including private data in an operating point data tree helps with model initialization. However, if you are having trouble initializing a model from a saved operating point and wonder whether the issue is in the hidden private data, try using `removePrivateData` to remove the hidden elements.

Open the Permanent Magnet DC Motor example model and create an `OperatingPoint` object named `op` using the `Start` values from the model:

```
ssc_dcmotor
op = simscape.op.create(gcs, 'Start')
```

```
op =
```

```
OperatingPoint with children:
```

```
-----
DC Motor
DC Voltage
ERef
Load Torque
MRRef Motor
MRRef Torque
Sensing
-----
```

Determine whether the operating point contains private data:

```
hasPrivateData(op)
```

```
ans =
```

```
logical
```

```
1
```

Create a new operating point, `op1`, by removing private data from `op`:

```
op1 = removePrivateData(op)
```

```
op1 =
```

```
OperatingPoint with children:
```

```
-----
DC Motor
DC Voltage
ERef
Load Torque
MRRef Motor
MRRef Torque
Sensing
-----
```

Verify that the new operating point does not contain private data:

```
hasPrivateData(op1)
```

```
ans =
```

```
logical
```

```
0
```

Input Arguments

op — Operating point

`OperatingPoint` object

Operating point in the workspace, specified as an `OperatingPoint` object.

See Also

`removePrivateData` | `simscape.op.OperatingPoint`

Topics

“Using Operating Point Data for Model Initialization”

“Initialize Model Using Operating Point from Logged Simulation Data”

Introduced in R2018a

removePrivateData

Package: `simscape.op`

Remove private data elements from operating point

Syntax

```
opNew = removePrivateData(op)
```

Description

`opNew = removePrivateData(op)` returns a copy of the `OperatingPoint` object `op`, with hidden private data removed from the data tree hierarchy. Private data corresponds to the Simscape language file members with the attribute `ExternalAccess = none`.

Simscape language members with the attribute `ExternalAccess = none` are not observable by definition. Therefore, you cannot see them in the operating point data. Use `hasPrivateData` to determine whether an operating point data tree contains private data elements. Then use `removePrivateData` to remove the hidden elements, if necessary.

Examples

Find and Remove Private Data from Operating Point

In general, including private data in an operating point data tree helps with model initialization. However, if you are having trouble initializing a model from a saved operating point and wonder whether the issue is in the hidden private data, try using `removePrivateData` to remove the hidden elements.

Open the Permanent Magnet DC Motor example model and create an `OperatingPoint` object named `op` using the `Start` values from the model:

```
ssc_dcmotor
op = simscape.op.create(gcs, 'Start')
```

```
op =
```

```
OperatingPoint with children:
-----
DC Motor
DC Voltage
ERef
Load Torque
MRRef Motor
MRRef Torque
Sensing
-----
```

Determine whether the operating point contains private data:

```
hasPrivateData(op)
```

```
ans =
    logical
    1
```

Create a new operating point, `op1`, by removing private data from `op`:

```
op1 = removePrivateData(op)
op1 =
    OperatingPoint with children:
    -----
    DC Motor
    DC Voltage
    ERef
    Load Torque
    MRRef Motor
    MRRef Torque
    Sensing
    -----
```

Verify that the new operating point does not contain private data:

```
hasPrivateData(op1)
ans =
    logical
    0
```

Input Arguments

op — Operating point
OperatingPoint object

Operating point in the workspace, specified as an `OperatingPoint` object.

Output Arguments

opNew — New operating point
OperatingPoint object

New `OperatingPoint` object, which is a copy of the original `OperatingPoint` object, `op`, with private data elements removed from the data tree hierarchy. You can remove elements recursively, that is, the name of the new `OperatingPoint` object, `opNew`, can be the same as the name of the original `OperatingPoint` object, `op`.

See Also

`hasPrivateData` | `simscape.op.OperatingPoint`

Topics

“Using Operating Point Data for Model Initialization”

“Initialize Model Using Operating Point from Logged Simulation Data”

Introduced in R2018a

remove

Package: `simscape.op`

Remove element from operating point

Syntax

```
opNew = remove(op, opPath)
```

Description

`opNew = remove(op, opPath)` returns a copy of the `OperatingPoint` object `op`, with an element at the specified location `opPath` removed from the data tree hierarchy. The element can be either a node or a target.

Examples

Remove an Element from Operating Point Data

Open the Permanent Magnet DC Motor example model and create an `OperatingPoint` object named `op` using the **Start** values from the model:

```
ssc_dcmotor
op = simscape.op.create(gcs, 'Start')
```

```
op =
```

```
OperatingPoint with children:
```

```
-----
DC Motor
DC Voltage
ERef
Load Torque
MRRef Motor
MRRef Torque
Sensing
-----
```

Select the Load Torque block and find the relative path to this block in the operating point data hierarchy:

```
relPath = relativePath(op, gcb)
```

```
relPath =
```

```
    'Load Torque'
```

Now remove this element from the operating point:

```
op = remove(op, relPath)
```

```
op =  
  
OperatingPoint with children:  
-----  
DC Motor  
DC Voltage  
ERef  
MRRef Motor  
MRRef Torque  
Sensing  
-----
```

Operating point `op` no longer has the `Load Torque` child.

Input Arguments

op — Operating point

`OperatingPoint` object

Operating point in the workspace, specified as an `OperatingPoint` object.

opPath — Relative path to element

slash-delimited character vector or string scalar

Relative path in the operating point data tree, specified as a slash-delimited character vector or string scalar. Use the `relativePath` function to determine the path to an element. The element can be a subsystem, block, or variable target.

Data Types: `char` | `string`

Output Arguments

opNew — New operating point

`OperatingPoint` object

New `OperatingPoint` object, which is a copy of the original `OperatingPoint` object, `op`, with element at the location specified by `opPath` removed from the data tree hierarchy. You can remove elements recursively, that is, the name of the new `OperatingPoint` object, `opNew`, can be the same as the name of the original `OperatingPoint` object, `op`.

See Also

`relativePath` | `simscape.op.OperatingPoint`

Topics

“Using Operating Point Data for Model Initialization”

“Initialize Model Using Operating Point from Logged Simulation Data”

Introduced in R2017b

move

Package: `simscape.op`

Move element from one path to another

Syntax

```
opNew = move(op, oldPath, newPath)
```

Description

`opNew = move(op, oldPath, newPath)` returns a copy of the `OperatingPoint` object `op`, with an element at the specified location `oldPath` moved to the new location, specified by `newPath`. Use this function to update the operating point data after restructuring your model or renaming a block or subsystem.

Examples

Rename Element to Match New Block Name

When you rename a block in your model, use the `move` function to update the operating point data.

Open the Permanent Magnet DC Motor example model and create an `OperatingPoint` object named `op` using the `Start` values from the model:

```
ssc_dcmotor
op = simscape.op.create(gcs, 'Start')
```

```
op =
```

```
OperatingPoint with children:
```

```
-----
DC Motor
DC Voltage
ERef
Load Torque
MRRef Motor
MRRef Torque
Sensing
-----
```

Select the DC Voltage block and find the relative path to this block in the operating point data hierarchy:

```
oldPath = relativePath(op, gcb)
```

```
oldPath =
```

```
    'DC Voltage'
```

Rename the DC Voltage block to 1.5V.

Select the block again and find the new relative path:

```
newPath = relativePath(op, gcb)
newPath =
    '1.5V'
```

Update the operating point data hierarchy to reflect the new block name:

```
op = move(op, oldPath, newPath)
op =
    OperatingPoint with children:
    -----
    1.5V
    DC Motor
    ERef
    Load Torque
    MRRef Motor
    MRRef Torque
    Sensing
    -----
```

Operating point `op` now lists `1.5V` as its child.

Input Arguments

op — Operating point

`OperatingPoint` object

Original operating point in the workspace, specified as an `OperatingPoint` object.

oldPath — Relative path to original element

slash-delimited character vector or string scalar

Relative path to the original element in the operating point data tree, specified as a slash-delimited character vector or string scalar. The element can be a subsystem, block, or variable target.

Data Types: `char` | `string`

newPath — Relative path to new element

slash-delimited character vector or string scalar

Relative path to the new element in the operating point data tree, specified as a slash-delimited character vector or string scalar. The element can be a subsystem, block, or variable target.

Data Types: `char` | `string`

Output Arguments

opNew — New operating point

`OperatingPoint` object

New `OperatingPoint` object, which is a copy of the original `OperatingPoint` object, `op`, with element specified by `oldPath` moved to the new location, `newPath`. You can move elements

recursively, that is, the name of the new `OperatingPoint` object, `opNew`, can be the same as the name of the original `OperatingPoint` object, `op`.

See Also

`relativePath` | `simscape.op.OperatingPoint`

Topics

“Using Operating Point Data for Model Initialization”

“Initialize Model Using Operating Point from Logged Simulation Data”

Introduced in R2017b

merge

Package: `simscape.op`

Create operating point by merging data from two operating points

Syntax

```
opNew = merge(op1, op2)
```

Description

`opNew = merge(op1, op2)` creates a new `OperatingPoint` object `opNew`, with children from two `OperatingPoint` objects, `op1` and `op2`.

The function starts by copying all children from `op1` into `opNew`. Then, if a child ID exists in `op2` but not in `op1`, the function adds the child to `opNew`. If a child ID exists both in `op1` and `op2`, then:

- If both children are `OperatingPoint` objects, the function merges them according to the same rules.
- If at least one of the two children is a `Target` object, the function retains the child of `op1` and discards the child of `op2`.

Examples

Merge Two Operating Points

Create the first `OperatingPoint` object, `op1`.

```
t1 = simscape.op.Target(1.5, 'V', 'High');
op1 = simscape.op.OperatingPoint;
op1 = set (op1, 'V0', t1);
op1 = set (op1, 'V1', t1)
```

```
op1 =
```

```
OperatingPoint with children:
```

```
-----
V0  1.5000|'V'  High
V1  1.5000|'V'  High
-----
```

This operating point has two children, `Target` objects `V0` and `V1`.

Create the second `OperatingPoint` object, `op2`.

```
t2 = simscape.op.Target(1, 'V');
op2 = simscape.op.OperatingPoint;
op2 = set (op2, 'V1', t2);
op2 = set (op2, 'V2', t2)
```

```
op2 =
```

```

OperatingPoint with children:
-----
V1  1|'V'  None
V2  1|'V'  None
-----

```

This operating point has two children, Target objects V1 and V2.

Merge the two operating points into a new `OperatingPoint` object, `op`.

```
op = merge (op1, op2)
```

```
op =
```

```

OperatingPoint with children:
-----
V0  1.5000|'V'  High
V1  1.5000|'V'  High
V2           1|'V'  None
-----

```

The new operating point has two children from the first operating point, `V0` and `V1`, and the `V2` child from the second operating point. The `V1` target from the second operating point is discarded because it conflicts with the child ID existing in the first operating point.

Input Arguments

op1 — First operating point to be merged

`OperatingPoint` object

First operating point to be merged, specified as an `OperatingPoint` object.

op2 — Second operating point to be merged

`OperatingPoint` object

Second operating point to be merged, specified as an `OperatingPoint` object.

Output Arguments

opNew — New operating point

`OperatingPoint` object

New `OperatingPoint` object, which contains children from `op1` and `op2`.

See Also

`simscape.op.OperatingPoint` | `simscape.op.Target`

Topics

“Using Operating Point Data for Model Initialization”

“Initialize Model Using Operating Point from Logged Simulation Data”

Introduced in R2017b

simscape.op.Target

Variable initialization target object

Description

`OperatingPoint` objects let you save sets of data necessary to initialize a model, manipulate this data, and then use it to initialize another model, or the same model before another simulation run. These sets of data contain a hierarchy of operating point `Target` objects, each target consisting of a variable value, unit, and initialization priority.

Creation

Syntax

```
t = simscape.op.Target()  
t = simscape.op.Target(value)  
t = simscape.op.Target(value, unit)  
t = simscape.op.Target(value, unit, priority)
```

Description

`t = simscape.op.Target()` creates an empty `Target` object.

`t = simscape.op.Target(value)` creates a `Target` object with the `Value` property set to provided value.

`t = simscape.op.Target(value, unit)` creates a `Target` object with the `Value` property set to provided value and the `Unit` property set to provided unit expression.

`t = simscape.op.Target(value, unit, priority)` creates a `Target` object with the `Value` property set to provided value, the `Unit` property set to provided unit expression, and the `Priority` property set to provided variable initialization priority.

Properties

Value — Variable initialization target value

scalar | vector | matrix

Variable initialization target value, specified as a scalar, vector, or matrix.

Example: 1.5

Data Types: double | int32

Unit — Physical unit expression

character vector | string scalar

Physical unit expression, specified as a character vector or string scalar. The expression can consist of valid physical unit names, numbers, math operators, such as +, -, *, /, and ^, and parentheses to specify the order of operations.

Example: 'm/s^2'

Data Types: char | string

Priority — Variable initialization priority

'High' | 'Low' | 'None'

Variable initialization priority, specified as 'High', 'Low', or 'None'.

Data Types: char

Description — Descriptive name of target variable

character vector | string scalar

Descriptive name of target variable, specified as a character vector or string scalar. If you create an operating point by extracting data from log or model, the extraction algorithms populate this target property with the user-friendly description of the variable. However, you do not have to set this property to be able to use the target for model initialization.

Data Types: char | string

Examples

Add Target to an Operating Point

Create an empty `OperatingPoint` object named `op`:

```
op = simscape.op.OperatingPoint
```

```
op =
```

```
    OperatingPoint with no children.
```

Create a `Target` object named `t`, consisting of a variable value, unit, and initialization priority:

```
t = simscape.op.Target(1.5, 'V', 'High')
```

```
t =
```

```
    Target with properties:
```

```
    Description: ''
    Value: 1.5000
    Unit: 'V'
    Priority: 'High'
    Attributes: [0x1 containers.Map]
```

Add the target `t` to the operating point `op` by assigning this target to the variable named `v0`:

```
op = set(op, 'v0', t)
```

```
op =
```

```
OperatingPoint with children:  
-----  
v0 1.5000|'V' High  
-----
```

You can create other `Target` objects or `OperatingPoint` objects and add them as children to the operating point `op`.

See Also

`simscape.op.OperatingPoint`

Topics

“Initialize Model Using Operating Point from Logged Simulation Data”

Introduced in R2017b

sl_postprocess

Make postprocessing customizations when building custom block library

Syntax

```
sl_postprocess(h)
```

Description

`sl_postprocess(h)` takes a handle to the custom block library, `h`, and allows you to make library postprocessing customizations (for example, add a forwarding table).

If a Simscape file package being built contains a `sl_postprocess.m` file, then `ssc_build` calls `sl_postprocess` once the block library (`package_name_lib`) is generated but before it is saved to disk. If `sl_postprocess` generates an error, the library does not build.

You can include a `sl_postprocess.m` file at any level in the library package. At the top level, it makes postprocessing changes to the whole custom block library. Similarly, if the `sl_postprocess.m` file resides in a sublibrary in the package, it takes a handle to that sublibrary and makes the corresponding changes.

Examples

If you rename a block or change a parameter name, you need to add a forwarding table to update old models that reference the block. However, if you manually add a forwarding table to a custom library, it will get overwritten every time you rebuild the library. Instead, include a `sl_postprocess.m` file in the library package, which will add the forwarding table automatically upon rebuilding the library:

```
- +MySimscape
| -- sl_postprocess.m
| -- +Mechanical
| | -- spring.ssc
| | -- ...
```

The `sl_postprocess.m` file contains a forwarding table:

```
function sl_postprocess(h)
% Forwarding table for the spring block
ft = { {'MySimscape_lib/Mechanical/Ideal Spring', 'MySimscape_lib/Mechanical/Rotational Spring'} }
set_param(h, 'ForwardingTable', ft);
end
```

This forwarding table indicates that the custom block name for the `spring.ssc` component has changed from 'Ideal Spring' to 'Rotational Spring'.

Note that if you have customized the library names using `lib.m` files, you have to use these custom names in the forwarding table (for example, 'Mechanical Library' instead of 'Mechanical').

See Also

`ssc_build`

Introduced in R2010a

SpectrumAnalyzerConfiguration

Configure Spectrum Analyzer for programmatic access

Description

The `spbscopes.SpectrumAnalyzerConfiguration` object contains the scope configuration information for the Spectrum Analyzer block.

Creation

`MyScopeConfiguration = get_param(gcbh, 'ScopeConfiguration')` constructs a new Spectrum Analyzer Configuration object. You must first select the block in the model or give the full path to the block.

Properties

Frequently Used

NumInputPorts — Number of input ports

"1" (default) | character vector | string scalar

Number of input ports on a scope block, specified by a character vector or string scalar. Maximum number of input ports is 96.

UI Use

Select **File > Number of Input Ports**.

