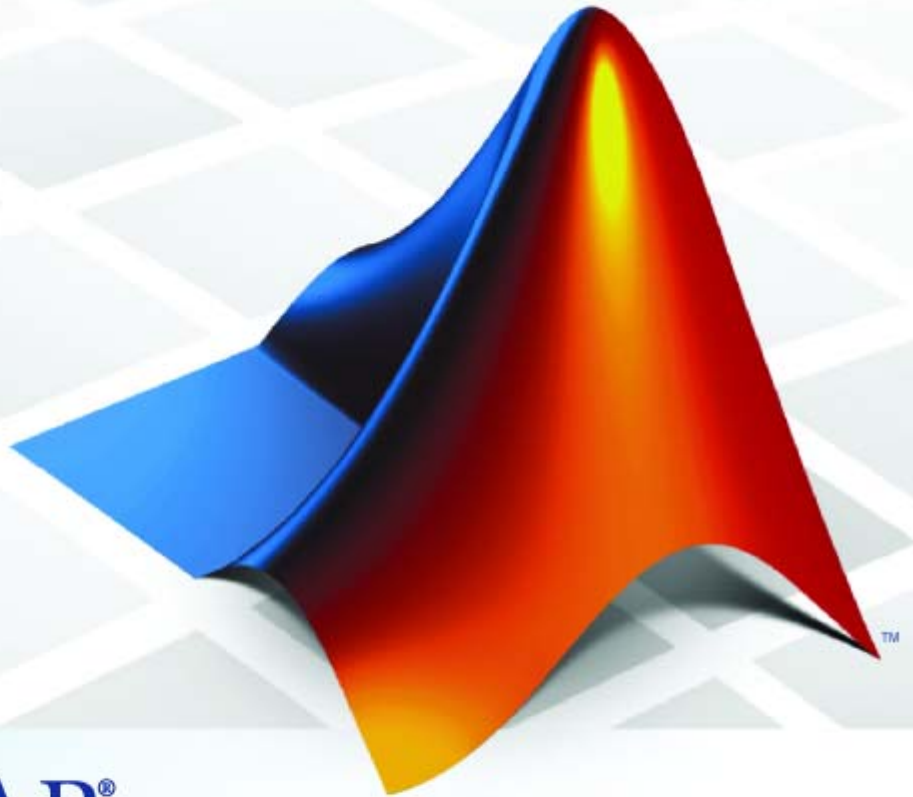


Simscape™ 3

Reference



MATLAB®
& **SIMULINK®**

How to Contact The MathWorks



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



suggest@mathworks.com Product enhancement suggestions
bugs@mathworks.com Bug reports
doc@mathworks.com Documentation error reports
service@mathworks.com Order status, license renewals, passcodes
info@mathworks.com Sales, pricing, and general information



508-647-7000 (Phone)



508-647-7001 (Fax)



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

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

Simscape™ Reference

© COPYRIGHT 2007–2009 by The MathWorks, Inc.

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

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

Trademarks

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

Patents

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

Revision History

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

Block Reference

1

Foundation	1-2
Electrical	1-2
Hydraulic	1-4
Mechanical	1-6
Physical Signals	1-9
Thermal	1-11
Utilities	1-12

Blocks — Alphabetical List

2

Command Reference

3

Language Reference

4

Simscape Foundation Domains

5

Domain Types and Directory Structure	5-2
---	-----

Electrical Domain	5-4
Hydraulic Domain	5-5
Mechanical Rotational Domain	5-7
Mechanical Translational Domain	5-8
Thermal Domain	5-9

Configuration Parameters

6

Simscape Pane: General	6-2
Simscape Pane Overview	6-3
Editing Mode	6-4
Explicit solver used in model containing Physical Networks blocks	6-6
Input filtering used in model containing Physical Networks blocks	6-8

Glossary

Index

Block Reference

Foundation (p. 1-2)

Basic electrical, hydraulic, mechanical, and physical signal blocks

Utilities (p. 1-12)

Essential environment blocks for creating Physical Networks models

Foundation

Electrical (p. 1-2)	Basic electrical diagram blocks, such as inductors, diodes, capacitors, sensors and sources
Hydraulic (p. 1-4)	Basic hydraulic diagram blocks, such as orifices, chambers, sensors and sources, and hydraulic utilities
Mechanical (p. 1-6)	Mechanical elements for rotational and translational motion, as well as mechanical sensors and sources
Physical Signals (p. 1-9)	Blocks for transmitting physical control signals
Thermal (p. 1-11)	Basic thermal blocks, such as heat transfer blocks, thermal mass, sensors and sources

Electrical

Electrical Elements (p. 1-2)	Electrical building blocks, such as inductors, diodes, and capacitors
Electrical Sensors (p. 1-3)	Current and voltage sensors
Electrical Sources (p. 1-4)	Current and voltage sources

Electrical Elements

Capacitor	Simulate linear capacitor in electrical systems
Diode	Simulate piecewise linear diode in electrical systems
Electrical Reference	Simulate connection to electrical ground

Gyrator	Simulate ideal gyrator in electrical systems
Ideal Transformer	Simulate ideal transformer in electrical systems
Inductor	Simulate linear inductor in electrical systems
Mutual Inductor	Simulate mutual inductor in electrical systems
Op-Amp	Simulate ideal operational amplifier
Resistor	Simulate linear resistor in electrical systems
Rotational Electromechanical Converter	Provide interface between electrical and mechanical rotational domains
Switch	Simulate switch controlled by external physical signal
Translational Electromechanical Converter	Provide interface between electrical and mechanical translational domains
Variable Resistor	Simulate linear variable resistor in electrical systems

Electrical Sensors

Current Sensor	Simulate current sensor in electrical systems
Voltage Sensor	Simulate voltage sensor in electrical systems

Electrical Sources

AC Current Source	Simulate ideal sinusoidal current source
AC Voltage Source	Simulate ideal constant voltage source
Controlled Current Source	Simulate ideal current source driven by input signal
Controlled Voltage Source	Simulate ideal voltage source driven by input signal
Current-Controlled Current Source	Simulate linear current-controlled current source
Current-Controlled Voltage Source	Simulate linear current-controlled voltage source
DC Current Source	Simulate ideal constant current source
DC Voltage Source	Simulate ideal constant voltage source
Voltage-Controlled Current Source	Simulate linear voltage-controlled current source
Voltage-Controlled Voltage Source	Simulate linear voltage-controlled voltage source

Hydraulic

Hydraulic Elements (p. 1-5)	Hydraulic building blocks, such as orifices, chambers, and hydro-mechanical converters
Hydraulic Sensors (p. 1-6)	Hydraulic sensors and sources
Hydraulic Sources (p. 1-6)	Hydraulic sensors and sources
Hydraulic Utilities (p. 1-6)	Basic hydraulic environment blocks, such as custom hydraulic fluid

Hydraulic Elements

Constant Area Orifice	Simulate hydraulic orifice with constant cross-sectional area
Constant Volume Chamber	Simulate hydraulic capacity of constant volume
Fluid Inertia	Simulate pressure differential across tube or channel due to change in fluid velocity
Hydraulic Reference	Simulate connection to atmospheric pressure
Linear Hydraulic Resistance	Simulate hydraulic pipeline with linear resistance losses
Piston Chamber	Simulate variable volume hydraulic capacity in cylinders
Resistive Tube	Simulate hydraulic pipeline which accounts for friction losses only
Rotational Hydro-Mechanical Converter	Simulate ideal hydro-mechanical transducer as building block for rotary actuators
Translational Hydro-Mechanical Converter	Simulate single chamber of hydraulic cylinder as building block for various cylinder models
Variable Area Orifice	Simulate hydraulic variable orifice created by cylindrical spool and sleeve
Variable Chamber	Simulate hydraulic capacity of variable volume with compressible fluid

Hydraulic Sensors

Ideal Hydraulic Flow Rate Sensor	Simulate ideal flow meter
Ideal Hydraulic Pressure Sensor	Simulate ideal pressure sensing device

Hydraulic Sources

Ideal Hydraulic Flow Rate Source	Simulate ideal source of hydraulic energy, characterized by flow rate
Ideal Hydraulic Pressure Source	Simulate ideal source of hydraulic energy, characterized by pressure

Hydraulic Utilities

Custom Hydraulic Fluid	Set working fluid properties by specifying parameter values
------------------------	---

Mechanical

Mechanical Sensors (p. 1-7)	Mechanical sensors and sources
Mechanical Sources (p. 1-7)	Mechanical sensors and sources
Mechanisms (p. 1-7)	Various simple mechanisms
Rotational Elements (p. 1-8)	Mechanical elements for rotational motion
Translational Elements (p. 1-8)	Mechanical elements for translational motion

Mechanical Sensors

Ideal Force Sensor	Simulate force sensor in mechanical translational systems
Ideal Rotational Motion Sensor	Simulate motion sensor in mechanical rotational systems
Ideal Torque Sensor	Simulate torque sensor in mechanical rotational systems
Ideal Translational Motion Sensor	Simulate motion sensor in mechanical translational systems

Mechanical Sources

Ideal Angular Velocity Source	Simulate ideal angular velocity source in mechanical rotational systems
Ideal Force Source	Simulate ideal source of mechanical energy that generates force proportional to the input signal
Ideal Torque Source	Simulate ideal source of mechanical energy that generates torque proportional to the input signal
Ideal Translational Velocity Source	Simulate ideal velocity source in mechanical translational systems

Mechanisms

Gear Box	Simulate gear boxes in mechanical systems
Lever	Simulate lever in mechanical systems
Wheel and Axle	Simulate wheel and axle mechanism in mechanical systems

Rotational Elements

Inertia	Simulate inertia in mechanical rotational systems
Mechanical Rotational Reference	Simulate reference for mechanical rotational ports
Rotational Damper	Simulate viscous damper in mechanical rotational systems
Rotational Friction	Simulate friction in contact between rotating bodies
Rotational Hard Stop	Simulate double-sided rotational hard stop
Rotational Spring	Simulate ideal spring in mechanical rotational systems

Translational Elements

Mass	Simulate mass in mechanical translational systems
Mechanical Translational Reference	Simulate reference for mechanical translational ports
Translational Damper	Simulate viscous damper in mechanical translational systems
Translational Friction	Simulate friction in contact between moving bodies
Translational Hard Stop	Simulate double-sided translational hard stop
Translational Spring	Simulate ideal spring in mechanical translational systems

Physical Signals

Functions (p. 1-9)	Perform math operations on physical signals
Linear Operators (p. 1-9)	Simulate continuous-time functions for physical signals
Lookup Tables (p. 1-10)	Perform one- and two-dimensional table lookup to generate physical signals
Nonlinear Operators (p. 1-10)	Simulate discontinuities, such as saturation or dead zone, for physical signals
Sources (p. 1-10)	Simulate physical signal sources

Functions

PS Add	Add two physical signal inputs
PS Divide	Compute simple division of two input physical signals
PS Gain	Multiply input physical signal by constant
PS Math Function	Apply mathematical function to input physical signal
PS Product	Multiply two physical signal inputs
PS Subtract	Compute simple subtraction of two input physical signals

Linear Operators

PS Integrator	Integrate physical signal
---------------	---------------------------

Lookup Tables

PS Lookup Table (1D)	Approximate one-dimensional function using specified lookup method
PS Lookup Table (2D)	Approximate two-dimensional function using specified lookup method

Nonlinear Operators

PS Abs	Output absolute value of input physical signal
PS Ceil	Output the smallest integer larger than or equal to input physical signal
PS Dead Zone	Provide region of zero output for physical signals
PS Fix	Round input physical signal toward zero
PS Floor	Output the largest integer smaller than or equal to input physical signal
PS Max	Output maximum of two input physical signals
PS Min	Output minimum of two input physical signals
PS Saturation	Limit range of physical signal
PS Sign	Output sign of input physical signal

Sources

PS Constant	Generate constant physical signal
-------------	-----------------------------------

Thermal

Thermal Elements (p. 1-11)	Thermal building blocks, such as thermal mass and various heat transfer blocks
Thermal Sensors (p. 1-11)	Temperature and heat flow sensors and sources
Thermal Sources (p. 1-11)	Temperature and heat flow sensors and sources

Thermal Elements

Conductive Heat Transfer	Simulate heat transfer by conduction
Convective Heat Transfer	Simulate heat transfer by convection
Radiative Heat Transfer	Simulate heat transfer by radiation
Thermal Mass	Simulate mass in thermal systems
Thermal Reference	Simulate reference for thermal ports

Thermal Sensors

Ideal Heat Flow Sensor	Simulate ideal heat flow meter
Ideal Temperature Sensor	Simulate ideal temperature sensor

Thermal Sources

Ideal Heat Flow Source	Simulate ideal source of thermal energy, characterized by heat flow
Ideal Temperature Source	Simulate ideal source of thermal energy, characterized by temperature

Utilities

Connection Port	Create Physical Modeling connector port for subsystem
PS-Simulink Converter	Convert physical signal into Simulink® output signal
Simulink-PS Converter	Convert Simulink input signal into physical signal
Solver Configuration	Represent Physical Networks environment and solver configuration
Two-Way Connection	Create two-way connector port for subsystem

Blocks — Alphabetical List

AC Current Source

Purpose Simulate 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:

$$I = I_0 \cdot \sin(\omega \cdot t + \varphi)$$

where

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

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

Dialog Box and Parameters

Block Parameters: AC Current Source

AC Current Source

The ideal AC current source maintains the sinusoidal current through it, independent of the voltage across its terminals. The output current is defined by $I = I_0 \cdot \sin(\omega \cdot t + \text{PHI})$, where I_0 is the peak amplitude, ω is the frequency in radians/s, and PHI is the phase shift in radians.

[View source for AC Current Source](#)

Parameters

Peak amplitude:	14.1421	A
Phase shift:	0	rad
Frequency:	60	Hz

OK Cancel Help Apply

Peak amplitude

Peak current amplitude. The default value is $10\sqrt{2}$, or 14.1421 A.

Phase shift

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

Frequency

Current frequency. The default value is 60 Hz.

Ports

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

See Also

AC Voltage Source

AC Voltage Source

Purpose Simulate 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:

$$V = V_0 \cdot \sin(\omega \cdot t + \varphi)$$

where

V Voltage

V_0 Peak amplitude

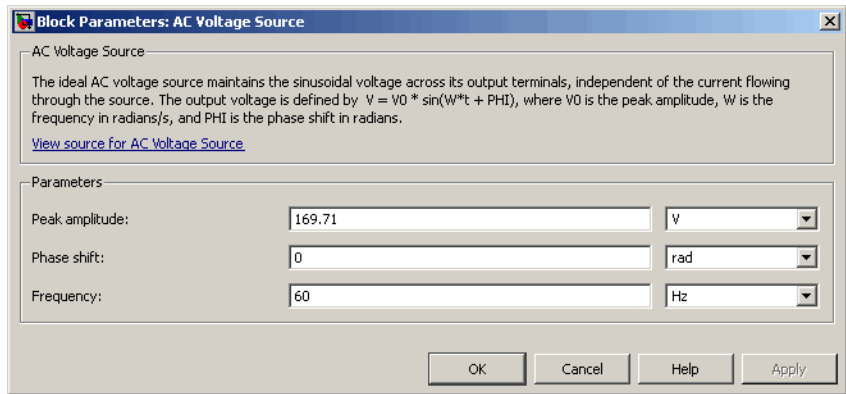
ω Frequency

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, $V(+)$ – $V(-)$.