Data Types: char | string

SpectrumType — Type of spectrum to show

"Power" (default) | "Power density" | "RMS"

Specify the spectrum type to display.

"Power" — Power spectrum

"Power density" — Power spectral density. The power spectral density is the magnitude squared of the spectrum normalized to a bandwidth of 1 hertz.

"RMS" — Root mean square. The root-mean-square shows the square root of the mean square. This option is useful when viewing the frequency of voltage or current signals.

Tunable: Yes

UI Use

Open the **Spectrum Settings**. In the **Main options** section, set **Type**.

Data Types: char | string

SampleRateSource — Source of input sample rate

"Inherited" (default) | "Property"

Specify the source of the input sample rate as:

- "Inherited" — Spectrum Analyzer inherits the input sample rate from the model.
- "Property" — Specify the sample rate input directly using the SampleRate property.

UI Use

Open the **Spectrum Settings**. In the **Main options** section, in the **Sample rate (Hz)** combo box, enter a custom sample rate or select Inherited.

Data Types: char | string

SampleRate — Sample rate of input

"10e3" (default) | character vector | string scalar

Specify the sample rate of the input signals in hertz as a character vector or string scalar.

Dependency

To enable this property, set SampleRateSource to "Property".

UI Use

Open the **Spectrum Settings**. In the **Main options** section, enter a **Sample rate (Hz)** in the combo box.

Data Types: char | string

PlotAsTwoSidedSpectrum — Two-sided spectrum flag

false (default) | true

- true — Compute and plot two-sided spectral estimates. When the input signal is complex-valued, you must set this property to true.
- false — Compute and plot one-sided spectral estimates. If you set this property to false, then the input signal must be real-valued.

When this property is false, Spectrum Analyzer uses power-folding. The y-axis values are twice the amplitude that they would be if this property were set to true, except at 0 and the Nyquist frequency. A one-sided power spectral density (PSD) contains the total power of the signal in the frequency interval from DC to half of the Nyquist rate. For more information, see pwelch.

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, select **Two-sided spectrum**.

Data Types: logical

FrequencyScale — Frequency scale

"Linear" (default) | "Log"

- "Log" — displays the frequencies on the x-axis on a logarithmic scale. To use the "Log" setting, you must also set the PlotAsTwoSidedSpectrum property to false.
- "Linear" — displays the frequencies on the x-axis on a linear scale. To use the "Linear" setting, you must also set the PlotAsTwoSidedSpectrum property to true.

Tunable: Yes

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, set **Scale**.

Data Types: char | string

Advanced

RBWSource — Source of resolution bandwidth value

"Auto" (default) | "Property" | "InputPort"

Specify the source of the resolution bandwidth (RBW) as "Auto", "Property", or "InputPort".

- "Auto" — The Spectrum Analyzer adjusts the spectral estimation resolution to ensure that there are 1024 RBW intervals over the defined frequency span.
- "Property" — Specify the resolution bandwidth directly using the RBW property.
- "InputPort" — An input port is added to the Spectrum Analyzer block to read the RBW. This option is only applicable to frequency input.

UI Use

Open the **Spectrum Settings**. In the **Frequency input options** section, set **RBW (Hz)**.

Data Types: char | string

RBW — Resolution bandwidth

"9.76" (default) | character vector | string scalar

RBW controls the spectral resolution of the Spectrum Analyzer. Specify the resolution bandwidth in hertz as a character vector or string scalar. You must specify a value to ensure that there are at least two RBW intervals over the specified frequency span. Thus, the ratio of the overall span to RBW must be greater than two:

$$\frac{span}{RBW} > 2$$

Dependency

To enable, set:

- RBWSource to "Property"

UI Use

Open the **Spectrum Settings**. In the **Main options** section, set **RBW (Hz)**.

Data Types: char | string

OverlapPercent — Overlap percentage

"0" (default) | character vector of a real scalar | string scalar of a real scalar

The percentage overlap between the previous and current buffered data segments, specified as a character vector or string scalar of a real scalar. The overlap creates a window segment that is used to compute a spectral estimate. The value must be greater than or equal to zero and less than 100.

UI Use

Open the **Spectrum Settings**. In the **Window options** section, set **Overlap (%)**.

Data Types: char | string

Window — Window function

"Hann" (default) | "Rectangular"

Specify a window function for the spectral estimation. The following table shows preset windows. For more information, follow the link to the corresponding function reference in the Signal Processing Toolbox™ documentation.

Window Option	Corresponding Signal Processing Toolbox Function
"Rectangular"	rectwin
"Hann"	hann

UI Use

Open the **Spectrum Settings**. In the **Window options** section, set **Window**.

Data Types: char | string

SpectrumUnits — Units of the spectrum

"dBm" (default)

This property is read-only.

Specify the units in which the Spectrum Analyzer displays power values. To change the spectrum units you must have DSP System Toolbox.

AveragingMethod — Smoothing method

"Running" (default) | "Exponential"

Specify the smoothing method as:

- **Running** — Running average of the last n samples. Use the `SpectralAverages` property to specify n .
- **Exponential** — Weighted average of samples. Use the `ForgettingFactor` property to specify the weighted forgetting factor.

For more information about the averaging methods, see "Averaging Method" (DSP System Toolbox).

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, set **Averaging method**.

Data Types: char | string

SpectralAverages — Number of spectral averages

"1" (default) | character vector | string scalar

Specify the number of spectral averages as a character vector or string scalar. The Spectrum Analyzer computes the current power spectrum estimate by computing a running average of the last N power spectrum estimates. This property defines N .

Dependency

To enable this property, set `AveragingMethod` to "Running".

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, set **Averages**.

Data Types: char | string

ForgettingFactor — Weighting forgetting factor

"0.9" (default) | string scalar of scalar in the range (0,1] | character vector of scalar in the range (0,1]

Specify the exponential weighting as a scalar value greater than 0 and less than or equal to 1, specified as a string scalar or character vector.

Dependency

To enable this property, set `AveragingMethod` to "Exponential".

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, set **Forgetting factor**.

Data Types: char | string

ReferenceLoad — Reference load

"1" (default) | character vector of a real positive scalar | string scalar of a real positive scalar

The load the scope uses as a reference to compute power levels.

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, set **Reference load**.

Data Types: char | string

FrequencyOffset — Frequency offset

"0" (default) | numeric scalar character vector | numeric vector character vector | numeric scalar string scalar | numeric vector string scalar

- Numeric scalar (specified as a character vector or string scalar) — Apply the same frequency offset to all channels, specified in hertz as a character vector.
- Numeric vector (specified as a character vector or string scalar) — Apply a specific frequency offset for each channel, specify a vector of frequencies. The vector length must be equal to number of input channels.

The frequency-axis values are offset by the values specified in this property. The overall span must fall within the "Nyquist frequency interval" on page 1-656.

UI Use

Open the **Spectrum Settings**. In the **Trace options** section, set **Offset (Hz)**.

Data Types: char | string

TreatMby1SignalsAsOneChannel — Treat unoriented sample-based input signal as a column vector`true (default) | false`

Set this property to `true` to treat M -by-1 and unoriented sample-based inputs as a column vector, or one channel. Set this property to `false` to treat M -by-1 and unoriented sample-based inputs as a 1-by- M row vector.

Data Types: `logical`

Measurements**MeasurementChannel** — Channel for which measurements are obtained`"1" (default) | character vector | string scalar`

Channel over which the measurements are obtained, specified as a character vector or a string scalar which evaluates to a positive integer greater than 0 and less than or equal to 100. The maximum number you can specify is the number of channels (columns) in the input signal.

Tunable: Yes

UI Use

Click on **Tools** > **Measurements** and open the **Trace Selection** settings.

Data Types: `char` | `string`

PeakFinder — Peak finder measurement`PeakFinderSpecification` object

Enable peak finder to compute and display the largest calculated peak values. The `PeakFinder` property uses the `PeakFinderSpecification` properties.

The `PeakFinderSpecification` properties are:

- `MinHeight` -- Level above which peaks are detected, specified as a scalar value.
Default: `-Inf`
- `NumPeaks` -- Maximum number of peaks to show, specified as a positive integer scalar less than 100.
Default: `3`
- `MinDistance` -- Minimum number of samples between adjacent peaks, specified as a positive real scalar.
Default: `1`
- `Threshold` -- Minimum height difference between peak and its neighboring samples, specified as a nonnegative real scalar.
Default: `0`
- `LabelFormat` -- Coordinates to display next to the calculated peak value, specified as a character vector or a string scalar. Valid values are "X", "Y", or "X + Y".
Default: `"X + Y"`


- **Enable** -- Set this property to `true` to enable peak finder measurements. Valid values are `true` or `false`.

Default: `false`

All `PeakFinderSpecification` properties are tunable.

Tunable: Yes

UI Use

Open the **Peak Finder** pane () and modify the **Settings** options.

CursorMeasurements – Cursor measurements

`CursorMeasurementsSpecification` object

Enable cursor measurements to display screen or waveform cursors. The `CursorMeasurements` property uses the `CursorMeasurementsSpecification` properties.

The `CursorMeasurementsSpecification` properties are:

- **Type** -- Type of the display cursors, specified as either "Screen cursors" or "Waveform cursors".

Default: "Waveform cursors"

- **ShowHorizontal** -- Set this property to `true` to show the horizontal screen cursors. This property applies when you set the `Type` property to "Screen cursors".

Default: `true`

- **ShowVertical** -- Set this property to `true` to show the vertical screen cursors. This property applies when you set the `Type` property to "Screen cursors".

Default: `true`

- **Cursor1TraceSource** -- Specify the waveform cursor 1 source as a positive real scalar. This property applies when you set the `Type` property to "Waveform cursors".

Default: 1

- **Cursor2TraceSource** -- Specify the waveform cursor 2 source as a positive real scalar. This property applies when you set the `Type` property to "Waveform cursors".

Default: 1

- **LockSpacing** -- Lock spacing between cursors, specified as a logical scalar.

Default: `false`

- **SnapToData** -- Snap cursors to data, specified as a logical scalar.

Default: `true`

- **XLocation** -- x-coordinates of the cursors, specified as a real vector of length equal to 2.

Default: [-2500 2500]

- **YLocation** -- y-coordinates of the cursors, specified as a real vector of length equal to 2. This property applies when you set the `Type` property to "Screen cursors".


Default: [-55 5]

- **Enable** -- Set this property to `true` to enable cursor measurements. Valid values are `true` or `false`.

Default: `false`

All `CursorMeasurementsSpecification` properties are tunable.

UI Use

Open the **Cursor Measurements** pane () and modify the **Settings** options.

DistortionMeasurements – Distortion measurements

`DistortionMeasurementsSpecification` object

Enable distortion measurements to compute and display the harmonic distortion and intermodulation distortion. The `DistortionMeasurements` property uses the `DistortionMeasurementsSpecification` properties.

The `DistortionMeasurementsSpecification` properties are:

- **Algorithm** -- Type of measurement data to display, specified as either "Harmonic" or "Intermodulation".

Default: "Harmonic"

- **NumHarmonics** -- Number of harmonics to measure, specified as a real, positive integer. This property applies when you set the `Algorithm` to "Harmonic".


Default: 6

- **Enable** -- Set this property to `true` to enable distortion measurements.

Default: `false`

All `DistortionMeasurementsSpecification` properties are tunable.

UI Use

Open the **Distortion Measurements** pane () and modify the **Distortion** and **Harmonics** options.

Visualization

Name – Window name

"Spectrum Analyzer" (default) | character vector | string scalar

Title of the scope window.

Tunable: Yes

Data Types: char | string

Position – Window position

screen center (default) | [left bottom width height]

Spectrum Analyzer window position in pixels, specified by the size and location of the scope window as a four-element double vector of the form [left bottom width height]. You can place the scope window in a specific position on your screen by modifying the values to this property.

By default, the window appears in the center of your screen with a width of 800 pixels and height of 450 pixels. The exact center coordinates depend on your screen resolution.

Tunable: Yes

PlotType — Plot type for normal traces

"Line" (default) | "Stem"

Specify the type of plot to use for displaying normal traces as either "Line" or "Stem". Normal traces are traces that display free-running spectral estimates.

Tunable: Yes

UI Use

Open the **Style** properties and set **Plot type**.

Data Types: char | string

ReducePlotRate — Improve performance with reduced plot rate

true (default) | false

The simulation speed is faster when this property is set to true.

- **true** — the scope logs data for later use and updates the display at fixed intervals of time. Data occurring between these fixed intervals might not be plotted.
- **false** — the scope updates every time it computes the power spectrum. Use the false setting when you do not want to miss any spectral updates at the expense of slower simulation speed.

UI Use

Select **Simulation > Reduce plot rate to improve performance**.

Data Types: logical

Title — Display title

' ' (default) | character vector | string scalar

Specify the display title as a character vector or string.

Tunable: Yes

UI Use

Open the **Configuration Properties**. Set **Title**.

Data Types: char | string

YLabel — Y-axis label

' ' (default) | character vector | string scalar

Specify the text for the scope to display to the left of the y-axis.

Regardless of this property, Spectrum Analyzer always displays power units as one of the `SpectrumUnits` values.

Tunable: Yes

UI Use

Open the **Configuration Properties**. Set **Y-label**.

Data Types: `char` | `string`

ShowLegend — Show legend

`false` (default) | `true`

To show a legend with the input names, set this property to `true`.

From the legend, you can control which signals are visible. This control is equivalent to changing the visibility in the **Style** dialog box. In the scope legend, click a signal name to hide the signal in the scope. To show the signal, click the signal name again. To show only one signal, right-click the signal name. To show all signals, press **Esc**.

Note The legend only shows the first 20 signals. Any additional signals cannot be viewed or controlled from the legend.

Tunable: Yes

UI Use

Open the **Configuration Properties**. On the **Display** tab, select **Show legend**.

Data Types: `logical`

ChannelNames — Channel names

`empty cell` (default) | `cell array of character vectors`

Specify the input channel names as a cell array of character vectors. The names appear in the legend, **Style** dialog box, and **Measurements** panels. If you do not specify names, the channels are labeled as `Channel 1`, `Channel 2`, etc.

Tunable: Yes

Dependency

To see channel names, set `ShowLegend` to `true`.

UI Use

On the legend, double-click the channel name.

Data Types: `char`

ShowGrid — Grid visibility

`true` (default) | `false`

Set this property to `true` to show gridlines on the plot.

Tunable: Yes

UI Use

Open the **Configuration Properties**. On the **Display** tab, set **Show grid**.

Data Types: logical

YLimits — Y-axis limits

`[-80, 20]` (default) | `[ymin ymax]`

Specify the y-axis limits as a two-element numeric vector, `[ymin ymax]`.

Example: `scope.YLimits = [-10,20]`

Tunable: Yes

UI Use

Open the **Configuration Properties**. Set **Y-limits (maximum)** and **Y-limits (minimum)**.

AxesScaling — Axes scaling mode

`"Auto"` (default) | `"Manual"` | `"OnceAtStop"` | `"Updates"`

Specify when the scope automatically scales the axes. Valid values are:

- `"Auto"` — The scope scales the axes as-needed to fit the data, both during and after simulation.
- `"Manual"` — The scope does not scale the axes automatically.
- `"OnceAtStop"` — The scope scales the axes when the simulation stops.
- `"Updates"` — The scope scales the axes once after 10 updates.

UI Use

Select **Tools > Axes Scaling**.

Data Types: char | string

AxesScalingNumUpdates — Number of updates before scaling

`"10"` (default) | integer character vector | integer string scalar

Set this property to delay auto scaling the y-axis.

Dependency

To enable this property, set `AxesScaling` to `"Updates"`.

UI Use

Open the **Axes Scaling** dialog box and set **Number of updates**.

Data Types: char | string

OpenAtSimulationStart — Open scope when starting simulation

`true` (default) | `false`

Set this property to `true` to open the scope when the simulation starts. Set this property to `false` to prevent the scope from opening at the start of simulation.

UI Use

Select **File > Open at Start of Simulation**.

Data Types: logical

Visible — Visibility of the Spectrum Analyzer

false | true

Set this property to `true` to show the spectrum analyzer window, or `false` to hide the spectrum analyzer window.

Data Types: logical

Examples

Construct a Spectrum Analyzer Configuration Object

Create the configuration object for a Spectrum Analyzer block.

Create a new Simulink® model with a randomly-generated name.

```
sysname=char(randi(26,1,7)+96);
new_system(sysname);
```

Add a new Spectrum Analyzer block to the model.

```
add_block('built-in/SpectrumAnalyzer',[sysname,'/SpectrumAnalyzer'])
```

Call the `get_param` function to retrieve the default Spectrum Analyzer block configuration properties.

```
config = get_param([sysname,'/SpectrumAnalyzer'],'ScopeConfiguration')
```

```
config =
```

```
SpectrumAnalyzerConfiguration with properties:
```

```
    NumInputPorts: '1'
    SpectrumType: 'Power'
    SampleRateSource: 'Inherited'
    PlotAsTwoSidedSpectrum: 1
    FrequencyScale: 'Linear'
```

```
Advanced
```

```
    RBWSource: 'Auto'
    OverlapPercent: '0'
    Window: 'Hann'
    SpectrumUnits: 'dBm'
    AveragingMethod: 'Running'
    SpectralAverages: '1'
    ReferenceLoad: '1'
    FrequencyOffset: '0'
    TreatMby1SignalsAsOneChannel: 1
```

```
Measurements
```

```
    MeasurementChannel: '1'
    PeakFinder: [1x1 PeakFinderSpecification]
    CursorMeasurements: [1x1 CursorMeasurementsSpecification]
    DistortionMeasurements: [1x1 DistortionMeasurementsSpecification]
```

```
Visualization
```

```
    Name: 'SpectrumAnalyzer'
    Position: [560 375 800 450]
    PlotType: 'Line'
    ReducePlotRate: 1
    Title: ''
    YLabel: ''
    ShowLegend: 0
    ChannelNames: {''}
    ShowGrid: 1
```

```
YLimits: [-80 20]  
AxesScaling: 'Auto'  
OpenAtSimulationStart: 1  
Visible: 0
```

See Also

Topics

“Control Scope Blocks Programmatically”

Introduced in R2016b

sscexplore

Open Simscape Results Explorer to interact with logged simulation data

Syntax

```
sscexplore(node)  
sscexplore(node,nodepath)
```

Description

`sscexplore(node)` opens a new Simscape Results Explorer window containing logged simulation data for the specified node in a simulation log variable. Before you call this function, you must have the simulation log variable in your current workspace. Create the simulation log variable by simulating the model with data logging turned on, or load a previously saved variable from a file. If `node` is the name of the simulation log variable, then the Simscape Results Explorer window contains the data for the whole model. If `node` is the name of a node in the simulation data tree, then the Simscape Results Explorer window contains the data for that node only.

`sscexplore(node,nodepath)` opens a new Simscape Results Explorer window that contains logged simulation data for the specified node, `node`, but opens at a subnode specified by `nodepath`.

Examples

Explore Logged Simulation Data for the Whole Model, Starting at the Root

Open the Permanent Magnet DC Motor example model:

```
ssc_dcmotor
```

This example model has data logging enabled for the whole model, with the **Workspace variable name** parameter set to `simlog_ssc_dcmotor`.

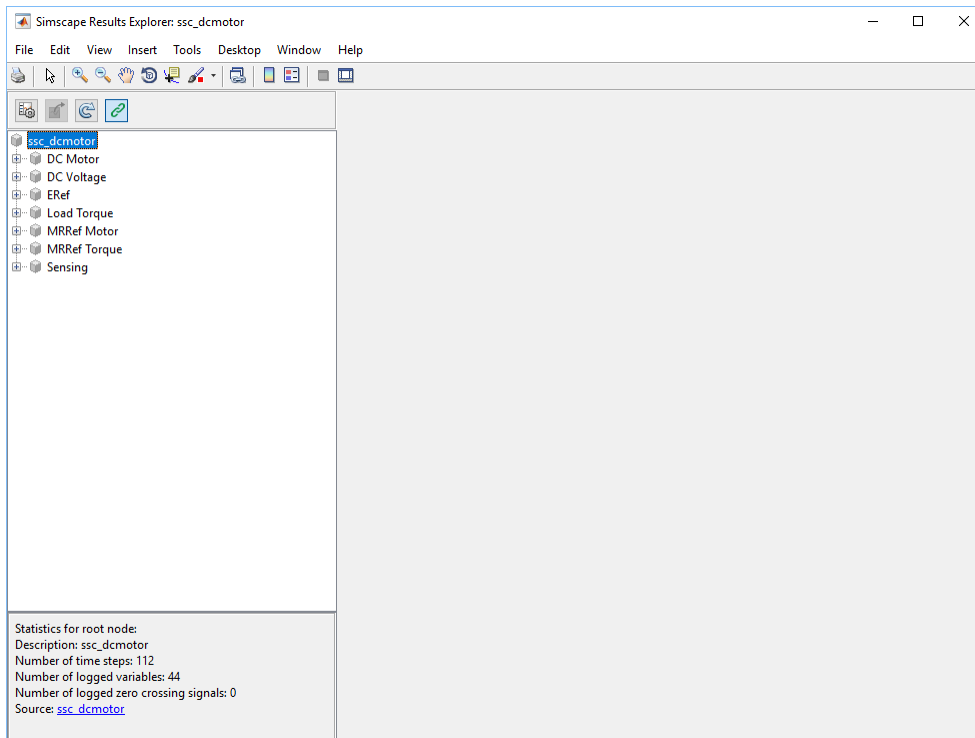
Simulate the model to log the simulation data:

```
sim('ssc_dcmotor');
```

Explore the simulation data:

```
sscexplore(simlog_ssc_dcmotor)
```

A new Simscape Results Explorer window opens. It contains logged simulation data for the whole model. The root node, `ssc_dcmotor`, is selected in the left pane by default. As you expand and select nodes in the left pane, the corresponding plots appear in the right pane.



Explore Logged Simulation Data for the Whole Model, Starting at a Specific Node

Open the Permanent Magnet DC Motor example model:

```
ssc_dcmotor
```

This example model has data logging enabled for the whole model, with the **Workspace variable name** parameter set to `simlog_ssc_dcmotor`.

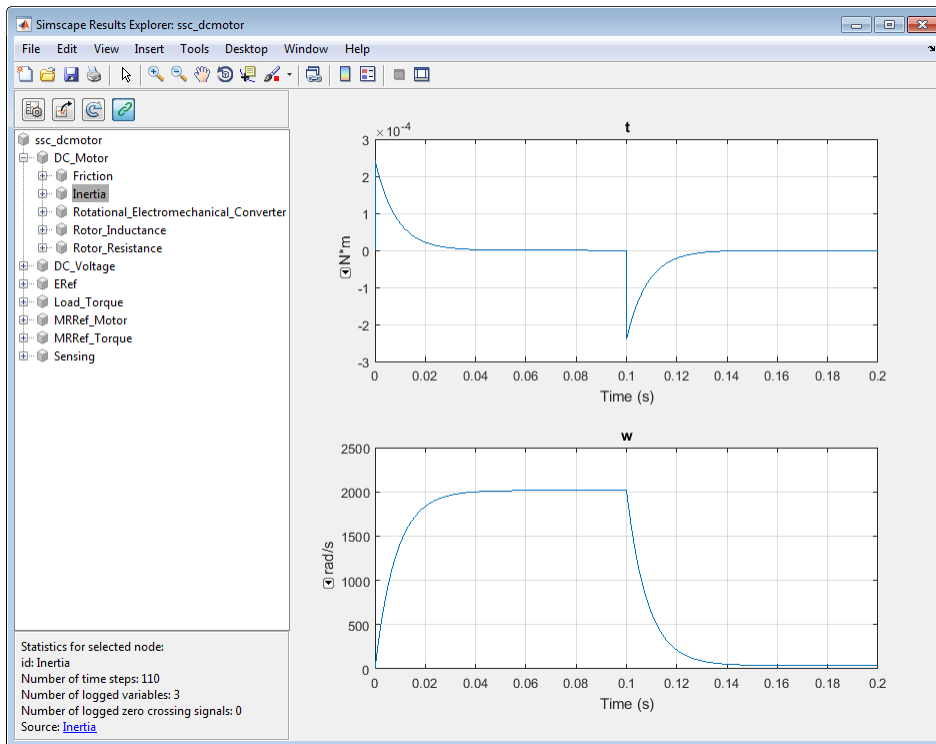
Simulate the model to log the simulation data:

```
sim('ssc_dcmotor');
```

Explore the simulation data:

```
sscexplore(simlog_ssc_dcmotor, 'DC_Motor.Inertia')
```

A new Simscape Results Explorer window opens. It contains logged simulation data for the whole model, but the data exploration starts with the node specified by the `nodepath` argument. This node corresponds to the Inertia block in the DC Motor subsystem.



Explore Logged Simulation Data Only for a Specific Node

Open the Permanent Magnet DC Motor example model:

```
ssc_dcmotor
```

This example model has data logging enabled for the whole model, with the **Workspace variable name** parameter set to `simlog_ssc_dcmotor`.

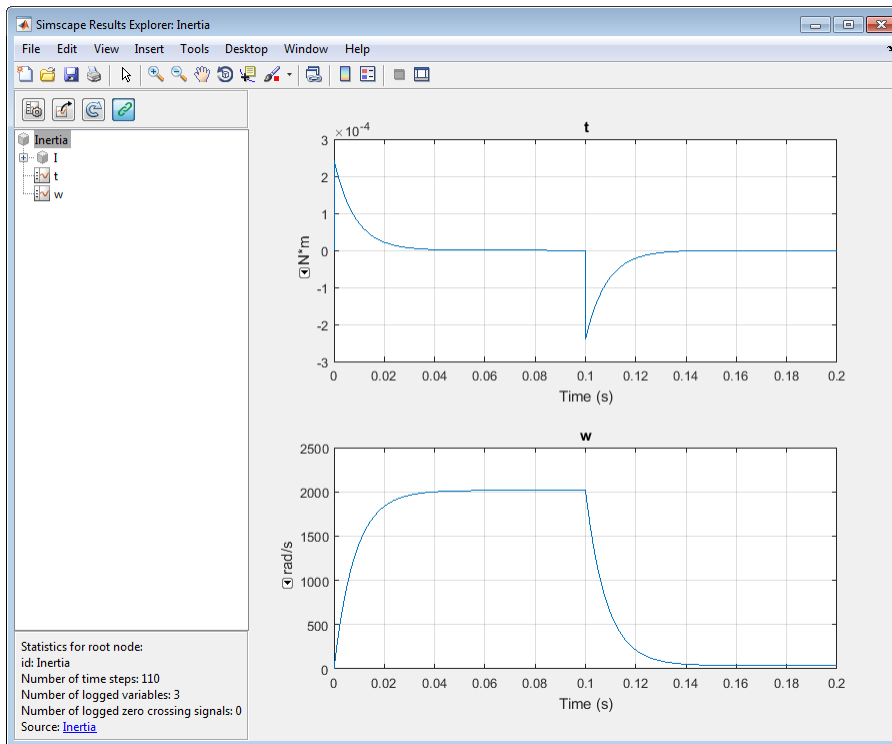
Simulate the model to log the simulation data:

```
sim('ssc_dcmotor');
```

Explore the simulation data:

```
sscexplore(simlog_ssc_dcmotor.DC_Motor.Inertia)
```

A new Simscape Results Explorer window opens. It contains just the logged simulation data for the node specified in the node argument. This node corresponds to the Inertia block in the DC Motor subsystem.



Input Arguments

node — Simulation log variable, or a specific node within the simulation log variable

Node object

Simulation log workspace variable, or a node within this variable, that contains the logged model simulation data, specified as a Node object. You specify the name of the simulation log variable by using the **Workspace variable name** parameter on the **Simscape** pane of the Configuration Parameters dialog box. To specify a node within the simulation log variable, provide the complete path to that node through the simulation data tree, starting with the top-level variable name.

Example: `simlog.DC_Motor.Motor_Inertia_J`

nodepath — Path to a subnode to open

character vector | string scalar

Path to a subnode to open, specified as a character vector or string scalar. If you omit the `nodepath` argument, then the Simscape Results Explorer window opens at the root of the specified node, `node`. If you specify `nodepath`, then the Simscape Results Explorer window still contains all the simulation data for the node, but opens at the subnode within it, specified by the `nodepath`. Use the `simscape.logging.findPath` function to find the `nodepath` value for a block or subsystem.

Example: `'DC_Motor.Motor_Inertia_J'`

Data Types: `char` | `string`

See Also

`simscape.logging.findPath`

Topics

“About Simulation Data Logging”

“Data Logging Options”

“About the Simscape Results Explorer”

Introduced in R2015a

sscprintzcs

Print zero crossing information for logged simulation data

Syntax

```
sscprintzcs(node)
sscprintzcs(node,verbosity)
```

Description

`sscprintzcs(node)` prints information about zero crossings detected during simulation, based on logged simulation data specified by `node`. Before you call this function, you must have the simulation log variable, which includes simulation statistics data, in your current workspace. Create the workspace variable by simulating the model with simulation statistics logging turned on, or load a previously saved variable from a file.

`sscprintzcs(node,verbosity)` prints detailed information about zero crossings. The `verbosity` argument controls the level of detail.

Examples

Print Information About Blocks That Produce Zero Crossings

Open the Mechanical System with Translational Hard Stop example model:

```
ssc_mechanical_system_translational_hardstop
```

This example model has data logging and simulation statistics logging enabled, with the **Workspace variable name** parameter set to `simlog_ssc_mechanical_system_translational_hardstop`.

Simulate the model to log the simulation data:

```
sim('ssc_mechanical_system_translational_hardstop');
```

Get the zero crossing information at the block level:

```
sscprintzcs(simlog_ssc_mechanical_system_translational_hardstop)
ssc_mechanical_system_translational_hardstop (2 signals, 28 crossings)
+-Translational_Hard_Stop (2 signals, 28 crossings)
```

The results show that the only block that can produce zero crossings is the Translational Hard Stop block. It has two signals that can produce zero crossings, and 28 actual zero crossings are detected.

Print Information About Signals That Produce Zero Crossings

Open the Mechanical System with Translational Hard Stop example model:

```
ssc_mechanical_system_translational_hardstop
```

This example model has data logging and simulation statistics logging enabled, with the **Workspace variable name** parameter set to `simlog_ssc_mechanical_system_translational_hardstop`.

Simulate the model to log the simulation data:

```
sim('ssc_mechanical_system_translational_hardstop');
```

Get the zero crossing information at the signal level:

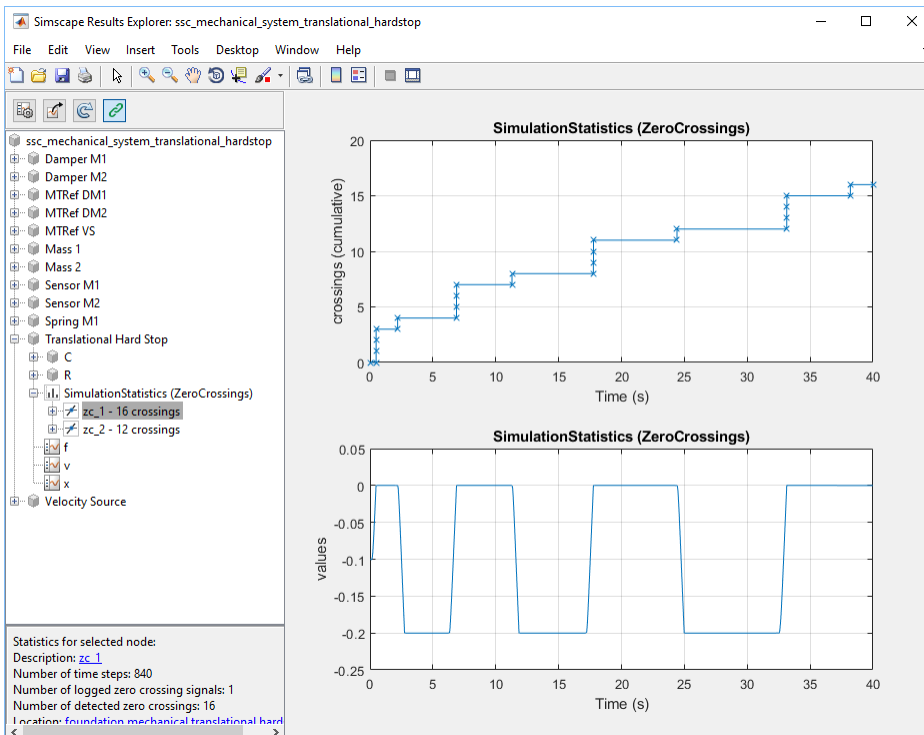
```
sscprintzcs(simlog_ssc_mechanical_system_translational_hardstop,1)

ssc_mechanical_system_translational_hardstop (2 signals, 28 crossings)
+-Translational_Hard_Stop (2 signals, 28 crossings)
  -zc_1      16
  -zc_2      12
```

The results show that the only block that can produce zero crossings is the Translational Hard Stop block. It has two signals that can produce zero crossings, `zc_1` and `zc_2`. A total of 28 actual zero crossings are detected. Signal `zc_1` produced 16 zero crossings during simulation, and signal `zc_2` produced 12.