Dialog Box and Parameters



Peak amplitude

Peak voltage amplitude. The default value is $120 \cdot \sqrt{2}$, or 169.71 V.

Phase shift

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

Frequency

Voltage frequency. 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

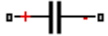
AC Current Source

Capacitor

Purpose Simulate 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 **Initial voltage** parameter sets the initial voltage across the capacitor.

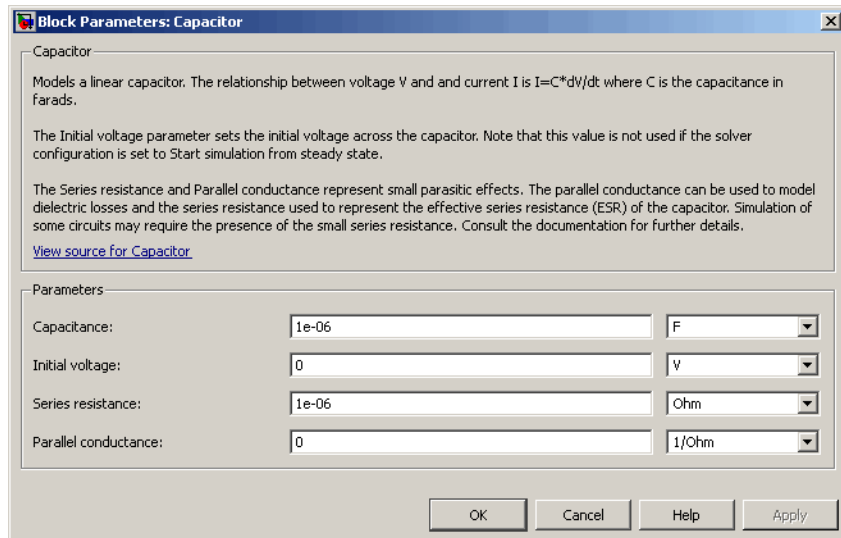
Note This value is not used if the solver configuration is set to **Start simulation from steady state**.

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” in the Simscape™ User’s Guide.

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, $V(+)-V(-)$.

Dialog Box and Parameters



Capacitance

Capacitance, in farads. The default value is $1\ \mu\text{F}$.

Initial voltage

Initial voltage across the capacitor. This parameter is not used if the solver configuration is set to **Start simulation from steady state**. The default value is 0.

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 $1\ \mu\Omega$.

Capacitor

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.

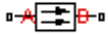
-

Electrical conserving port associated with the capacitor negative terminal.

Purpose Simulate 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

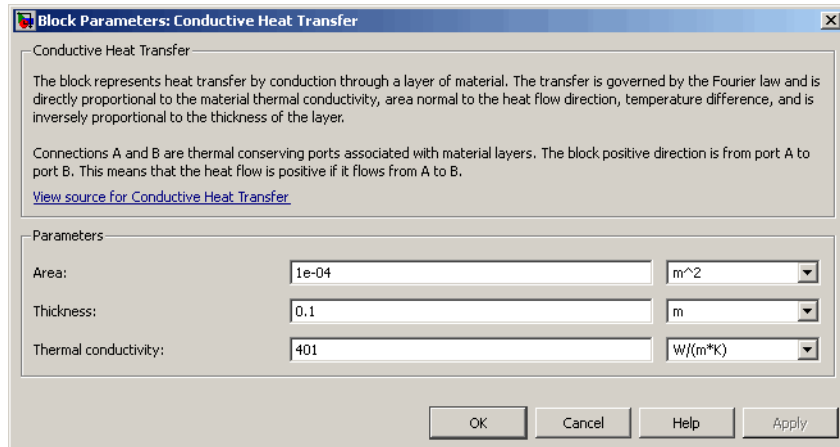
D Distance between layers

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.

Conductive Heat Transfer

Dialog Box and Parameters



Area

Area of heat transfer, normal to the heat flow direction. The default value is 0.0001 m².

Thickness

Thickness 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.

See Also

Convective Heat Transfer




Radiative Heat Transfer

Purpose Create Physical Modeling connector port for subsystem

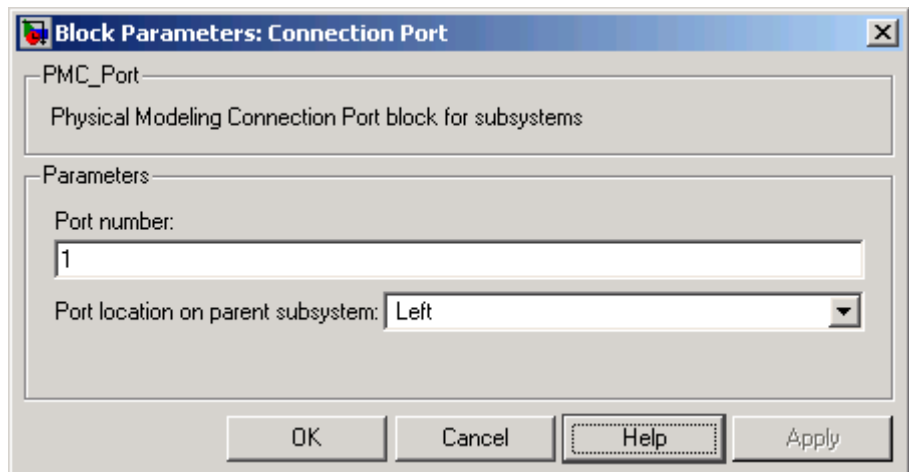
Library Utilities

Description The Connection Port block is used to export both the conserving and the physical signal connections to the outside boundary of a subsystem block, similar to the Inport and Outport blocks in Simulink models.



The ports on the subsystem boundary appear as the square Conserving ports , as triangular Physical Signal ports , or as two-way connector ports , depending on the type of port to which the Connection Port block is connected inside the subsystem. In other words, if a Connection Port block is connected to a Conserving port in a subsystem, it appears as a Conserving port on the outside boundary of the subsystem. If it is connected to a Physical Signal inport or outport inside the subsystem, it appears as a Physical Signal inport or outport, respectively, on the outside boundary of the subsystem. If it is connected to a two-way connector port of the Two-Way Connection block inside the subsystem, it appears as a two-way connector port on the outside boundary of the subsystem.

Dialog Box and Parameters



Connection Port

Port number

Labels the subsystem connector port created by this block. Each connector port on the boundary of a single subsystem requires a unique number as a label. The default value for the first port is 1.

Port location on parent subsystem

Choose here on which side of the parent subsystem boundary the port is placed. The choices are `Left` or `Right`. The default choice is `Left`.

See Also

In the Using Simulink documentation, see “Working with Block Masks”.

Purpose Simulate hydraulic orifice with constant cross-sectional area

Library Hydraulic Elements

Description



The Constant Area Orifice block models a sharp-edged constant-area orifice. The model distinguishes between the laminar and turbulent flow regimes by comparing the Reynolds number with its critical value. The flow rate through the orifice is proportional to the pressure differential across the orifice, and is determined according to the following equations:

$$q = \begin{cases} C_D \cdot A \sqrt{\frac{2}{\rho} |p| \cdot \text{sign}(p)} & \text{for } Re \geq Re_{cr} \\ 2C_{DL} \cdot A \frac{D_H}{v \cdot \rho} p & \text{for } Re < Re_{cr} \end{cases}$$

$$p = p_A - p_B$$

$$Re = \frac{q \cdot D_H}{A \cdot v}$$

$$C_{DL} = \left(\frac{C_D}{\sqrt{Re_{cr}}} \right)^2$$

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

where

q Flow rate

p Pressure differential

p_A, p_B Gauge pressures at the block terminals

Constant Area Orifice

C_D	Flow discharge coefficient
A	Orifice passage area
D_H	Orifice hydraulic diameter
ρ	Fluid density
ν	Fluid kinematic viscosity
Re	Reynolds number
Re_{cr}	Critical Reynolds number

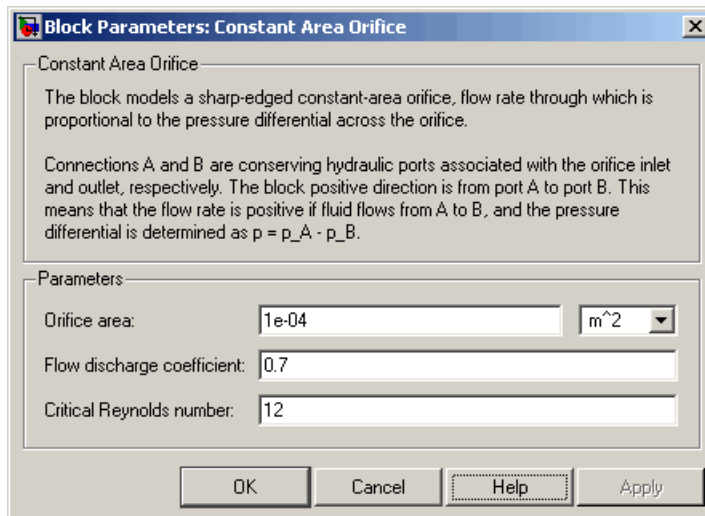
The block positive direction is from port A to port B. This means that the flow rate is positive if it flows from A to B, and the pressure differential is determined as $p = p_A - p_B$.

Basic Assumptions and Limitations

The model is based on the following assumptions:

- Fluid inertia is not taken into account.
- The transition between laminar and turbulent regimes is assumed to be sharp and taking place exactly at $Re=Re_{cr}$.

Dialog Box and 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.

Critical Reynolds number

The maximum Reynolds number for laminar flow. The transition from laminar to turbulent regime is supposed to take place when the Reynolds number reaches this value. The value of the parameter depends on orifice geometrical profile, and the recommendations on the parameter value can be found in hydraulic textbooks. The default value is 12, which corresponds to a round orifice in thin material with sharp edges.

Constant Area Orifice

Global Parameters

Fluid density

The parameter is determined by the type of working fluid selected for the system under design. Use the Custom Hydraulic Fluid block, or the Hydraulic Fluid block available with SimHydraulics® block libraries, to specify the fluid properties.

Fluid kinematic viscosity

The parameter is determined by the type of working fluid selected for the system under design. Use the Custom Hydraulic Fluid block, or the Hydraulic Fluid block available with SimHydraulics® block libraries, 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.

See Also

Variable Area Orifice

Purpose

Simulate hydraulic capacity of constant volume

Library

Hydraulic Elements

Description



The Constant Volume 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:

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

$$q = \frac{dV_f}{dt}$$

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. The mixture bulk modulus is determined as:

Constant Volume Chamber

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

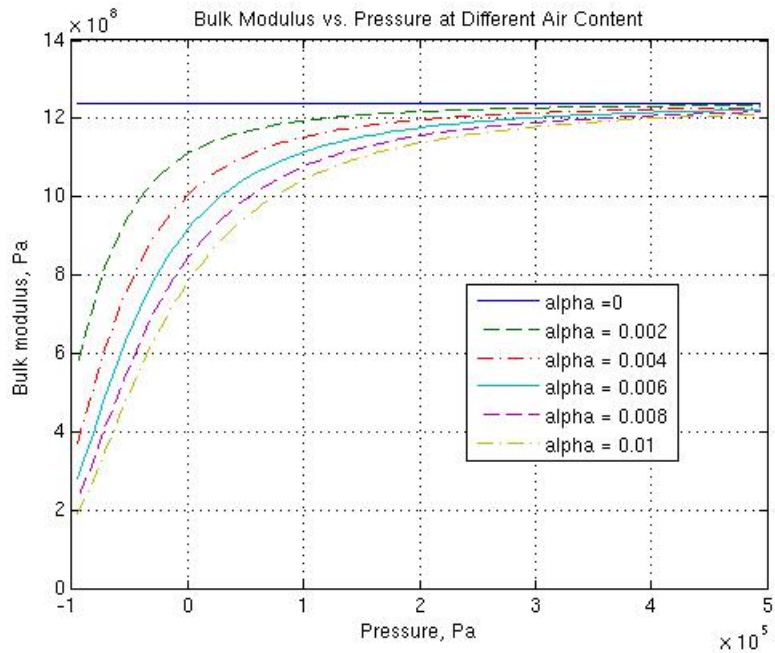
where

- E_l Pure liquid bulk modulus
- p_a 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 at $p \rightarrow p_a$, thus considerably slowing down further pressure change.

At high pressure, $p \gg p_a$, a small amount of nondissolved gas has practically no effect on the system behavior.

Constant Volume 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 it is known that cavitation is unlikely in the system under design, you can set the relative gas content in the fluid properties to zero, thus increasing the speed of computations. Use the Hydraulic Fluid or the Custom Hydraulic Fluid block to set the fluid properties.

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

Constant Volume Chamber

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

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

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 [1]):

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

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 [2]).

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.

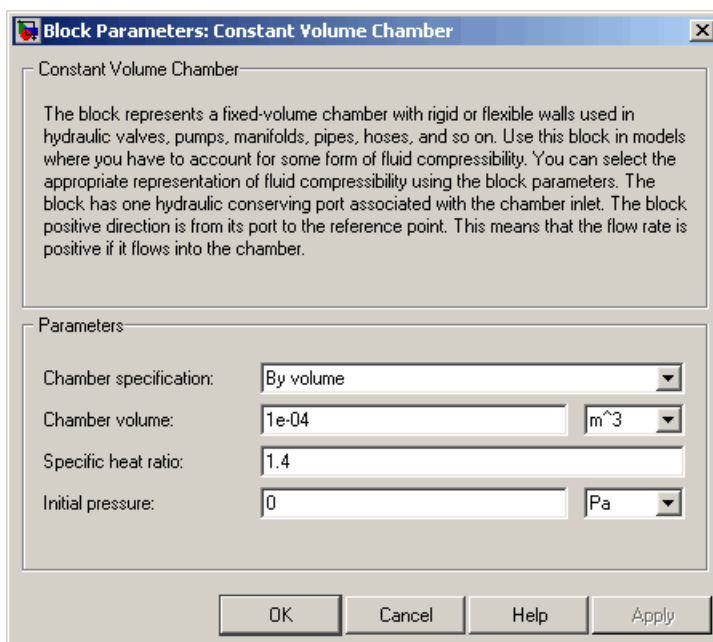
Basic Assumptions and Limitations

The model is based on the following assumptions:

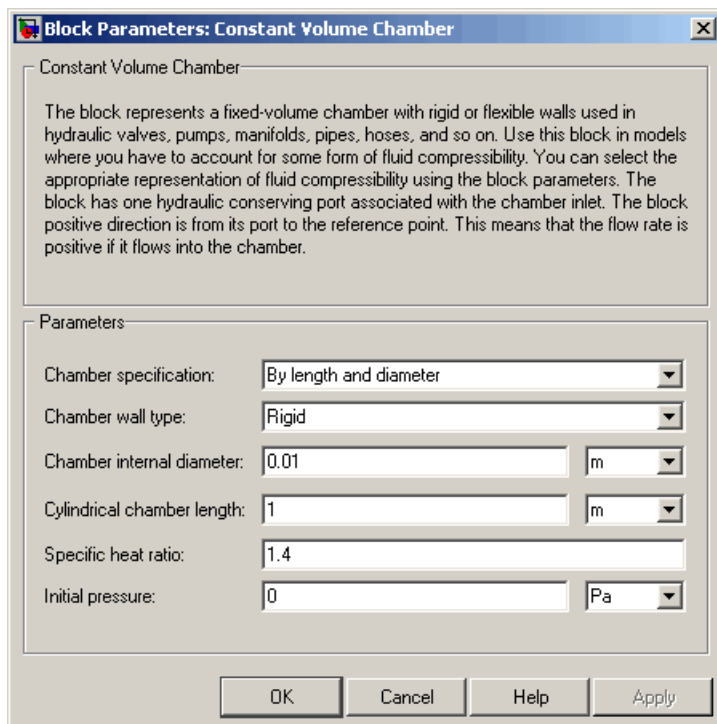
- No inertia associated with pipe walls is taken into account.
- Chamber with compliant walls is assumed to have a cylindrical shape. Chamber with rigid wall can have any shape.