Use the `sscexplore` function to further explore the zero crossing data for signal `zc_1`.

```
sscexplore(simlog_ssc_mechanical_system_translational_hardstop,...
'Translational_Hard_Stop.SimulationStatistics.zc_1')
```



Input Arguments

node — Simulation log variable, or a specific node within the simulation log variable
Node object

Simulation log workspace variable that contains the logged model simulation data, including simulation statistics, specified as a **Node** object. You specify the name of the simulation log variable by using the **Workspace variable name** parameter on the **Simscape** pane of the Configuration Parameters dialog box. You can also specify a node within the simulation log variable by providing the complete path to that node through the simulation data tree. In this case, the function prints information only about zero crossings found in that particular node.

Example: `simlog.DC_Motor`

verbosity — Level of detail in printed information about zero crossings

0 (default) | 1 | 2

Level of detail in printed information about zero crossings, specified as a number:

- 0 — Block-level information
- 1 — Signal-level information
- 2 — Signal-level information, including location

Data Types: `double`

See Also

`sscexplore`

Topics

“About Simulation Data Logging”

“Data Logging Options”

“About the Simscape Results Explorer”

Introduced in R2015a

ssc_build

Build custom library from collection of Simscape files

Syntax

```
ssc_build(package)
ssc_build(package, '-output', outputlibrary)
ssc_build package
ssc_build
```

Description

`ssc_build(package)` generates a custom Simscape library file from the specified package, `package`. Call `ssc_build` from the package parent directory, that is, from the directory containing the top-level package directory. For more information on package directory structure, see “Organizing Your Simscape Files”.

When you call `ssc_build` with one argument, the library file is named `package_lib` and is located in the package parent directory. The library contains all the sublibraries and blocks generated from the Simscape files (either source or protected) located in the package and its subdirectories. Simscape protected files have higher precedence than the source files when you build a library. If both the protected and the source files are present in the package and the source files are newer than the protected files, `ssc_build` uses the protected files to build the library, but issues a warning.

`ssc_build(package, '-output', outputlibrary)` generates a custom Simscape library file from the specified package, `package`, with `outputlibrary` defining the library file name and location. This syntax uses a name-value argument pair, where `'-output'` is the name of the optional argument and `outputlibrary` is the argument value. The function implements partial argument name matching, therefore specifying `'-o'` as the argument name also works.

`ssc_build package` is the command form of the syntax. Command form requires fewer special characters. You do not need to type parentheses or enclose the input in single or double quotes. Separate inputs with spaces instead of commas.

For example, to build a package named `+MyPackage`, these statements are equivalent:

```
ssc_build MyPackage          % command form
ssc_build('MyPackage')     % function form
```

You can also use command form with the name-value argument pair, described in the previous syntax, as long as the path and name of the output library is a character vector. For example, to build a package named `+MyPackage` and save the output library as `'C:\Work\MyLibrary'`, these statements are equivalent:

```
ssc_build MyPackage -output C:\Work\MyLibrary          % command form
ssc_build('MyPackage', '-output', 'C:\Work\MyLibrary') % function form
```

Do not use command form when `outputlibrary` uses variables, or functions like `fullfile`, to specify the output library name and location. For more information on the command-function duality, see “Command vs. Function Syntax”.

`ssc_build` is a special syntax, with no arguments, that you can use to call the function from inside the package directory structure. It builds a library from the current package, with default library name and location. To specify a different name or location for the output library, call `ssc_build` from the package parent directory using either the command or the function form of the syntax with the name-value argument pair.

Examples

Build Custom Block Library with Default Name

Suppose your top-level package directory, where you store your Simscape files, is named `+SimscapeCustomBlocks` and is located in `C:\Work\MyLibraries`.

To generate a custom block library, change your current working directory to `C:\Work\MyLibraries`. Then, at the MATLAB Command prompt, type:

```
ssc_build('SimscapeCustomBlocks');
```

You can also use the command form of this syntax, which requires fewer special characters:

```
ssc_build SimscapeCustomBlocks;
```

This command generates a Simulink model file called `SimscapeCustomBlocks_lib` in the package parent directory, `C:\Work\MyLibraries` (that is, in the same directory that contains your `+SimscapeCustomBlocks` package).

Specify Name and Location when Building Custom Block Library

Suppose your top-level package directory, where you store your Simscape files, is named `+CustomElectricalBlocks` and is located in `C:\Work`. You want to generate a custom block library from this package, place it in `C:\Work\MyLibraries`, and give it a shorter and more meaningful name, `CustomDiodes`. You do this by providing the name-value argument pair to the `ssc_build` function.

To generate the custom block library, change your current working directory to `C:\Work`. Then, at the MATLAB Command prompt, type:

```
ssc_build('CustomElectricalBlocks', '-output', 'C:\Work\MyLibraries\CustomDiodes')
```

```
Generating Simulink library 'CustomDiodes' in the output directory 'C:\Work\MyLibraries' ...
```

This function call generates a Simulink model file called `CustomDiodes.slx` in the specified directory. Add the `C:\Work\MyLibraries` directory to the MATLAB path to facilitate using these custom blocks in various models.

Build Custom Block Library from Read-Only Source Files

Suppose your package with source files is located in a read-only directory. For example, the `+BatteryPack` package is used in the “Lithium-Ion Battery Pack With Fault Using Arrays” example. For more information on this package, see “Case Study — Battery Pack with Fault Using Arrays”.

To generate a custom block library from the `+BatteryPack` package, you must specify a location for the output library other than the read-only parent directory of the package. You do this by providing the name-value argument pair to the `ssc_build` function. In this example, the output directory name is a character vector, which allows you to use the command form of the syntax.

Change your current working directory to the directory containing the `+BatteryPack` package.

```
cd(matlabroot)
cd toolbox/physmod/simscape/simscapedemos
```

At the MATLAB Command prompt, type:

```
ssc_build BatteryPack -output C:\Work\BatteryPack_lib
```

```
Generating Simulink library 'BatteryPack_lib' in the output directory 'C:\Work' ...
```

This command generates a Simulink model file called `BatteryPack_lib` in the specified output directory, `C:\Work`.

You can also take advantage of the partial argument name matching and shorten the command syntax even more:

```
ssc_build BatteryPack -o C:\Work\BatteryPack_lib
```

This command is equivalent to the previous one.

Input Arguments

package — Name of package containing Simscape files

character vector | string scalar

Name of package containing Simscape files, specified as a character vector or a string scalar and located in the directory from which you call the `ssc_build` function. `package` is the name of the top-level package directory without the leading `+` character. When you call `ssc_build` using the command syntax, do not use quotes around `package`. For more information on the command-function duality, see “Command vs. Function Syntax”.

Example: `ssc_build MyPackage` is the command syntax to build a library from the package `+MyPackage`. The equivalent function syntax is `ssc_build('MyPackage')`.

outputLibrary — Full path and name of generated block library file

character vector | string scalar

Name and location of the block library file generated from the package, specified as a character vector or a string scalar. When you call `ssc_build` using the function syntax, you can also use MATLAB variables and functions, such as `fullfile` or `genpath`, to specify the path to the library file.

If you specify a file name without the path, the library is generated in your current working directory. When you specify the file name, you can include the file extension, `.slx`, or omit it.

Example: `fullfile(tempdir, 'MyLibrary.slx')` with function syntax, you can use MATLAB variables and functions to specify the path.

See Also

`addpath` | `fullfile` | `genpath` | `sl_postprocess` | `ssc_clean` | `ssc_mirror` | `ssc_protect`

Topics

“Building Custom Block Libraries”

Introduced in R2008b

ssc_clean

Clean all derived files generated by library build process

Syntax

```
ssc_clean package
```

Description

`ssc_clean package` deletes all derived files generated by `ssc_build` in the package named *package*, including the library file.

The argument, *package*, must be a top-level package name.

Note The package directory name begins with a leading + character, whereas the argument to `ssc_clean` must omit the + character.

Running `ssc_clean` before rebuilding a library forces `ssc_build` to generate all derived files in the package, rather than regenerate only those files that have changed. You do not need to run `ssc_clean` before regular iterative library builds.

When you upgrade to a new version of Simscape software, run `ssc_clean` and then rebuild the custom block libraries.

When deploying your libraries on multiple platforms, you do not need to run `ssc_clean` on each platform. Use `ssc_clean` at the beginning of deployment and then just run `ssc_build` on each platform.

Examples

To clean all derived files from the package directory `+MyPackage`, invoke the following from the directory containing the package directory `+MyPackage`:

```
ssc_clean MyPackage;
```

See Also

`ssc_build`

Introduced in R2008b

ssc_mirror

Create protected mirror of library of Simscape files

Syntax

```
ssc_mirror package mirrordir buildmirror
```

Description

The `ssc_mirror` command lets you protect and build a whole package of Simscape files in one step.

`ssc_mirror package mirrordir buildmirror` creates a protected mirror of a package of Simscape files in a specified directory *mirrordir*, and also optionally builds a custom library from these files.

The first argument, *package*, must be a top-level package name.

Note The package directory name begins with a leading + character, whereas the argument to `ssc_mirror` must omit the + character.

The second argument, *mirrordir*, is the directory where the protected package is placed. The `ssc_mirror` command creates this directory, if it does not exist, recreates the whole package structure under it, generates the protected files, and places them in the appropriate mirror locations.

If the `buildmirror` flag is set to `true`, the `ssc_mirror` command also builds a custom Simscape library file, named *package_lib*, containing all the sublibraries and blocks generated from the Simscape files in the mirrored package (similar to the `ssc_build` command), and places the *package_lib* file in the *mirrordir* directory. The `buildmirror` flag is optional and the default is `false`, that is, by default the package is mirrored and protected but the library is not built.

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

Examples

For example, your top-level package directory, where you store your Simscape files, is named `+SimscapeCustomBlocks`. To protect, mirror, and generate a custom block library from this package in the directory `C:\Work\deploy`, at the MATLAB Command prompt, type:

```
ssc_mirror SimscapeCustomBlocks C:\Work\deploy true;
```

This command creates a mirror package, equivalent to the `+SimscapeCustomBlocks` package but consisting of Simscape protected files, in the directory `C:\Work\deploy`, and generates a file called `SimscapeCustomBlocks_lib` in the `C:\Work\deploy` directory.

See Also

`sl_postprocess` | `ssc_build` | `ssc_clean` | `ssc_protect`

Introduced in R2009a

ssc_new

Create new Simscape model populated by required and commonly used blocks

Syntax

```
ssc_new
ssc_new('modelname')
ssc_new('modelname','domain')
```

Description

`ssc_new` creates a new Simscape model, with required and commonly used blocks already on the model canvas. The model uses the recommended solver `VariableStepAuto` with the absolute tolerance, `AbsTol`, set to $1e-3$.

The function also turns on simulation data logging for the whole model, using the default workspace variable name `simlog` and limiting the logged simulation data to 10000 points. For more information, see “Data Logging”.

By default, the function uses the Simulink default new model name `untitled` and the recommended solver `VariableStepAuto`.

`ssc_new('modelname')` creates a new Simscape model with the specified name.

`ssc_new('modelname','domain')` creates a new Simscape model with the specified name and with domain-specific blocks added to the model canvas. Valid domains types are `'electrical'`, `'gas'`, `'hydraulic'`, `'isothermal_liquid'`, `'magnetic'`, `'moist_air'`, `'rotational'`, `'translational'`, `'thermal'`, `'thermal_liquid'`, and `'two_phase_fluid'`.

Using the `ssc_new` to create a model is equivalent to opening the corresponding Simscape template from the Simulink Start Page.

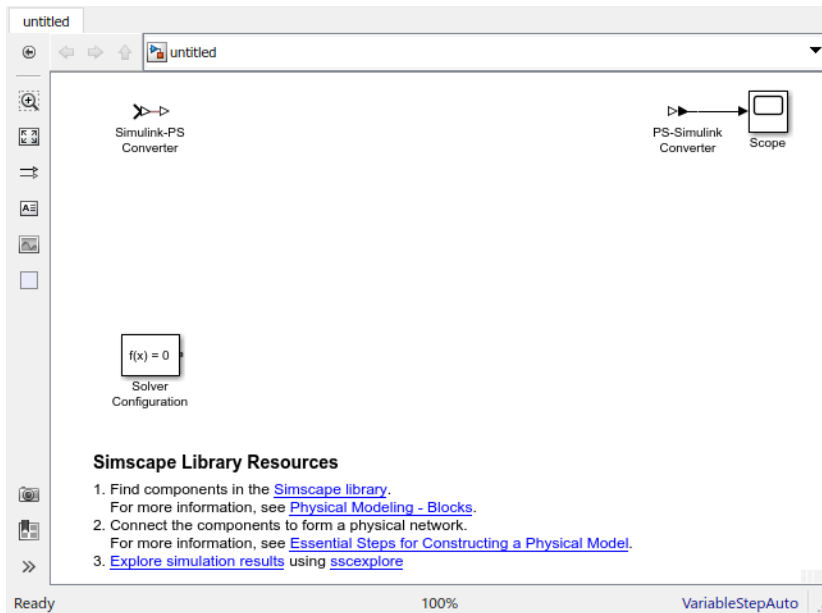
Examples

Create a Generic Simscape Model

To create a generic Simscape model, type:

```
ssc_new
```

The software creates a new untitled model, with the default solver set to `VariableStepAuto`. The model contains a Solver Configuration block, a Simulink-PS Converter block, and a PS-Simulink Converter block connected to a Scope block.



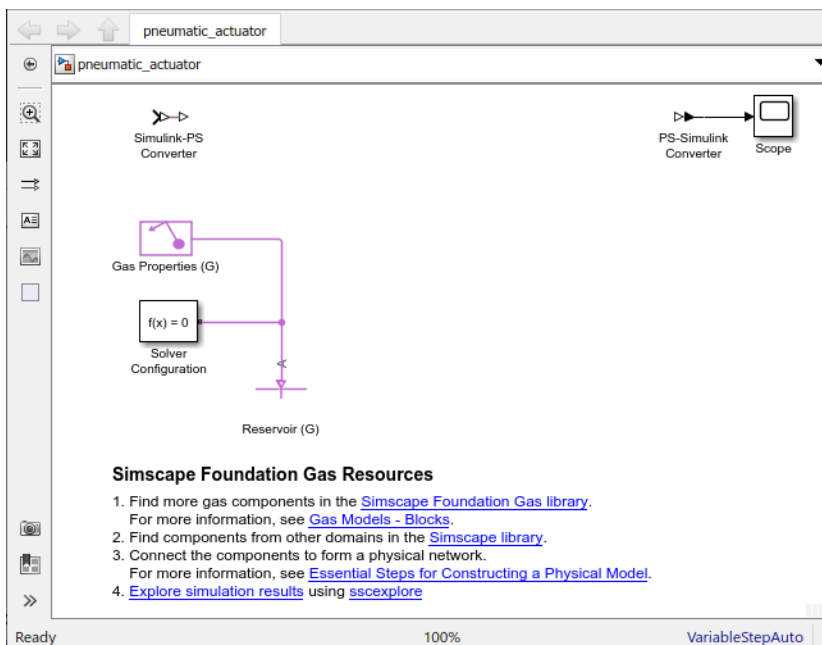
After using `ssc_new`, continue developing your model by copying the blocks, as needed, and adding other blocks from the Simscape libraries. Use the resources links at the bottom of the model window to open the main Simscape library and access documentation topics that help you get started.

Create a Gas Model with Specified Name

To create a gas model, called `pneumatic_actuator`, type:

```
ssc_new('pneumatic_actuator', 'gas')
```

The software creates the following model.



After using `ssc_new`, continue developing your model by copying the blocks, as needed, and adding other blocks from the Simscape libraries. Use the resources links at the bottom of the model window to open the relevant block libraries and access documentation topics that help you get started.

See Also

Topics

“Creating a New Simscape Model”

“Select Solver Using Auto Solver”

Introduced in R2009a

sscnewfile

Create new Simscape file populated by required and commonly used keywords

Syntax

```
sscnewfile(name)
sscnewfile(name,template_keyword)
sscnewfile(name,template_file)
sscnewfile -list
```

Description

`sscnewfile(name)` creates a component with the specified `name` using the default component template. The function automatically saves the new Simscape file, `name`, in the current folder.

`sscnewfile(name,template_keyword)` creates a component with the specified `name` using the domain-specific template specified by `template_keyword`. For a list of available keywords, use the `-list` option. The function automatically saves the new Simscape file, `name`, in the current folder.

`sscnewfile(name,template_file)` creates a component, domain, or function file with the specified `name` using a file of the same type, `template_file`, as the template. The template file name must include the file path. Various template files are available in the `simscape.template` package; however, you can use any Simscape file as a template. The function copies the contents of the template file into the new file, replaces the name of the component, domain, or function with `name`, and saves the new file in the current folder.

`sscnewfile -list` returns a list of available template keywords and lists all the files available in the `simscape.template` package.

Examples

Create a Custom Component Using the Default Template

Create a component named `MyComponent` using the default component template and save it as `MyComponent.ssc` in the current folder.

```
sscnewfile('MyComponent')
```

The new file opens in the MATLAB Editor.

```
component MyComponent
% Simple Simscape component

parameters
    % Add parameters here
    % p = { value , 'unit' }; % Parameter name
end

nodes
    % A = package_name.domain_name; % A:left
```

```

    % B = package_name.domain_name; % B:right
end

variables
    % x = { value , 'unit' }; % Through variable name
    % y = { value , 'unit' }; % Across variable name
end

branches
    % x : A.x -> B.x;
end

equations
    % Add equations here
    % y == A.y - B.y;
    % x == fcn(y);
end

end

```

Use this file as a starting point for authoring the new component. Lines that start with % are comments. Replace them with actual declarations and equations, as needed. You can also delete unnecessary sections and add other sections, such as components, connections, or intermediates.

Create a Custom Component Using a Domain-Specific Template

Create a component named MyResistor using the default component template for the electrical domain and save it as MyResistor.ssc in the current folder.

```
sscnewfile('MyResistor','electrical')
```

The new file opens in the MATLAB Editor.

```

component MyResistor
% Two-port electrical component

parameters
    % Add parameters here
    % R = { 1, 'Ohm' }; % Resistance
end

nodes
    p = foundation.electrical.electrical; % +:left
    n = foundation.electrical.electrical; % -:right
end

variables
    i = { 0, 'A' }; % Current
    v = { 0, 'V' }; % Voltage
end

branches
    i : p.i -> n.i;
end

```

```

equations
    % Voltage difference between nodes
    v == p.v - n.v;

    % Add equations here
    % v == i*R;
end

end

```

Use this file as a starting point for authoring the new component. Lines that start with % are comments. Replace them with actual declarations and equations, as needed. For an example of creating a custom resistor, see “Model Linear Resistor in Simscape Language”.

Create a Custom Domain Using a Foundation Domain as the Template

Create a domain named `MyGasDomain` using the Foundation gas domain as a template and save it as `MyGasDomain.ssc` in the current folder.

```
sscnewfile('MyGasDomain', 'foundation.gas.gas')
```

The new file opens in the MATLAB Editor. The name of the new domain is `MyGasDomain`, matching the name argument. The rest of the file is a copy of the Foundation gas domain definition.

Use this file as a starting point for authoring the new domain. Modify the domain parameters and properties to suit your application.

List Existing Template Keywords and Files

List the template keywords and the template files available in the `simscape.template` package.

```
sscnewfile -list
```

```

-----
Template keywords
-----
default          simscape.template.simple_component
electrical       simscape.template.electrical.two_port
gas              simscape.template.gas.two_port_steady
hydraulic        simscape.template.hydraulic.two_port
magnetic         simscape.template.magnetic.two_port
moist_air        simscape.template.moist_air.two_port_steady
rotational       simscape.template.mechanical.rotational.two_port
signal           simscape.template.signal.simple_component
thermal          simscape.template.thermal.two_port
thermal_liquid   simscape.template.thermal_liquid.two_port_steady
translational    simscape.template.mechanical.translational.two_port
two_phase_fluid  simscape.template.two_phase_fluid.two_port_steady
-----
Files in package simscape.template
-----
simple_component  Simple Simscape component
simple_function
-----
Files in package simscape.template.two_phase_fluid
-----
two_port_dynamic  Two-port dynamic two-phase fluid component
two_port_steady   Two-port steady two-phase fluid component
-----

```

```

Files in package simscape.template.mechanical.translational
-----
    two_port    Two-port translational component
-----
Files in package simscape.template.mechanical.rotational
-----
    two_port    Two-port rotational component
-----
Files in package simscape.template.hydraulic
-----
    two_port    Two-port hydraulic component
-----
Files in package simscape.template.gas
-----
    two_port_dynamic    Two-port dynamic gas component
    two_port_steady     Two-port steady gas component
-----
Files in package simscape.template.thermal
-----
    two_port    Two-port thermal component
-----
Files in package simscape.template.signal
-----
    simple_component    Simple component with physical signals
-----
Files in package simscape.template.magnetic
-----
    two_port    Two-port magnetic component
-----
Files in package simscape.template.thermal_liquid
-----
    two_port_dynamic    Two-port dynamic thermal liquid component
    two_port_steady     Two-port steady thermal liquid component
-----
Files in package simscape.template.moist_air
-----
    two_port_dynamic    Two-port dynamic moist_air component
    two_port_steady     Two-port steady moist air component
-----
Files in package simscape.template.electrical
-----
    two_port    Two-port electrical component

```

To use one of these files as the template for a new component, specify the full path and name of the file as the second input argument, for example, 'simscape.template.gas.two_port_dynamic'.

Input Arguments

name — Name of component, domain, or function

character vector | string scalar

Name of the new component, domain, or function, specified as a character vector or string scalar. This name also serves as the name of the new Simscape file being created. The file is automatically saved in the current folder.

Example: 'MyResistor'

Data Types: char | string

template_keyword — Keyword specifying the template to use

character vector | string scalar

Type of component file to be used as the template for the new file, specified as a character vector or string scalar. Use these keywords to create component files for a specific domain type. For a list of available keywords, use the `-list` option.

Example: 'electrical'

Data Types: char | string

template_file — Name of file to use as template

character vector | string scalar

Name of a Simscape component, domain, or function, specified as a character vector or string scalar. The file name must include the path to the file from the top-level package folder. If the file is on the MATLAB path, you can specify the absolute file path instead. The new file being created will use this Simscape file as the template.

Example: 'foundation.electrical.elements.resistor'

Data Types: char | string

See Also

component | domain | function

Topics

“Model Linear Resistor in Simscape Language”

“Creating Custom Components”

“How to Define a New Physical Domain”

“Simscape Functions”

Introduced in R2019b

ssc_protect

Generate Simscape protected files from source files

Syntax

```
ssc_protect filename  
ssc_protect filename -inplace  
ssc_protect dirname  
ssc_protect dirname -inplace
```

Description

The `ssc_protect` command creates content-obscured files (Simscape protected files) from Simscape source files, to enable model sharing without disclosing the component or domain source. While Simscape source files have the extension `.ssc`, Simscape protected files have the extension `.sscp`.

`ssc_protect filename` generates a Simscape protected file, named `filename.sscp`, from the Simscape source file named `filename.ssc`, and places the protected file in your current working directory. `filename` can include absolute path to the file, or relative path if the file is in a subfolder of the current working directory. If this path includes package directories, the package structure will be recreated under the current working directory (unless it already exists) and the protected file placed in the package (see examples on page 2-166). The extension `.ssc` in `filename` is optional.

`ssc_protect filename -inplace` generates a Simscape protected file, named `filename.sscp`, from the Simscape source file named `filename.ssc`, and places the protected file in the same directory as the source file.

`ssc_protect dirname` generates Simscape protected files from all the Simscape source files in the directory named `dirname`, and places the protected files under your current working directory. If the path to `dirname` includes package directories, the package structure will be recreated under the current working directory (unless it already exists) and the protected files placed in the package, similar to when protecting a single file.

`ssc_protect dirname -inplace` generates Simscape protected files from all the Simscape source files in the directory named `dirname`, and places the protected files in the same directory as the source files.

Note Existing Simscape protected files are overwritten without warning.

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

Simscape protected files have higher precedence than the source files when you build a library. If the protected and the source files are in the same directory, and protected files are out of date, `ssc_build` will use the protected files to build the library, but you will get a warning.

Examples

To protect a single file, with the protected file placed under your current working directory, at the MATLAB Command prompt, type:

```
ssc_protect C:\Work\libraries\source\+SimscapeLibrary\+MechanicalElements\my_spring.ssc
```

This command creates a folder called `+SimscapeLibrary` and a subfolder called `+MechanicalElements` in your current working directory (unless these folders already exist) and generates a file called `my_spring.ssc` in the `+MechanicalElements` folder.

To protect a single file, with the protected file placed in the same directory as the source file, type:

```
ssc_protect C:\Work\libraries\source\+SimscapeLibrary\+MechanicalElements\my_spring.ssc -inplace
```

This command generates a file called `my_spring.ssc` in the `C:\Work\libraries\source\+SimscapeLibrary\+MechanicalElements` folder.

To protect all files in a directory, with the protected files placed under your current working directory, type:

```
ssc_protect C:\Work\libraries\source\+SimscapeLibrary\+MechanicalElements
```

This command generates protected files for each source file in the `C:\Work\libraries\source\+SimscapeLibrary\+MechanicalElements` folder, and places the protected files in a folder called `+SimscapeLibrary\+MechanicalElements` in your current working directory (creating this folder structure, if it does not exist).

To protect all files in a directory, with the protected files placed in the same directory as the source files, type:

```
ssc_protect C:\Work\libraries\source\+SimscapeLibrary\+MechanicalElements -inplace
```

This command generates protected files for each source file in the `C:\Work\libraries\source\+SimscapeLibrary\+MechanicalElements` folder, and places the protected files in the same folder.

See Also

`ssc_build` | `ssc_clean` | `ssc_mirror`

Introduced in R2009a

ssc_update

Update Simscape component files to use new syntax

Syntax

`ssc_update package`

Description

The `ssc_update` command runs a script that updates the legacy component files containing `across` and `through` statements. Upon encountering a Simscape component file written in the old format, the script creates a backup copy of the file (`filename.ssc.bak`), removes the `through` and `across` statements from the `setup` section, 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.

`ssc_update package` updates all the legacy component files located in the package. The argument, `package`, must be a top-level package name.

Note The package directory name begins with a leading `+` character, whereas the argument to `ssc_update` must omit the `+` character.

If you run the `ssc_update` command from inside the package directory structure, you can omit the argument.

Examples

For example, you have a custom package `+MyCapacitors`, which contains a component file `IdealCapacitor.ssc`, written in the old format:

```
component IdealCapacitor
% Ideal Capacitor
% Models an ideal (lossless) capacitor.

nodes
    p = foundation.electrical.electrical; % +:top
    n = foundation.electrical.electrical; % -:bottom
end
parameters
    C = { 1, 'F' }; % Capacitance
    V0 = { 0, 'V' }; % Initial voltage
end
variables
    i = { 0, 'A' }; % Current through variable
    v = { 0, 'V' }; % Voltage across variable
end
function setup
    if C <= 0
        error( 'Capacitance must be greater than zero' )
    end
end
```

```

    end
    through( i, p.i, n.i ); % Through variable i from node p to node n
    across( v, p.v, n.v ); % Across variable v from p to n
    v = V0;
end
equations
    i == C*v.der;    % Equation
end
end

```

To update the file to the new format, at the MATLAB command prompt, type:

```
ssc_update MyCapacitors;
```

This command creates a backup copy of the component file, `IdealCapacitor.ssc.bak`, in the same folder where the original file resides, and rewrites the `IdealCapacitor.ssc` file as follows:

```

component IdealCapacitor
% Ideal Capacitor
% Models an ideal (lossless) capacitor.

nodes
    p = foundation.electrical.electrical; % +:top
    n = foundation.electrical.electrical; % -:bottom
end
parameters
    C = { 1, 'F' };    % Capacitance
    V0 = { 0, 'V' };  % Initial voltage
end
variables
    i = { 0, 'A' };    % Current through variable
    v = { 0, 'V' };    % Voltage across variable
end
function setup
    if C <= 0
        error( 'Capacitance must be greater than zero' )
    end
    v = V0;
end

branches
    i : p.i -> n.i; % Through variable i from node p to node n
end

equations
    v == p.v - n.v; % Across variable v from p to n

    i == C*v.der;    % Equation
end
end

```

As you can see, the original through statement

```
through( i, p.i, n.i ); % Through variable i from node p to node n
```

has been replaced with the branches section:

```
branches
  i : p.i -> n.i; % Through variable i from node p to node n
end
```

The across statement

```
across( v, p.v, n.v ); % Across variable v from p to n
```

has been replaced with the equation

```
v == p.v - n.v; % Across variable v from p to n
```

in the equations section.

The other two statements in the `setup` section have been left unchanged.

Note Starting in R2019a, using `setup` is not recommended. Other constructs available in Simscape language let you achieve the same results without compromising run-time capabilities. For more information, see “`setup` is not recommended”.

Introduced in R2014a

subsystem2ssc

Convert subsystem containing Simscape blocks into equivalent Simscape file or files

Syntax

```
subsystem2ssc(subsystem)
subsystem2ssc(subsystem,targetFolder)
```

Description

`subsystem2ssc(subsystem)` converts a subsystem consisting entirely of Simscape blocks into a single Simscape component file, located in the current working folder. The function generates a composite component file based on the subsystem configuration. If the subsystem being converted contains nested subsystems, then the function generates several Simscape files, one for each subsystem.

You can mark member block and subsystem parameters for promotion to the top level, and the function automatically generates the corresponding code, similar to composite components. For more information, see “Promote Underlying Parameters to Subsystem Mask”.

The subsystem being converted cannot contain blocks from the Simscape “Utilities” library (such as Solver Configuration, PS-Simulink Converter, Simulink-PS Converter, Simscape Bus, and so on) because they are not authored in Simscape language and therefore have no equivalent textual representation. The exception is the Connection Port block, because it can be represented by the `connect` statements in Simscape language. If the subsystem contains a Simscape Component block, then during the conversion this block is replaced by its source component.

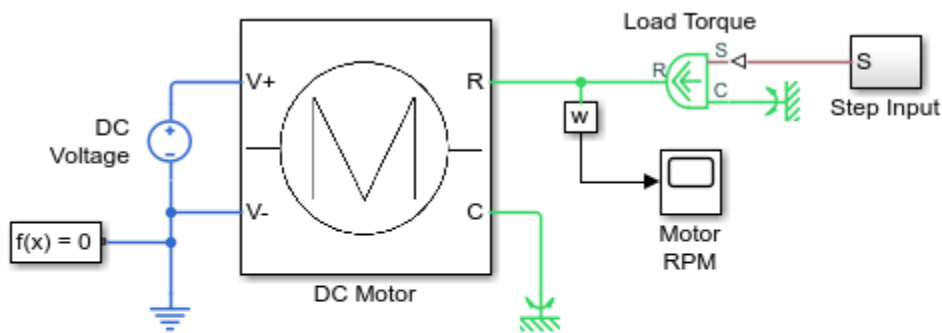
`subsystem2ssc(subsystem,targetFolder)` converts a subsystem containing Simscape blocks into equivalent Simscape component file, or files, located in `targetFolder`.

Examples

Convert Subsystem into Composite Component

Open the DC Motor example model.

```
ssc_dcmotor
```



Permanent Magnet DC Motor

1. [Plot](#) current and load torque ([see code](#))
2. [Explore simulation results](#) using [sscexplore](#)
3. [Learn more](#) about this example

This example model contains a subsystem named DC Motor. Convert this subsystem into a Simscape component file and place this file in your current working folder.

```
subsystem2ssc('ssc_dcmotor/DC Motor')
```

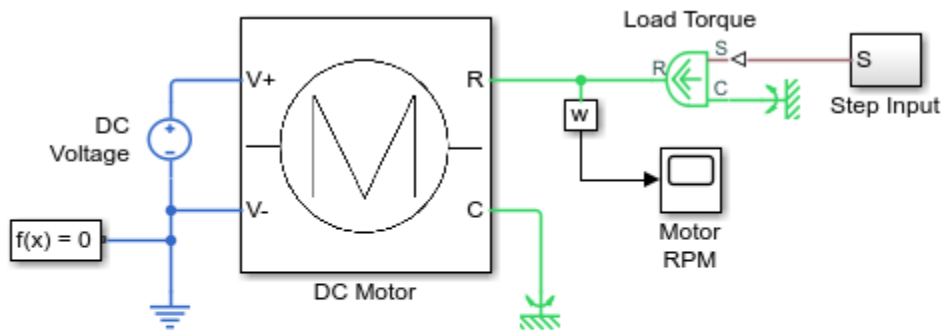
The function creates a file named DC_Motor.ssc in the current folder. Double-click the DC_Motor.ssc file to open it in the editor.

Double-click the DC Motor subsystem in the example model and compare it to the generated file. The generated composite component contains the same components, parameters, and connections as the original subsystem.

Convert Subsystem into Component in Target Folder

Open the DC Motor example model.

```
ssc_dcmotor
```



Permanent Magnet DC Motor

1. [Plot](#) current and load torque ([see code](#))
2. [Explore simulation results](#) using [sscexplore](#)
3. [Learn more](#) about this example

This example model contains a subsystem named DC Motor. Convert this subsystem into a Simscape component file and place the generated file in a designated folder.

```
subsystem2ssc('ssc_dcmotor/DC Motor', './MotorsLibrary')
```

The function creates a file named DC_Motor.ssc and places it into the folder named MotorsLibrary.

If the specified target folder does not exist, the function creates it.