Constant Volume Chamber

Dialog Box and Parameters



Constant Volume Chamber



Block Parameters: Constant Volume Chamber [X]

Constant Volume Chamber

The block represents a fixed-volume chamber with rigid or flexible walls 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. 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.

Parameters

Chamber specification: [v]

Chamber wall type: [v]

Chamber internal diameter: [v]

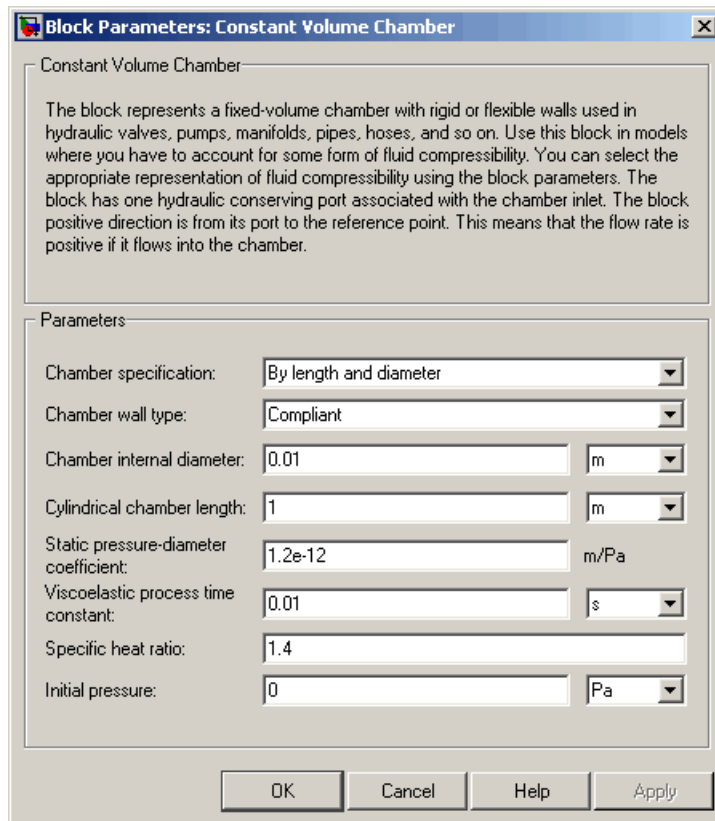
Cylindrical chamber length: [v]

Specific heat ratio:

Initial pressure: [v]

OK Cancel Help Apply

Constant Volume Chamber



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.

Constant Volume Chamber

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. For more information, see "Computing Initial Conditions". The default value is 0.

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

Fluid bulk modulus

The parameter is determined by the type of working fluid selected for the system under design. Use the Hydraulic Fluid block or the Custom Hydraulic Fluid block to specify the fluid properties.

Nondissolved gas ratio

Nondissolved gas relative content determined as a ratio of gas volume to the liquid volume. The parameter is determined by the type of working fluid selected for the system under design. 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] Meritt, H.E., *Hydraulic Control Systems*, John Wiley & Sons, New York, 1967

[2] Holcke, Jan, *Frequency Response of Hydraulic Hoses*, RIT, FTH, Stockholm, 2002

See Also

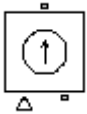
Variable Volume Chamber

Controlled Current Source

Purpose Simulate ideal current source driven by input signal

Library Electrical Sources

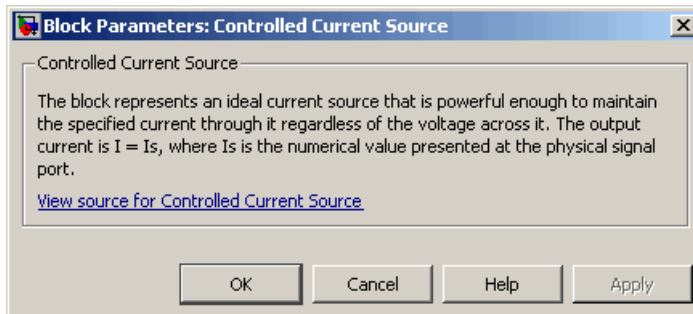
Description The Controlled Current Source block represents an ideal current source that is powerful enough to maintain the specified current through it regardless of the voltage across the source.



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.

Dialog Box and Parameters



The block has no parameters.

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

See Also Controlled Voltage Source

Controlled Voltage Source

Purpose Simulate ideal voltage source driven by input signal

Library Electrical Sources

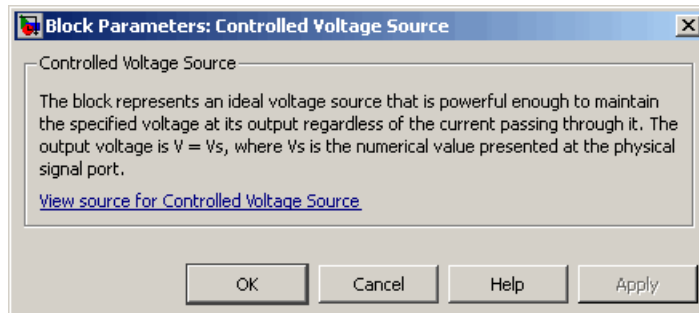
Description



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 current is $V = V_s$, where V_s is the numerical value presented at the physical signal port.

Dialog Box and Parameters



The block has no parameters.

Ports

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

See Also

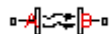
Controlled Current Source

Convective Heat Transfer

Purpose Simulate 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

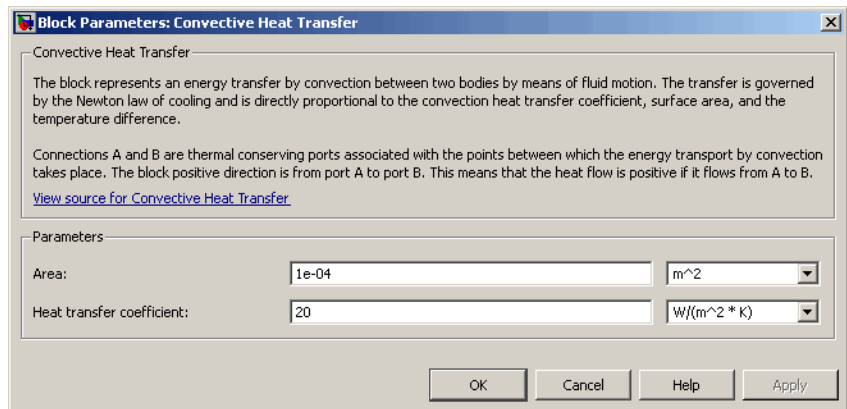
k Convection heat transfer coefficient

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.

Dialog Box and Parameters



Area

Surface area of heat transfer. The default value is 0.0001 m^2 .

Heat transfer coefficient

Convection heat transfer coefficient. The default value is $20 \text{ W/m}^2/\text{K}$.

Ports

The block has the following ports:

- A
Thermal conserving port associated with body A.
- B
Thermal conserving port associated with body B.

See Also

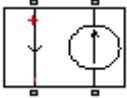
Conductive Heat Transfer
Radiative Heat Transfer

Current-Controlled Current Source

Purpose Simulate 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

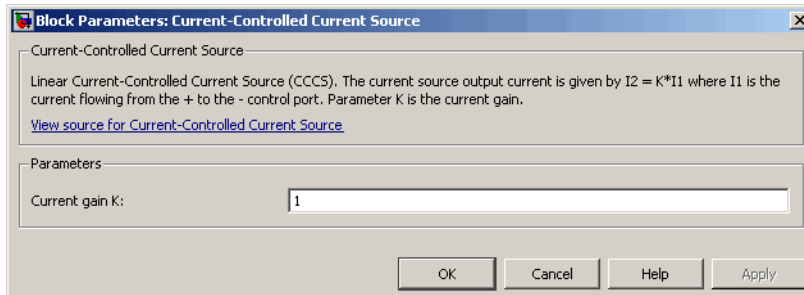
$I2$ Output current

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.

Dialog Box and Parameters



Current-Controlled Current Source

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.

See Also

Current-Controlled Voltage Source

Voltage-Controlled Current Source

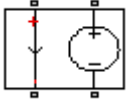
Voltage-Controlled Voltage Source

Current-Controlled Voltage Source

Purpose Simulate 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

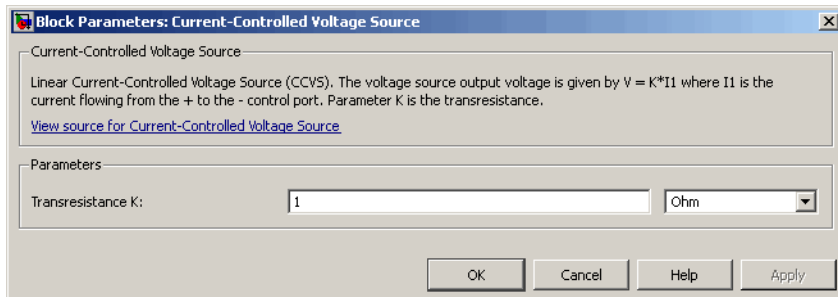
V Voltage

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.

Dialog Box and 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 1 Ω .

Current-Controlled Voltage Source

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.

See Also

Current-Controlled Current Source

Voltage-Controlled Current Source

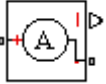
Voltage-Controlled Voltage Source

Current Sensor

Purpose Simulate current sensor in electrical systems

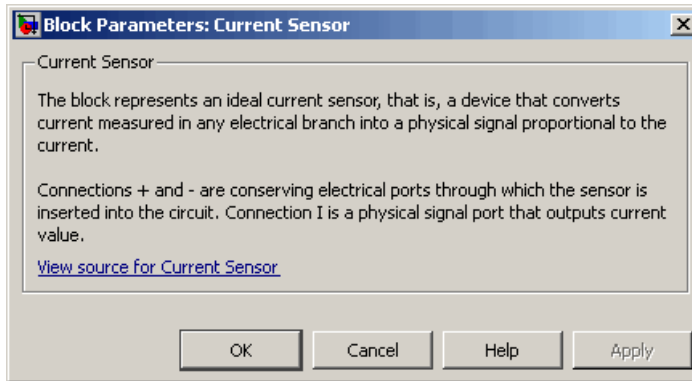
Library Electrical Sensors

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.

Dialog Box and Parameters



The block has no parameters.

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.

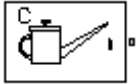
See Also Voltage Sensor

Custom Hydraulic Fluid

Purpose Set working fluid properties 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.

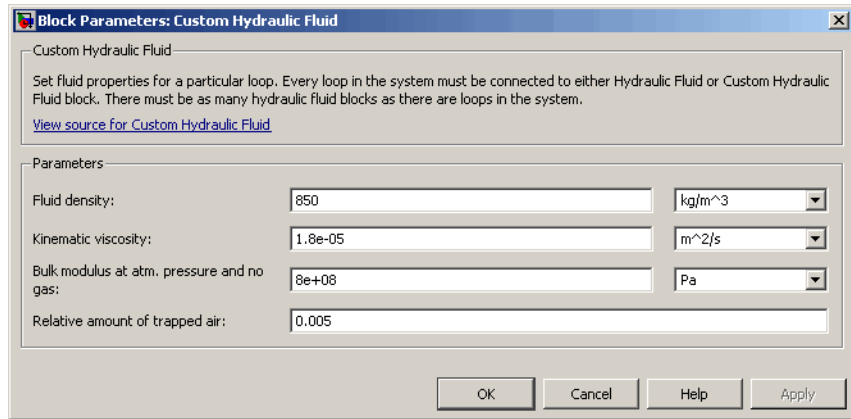
The Custom Hydraulic Fluid block lets you specify the fluid properties, such as kinematic viscosity, density, bulk modulus, and relative amount of entrapped air, as block parameters.

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 exactly one Custom Hydraulic Fluid block or Hydraulic Fluid block, available with SimHydraulics libraries, to be connected to it. Therefore, there must be as many Custom Hydraulic Fluid blocks (or Hydraulic Fluid blocks) as there are loops in the system.

Note 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 Skydrol LD-4 at 60°C and with a 0.005 ratio of entrapped air. See the Hydraulic Fluid block reference page for more information.

Dialog Box and Parameters



Fluid density

Density of the working fluid. The default value is 850 kg/m³.

Kinematic viscosity

Kinematic viscosity of the working fluid. The default value is 1.8e-5 m²/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 8e8 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. The default value is 0.005.

Ports

The block has one hydraulic conserving port.

See Also

Hydraulic Fluid

DC Current Source

Purpose Simulate ideal constant current source

Library Electrical Sources

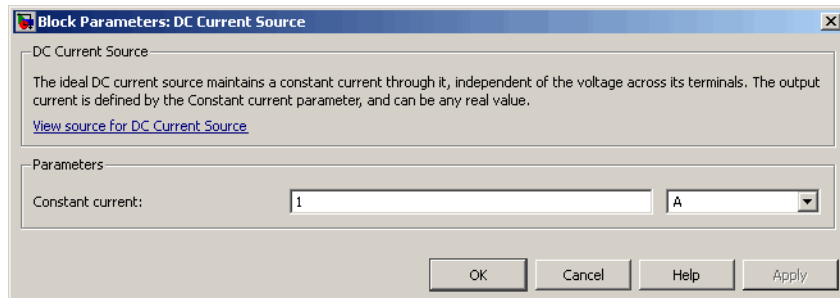
Description The DC Current Source block represents an ideal current source that is powerful enough to maintain specified current through it regardless of the voltage across the source.



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.

Dialog Box and 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.

See Also DC Voltage Source

Purpose Simulate ideal constant voltage source

Library Electrical Sources

Description

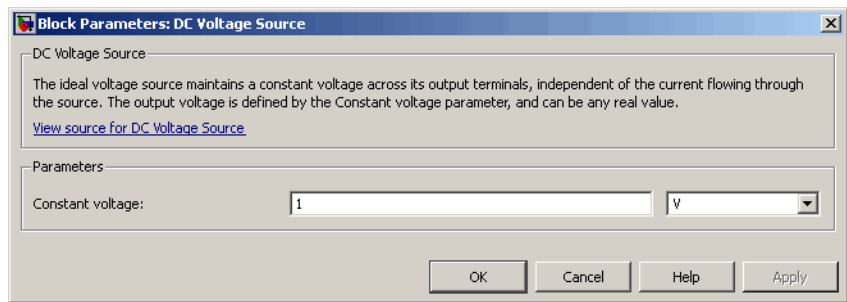


The DC Voltage Source block represents an ideal voltage source that is powerful enough to maintain specified voltage at its output regardless of the current flowing through the source.