Input Arguments

subsystem — Subsystem name and path

handle | character vector | string scalar

Subsystem name or identifier, specified as a block handle or a full name, including the path to the subsystem from the root of the model.

Example: 'ssc_dcmotor/DC Motor'

Data Types: double | char | string

targetFolder — Location of generated component files

character vector | string scalar

Location of generated component files, specified as a character vector or string scalar. The function places the generated component files in this folder. If the folder does not exist, the function creates it.

Example: './files'

Data Types: char | string

See Also

components | connections | ssc_build

Topics

“Converting Subsystems into Composite Components”

“About Composite Components”

Introduced in R2018b

twoPhaseFluidTables

Generate fluid property tables from REFPROP or CoolProp database

Syntax

```
fluidTables = twoPhaseFluidTables(uRange,pRange,mLiquid,mVapor,n,substance,
installPath)
twoPhaseFluidTables(block,fluidTables)
```

Description

`fluidTables = twoPhaseFluidTables(uRange,pRange,mLiquid,mVapor,n,substance,installPath)` retrieves the properties of a substance from a database and tabulates them for use in the Two-Phase Fluid Properties (2P) block. The substance can be a pure fluid such as R-134a or a predefined mixture such as R-404a, a ternary mixture of R-125, R-143a, and R-134a. The database can be REFPROP, an industry standard developed by NIST, or the open-source CoolProp.

The tables are stored in `fluidTables` as a structure array. A `liquid` substructure contains the data for the **Liquid Properties** tab and a `vapor` substructure contains that for the **Vapor Properties** tab. The fields of the substructures contain the fluid properties themselves—specific volume, specific entropy, kinematic viscosity, thermal conductivity, and others needed for simulation.

The tabulated data is in the two-dimensional space of `block`. A normalized specific internal energy varies across the rows, and absolute pressure varies across the columns. These variables have the special property that, when plotted against them, phase boundaries are vertical and straight, and `block` calculations are simpler and faster.

Normalized specific internal energy spans between the bounds in `uRange` across `mLiquid` rows for the liquid phase and `mVapor` rows for the vapor phase. Pressure spans between the bounds in `pRange` across `n` rows for both liquid and vapor phases. The properties of liquid-vapor mixtures are determined by interpolation between the pure phases.

`twoPhaseFluidTables(block,fluidTables)` assigns the properties stored in the structure array `fluidTables` to the parameters of the Two-Phase Fluid Properties (2P) block in the path `block`. Use the alternative syntax of this function if necessary to generate `fluidTables`.

Examples

Retrieve the Properties of Water from REFPROP

Get the properties of water from REFPROP and save them as tables in a structure named `waterTables`. Assume the REFPROP root folder to be `C:\REFPROP`. Specify a specific internal energy range of 25-4,000 kJ/kg split over 25 rows and a pressure range of 0.01-15 MPa split over 60 columns:

```
waterTables = twoPhaseFluidTables([25,4000],[0.01,15],25,25,60,...
'water','C:\Program Files\REFPROP\')
```


Retrieve the Properties of R-134a from CoolProp

Get the properties of R-134a from CoolProp and save them as tables in a structure named `r134aTables`. Assume the CoolProp root folder to be `C:\CoolProp`. Specify a specific internal energy range of 80-500 kJ/kg split over 25 rows and a pressure range of 0.001-3 MPa split over 60 columns:

```
r134aTables = twoPhaseFluidTables([80,500],[0.001,3],25,25,60,...
'R134a','py.CoolProp.CoolProp.PropsSI')
```

Assign the Properties of R-134a to a Block

Populate the parameter fields of a Two-Phase Fluid Properties (2P) block with the property tables of R-134a (stored previously in the structure `r134aTables`). Select the block and get its path name:

```
gcb
```

Assign to the block the tables of R-134a:

```
twoPhaseFluidTables(gcb,r134aTables)
```

Open the block dialog box and check that the parameter fields are specified in terms of `r134aTables` data.

Input Arguments

uRange — Lower and upper bounds of the specific internal energy range

two-element array in units of kJ/kg

Lower and upper bounds of the specific internal energy range onto which to map the fluid properties. The liquid tables range in specific internal energy from the lower bound to the liquid saturation value. The vapor properties range from the vapor saturation value to the upper bound. The bounds must encompass a range broad enough to include both liquid and vapor saturation values (both retrieved from the database).

Example: [30,4000]

pRange — Lower and upper bounds of the absolute pressure range

two-element array in units of MPa

Lower and upper bounds of the (absolute) pressure range onto which to map the fluid properties. The upper bound can be above the critical pressure of the fluid.

Example: [0.01,100]

mLiquid — Number of rows to include in the liquid tables

unitless scalar

Number of rows to include in the fluid tables for the liquid phase. Each row gives the fluid properties at a fixed value of the normalized specific internal energy, with the normalized specific internal energy increasing from left to right between the lower bound of `uRange` and the liquid saturation value.

Example: 25

mVapor — Number of rows to include in the vapor tables

unitless scalar

Number of rows to include in the fluid tables for the vapor phase. Each row gives the fluid properties at a fixed value of the normalized specific internal energy, with the normalized specific internal energy increasing from left to right between the vapor saturation value and the upper bound of `uRange`.

Example: 25

n — Number of columns to include in the fluid tables

unitless scalar

Number of columns to include in the fluid tables. Each column gives the fluid properties at a fixed pressure, with the pressure increasing from left to right between the bounds given in `pRange`. The number of columns is the same whether for the liquid or vapor phase.

Example: 60

substance — Database name for the fluid whose properties to retrieve

string scalar or character vector

Name of the fluid whose property tables the function is to construct. The name must be one recognized by the database specified. Refer to the database documentation for a list of valid fluid names.

Example: 'water'

installPath — File path to REFPROP installation folder or Python package path to CoolProp PropsSI function

string scalar or character vector

File path to the installation folder for REFPROP or Python package path to the CoolProp PropsSI function. If you are using CoolProp version 6.1.0 or earlier, then `InstallPath` is the file path to the folder in which the CoolProp MEX files are stored.

Example: 'C:\Program Files\REFPROP\'

Example: 'py.CoolProp.CoolProp.PropsSI'

block — Simulink path to the block to which to assign the fluid tables

string scalar or character vector

Simulink path to the Two-Phase Fluid Properties (2P) block whose fluid tables the function is to specify. To obtain the path to a block, click the block in the model canvas and, at the MATLAB command prompt, enter `gcb`.

Example: 'ssc_refrigeration/Two-Phase Fluid Properties (2P)'

fluidTables — Name of the structure array in which the fluid tables are stored

string scalar or character vector

Name of the structure array in which the fluid tables that the function is to specify are stored. The tables must have been generated in a prior call to this function. The structure array must be currently in the MATLAB workspace.

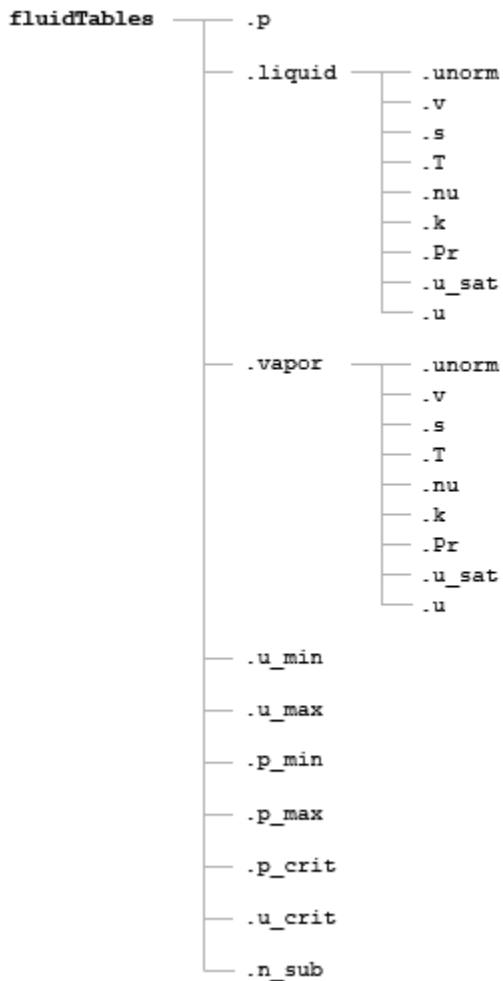
Example: 'waterTables'

Output Arguments

fluidTables — Name of the structure in which to save the fluid tables

structure array

Name of the structure array in which to save the fluid property tables. The array reflects in its hierarchy the structure of the Two-Phase Fluid Properties (2P) block.



Contents of the fluidTables Structure Array

See the table for more on the fields of the fluidTables structure array.

Field	Contents	Dimensions	Units
p	Pressure	n-by-1	MPa
unorm	Normalized specific internal energy	mLiquid- or mVapor-by-1	1

Field	Contents	Dimensions	Units
v	Specific volume	mLiquid- or mVapor-by-n	m ³ /kg
s	Specific entropy	mLiquid- or mVapor-by-n	kJ/(kg*K)
T	Temperature	mLiquid- or mVapor-by-n	K
nu	Kinematic viscosity	mLiquid- or mVapor-by-n	mm ² /s
k	Thermal conductivity	mLiquid- or mVapor-by-n	W/(m*K)
Pr	Prandtl number	mLiquid- or mVapor-by-n	1
u_sat	Saturation specific internal energy	mLiquid- or mVapor-by-1	kJ/kg
u	Specific internal energy	mLiquid- or mVapor-by-1	kJ/kg
u_min	Minimum specific internal energy	1-by-1	kJ/kg
u_max	Maximum specific internal energy	1-by-1	kJ/kg
p_min	Minimum pressure	1-by-1	MPa
p_max	Maximum pressure	1-by-1	MPa
p_crit	Critical pressure	1-by-1	MPa
u_crit	Specific internal energy at the critical point	1-by-1	kJ/kg
n_sub	Number of elements in the pressure vector below the critical pressure	1-by-1	1

Database Installation

Install REFPROP as described by NIST (<https://www.nist.gov/srd/refprop>). The root folder should contain a DLL file and a subfolder with FLD files—the fluid definitions. Only the 64-bit Windows

version of REFPROP is supported. This function has been tested with REFPROP versions 9.1, 9.1.1, and 10.

Install CoolProp as described by the CoolProp development team (<http://www.coolprop.org/coolprop/wrappers/MATLAB/>). CoolProp version 6.2 uses a Python wrapper and requires that you install Python prior to use. CoolProp is compatible with Windows, Linux, and Macintosh systems. `twoPhaseFluidTables` has been tested with CoolProp versions 6.0.0, 6.1.0, and 6.2.0.

See Also

Two-Phase Fluid Properties (2P)

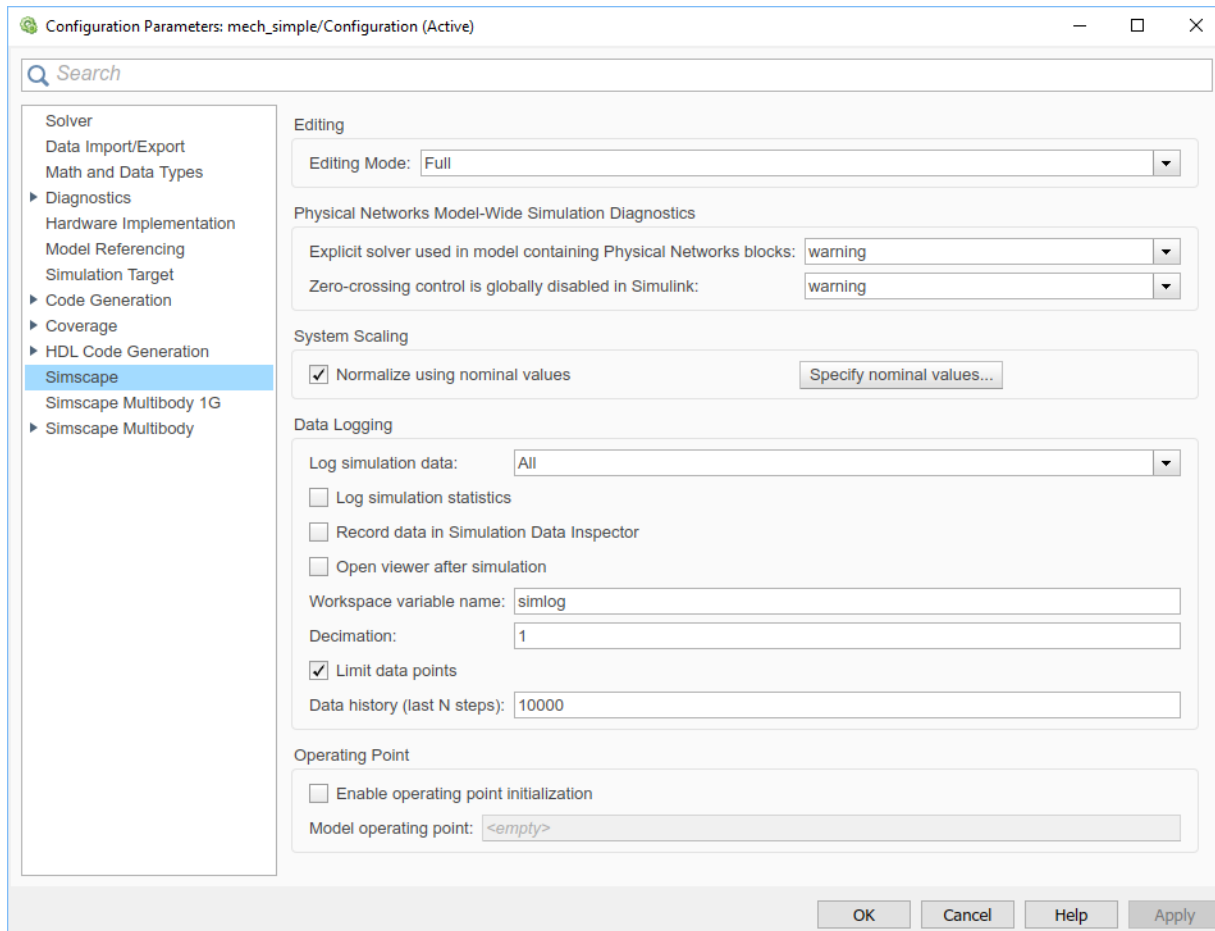
Topics

“Manually Generate Fluid Property Tables”

Introduced in R2015b

Configuration Parameters

Simscape Pane: General



In this section...

- “Simscape Pane Overview” on page 3-3
- “Editing Mode” on page 3-3
- “Explicit solver used in model containing Physical Networks blocks” on page 3-4
- “Zero-crossing control is globally disabled in Simulink” on page 3-5
- “Normalize using nominal values” on page 3-5
- “Log simulation data” on page 3-6
- “Log simulation statistics” on page 3-7
- “Record data in Simulation Data Inspector” on page 3-7
- “Open viewer after simulation” on page 3-8
- “Workspace variable name” on page 3-8
- “Decimation” on page 3-9
- “Limit data points” on page 3-9
- “Data history (last N steps)” on page 3-10

In this section...

“Enable operating point initialization” on page 3-11

“Model operating point” on page 3-11

Simscape Pane Overview

The **Editing Mode** parameter controls the Simscape Editing Mode functionality, which 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. Simscape add-on products include Simscape Driveline™, Simscape Electrical, Simscape Fluids, and Simscape Multibody. Use this functionality to perform multidomain physical modeling and simulation while minimizing the number of required licenses.

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 parameters in the **Physical Networks Model-Wide Simulation Diagnostics** section let you configure your preferences for solver-related warnings when you simulate models containing blocks from Simscape libraries.

The parameters in the **System Scaling** section let you apply scaling to model variables individually, based on their expected magnitude, and specify model-wide nominal values.

The parameters in the **Data Logging** section let you log simulation data to workspace.

The parameters in the **Operating Point** section let you initialize model from a previously saved operating point.

Configuration

This pane appears only if your model contains a block from the Simscape libraries (including Simscape add-on products).

See Also

- “About the Simscape Editing Mode”
- Working with Restricted and Full Modes
- “Harmonizing Simulink and Simscape Solvers”
- “System Scaling by Nominal Values”
- “About Simulation Data Logging”

Editing Mode

Set the editing mode of the model to either Full or Restricted.

Settings

Default: Full

Full

Sets the editing mode of the model to Full. In this mode, you can make any modifications to the model.

When you open a model in Full mode, the license manager checks out all the add-on product licenses for the blocks present in the model.

When you switch from Restricted to Full mode, the license manager checks whether the required add-on product licenses are available and checks them out. If some of the add-on product licenses are not available, the license manager issues an error and the model stays in Restricted mode.

Restricted

Sets the editing mode of the model to Restricted. In this mode, you can simulate the model, generate code, and make limited modifications.