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, $V(+)-V(-)$.

Dialog Box and 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.

DC Voltage Source

-
Electrical conserving port associated with the source negative terminal.

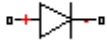
See Also DC Current Source

Purpose

Simulate 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:

$$V = Vf(1 - R_{on} \cdot G_{off})$$

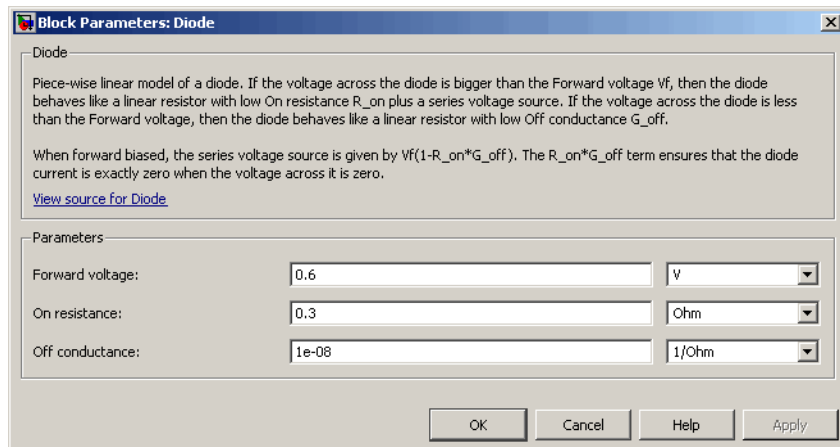
where

V	Voltage
Vf	Forward voltage
R_{on}	On resistance
G_{off}	Off conductance

The $R_{on} \cdot G_{off}$ term ensures that the diode current is exactly zero when the voltage across it is zero.

Diode

Dialog Box and 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 $1e-8$ $1/\Omega$.

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.

Purpose Simulate connection to electrical ground

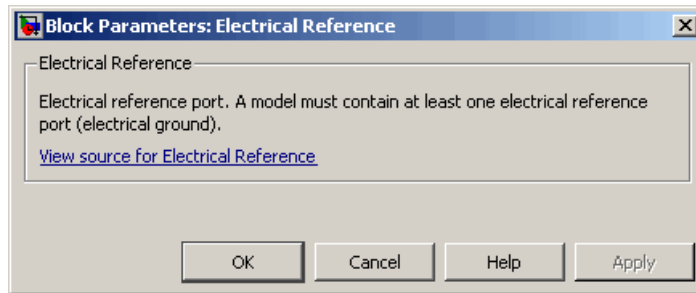
Library Electrical Elements

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.

Dialog Box and Parameters



The Electrical Reference block has no parameters.

Ports The block has one electrical conserving port.

See Also

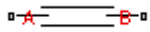
- Hydraulic Reference
- Mechanical Rotational Reference
- Mechanical Translational Reference
- Thermal Reference

Fluid Inertia

Purpose Simulate 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

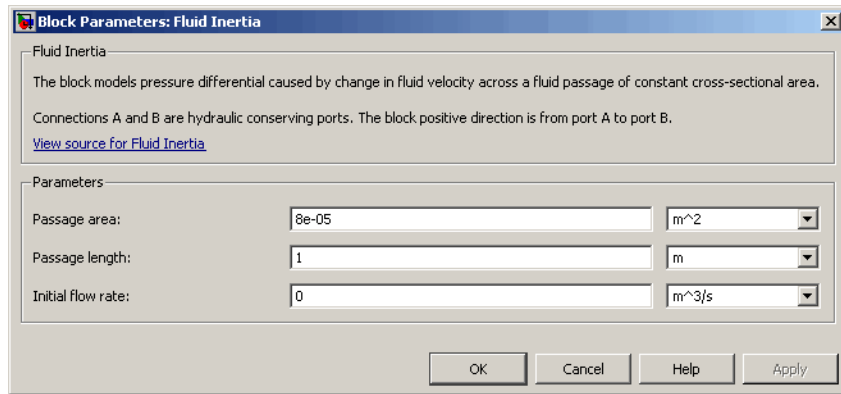
Use this block in various pipe or channel models that require fluid inertia to be accounted for.

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.

Assumptions and Limitations The model is based on the following assumptions:

- Fluid density remains constant.

Dialog Box and 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 "Computing Initial Conditions". The default value is 0.

Global Parameters

Fluid density

The parameter is determined by the type of working fluid selected for the system under design. 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.

Gear Box

Purpose Simulate gear boxes in mechanical systems

Library 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:

$$\omega_1 = N \cdot \omega_2$$

$$T_2 = N \cdot T_1$$

$$P_1 = \omega_1 \cdot T_1$$

$$P_2 = -\omega_2 \cdot T_2$$

where

ω_1 Input shaft angular velocity

ω_2 Output shaft angular velocity

N Gear ratio

T_1 Torque on the input shaft

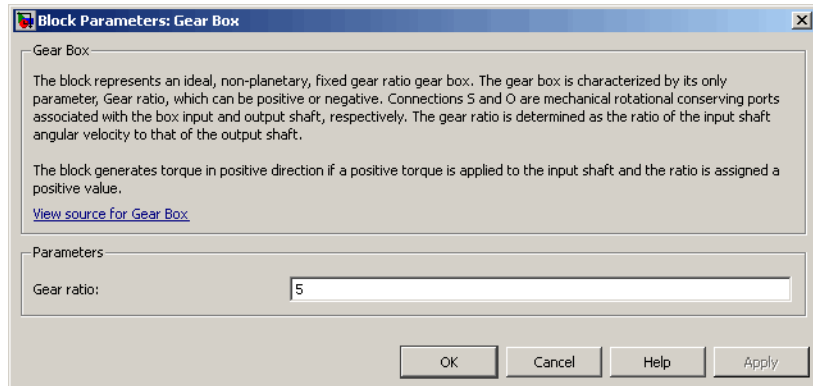
T_2 Torque on the output shaft

P_1 Power on the input shaft

P_2 Power on the output shaft. Notice the minus sign in computing P_2 . 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.

Dialog Box and Parameters



Gear ratio

The ratio of the input shaft angular velocity to that of the output shaft. You can specify both positive and negative values. The default value is 5.

Ports

The block has the following ports:

S

Mechanical rotational conserving port associated with input shaft.

O

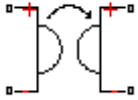
Mechanical rotational conserving port associated with the output shaft.

Gyrator

Purpose Simulate 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

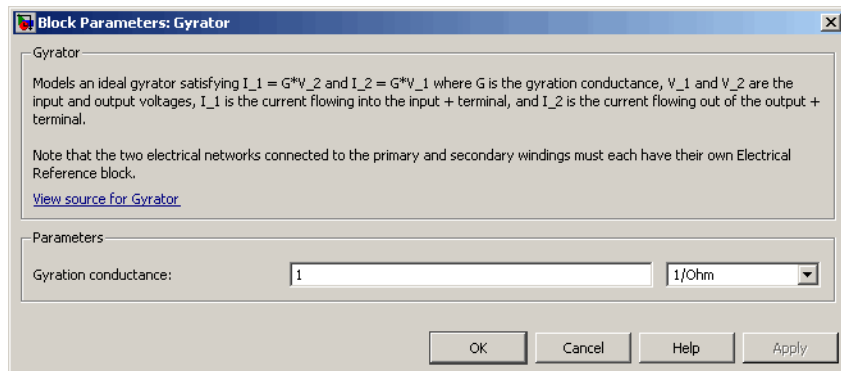
I1 Current flowing into the input + terminal

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.

Dialog Box and 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.

Hydraulic Reference

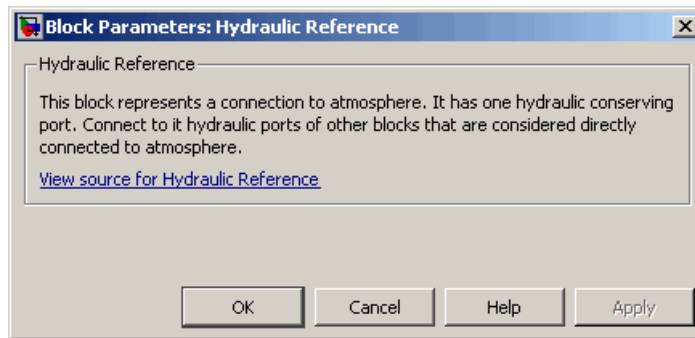
Purpose Simulate connection to atmospheric pressure

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.



Dialog Box and Parameters



The Hydraulic Reference block has no parameters.

Ports The block has one hydraulic conserving port.

See Also

- Electrical Reference
- Mechanical Rotational Reference
- Mechanical Translational Reference
- Thermal Reference

Purpose Simulate ideal angular velocity source in mechanical rotational systems

Library Mechanical Sources

Description

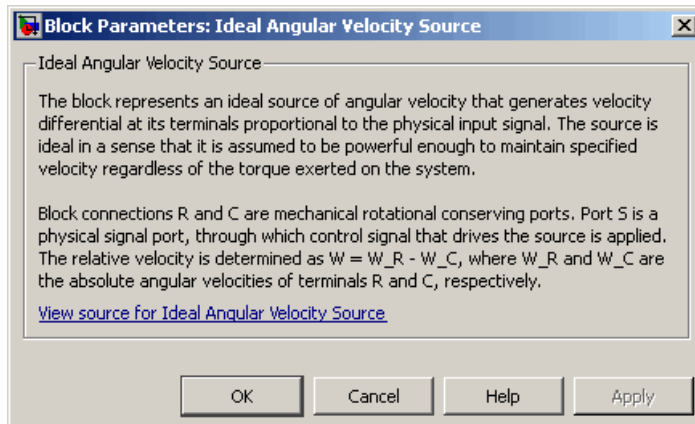


The Ideal Angular Velocity Source block represents an ideal source of angular velocity that generates velocity differential at its terminals proportional to the input physical signal. The source is ideal in a sense that it is assumed to be powerful enough to maintain specified velocity regardless of 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.

Dialog Box and Parameters



Ideal Angular Velocity Source

The block has no parameters.

Ports

The block has the following ports:

R

Mechanical rotational conserving port.

C

Mechanical rotational conserving port associated with the source reference point (case).

S

Physical signal input port, through which the control signal that drives the source is applied.

See Also

Ideal Force Source

Ideal Torque Source

Ideal Translational Velocity Source

Purpose

Simulate force sensor in mechanical translational systems

Library

Mechanical Sensors

Description

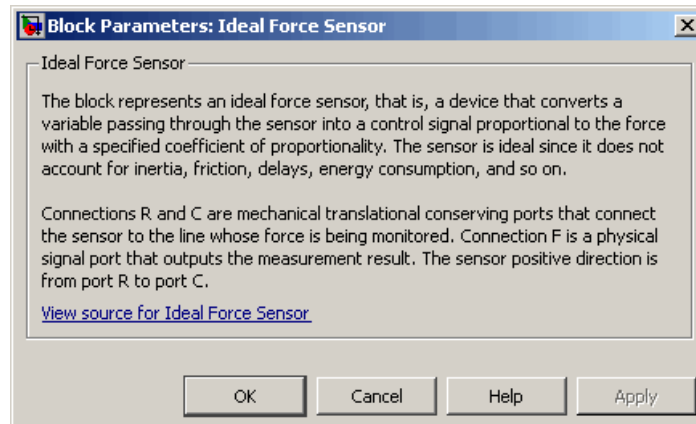


The Ideal Force Sensor block represents a device that converts a variable passing through the sensor into a control signal proportional to the force. The sensor is ideal since it does not account for inertia, friction, delays, energy consumption, and so on.

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.

Dialog Box and Parameters



The block has no parameters.

Ports

The block has the following ports:

Ideal Force Sensor

- R
Mechanical translational conserving port associated with the sensor positive probe.
- C
Mechanical translational conserving port associated with the sensor negative (reference) probe.
- F
Physical signal output port for force.

See Also

- Ideal Rotational Motion Sensor
- Ideal Torque Sensor
- Ideal Translational Motion Sensor

Purpose Simulate ideal source of mechanical energy that generates force proportional to the input signal

Library Mechanical Sources

Description

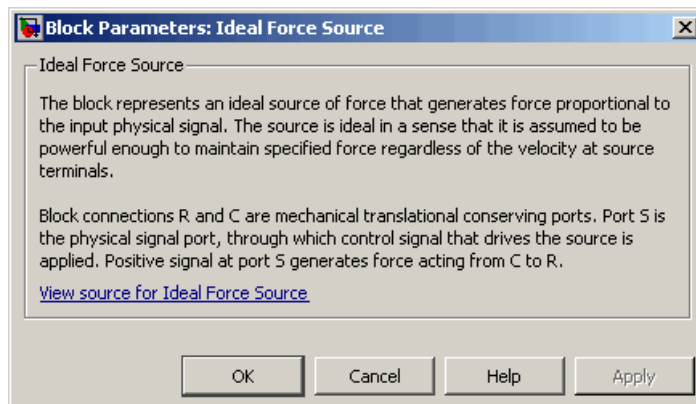


The Ideal Force Source block represents an ideal source of mechanical energy that generates force proportional to the input physical signal. The source is ideal in a sense that it is assumed to be powerful enough to maintain specified force at its output regardless of the velocity at source terminals.

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 $v = v_R - v_C$, 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.

Dialog Box and Parameters



Ideal Force Source

The block has no parameters.

Ports

The block has the following ports:

R

Mechanical translational conserving port.

C

Mechanical translational conserving port associated with the source reference point (case).

S

Physical signal input port, through which the control signal that drives the source is applied.

See Also

Ideal Angular Velocity Source

Ideal Torque Source

Ideal Translational Velocity Source

Purpose Simulate ideal heat flow meter

Library Thermal Sensors

Description

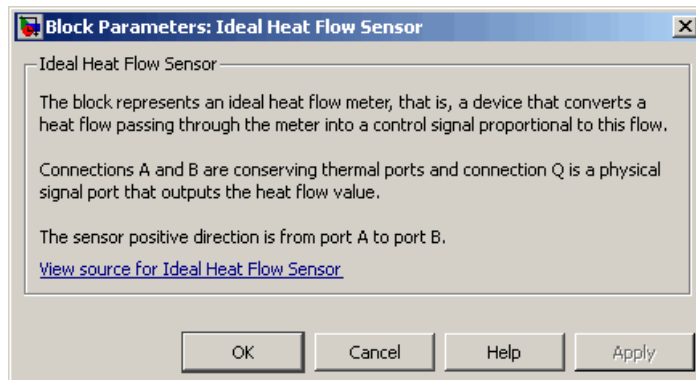


The Ideal Heat Flow Sensor block represents an ideal heat flow meter, that is, a device that converts a heat flow passing through the meter into a control signal proportional to this flow. The meter must be connected in series with the component whose heat flow is being monitored.

Connections A and B are thermal conserving ports. Port Q is a physical signal port that outputs the heat flow value.

The block positive direction is from port A to port B.

Dialog Box and Parameters



The block has no parameters.

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.