When you open a model in Restricted mode, the license manager does not check out the add-on product licenses.

When you switch from Full to Restricted mode, all the add-on product licenses for the blocks present in the model remain checked out until the end of the MATLAB session.

Command-Line Information

Parameter: EditingMode

Type: character vector

Value: 'Full' | 'Restricted'

Default: 'Full'

See Also

- Saving a Model in Restricted Mode
- Switching from Restricted to Full Mode

Explicit solver used in model containing Physical Networks blocks

Specify whether or not the system will issue a warning or error upon simulation if the model uses an explicit solver.

Settings

Default: warning

warning

Makes the system issue a warning upon simulation if the model uses an explicit solver.

It is possible to choose any variable-step or fixed-step solver for models containing Simscape blocks. When you first create a model, the default Simulink solver is ode45. 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. To alert you to a potential issue, the system issues a warning when you use an explicit solver in a model containing Simscape blocks.

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.

Command-Line Information

Parameter: ExplicitSolverDiagnosticOptions

Type: character vector

Value: 'warning' | 'error' | 'none'

Default: 'warning'

See Also

“Switching from the Default Explicit Solver to Other Simulink Solvers”

Zero-crossing control is globally disabled in Simulink

Specify whether or not the system will issue a warning or error upon simulation if the **Zero-crossing control** parameter in the **Solver** pane is set to `Disable all`, which means that zero-crossing control is globally disabled.

Settings

Default: warning

warning

Makes the system issue a warning upon simulation if zero-crossing control is globally disabled.

error

Makes the system issue an error upon simulation if zero-crossing control is globally disabled.

Command-Line Information

Parameter: GlobalZcOffDiagnosticOptions

Type: character vector

Value: 'warning' | 'error'

Default: 'warning'

See Also

“Enabling or Disabling Simulink Zero-Crossing Detection”

Normalize using nominal values

Specify whether or not to scale the system based on nominal values.

Settings

Default: off

On

Applies scaling by nominal values. To view, add, and edit the value-unit pairs for the model, click the **Specify nominal values** button.

Off

Does not apply scaling by nominal values.

Command-Line Information

Parameter: SimscapeNormalizeSystem

Type: character vector

Value: 'on' | 'off'

Default: 'on'

Parameter: SimscapeNominalValues

Type: array of character vectors

Value: array of value-unit pairs

Default: `' [{"value": "1", "unit": "A"}, {"value": "1", "unit": "bar"}, {"value": "1", "unit": "cm^2"}, {"value": "1", "unit": "cm^3/s"}, {"value": "1", "unit": "kJ/kg"}, {"value": "1", "unit": "kW"}, {"value": "1", "unit": "l"}, {"value": "1", "unit": "N"}, {"value": "1", "unit": "N*m"}, {"value": "1", "unit": "V"}]'`

See Also

“System Scaling by Nominal Values”

Log simulation data

Specify whether or not the system logs simulation data to workspace.

Settings

Default: None

None

Performs no data logging upon simulation.

All

Upon simulating the model, logs simulation data from all the Simscape blocks in the model to a workspace variable specified by the **Workspace variable name** parameter.

Use local settings

Upon simulating the model, logs simulation data from selected Simscape blocks only to a workspace variable specified by the **Workspace variable name** parameter. If using this setting, you have to select blocks for data logging prior to simulating the model, otherwise no data will be logged.

Command-Line Information

Parameter: SimscapeLogType

Type: character vector

Value: 'none' | 'all' | 'local'

Default: 'none'

See Also

“Data Logging”

Log simulation statistics

Specify whether to log simulation statistics as part of simulation data.

Settings

Default: off



On

Logs simulation statistics.



Off

Does not log simulation statistics.

Command-Line Information

Parameter: SimscapeLogSimulationStatistics

Type: character vector

Value: 'on' | 'off'

Default: 'off'

See Also

“Data Logging”

Record data in Simulation Data Inspector

Specify whether to stream the data to Simulation Data Inspector, which allows you to view the data during and after simulation.

Settings

Default: off



On

Streams the data to Simulation Data Inspector. When you simulate the model, as soon as the streamed data becomes available, the Simulation Data Inspector button in the model toolbar highlights. Open the Simulation Data Inspector to view the data during simulation and to compare data for different simulation runs.



Off

Does not stream the data to Simulation Data Inspector.

Tip

This option lets you view the data both during and after the simulation. The **Record logged workspace data in Simulation Data Inspector** option on the **Data Import/Export** pane of the Configuration Parameters dialog box lets you view the logged Simscape data in the Simulation Data Inspector only after the simulation is over.

Command-Line Information

Parameter: SimscapeLogToSDI

Type: character vector

Value: 'on' | 'off'

Default: 'off'

See Also

“About Simulation Data Logging”

Open viewer after simulation

Specify whether to open Simscape Results Explorer at the end of simulation run.

Settings

Default: off

On

Automatically opens Simscape Results Explorer at the end of simulation run. If there is an open Simscape Results Explorer window linked to the session, reloads the simulation data into this window at the end of simulation run.

Off

Does not open Simscape Results Explorer automatically.

Tip

If the **Record data in Simulation Data Inspector** check box is also selected, then the Simulation Data Inspector opens instead of the Simscape Results Explorer.

Command-Line Information

Parameter: SimscapeLogOpenViewer

Type: character vector

Value: 'on' | 'off'

Default: 'off'

See Also

“Data Logging”

Workspace variable name

Specify the name of the workspace variable for simulation data logging.

Settings

Default: simlog

- The default value logs all the simulation data to a workspace variable named `simlog`.
- You can specify any other valid character vector as the workspace variable name.

Command-Line Information**Parameter:** SimscapeLogName**Type:** character vector**Value:** any valid value**Default:** 'simlog'**See Also**

"Data Logging"

Decimation

Lets you limit data points being logged, by skipping time steps. Logs data points for the first time step and every n th time step thereafter, where n is the decimation factor.

Settings**Default:** 1

- The default value logs simulation data for each step.
- You can specify any other positive integer number. For example, specifying 2 logs data points for every other time step, while specifying 10 logs data points for just one in ten steps.

Tips

- Saving data to workspace can slow down the simulation and consume memory. Use this parameter to limit the number of data points saved.
- Another way to limit the number of data points saved is using the **Limit data points** check box in conjunction with the **Data history (last N steps)** parameter. The two methods work independently from each other and can be used separately or together.

Command-Line Information**Parameter:** SimscapeLogDecimation**Type:** numeric**Value:** any positive integer value**Default:** 1**See Also**

"Data Logging"

Limit data points

Specify that the number of data points logged to workspace is limited to the value corresponding to the number of simulation steps specified by the **Data history (last N steps)** parameter.

Settings**Default:** on On

Limits the number of data points exported to workspace to those for the number of steps specified by the **Data history (last N steps)** parameter.

Off

Does not limit the number of data points.

Tips

- Saving data to workspace can slow down the simulation and consume memory. Use this parameter to limit the number of data points saved.
- Another way to limit the number of data points saved is using the **Decimation** parameter. The two methods work independently from each other and can be used separately or together.
- You must select the **Limit data points** check box before specifying the number of steps in the **Data history (last N steps)** parameter.

Command-Line Information

Parameter: SimscapeLogLimitData

Type: character vector

Value: 'on' | 'off'

Default: 'on'

See Also

“Data Logging”

Data history (last N steps)

Specify the number of simulation steps to limit the number of data points output to workspace. The workspace variable defined by the **Workspace variable name** parameter contains the data points corresponding to the last N steps of the simulation, where N is the value you specify for the **Data history (last N steps)** parameter. If the simulation contains fewer steps than the number specified, the workspace variable contains the data points for the whole simulation.

Settings

Default: 5000

- The default value logs simulation data for the last 5000 steps.
- You can specify any other positive integer number.

Tips

- Saving data to workspace can slow down the simulation and consume memory. Use this parameter to limit the number of data points saved.
- You must select the **Limit data points** check box before specifying the number of steps in the **Data history (last N steps)** parameter.

Command-Line Information

Parameter: SimscapeLogDataHistory

Type: numeric

Value: any positive integer value

Default: 5000

See Also

“Data Logging”

Enable operating point initialization

Specify whether to use operating point data to initialize the model.

Settings

Default: off



On

Initialize model from previously saved operating points.



Off

Do not use operating points to initialize the model.

Command-Line Information

Parameter: SimscapeUseOperatingPoints

Type: character vector

Value: 'on' | 'off'

Default: 'off'

See Also

“Using Operating Point Data for Model Initialization”

Model operating point

Specify the name of the operating point to use for model initialization.

Settings

Default: none

- There is no default value.
- Specify name of a workspace variable that contains operating point data.

Command-Line Information

Parameter: SimscapeOperatingPoint

Type: character vector

Value: any valid value

Default: none

See Also

“Using Operating Point Data for Model Initialization”

Model Advisor Checks

Simscape Model Advisor Checks

In this section...

“Simscape Checks Overview” on page 4-2
“Modeling Physical Systems Checks Overview” on page 4-3
“Check consistency of block parameter units” on page 4-3
“Check for outdated AC source blocks” on page 4-3
“Check for dry hydraulic nodes” on page 4-4
“Simscape Checks Overview” on page 4-5
“Check Simscape Solver Configuration block settings” on page 4-5
“Check Fluid dynamic compressibility option” on page 4-6
“Electrical Checks Overview” on page 4-7
“Check Model dynamics option” on page 4-7
“Check Noise mode option” on page 4-8
“Check Parasitic ground conductance” on page 4-9
“Check Resolver parameterization option” on page 4-10
“Check Simulation mode option” on page 4-11
“Check Transmission Line blocks” on page 4-12
“Check Zero sequence” on page 4-13
“Fluids Checks Overview” on page 4-14
“Check Valve opening dynamics option” on page 4-14
“Driveline Checks Overview” on page 4-15
“Check Gear friction model option” on page 4-15
“Check Tire compliance option” on page 4-16
“Check Engine time constant option” on page 4-17
“Check Dog clutch model option” on page 4-17
“Check Losses model option” on page 4-18
“Check Model transmission lag option” on page 4-19
“Check Hard stop model option” on page 4-20
“Check and update outdated Simscape Physical Signal blocks” on page 4-21
“Check usage of Simscape event variables with unspecified priority” on page 4-22

Simscape Checks Overview

Use Simscape Model Advisor checks to identify Simscape blocks with ambiguous setting of parameter units, or outdated Simscape blocks in your model.

See Also

- “Run Model Advisor Checks”

Modeling Physical Systems Checks Overview

Use the Modeling Physical Systems Model Advisor checks to identify Simscape blocks with ambiguous setting of parameter units.

See Also

- “Run Model Advisor Checks”

Check consistency of block parameter units

Check model for Simscape blocks with ambiguous setting of parameter units.

Description

This check identifies blocks in your model that have an ambiguous setting of parameter units. This situation most often applies to frequency and angular velocity units.

For example, a parameter expected in Hz (1/s) may be specified in the block dialog with unit of rad/s. These units are commensurate, but not directly convertible, and using one instead of the other may result in unexpected conversion factors applied to the numerical value by the block equations. The purpose of the check is to verify that the specified unit matches your design intent.

Available with Simscape.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains blocks where parameter units are not directly convertible to those expected by the block.</p> <p>After running the check, you get a table of results 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.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>Double-click the highlighted block, verify the parameter unit setting and correct it, if necessary. Then save and reload the model.</p>

See Also

- “Units for Angular Velocity and Frequency”

Check for outdated AC source blocks

Check model for AC source blocks that should be updated to the current version of the product.

Description

This check identifies AC source blocks in your model that do not match the latest version of the block in the Simscape block libraries.

Blocks from previous versions may be missing parameters available in the latest version. In this case, simulating the model may produce warnings or unexpected results.

Available with Simscape.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains outdated AC source blocks.</p> <p>After running the check, you get a list of links to the outdated blocks in the right pane of the Model Advisor window. Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Model Advisor window and click the Update button.</p> <ul style="list-style-type: none"> • If the automatic update is successful, the Result 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. <p>Alternatively, you can consult the Upgrade Advisor to migrate the model to the latest version of Simscape software.</p>

See Also

- “Model Upgrades”

Check for dry hydraulic nodes

Check model for hydraulic nodes that are considered dry due to a lack of compliance.

Description

This check identifies hydraulic nodes in a Simscape model that are considered dry due to a lack of compliance.

The presence of dry hydraulic nodes can reduce the solver robustness in complex Simscape models. By adding a hydraulic chamber to a node, you can considerably improve the convergence and computational efficiency of a model. Adding a chamber adds a degree of freedom. By adding a chamber, you replace a complex algebraic constraint (the dry node) with a dynamic constraint. The hydraulic chamber is represented by the Constant Volume Hydraulic Chamber block.

Available with Simscape.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains hydraulic nodes that are considered dry due to a lack of compliance.</p> <p>After running the check, you get a table of results in the right pane of the Model Advisor window. The first column lists the dry nodes found, with the middle column listing the blocks connected to each dry node. Each cell in the middle column of the table contains a link to the block in question, and the corresponding cell in the third column contains the name of port that connects to the dry node.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>Consider adding one Constant Volume Hydraulic Chamber block to each dry node in the list.</p>

See Also

- “Considerations For Building Fluids Models” (Simscape Fluids)

Simscape Checks Overview

Use these checks to optimize real-time simulation performance of models containing Simscape blocks. The top-level **Simscape checks** are applicable to all physical models. If you have add-on product licenses, the **Simscape checks** folder also includes corresponding subfolders, such as **Driveline checks** or **Electronics checks**. Each of the subfolders contains checks that target specific blocks from that add-on product. If your model contains blocks from an add-on product, run the checks in the respective subfolder in addition to the top-level **Simscape checks**.

See Also

- “Improve Simulation Performance Using Performance Advisor”

Check Simscape Solver Configuration block settings

Check model for Solver Configuration blocks with settings that are suboptimal for real-time simulation.

Description

This check identifies Solver Configuration blocks in your model where settings are suboptimal for real-time simulation.

For optimal results, Solver Configuration blocks should have the following options selected: **Use local solver** and **Use fixed-cost runtime consistency iterations**.

Available with Simscape.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains Solver Configuration blocks that do not use local solver or fixed-cost runtime consistency iterations.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic Solver Configuration block. The corresponding cells in the second and third columns contain the current setting for Use local solver and Use fixed-cost runtime consistency iterations, respectively.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>Alternately, double-click the highlighted block, select the Use local solver and Use fixed-cost runtime consistency iterations check boxes, verify the Sample time parameter value and correct it, if necessary. Then save and reload the model.</p> <p>After updating the blocks, either manually or automatically, rerun the check.</p> <p>If the check passes, the table of results contains links to all the Solver Configuration blocks in the model, along with the local solver Sample time parameter value for each block. Check these values for consistency.</p>

See Also

- Solver Configuration

Check Fluid dynamic compressibility option

Check model for blocks with **Fluid dynamic compressibility** settings that are suboptimal for real-time simulation.

Description

This check identifies blocks in your model where the **Fluid dynamic compressibility** parameter setting is suboptimal for real-time simulation. This parameter exists in Thermal Liquid blocks, such as pipes and energy converters.

For optimal results, **Fluid dynamic compressibility** should be set to Off.

Available with Simscape.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains Thermal Liquid blocks that account for fluid dynamic compressibility.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column contains the current setting for the Fluid dynamic compressibility parameter.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all blocks with a Fluid dynamic compressibility option have been updated.</p>

See Also

- “Modeling Thermal Liquid Systems”

Electrical Checks Overview

Use these checks to optimize real-time simulation performance of your Simscape Electrical model. These checks target specific Simscape Electrical blocks. Run these checks in addition to the top-level **Simscape checks**, which are applicable to all physical models.

See Also

- “Improve Simulation Performance Using Performance Advisor”

Check Model dynamics option

Check model for blocks with **Model dynamics** settings that are suboptimal for real-time simulation.

Description

This check identifies blocks in your model where the **Model dynamics** or **Dynamics** parameter setting is suboptimal for real-time simulation. This parameter exists in several types of blocks, and the drop-down options vary between block types. For optimal results, the option that corresponds to not modeling the dynamics should be selected in all cases:

- For switches, Voltage-Controlled Oscillator, DC-DC Converter — No dynamics
- For Accelerometer, Gyro, Pressure Transducer blocks — No dynamics - Suitable for HIL

Available with Simscape Electrical.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains blocks with dynamics modeling enabled.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column contains the current setting for the Model dynamics or Dynamics parameter.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all blocks with a Model dynamics option have been updated.</p>

See Also

- Accelerometer
- DC-DC Converter
- DPDT Switch
- DPST Switch
- Gyro
- Pressure Transducer
- SPDT Switch
- SPST Switch
- Voltage-Controlled Oscillator

Check Noise mode option

Check model for blocks with **Noise mode** settings that are suboptimal for real-time simulation.