Ideal Heat Flow Sensor

Q

Physical signal output port for heat flow.

See Also

Ideal Heat Flow Source

Ideal Temperature Sensor

Ideal Temperature Source

Purpose Simulate ideal source of thermal energy, characterized by heat flow

Library Thermal Sources

Description

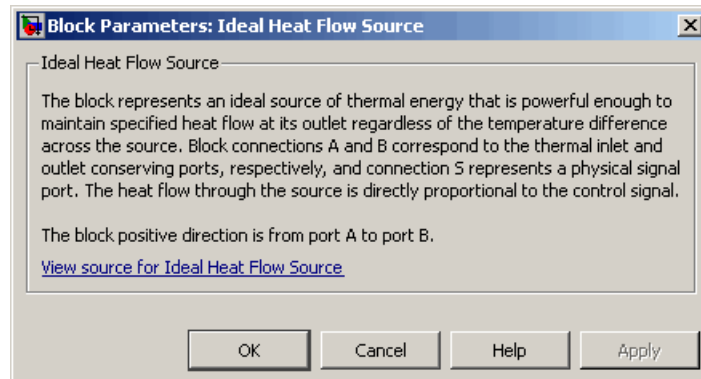


The Ideal Heat Flow Source block represents an ideal source of thermal energy that is powerful enough to maintain specified heat flow at its outlet regardless of the temperature difference across the source.

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.

Dialog Box and Parameters



The block has no parameters.

Ports

The block has the following ports:

- A Thermal conserving port associated with the source inlet.

Ideal Heat Flow Source

- B Thermal conserving port associated with the source outlet.
- S Physical signal input port, through which the control signal that drives the source is applied.

See Also

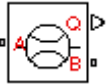
Ideal Heat Flow Sensor
Ideal Temperature Sensor
Ideal Temperature Source

Ideal Hydraulic Flow Rate Sensor

Purpose Simulate ideal flow meter

Library Hydraulic Sensors

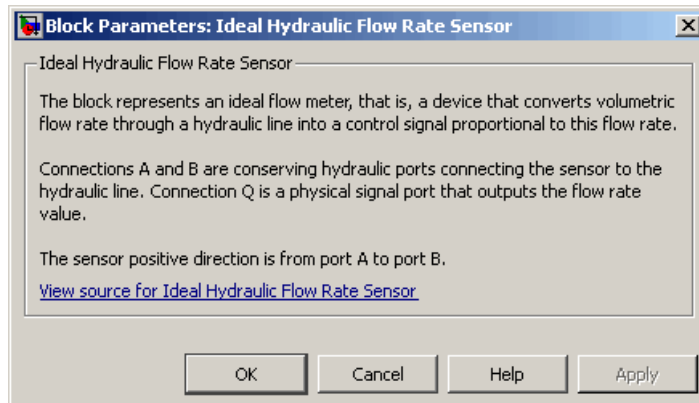
Description



The Ideal 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. 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 Q is a physical signal port that outputs the flow rate value. The sensor positive direction is from A to B. This means that the flow rate is positive if it flows from A to B.

Dialog Box and Parameters



The block has no parameters.

Ports

The block has the following ports:

A

Hydraulic conserving port associated with the sensor positive probe.

Ideal Hydraulic Flow Rate Sensor

- B Hydraulic conserving port associated with the sensor negative (reference) probe.
- Q Physical signal port that outputs the flow rate value.

See Also

- Ideal Hydraulic Flow Rate Source
- Ideal Hydraulic Pressure Sensor

Ideal Hydraulic Flow Rate Source

Purpose Simulate ideal source of hydraulic energy, characterized by flow rate

Library Hydraulic Sources

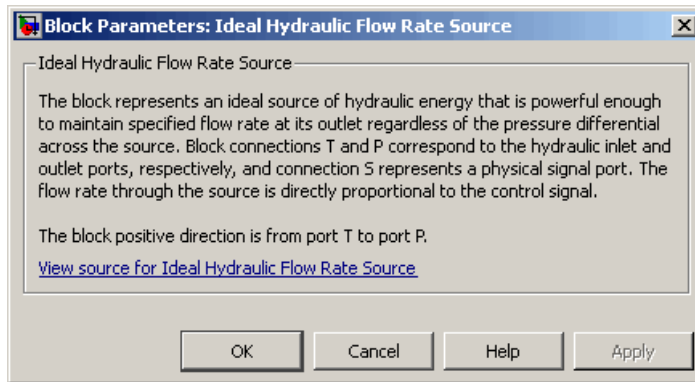
Description



The Ideal 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 $p = p_T - p_P$ 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.

Dialog Box and Parameters



The block has no parameters.

Ports

The block has the following ports:

Ideal Hydraulic Flow Rate Source

- T Hydraulic conserving port associated with the source inlet.
- P Hydraulic conserving port associated with the source outlet.
- S Control signal port.

See Also

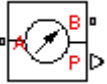
Ideal Hydraulic Flow Rate Sensor
Ideal Hydraulic Pressure Source

Ideal Hydraulic Pressure Sensor

Purpose Simulate ideal pressure sensing device

Library Hydraulic Sensors

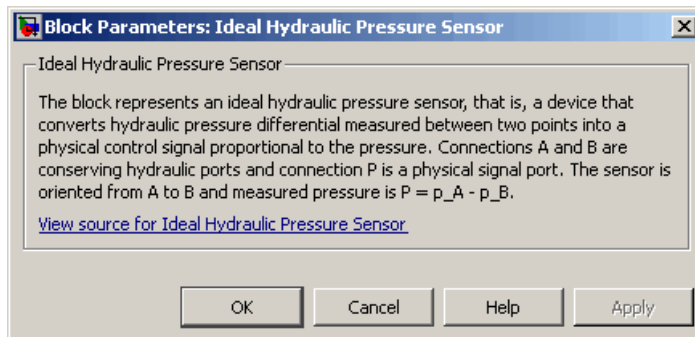
Description



The Ideal 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 flow rate is positive if it flows from A to B.

Dialog Box and Parameters



The block has no parameters.

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.

Ideal Hydraulic Pressure Sensor

P

Physical signal port that outputs the pressure value.

See Also

Ideal Hydraulic Flow Rate Sensor

Ideal Hydraulic Pressure Source

Ideal Hydraulic Pressure Source

Purpose Simulate ideal source of hydraulic energy, characterized by pressure

Library Hydraulic Sources

Description



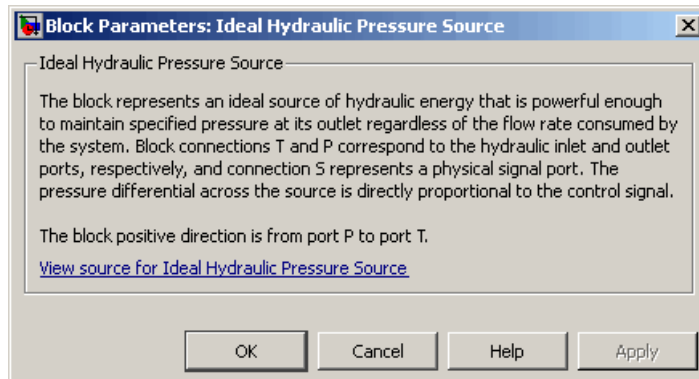
The Ideal Hydraulic Pressure Source block represents an ideal source of hydraulic energy that is powerful enough to maintain specified pressure at its outlet regardless of the flow rate consumed by the system. Block connections T and P correspond to the hydraulic inlet and outlet ports, respectively, 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.

Dialog Box and Parameters



The block has no parameters.

Ideal Hydraulic Pressure Source

Ports

The block has the following ports:

- P
Hydraulic conserving port associated with the source inlet.
- T
Hydraulic conserving port associated with the source outlet.
- S
Control signal port.

See Also

Ideal Hydraulic Flow Rate Source
Ideal Hydraulic Pressure Sensor

Ideal Rotational Motion Sensor

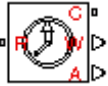
Purpose

Simulate motion sensor in mechanical rotational systems

Library

Mechanical Sensors

Description



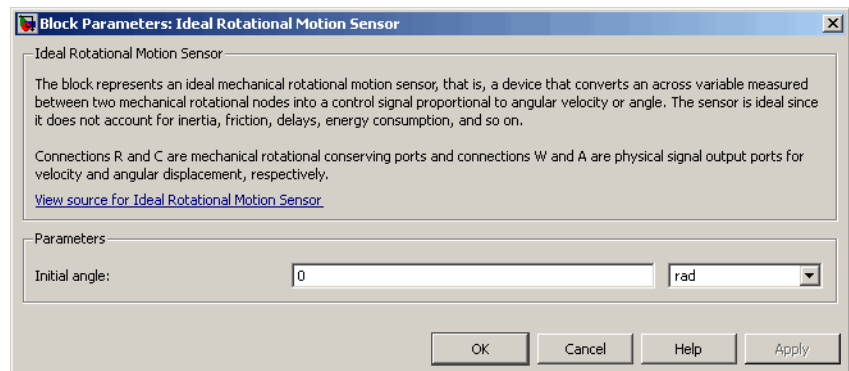
The Ideal Rotational Motion Sensor block represents an ideal mechanical rotational motion sensor, that is, a device that converts an across variable measured between two mechanical rotational nodes into a control signal proportional to angular velocity or angle. You can specify the initial angular position (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.

Dialog Box and Parameters



Initial angle

Sensor initial angle, or offset (rad). The default value is 0.

Ideal Rotational Motion Sensor

Ports

The block has the following ports:

- R
Mechanical rotational conserving port associated with the sensor positive probe.
- C
Mechanical rotational conserving port associated with the sensor negative (reference) probe.
- W
Physical signal output port for angular velocity.
- A
Physical signal output port for angular displacement.

See Also

- Ideal Force Sensor
- Ideal Torque Sensor
- Ideal Translational Motion Sensor

Purpose Simulate ideal temperature sensor

Library Thermal Sensors

Description

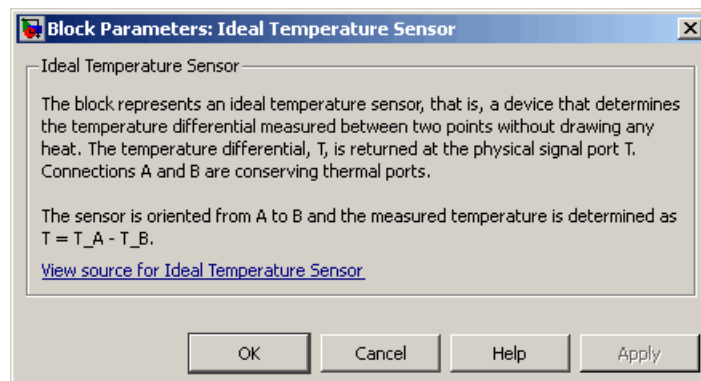


The Ideal Temperature Sensor block represents an ideal temperature sensor, that is, a device that determines the temperature differential measured between two points without drawing any heat.

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 $T = T_A - T_B$.

Dialog Box and Parameters



The block has no parameters.

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.

Ideal Temperature Sensor

T

Physical signal output port for temperature.

See Also

Ideal Heat Flow Sensor

Ideal Heat Flow Source

Ideal Temperature Source

Purpose Simulate ideal source of thermal energy, characterized by temperature

Library Thermal Sources

Description

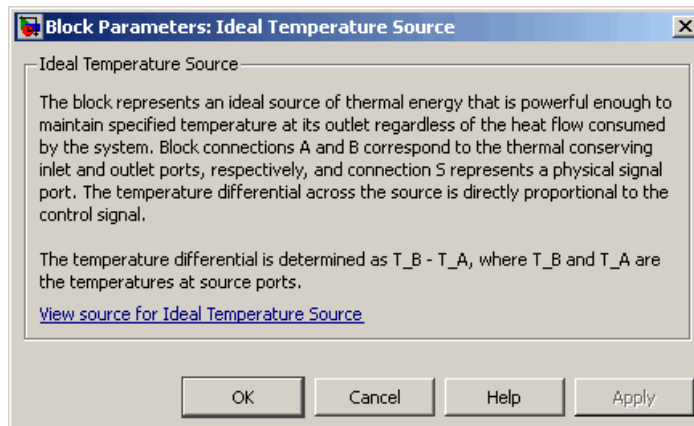


The Ideal Temperature Source block represents an ideal source of thermal energy that is powerful enough to maintain specified temperature at its outlet regardless of the heat flow consumed by the system.

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.

Dialog Box and Parameters



The block has no parameters.

Ports

The block has the following ports:

Ideal Temperature Source

- A Thermal conserving port associated with the source inlet.
- B Thermal conserving port associated with the source outlet.
- S Physical signal input port, through which the control signal that drives the source is applied.

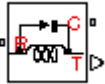
See Also

Ideal Heat Flow Sensor
Ideal Heat Flow Source
Ideal Temperature Sensor

Purpose Simulate torque sensor in mechanical rotational systems

Library Mechanical Sensors

Description

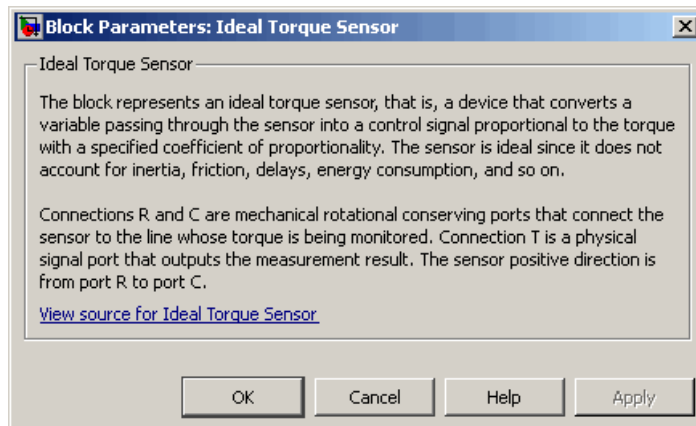


The Ideal Torque Sensor block represents a device that converts a variable passing through the sensor into a control signal proportional to the torque. The sensor is ideal since it does not account for inertia, friction, delays, energy consumption, and so on.

Connections R and C are mechanical rotational conserving ports that connect the block to the 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.

Dialog Box and Parameters



The block has no parameters.

Ports The block has the following ports:

R Mechanical rotational conserving port associated with the sensor positive probe.

Ideal Torque Sensor

C Mechanical rotational conserving port associated with the sensor negative (reference) probe.

T Physical signal output port for torque.

See Also

Ideal Force Sensor

Ideal Rotational Motion Sensor

Ideal Translational Motion Sensor

Purpose Simulate ideal source of mechanical energy that generates torque proportional to the input signal

Library Mechanical Sources

Description

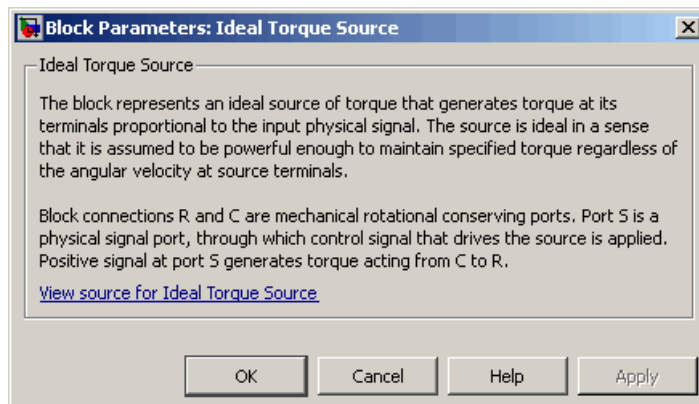


The Ideal Torque Source block represents an ideal source of mechanical energy that generates torque proportional to the input physical signal. The source is ideal in a sense that it is assumed to be powerful enough to maintain specified torque regardless of the angular velocity at source terminals.

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.

Dialog Box and Parameters



Ideal Torque Source

The block has no parameters.

Ports

The block has the following ports:

R

Mechanical rotational conserving port.

C

Mechanical rotational conserving port associated with the source reference point (case).

S

Physical signal input port, through which the control signal that drives the source is applied.

See Also

Ideal Angular Velocity Source

Ideal Force Source

Ideal Translational Velocity Source

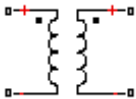
Purpose

Simulate 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

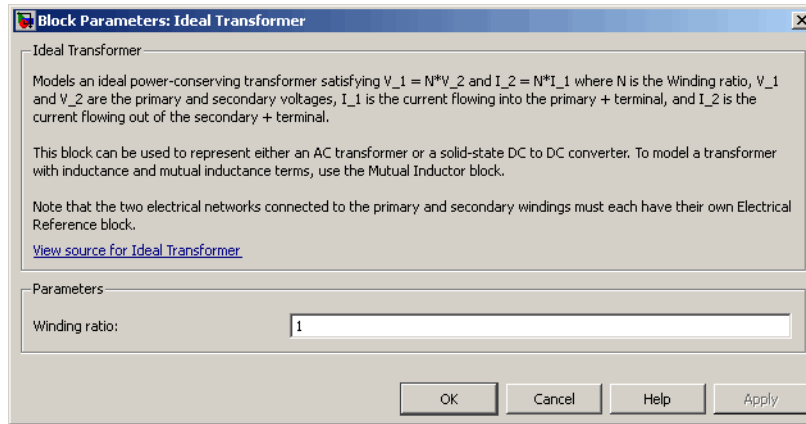
N Winding ratio

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.

Ideal Transformer

Dialog Box and 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.

See Also

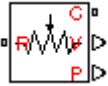
Mutual Inductor

Ideal Translational Motion Sensor

Purpose Simulate motion sensor in mechanical translational systems

Library Mechanical Sensors

Description



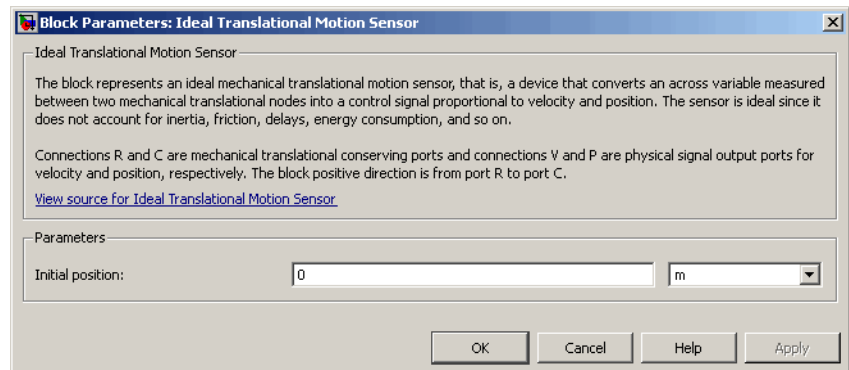
The Ideal Translational Motion Sensor block represents a device that converts an across variable measured between two mechanical translational nodes into a control signal proportional to velocity or position. You can specify the initial position (offset) as a block parameter.

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.

Dialog Box and Parameters



Initial position

Sensor initial position, or offset (m). The default value is 0.

Ideal Translational Motion Sensor

Ports

The block has the following ports:

R

Mechanical translational conserving port associated with the sensor positive probe.

C

Mechanical translational conserving port associated with the sensor negative (reference) probe.

V

Physical signal output port for velocity.

P

Physical signal output port for position.

See Also

Ideal Force Sensor

Ideal Rotational Motion Sensor

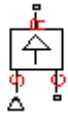
Ideal Torque Sensor

Ideal Translational Velocity Source

Purpose Simulate ideal velocity source in mechanical translational systems

Library Mechanical Sources

Description

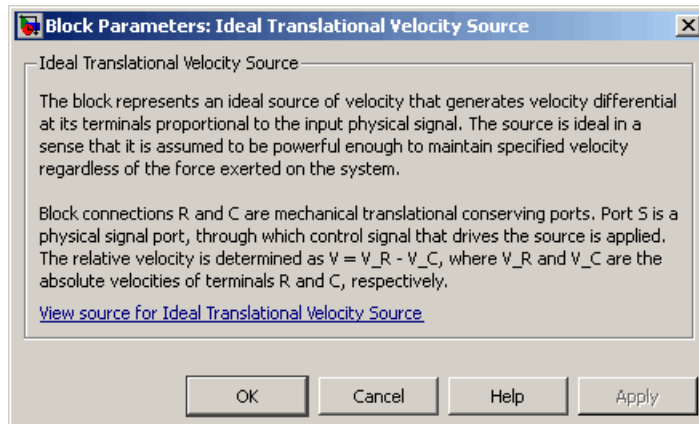


The Ideal Translational Velocity Source block represents an ideal source of velocity that generates velocity differential at its terminals proportional to the input physical signal. The source is ideal in a sense that it is assumed to be powerful enough to maintain specified velocity regardless of the 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.

Dialog Box and Parameters



Ideal Translational Velocity Source

The block has no parameters.

Ports

The block has the following ports:

R

Mechanical translational conserving port.

C

Mechanical translational conserving port associated with the source reference point (case).

S

Physical signal input port, through which the control signal that drives the source is applied.

See Also

Ideal Angular Velocity Source

Ideal Force Source

Ideal Torque Source

Purpose Simulate 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 **Initial current** parameter sets the initial current through the inductor.

Note This value is not used if the solver configuration is set to **Start simulation from steady state**.

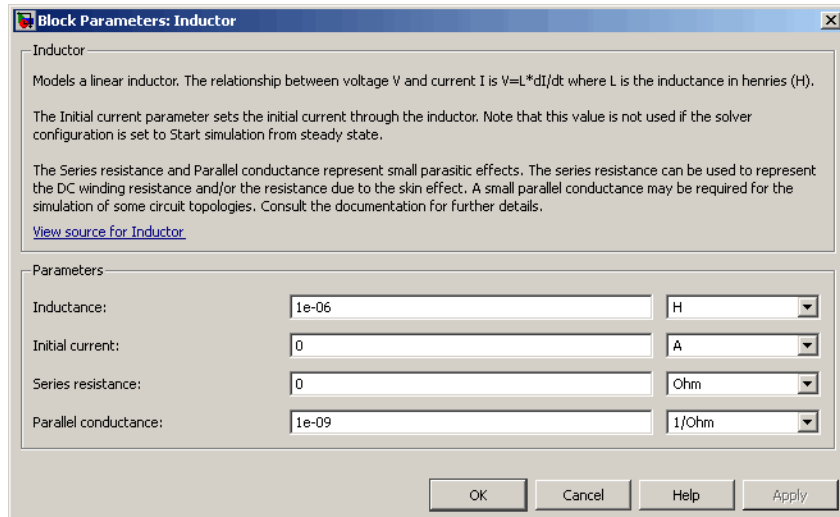
The **Series resistance** and **Parallel conductance** parameters represent small parasitic effects. The series resistance can be used to represent the DC winding resistance or the resistance due to the skin effect. Simulation of some circuits may require the presence of a small parallel conductance. For more information, see “Modeling Best Practices” in the Simscape User’s Guide.

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

Inductor

across the inductor is equal to the difference between the voltage at the positive and the negative terminal, $V(+)$ – $V(-)$.

Dialog Box and Parameters



Inductance

Inductance, in henries. The default value is 1 μ H.

Initial current

Initial current through the inductor. This parameter is not used if the solver configuration is set to **Start simulation from steady state**. The default value is 0.

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.

Inertia

Purpose

Simulate inertia in mechanical rotational systems

Library

Mechanical Rotational Elements

Description

The Inertia block represents an ideal mechanical rotational inertia, described with the following equation:



$$T = J \frac{d\omega}{dt}$$

where

T	Inertia torque
J	Inertia
ω	Angular velocity
t	Time

The block has one mechanical rotational 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.

Dialog Box and Parameters

Block Parameters: Inertia

Inertia

The block represents an ideal mechanical rotational inertia.

The block has one mechanical rotational conserving port. The block positive direction is from its port to the reference point. This means that the inertia torque is positive if the inertia is accelerated in the positive direction.

[View source for Inertia](#)

Parameters

Inertia:

Initial velocity:

OK Cancel Help Apply

Inertia

Inertia. The default value is $0.001 \text{ kg}\cdot\text{m}^2$.

Initial velocity

Initial angular velocity of the inertia. 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 "Computing Initial Conditions". The default value is 0.

Ports

The block has one mechanical rotational conserving port, associated with the inertia connection to the system.

See Also

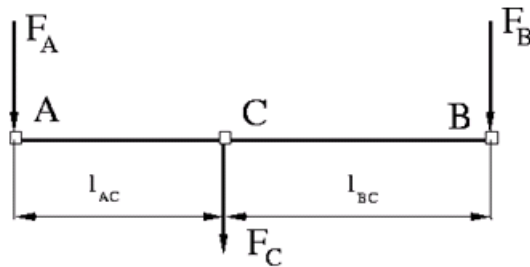
Mass

Lever

Purpose Simulate lever in mechanical systems

Library 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

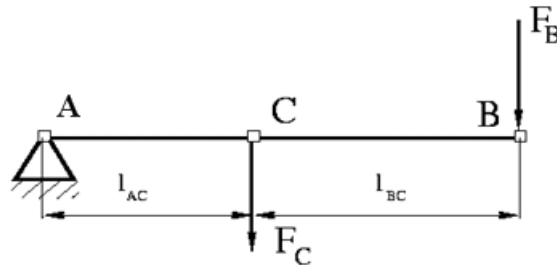
v_A, v_B, v_C Lever joints velocities

F_A, F_B, F_C Lever joints forces

l_{AC}, l_{BC} Arm lengths

The above equations were derived with the assumption that the lever sums forces and motions at node C. The assumption was 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 at the end, or a second-class lever, with the fulcrum in the middle.

The following illustration shows a schematic of a two-node first-class lever, with the fulcrum at node A.



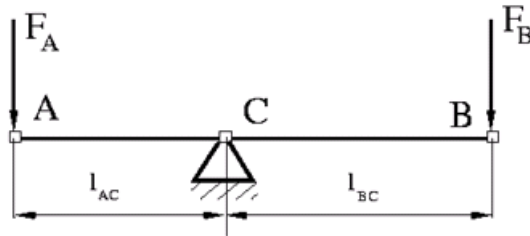
It is described with the following equations:

$$v_C = K_{BC} \cdot v_B$$

$$F_B = K_{BC} \cdot F_C$$

The next illustration shows a schematic of a second-class lever, with the fulcrum in the middle.

Lever



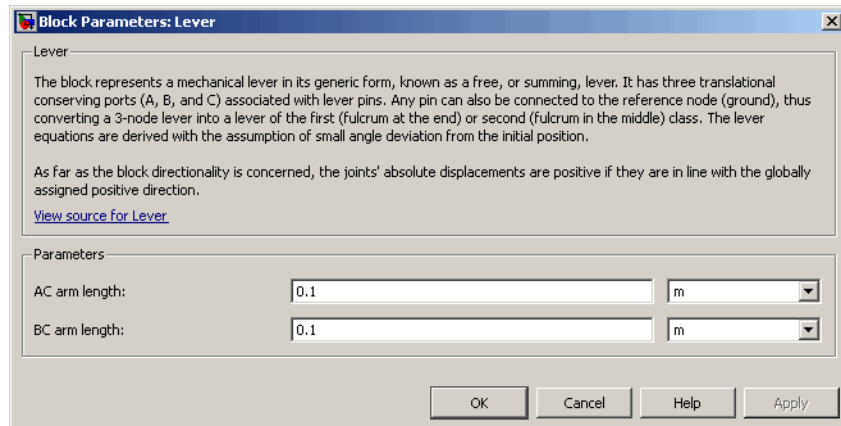
It is described with the following equations:

$$v_A = -\frac{l_{AC}}{l_{BC}} \cdot v_B$$

$$F_B = -\frac{l_{AC}}{l_{BC}} \cdot F_A$$

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.

Dialog Box and Parameters



AC arm length

Arm length between nodes A and C. The default value is 0.1 m.

BC arm length

Arm length between nodes B and C. The default value is 0.1 m.

Ports

The block has the following ports:

A

Mechanical translational conserving port associated with the node A of the lever.

B

Mechanical translational conserving port associated with the node B of the lever.

C

Mechanical translational conserving port associated with the node C of the lever.

Examples

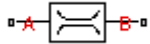
The Linkage Mechanism demo (`ssc_linkage_mechanism`) illustrates the use of the Lever block in three different modes. Linkages L_1 and L_4 simulate first-class levers with the fulcrum at the end. Linkage L_2 represents a summing lever. Linkage L_3 simulates a second-class lever with the fulcrum in the middle.

Linear Hydraulic Resistance

Purpose Simulate hydraulic pipeline with linear resistance losses

Library Hydraulic Elements

Description

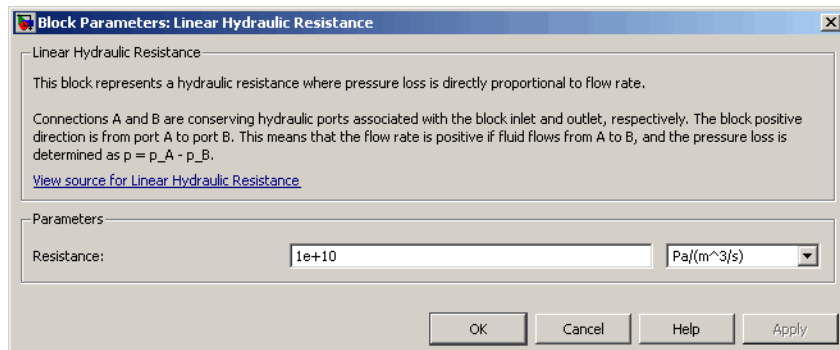


The Linear Hydraulic Resistance block represents a hydraulic resistance where pressure loss is directly proportional to flow rate. This block can be useful at preliminary stages of development, or as a powerful means to speed up the simulation, especially if the flow rate varies insignificantly with 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 $p = p_A - p_B$.

Dialog Box and Parameters



Resistance

The linear resistance coefficient. The default value is 10e9 Pa/(m³/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.