Description

This check identifies blocks in your model where the **Noise mode** parameter setting is suboptimal for real-time simulation. This parameter exists in blocks that can generate thermal noise, such as resistors and electrical sources, and you can enable or disable this option. Simulating with noise enabled slows down simulation. For optimal results, **Noise mode** should be set to **Disabled**.

Available with Simscape Electrical.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains blocks where thermal noise generation is enabled.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column contains the current setting for the Noise mode parameter.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all blocks with a Noise mode option have been updated.</p>

See Also

- Current Source
- Resistor
- Voltage Source

Check Parasitic ground conductance

Check model for blocks with **Parasitic ground conductance** settings that are suboptimal for real-time simulation.

Description

This check identifies blocks in your model where the **Parasitic ground conductance** parameter setting is suboptimal for real-time simulation. Simulating with noise enabled slows down simulation. For optimal results, **Parasitic ground conductance** should be set to 0.

Available with Simscape Electrical.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains blocks with nonzero values for Parasitic ground conductance.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column contains the current setting for the Parasitic ground conductance parameter.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all blocks with a Parasitic ground conductance parameter have been updated.</p>

See Also

- Floating Neutral
- Neutral Port

Check Resolver parameterization option

Check model for Resolver blocks with **Parameterization** settings that are suboptimal for real-time simulation.

Description

This check identifies Resolver blocks in your model where the **Parameterization** parameter setting is suboptimal for real-time simulation. For optimal results, **Parameterization** should be set to Specify transformation ratio and omit dynamics.

Available with Simscape Electrical.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains Resolver blocks where the Parameterization setting is other than Specify transformation ratio and omit dynamics.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column contains the current setting for the Parameterization parameter.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all Resolver blocks have been updated.</p>

See Also

- Resolver

Check Simulation mode option

Check model for blocks with **Simulation mode** settings that are suboptimal for real-time simulation.

Description

This check identifies blocks in your model where the **Simulation mode** parameter setting is suboptimal for real-time simulation. This parameter exists in two types of blocks:

- Pulse-width modulated (PWM) actuators and drivers, such as H-Bridge, where you can choose between PWM and Averaged modes.
- Stepper motors and drivers, where you can choose between Stepping and Averaged modes.

For optimal results, **Simulation mode** should be set to Averaged.

Available with Simscape Electrical.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains blocks where the Simulation mode parameter setting is other than Averaged.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column contains the current setting for the Simulation mode parameter.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all blocks with a Simulation mode option have been updated.</p> <p>After updating the blocks, verify that parameters specific to the Averaged option (for example, Step rate sensitivity) have suitable values.</p>

See Also

- Controlled PWM Voltage
- Generic Linear Actuator
- Generic Rotary Actuator
- H-Bridge
- Stepper Motor
- Stepper Motor Driver
- Unipolar Stepper Motor
- Unipolar Stepper Motor Driver

Check Transmission Line blocks

Check model for Transmission Line blocks with **Model type** settings that are suboptimal for real-time simulation.

Description

This check identifies Transmission Line blocks in your model where the **Model type** parameter setting is suboptimal for real-time simulation. For optimal results, **Model type** should be set to **Delay-based** and **lossless**. This option is the most efficient numerically, because the other options need multiple segments (typically several tens of segments) to get reasonable accuracy.

Available with Simscape Electrical.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains Transmission Line blocks where the Model type setting is other than Delay-based and lossless.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column contains the current setting for the Model type parameter.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all Transmission Line blocks have been updated.</p> <p>After updating the blocks, verify that parameters specific to the Delay-based and lossless option (such as Transmission delay and Characteristic impedance) have suitable values.</p>

See Also

- Transmission Line

Check Zero sequence

Check model for Simscape Electrical blocks with **Zero sequence** settings that are suboptimal for real-time simulation.

Description

This check identifies blocks in your model where the **Zero sequence** parameter setting is suboptimal for real-time simulation. These parameters exist in multiple blocks in the Machines library. For optimal results, set the **Zero sequence** option to Exclude.

Available with Simscape Electrical.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains blocks where the Zero sequence parameter has a value of Include.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column contains the current setting for the Zero sequence parameter.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all blocks with a suboptimal Zero sequence value have been updated.</p>

See Also

- “Electromechanical” (Simscape Electrical)

Fluids Checks Overview

Use these checks to optimize real-time simulation performance of your Simscape Fluids model. These checks target specific Simscape Fluids blocks. Run these checks in addition to the top-level **Simscape checks**, which are applicable to all physical models.

See Also

- “Improve Simulation Performance Using Performance Advisor”

Check Valve opening dynamics option

Check model for valve blocks with **Opening dynamics** settings that are suboptimal for real-time simulation.

Description

This check identifies blocks in your model where the **Opening dynamics** parameter setting is suboptimal for real-time simulation. This parameter exists in several directional and pressure control valves. By default, these valve models do not include opening dynamics. For optimal results, **Opening dynamics** should be set to **Include valve opening dynamics**. This option avoids instantaneous area changes, which is important in simulations with the local solver, and provides continuous behavior that is more physically realistic.

Available with Simscape Fluids.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains valves with opening dynamics modeling disabled.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column indicates the current setting for modeling the valve opening dynamics.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all blocks with an Opening dynamics option have been updated.</p>

See Also

- Check Valve
- Hydraulically Operated Remote Control Valve
- Pilot-Operated Check Valve

- Pressure Compensator
- Pressure Reducing 3-Way Valve
- Pressure Reducing Valve
- Pressure Relief Valve
- Shuttle Valve

Driveline Checks Overview

Use these checks to optimize real-time simulation performance of your Simscape Driveline model. These checks target specific Simscape Driveline blocks. Run these checks in addition to the top-level **Simscape checks**, which are applicable to all physical models.

See Also

- “Improve Simulation Performance Using Performance Advisor”

Check Gear friction model option

Check model for gear blocks with **Friction model** settings that are suboptimal for real-time simulation.

Description

This check identifies gear blocks in your model where the **Friction model** parameter setting is suboptimal for real-time simulation. This parameter exists in all gears, and the drop-down options vary between blocks. For optimal results, the option that corresponds to not modeling the friction losses should be selected in all cases:

- For Worm Gear, Sun-Planet Worm Gear, and Leadscrew blocks — `No friction losses – Suitable for HIL simulation`
- For all other gears — `No meshing losses – Suitable for HIL simulation`

This check does not apply to thermal variants of gear blocks, because these variants always model friction losses.

Available with Simscape Driveline.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains gear blocks where modeling of the friction or meshing losses is enabled.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column contains the current setting for the Friction model parameter.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all gear blocks with a friction losses option have been updated.</p>

See Also

- “Gears” (Simscape Driveline)

Check Tire compliance option

Check model for tire blocks with **Compliance** settings that are suboptimal for real-time simulation.

Description

This check identifies tire blocks in your model where the **Compliance** parameter setting is suboptimal for real-time simulation. This parameter exists in all tire blocks and specifies whether the model includes longitudinal stiffness and damping. For optimal results, **Compliance** should be set to **No compliance – Suitable for HIL simulation**.

Available with Simscape Driveline.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains tire blocks where the model includes longitudinal stiffness and damping.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column indicates the current setting for the Compliance parameter.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all tire blocks with a compliance option have been updated.</p>

See Also

- Tire (Friction Parameterized)
- Tire (Magic Formula)
- Tire (Simple)

Check Engine time constant option

Check model for Generic Engine blocks with **Engine time constant** settings that are suboptimal for real-time simulation.

Description

This check identifies Generic Engine blocks in your model where the **Engine time constant** parameter setting is suboptimal for real-time simulation. This parameter lets you model engine dynamics, that is, the time lag of the engine response. For optimal results, **Engine time constant** should be set to **No time constant – Suitable for HIL simulation**.

Available with Simscape Driveline.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains Generic Engine blocks where the model includes the time lag of the engine response.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column indicates the current setting for modeling the engine dynamics.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all the Generic Engine blocks with an Engine time constant option have been updated.</p>

See Also

- Generic Engine

Check Dog clutch model option

Check model for dog clutch blocks with **Torque transmission model** settings that are suboptimal for real-time simulation.

Description

This check identifies blocks in your model where the **Torque transmission model** parameter setting is suboptimal for real-time simulation. This parameter exists in all blocks that let you model a dog clutch, such as Dog Clutch, Synchronizer, and Double-Sided Synchronizer. The parameter controls

whether the torque transmission model accounts for backlash, torsional compliance, and contact forces between ring and hub teeth. For optimal results, **Torque transmission model** should be set to **Friction clutch approximation - Suitable for HIL and linearization**, which models clutch engagement simply as a friction phenomenon between the ring and the hub.

Available with Simscape Driveline.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains dog clutch blocks where the torque transmission model accounts for backlash and other special effects.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column indicates the current torque transmission model setting.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all the blocks with a Torque transmission model option have been updated.</p>

See Also

- Dog Clutch
- Double-Sided Synchronizer
- Synchronizer

Check Losses model option

Check model for Variable Ratio Transmission blocks with **Losses model** settings that are suboptimal for real-time simulation.

Description

This check identifies Variable Ratio Transmission blocks in your model where the **Losses model** parameter setting is suboptimal for real-time simulation. This parameter specifies how to implement friction losses from nonideal torque transfer. For optimal results, **Losses model** should be set to **No losses – Suitable for HIL simulation**.

Available with Simscape Driveline.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains Variable Ratio Transmission blocks where the torque transmission model accounts for friction losses from nonideal torque transfer.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column indicates the current setting for modeling the friction losses.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all Variable Ratio Transmission blocks with a Losses model option have been updated.</p>

See Also

- Variable Ratio Transmission

Check Model transmission lag option

Check model for Torque Converter blocks with **Model transmission lag** settings that are suboptimal for real-time simulation.

Description

This check identifies Torque Converter blocks in your model where the **Model transmission lag** parameter setting is suboptimal for real-time simulation. This parameter specifies how to model transmission lag from input to output driveshaft. For optimal results, **Model transmission lag** should be set to **No lag – Suitable for HIL simulation**.

Available with Simscape Driveline.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains Torque Converter blocks where torque is transferred with a time lag.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column indicates the current setting for modeling the transmission lag.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all Torque Converter blocks with a Model transmission lag option have been updated.</p>

See Also

- Torque Converter

Check Hard stop model option

Check model for blocks with **Hard stop model** settings that are suboptimal for real-time simulation.

Description

This check identifies Shock Absorber and Torsional Spring-Damper blocks in your model where the **Hard stop model** parameter setting is suboptimal for real-time simulation. This parameter controls whether the model includes hard stops. For optimal results, **Hard stop model** should be set to **No hard-stops – Suitable for HIL simulation**, which eliminates the hard stop force contribution and enhances simulation speed.

Available with Simscape Driveline.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains Shock Absorber or Torsional Spring-Damper blocks with hard stops.</p> <p>After running the check, you get a table of results in the right pane of the Performance Advisor window. Each cell in the first column of the table contains a link to the problematic block. The corresponding cell in the second column indicates the current hard stop model setting.</p> <p>Clicking a link highlights the corresponding block in the model.</p>	<p>To update the blocks, scroll down the right pane of the Performance Advisor window and click the Update button.</p> <p>If the automatic update is successful, the Result box displays a message that all the blocks with a Hard stop model option have been updated.</p>

See Also

- Shock Absorber
- Torsional Spring-Damper

Check and update outdated Simscape Physical Signal blocks

Check model for Physical Signal blocks that should be updated to the current version of the product.

Description

This check identifies Physical Signal blocks in your model that do not match the latest version of the block in the Simscape block libraries. Blocks from previous versions do not propagate physical signal size and units.

Available with Simscape.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains legacy Simscape Physical Signal blocks that do not propagate units.</p> <p>After running the check, you get a list of links to the outdated blocks in the right pane of the Upgrade Advisor window. Clicking a link highlights the corresponding block in the model.</p> <p>The links can be divided in multiple groups:</p> <ul style="list-style-type: none"> • The first group contains outdated blocks that can be updated automatically, if any. The Upgrade link under this group converts all these blocks to the latest version. • Sometimes the legacy blocks cannot be converted automatically because direct conversion will result in a compilation error or a different answer. These blocks are listed in a table, grouped by the underlying issue. Each row of the table contains: <ol style="list-style-type: none"> 1 A list of links to blocks affected by the issue 2 Issue description 3 Switch to new version link 	<p>To update the blocks:</p> <ul style="list-style-type: none"> • If the model contains outdated blocks that can be updated automatically, click the Upgrade link under the list of block links. • If the message says that some of the blocks, when switched to propagate signal units, will result in a compilation error or different answer, review the table that groups the blocks based on the underlying issue. For each table row, click the Switch to new version link to convert all blocks listed in the first cell of this row, and then visit the affected blocks individually to resolve the issue.

See Also

- “Model Upgrades”
- “Upgrading Models with Legacy Physical Signal Blocks”

Check usage of Simscape event variables with unspecified priority

Check model for event variables with unspecified priority that can affect initialization results.

Description

This check identifies custom components in your model that have event variables declared with unspecified priority and used outside of a when clause. This situation can affect model initialization results.

Prior to R2019b, event variables were not part of the initial solve and event variable targets always had high priority. Now, event variables are treated the same as any other type of variables during initial solve. Therefore, their default priority is none, and you might have to explicitly declare them as high-priority to achieve the previous model behavior during initialization.

Available with Simscape.

Results and Recommended Actions

Condition	Recommended Action
<p>This model contains custom components with event variables that are:</p> <ul style="list-style-type: none"> • Declared with unspecified priority • Used outside of a when clause <p>After running the check, you get a table of results in the right pane of the Upgrade Advisor window. Clicking a link highlights the corresponding block in the model. The results also point to the location of the variable declaration in the source code.</p>	<p>To update the model, inspect variables flagged by the check in the Variable Viewer. For variables that initialize to a value other than the target, change the priority to high, either in the underlying source code or in the Variables section of the block interface.</p>

See Also

- “Model Upgrades”
- “Block-Level Variable Initialization”
- `variables`

Bibliography

- [1] Andersen, B. W. *The Analysis and Design of Pneumatic Systems*. New York: John Wiley & Sons, 1967.
- [2] Armstrong, B., and C. C. de Wit. "Friction Modeling and Compensation." *The Control Handbook*. Boca Raton, Florida: CRC Press, 1995.
- [3] Beater, P. *Pneumatic Drives. System Design, Modeling and Control*. New York: Springer, 2007.
- [4] Brauer J.R. *Magnetic Actuators and Sensors*. Piscataway, NJ: Wiley-IEEE Press, 2006.
- [5] Fitzgerald A.E., Kingsley C., Jr., and Umans S.D. *Electric Machinery*. Sixth edition. New Delhi: Tata McGraw Hill, 2002.
- [6] Holcke, Jan. "Frequency Response of Hydraulic Hoses." Licentiate Thesis. Royal Institute of Technology, KTH, Stockholm, 2002.
- [7] Kahaner, D., Cleve Moler, and Stephen Nash. *Numerical Methods and Software*. Englewood Cliffs, NJ: Prentice Hall, 1988.
- [8] Lorenz R.D. and Haines L.P. *Understanding Modern Power Conversion*. Third edition. Madison: University of Wisconsin, 2000.
- [9] Martin, H. *The Design of Hydraulic Components and Systems*. New York: Ellis Horwood, 1995.
- [10] Meritt, H. E. *Hydraulic Control Systems*. New York: John Wiley & Sons, 1967.
- [11] Moran M.J. and Shapiro H.N. *Fundamentals of Engineering Thermodynamics*. Second edition. New York: John Wiley & Sons, 1992.
- [12] Pêcheux, F., B. Allard, C. Lallement, A. Vachoux, and H. Morel. "Modeling and Simulation of Multi-Discipline Systems using Bond Graphs and VHDL-AMS." International Conference on Bond Graph Modeling and Simulation (ICBGM). New Orleans, USA, 23-27 Jan. 2005.
- [13] Press, W. H., B. P. Flannery, S. A. Teulkolsky, and W. T. Wetterling. *Numerical Recipes in C: The Art of Scientific Computing*. New York: Cambridge University Press, 1992.
- [14] Sanville, F. E. "A New Method of Specifying the Flow Capacity of Pneumatic Fluid Power Valves." Paper D3, p.37-47. BHRA. Second International Fluid Power Symposium, Guildford, England, 1971.
- [15] Shapiro, A. H. *The Dynamics and Thermodynamics of Compressible Fluid Flow*. Vol. 1. New York: John Wiley & Sons, 1953.
- [16] White, F. M. *Viscous Fluid Flow*. New York: McGraw-Hill, 1991.
- [17] Yeapple, F. *Fluid Power Design Handbook*, Third edition. New York: Marcel Dekker, 1996.