See Also

Resistive Tube

Mass

Purpose

Simulate mass in mechanical translational systems

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 Inertia force

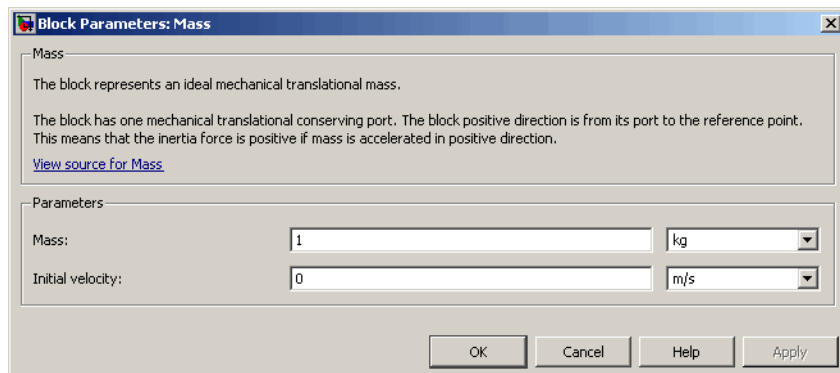
m Mass

v Velocity

t Time

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.

Dialog Box and Parameters



Mass

Mass. The default value is 1 kg.

Initial velocity

Initial velocity of the mass. 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 "Computing Initial Conditions". The default value is 0.

Ports

The block has one mechanical translational conserving port, associated with the mass connection to the system.

See Also

Inertia

Mechanical Rotational Reference

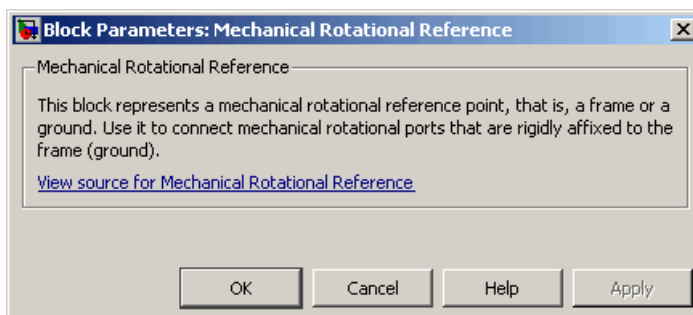
Purpose Simulate reference for mechanical rotational ports

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.



Dialog Box and Parameters



The Mechanical Rotational Reference block has no parameters.

Ports The block has one mechanical rotational port.

See Also

- Electrical Reference
- Hydraulic Reference
- Mechanical Translational Reference
- Thermal Reference

Mechanical Translational Reference

Purpose Simulate reference for mechanical translational ports

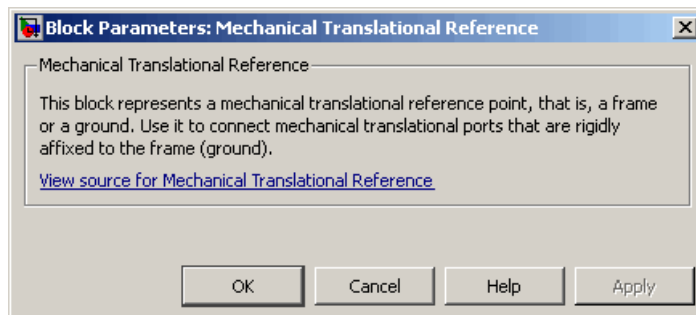
Library Mechanical Translational Elements

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.

Dialog Box and Parameters



The Mechanical Translational Reference block has no parameters.

Ports The block has one mechanical translational port.

See Also

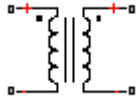
- Electrical Reference
- Hydraulic Reference
- Mechanical Rotational Reference
- Thermal Reference

Mutual Inductor

Purpose Simulate 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

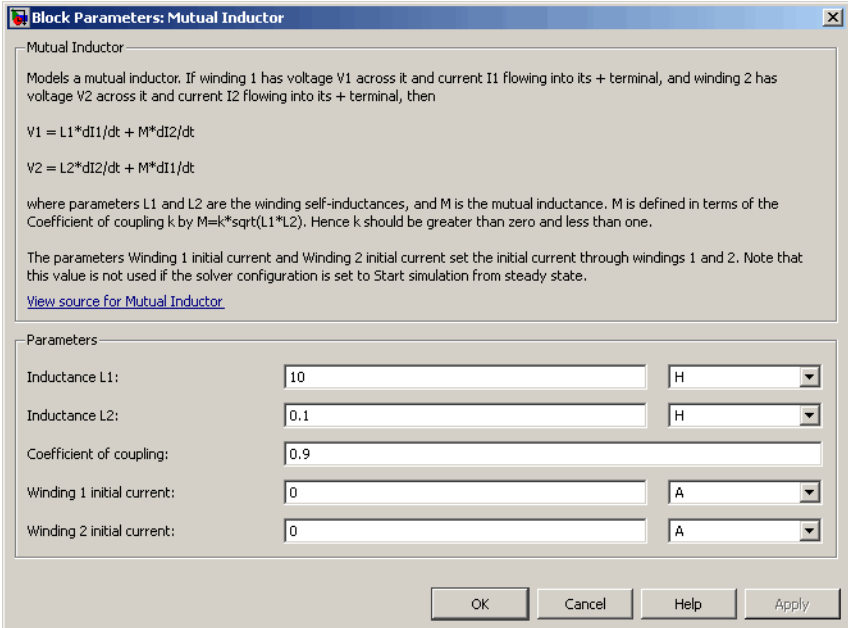
The **Winding 1 initial current** and **Winding 2 initial current** parameters set the initial current through windings 1 and 2.

Note These values are not used if the solver configuration is set to **Start simulation from steady state**.

Dialog Box and Parameters

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.



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.

Mutual Inductor

Winding 1 initial current

Initial current through the first winding. This parameter is not used if the solver configuration is set to **Start simulation from steady state**. The default value is 0.

Winding 2 initial current

Initial current through the second winding. This parameter is not used if the solver configuration is set to **Start simulation from steady state**. The default value is 0.

Ports

The block has four electrical conserving ports. Polarity is indicated by the + and – signs.

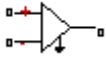
See Also

Ideal Transformer

Purpose Simulate ideal operational amplifier

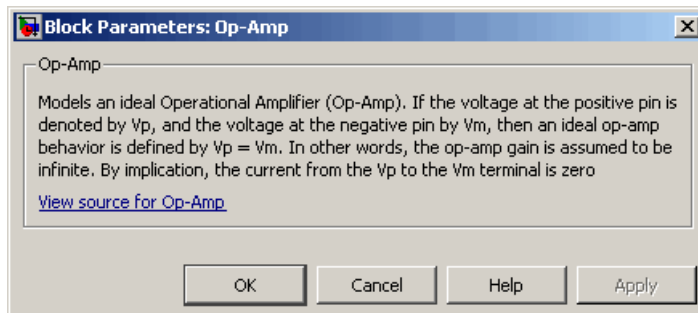
Library Electrical Elements

Description



The Op-Amp block models an ideal operational amplifier (op-amp). If the voltage at the positive pin is denoted by V_p , and the voltage at the negative pin by V_m , then an ideal op-amp behavior is defined by $V_p = V_m$. In other words, the op-amp gain is assumed to be infinite. By implication, the current from the V_p to the V_m terminal is zero.

Dialog Box and Parameters



The Op-Amp block has no parameters.

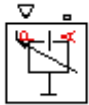
Ports The block has three electrical conserving ports.

Piston Chamber

Purpose Simulate variable volume hydraulic capacity in cylinders

Library Hydraulic Elements

Description



The 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.

Note The 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:

$$q = \frac{A(x_0 + x \cdot or)}{E} \cdot \frac{dp}{dt}$$
$$E = E_l \frac{1 + \alpha \left(\frac{p_a}{p_a + p} \right)^{1/n}}{1 + \alpha \frac{p_a^{1/n}}{n \cdot (p_a + p)^{\frac{n+1}{n}}} E_l}$$

where

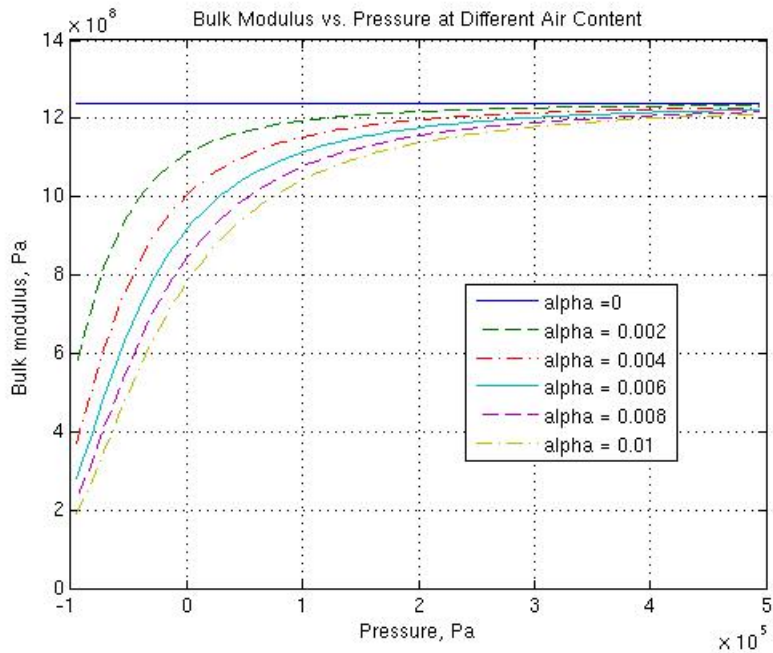
- q Flow rate due to fluid compressibility
- A Effective piston area
- x_0 Piston initial position
- x Piston displacement from initial position

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_a	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 at $p \rightarrow p_a$, thus considerably slowing down further pressure change.

At high pressure, $p \gg p_a$, a small amount of nondissolved gas has practically no effect on the system behavior.

Piston 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.

Port A is a hydraulic conserving port associated with the chamber inlet. Port P is a physical signal port that controls piston displacement.

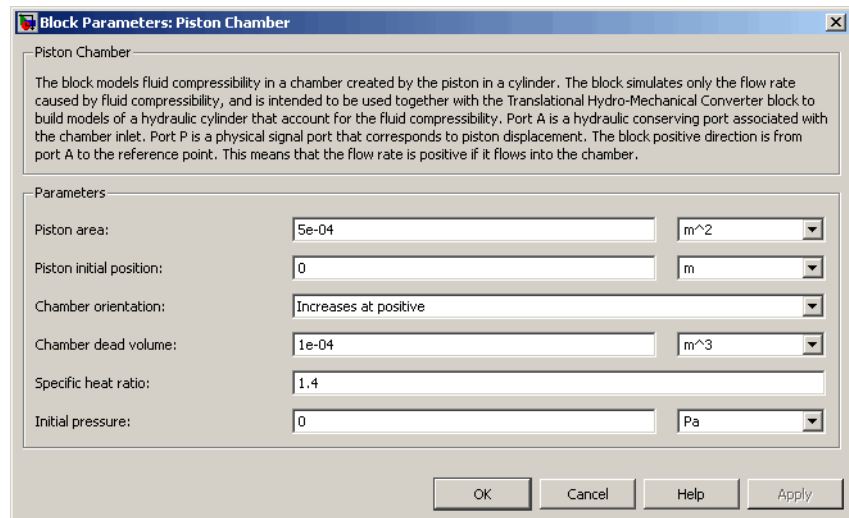
The block positive direction is from port A to the reference point. This means that the flow rate is positive if it flows into the chamber.

Basic Assumptions and Limitations

The model is based on the following assumptions:

- Fluid density remains constant.
- Chamber volume can not be less than the dead volume.
- Fluid fills the entire chamber volume.

Dialog Box and 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

Piston Chamber

chamber volume, set the parameter to `Decreases at positive`. The default value is `Increases at positive`.

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`.

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. For more information, see "Computing Initial Conditions". The default value is `0`.

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

Fluid bulk modulus

The parameter is determined by the type of working fluid selected for the system under design. Use the Hydraulic Fluid block or the Custom Hydraulic Fluid block to specify the fluid properties.

Nondissolved gas ratio

Nondissolved gas relative content determined as a ratio of gas volume to the liquid volume. The parameter is determined by the type of working fluid selected for the system under design. 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.

See Also

Constant Volume Chamber

Translational Hydro-Mechanical Converter

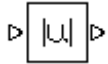
Variable Chamber

PS Abs

Purpose Output absolute value of input physical signal

Library Physical Signals/Nonlinear Operators

Description The PS Abs block returns the absolute value of the input physical signal:



$$y = |u|$$

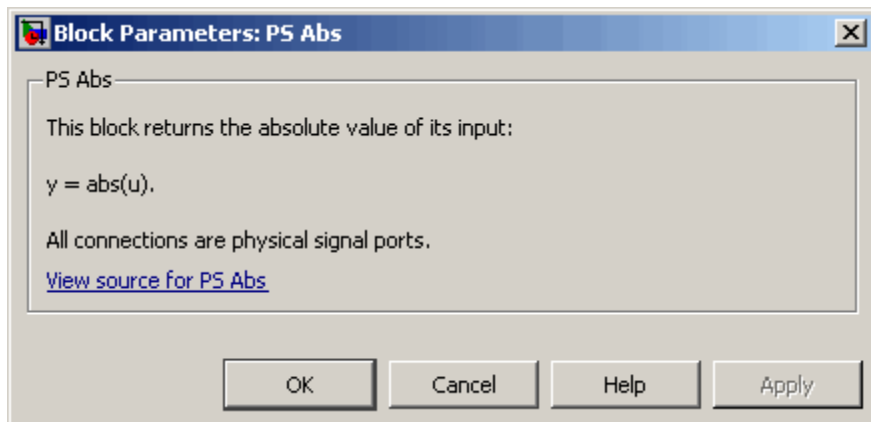
where

u Physical signal at the input port

y Physical signal at the output port

Both the input and the output are physical signals.

Dialog Box and Parameters



The PS Abs block has no parameters.

Ports The block has one physical signal input port and one physical signal output port.

See Also

PS Dead Zone

PS Max

PS Min

PS Saturation

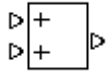
PS Sign

PS Add

Purpose Add two physical signal inputs

Library Physical Signals/Functions

Description The PS Add block outputs the sum of two input physical signals:



$$y = u_1 + u_2$$

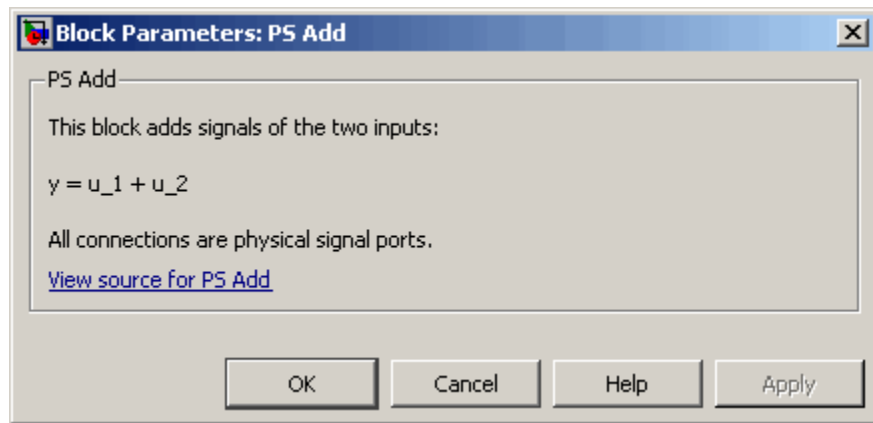
where

u_1 Physical signal at the first input port

u_2 Physical signal at the second input port

y Physical signal at the output port

Dialog Box and Parameters



The PS Add block has no parameters.

Ports The block has two physical signal input ports and one physical signal output port.

See Also PS Divide

PS Gain

PS Math Function

PS Product

PS Subtract

PS Ceil

Purpose Output the smallest integer larger than or equal to input physical signal

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:



$$y = \text{ceil}(u)$$

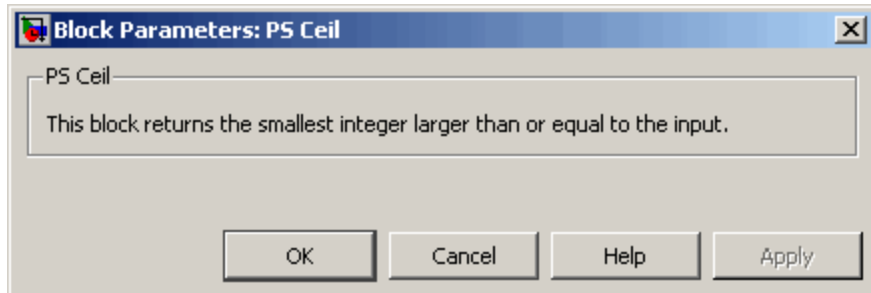
where

u Physical signal at the input port

y Physical signal at the output port

Both the input and the output are physical signals.

Dialog Box and Parameters



The PS Ceil block has no parameters.

Ports The block has one physical signal input port and one physical signal output port.

See Also

- ceil
- PS Fix
- PS Floor

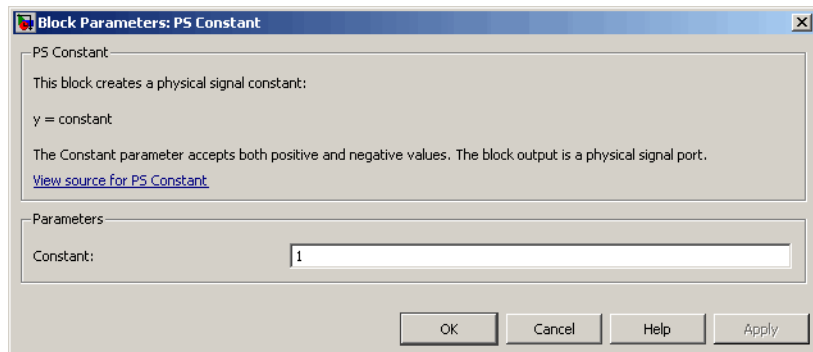
Purpose Generate constant physical signal

Library Physical Signals/Sources

Description The PS Constant block generates a physical signal of a constant value. You specify the value of the signal as the **Constant** parameter.



Dialog Box and Parameters



Constant The signal value. You can specify both positive and negative values.

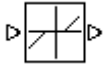
Ports The block has one physical signal output port.

PS Dead Zone

Purpose Provide region of zero output for physical signals

Library Physical Signals/Nonlinear Operators

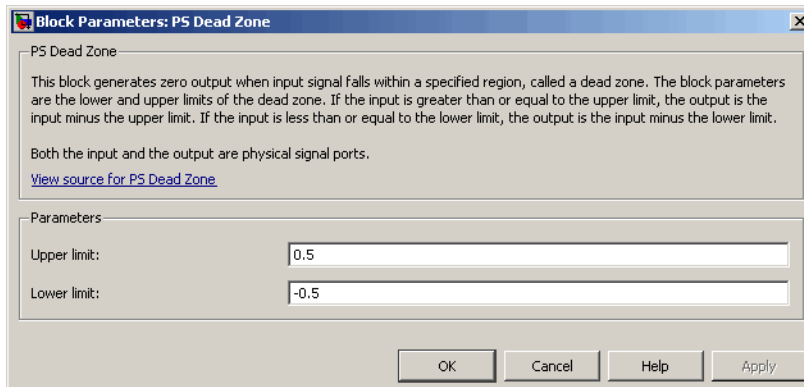
Description The PS Dead Zone block generates zero output when input signal falls within a specified region, called a dead zone. You can specify the lower and upper limits of the dead zone as block parameters. The block output depends on the input and dead zone:



- 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.

Dialog Box and Parameters



Upper limit

The upper limit, or end, of the dead zone. The default value is 0.5.

Lower limit

The lower limit, or start, of the dead zone. The default value is -0.5.

Ports

The block has one physical signal input port and one physical signal output port.

See Also

PS Abs

PS Max

PS Min

PS Saturation

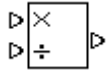
PS Sign

PS Divide

Purpose Compute simple division of two input physical signals

Library Physical Signals/Functions

Description The PS Divide block divides one physical signal input by another and outputs the difference:



$$y = u_1 \div u_2$$

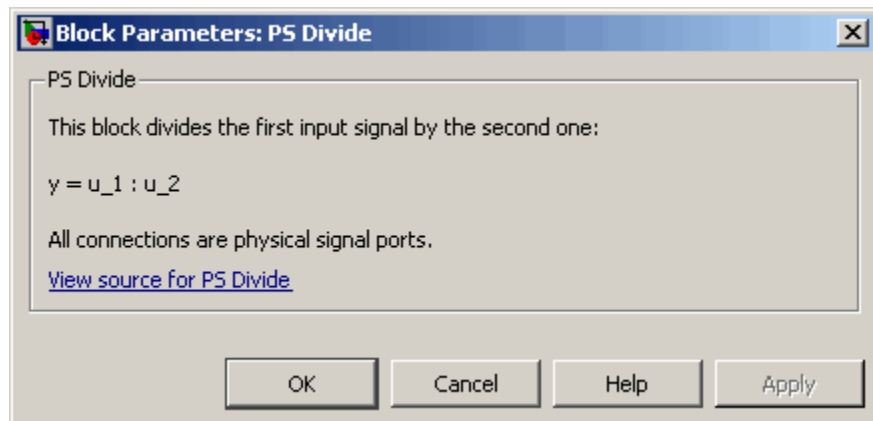
where

u_1 Physical signal at the first input port (marked with the x sign)

u_2 Physical signal at the second input port (marked with the ÷ sign)

y Physical signal at the output port

Dialog Box and Parameters



The PS Divide block has no parameters.

Ports The block has two physical signal input ports and one physical signal output port.

See Also

PS Add

PS Gain

PS Math Function

PS Product

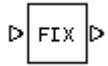
PS Subtract

PS Fix

Purpose Round input physical signal toward zero

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:



$$y = \text{fix}(u)$$

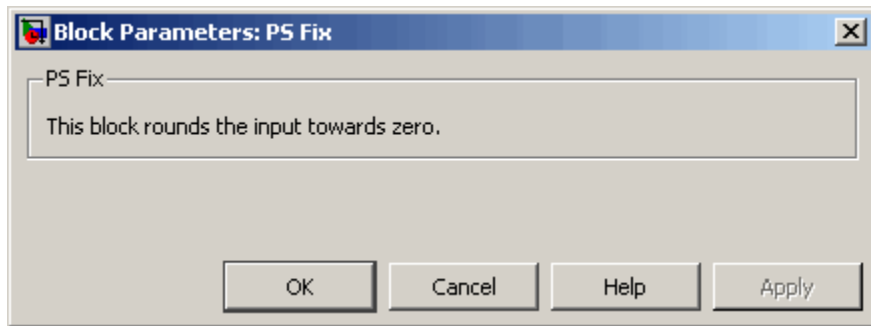
where

u Physical signal at the input port

y Physical signal at the output port

Both the input and the output are physical signals.

Dialog Box and Parameters



The PS Fix block has no parameters.

Ports The block has one physical signal input port and one physical signal output port.

See Also

fix

PS Ceil

PS Floor

PS Floor

Purpose Output the largest integer smaller than or equal to input physical signal

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:



$$y = \text{floor}(u)$$

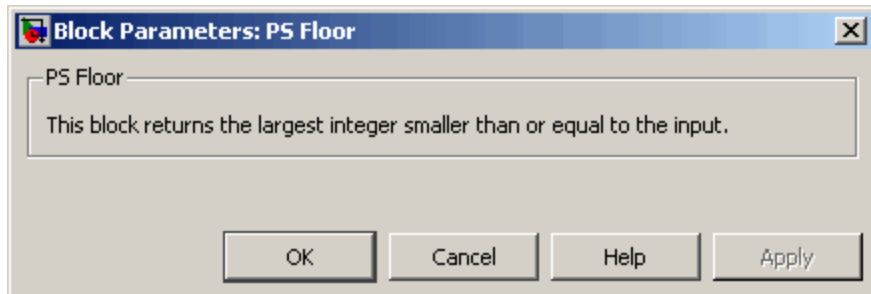
where

u Physical signal at the input port

y Physical signal at the output port

Both the input and the output are physical signals.

Dialog Box and Parameters



The PS Floor block has no parameters.

Ports The block has one physical signal input port and one physical signal output port.

See Also floor
PS Ceil
PS Fix

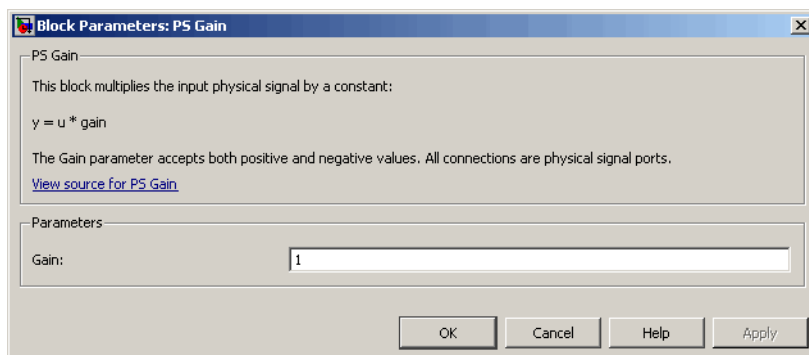
Purpose Multiply input physical signal by constant

Library Physical Signals/Functions

Description The PS Gain block multiplies the input physical signal by a constant value (gain). You specify the value of the gain as the **Gain** parameter.



Dialog Box and Parameters



Gain The multiplication coefficient. You can specify both positive and negative values.

Ports The block has one physical signal input port and one physical signal output port.

See Also

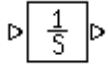
- PS Add
- PS Divide
- PS Math Function
- PS Product
- PS Subtract

PS Integrator

Purpose Integrate physical signal

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
- y_0 Initial condition
- y Physical signal at the output port
- 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:

$$x = y(t)$$

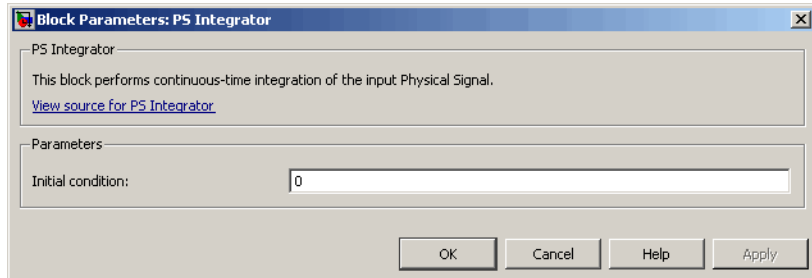
$$x_0 = y_0$$

$$\dot{x} = u(t)$$

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. You can specify another value for the initial condition as a parameter on the block dialog box.

Dialog Box and Parameters



Initial Condition

Specify the initial condition for use in computing the block's initial state at the beginning of a simulation run. For more information, see "Computing Initial Conditions". The default value is 0.

Ports

The block has one physical signal input port and one physical signal output port.

PS Lookup Table (1D)

Purpose Approximate one-dimensional function using specified lookup method

Library Physical Signals/Lookup Tables

Description The PS Lookup Table (1D) block computes an approximation to some function $y=f(x)$ given data vectors x and y . Both the input and the output are physical signals.

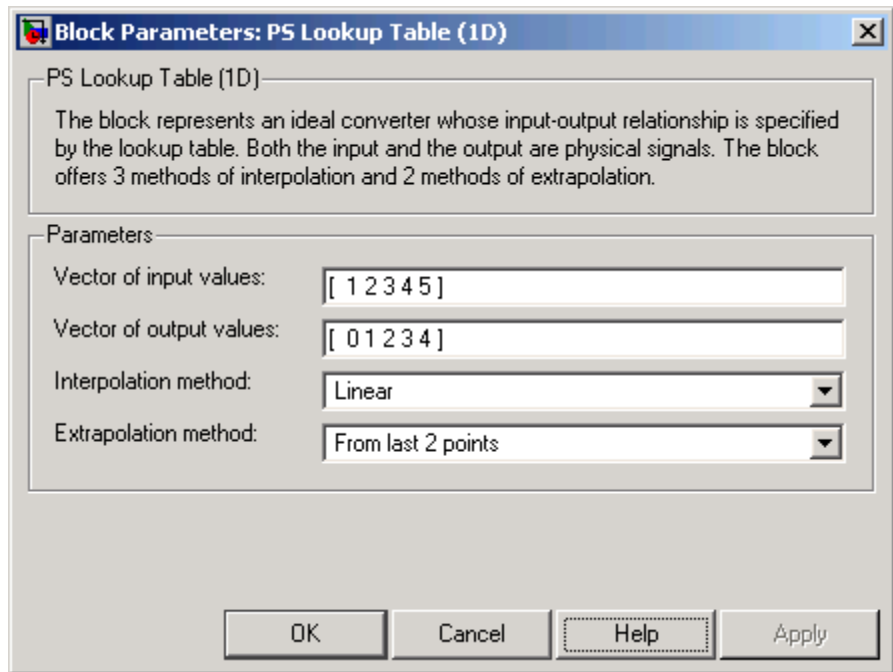


Note To map two physical signal inputs to an output, use the PS Lookup Table (2-D) block.

The length of the x and y data vectors provided to this block must match. Also, the x data vector must be *strictly monotonically increasing* (i.e., the value of the next element in the vector is greater than the value of the preceding element).

You define the lookup table by specifying the **Vector of input values** parameter as a 1-by- n vector and the **Vector of output values** parameter as a 1-by- n vector. The block generates output based on the input values using the selected interpolation and extrapolation methods. You have a choice of three interpolation methods and two extrapolation methods.

Dialog Box and Parameters



Vector of input values

Specify the vector of input values as a tabulated 1-by-n array. The input values vector must be strictly monotonically increasing. The values can be non-uniformly spaced.

Vector of output values

Specify the vector of output values as a tabulated 1-by-n array. The output values vector must be the same size as the input values vector.

Interpolation method

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.

PS Lookup Table (1D)

- **Cubic** — Uses the Piecewise Cubic Hermite Interpolation Polynomial (PCHIP). For more information, see [1] and the `pchip` MATLAB® function.
- **Spline** — Uses the cubic spline interpolation algorithm described in [2].

Extrapolation method

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:

- **From last 2 points** — Extrapolates using the linear method (regardless of the interpolation method specified), based on the last two output values at the appropriate end of the range. That is, the block uses the first and second specified output values if the input value is below the specified range, and the two last specified output values if the input value is above the specified range.
- **From last point** — Uses the last specified output value at the appropriate end of the range. That is, the block uses the last specified output value for all input values greater than the last specified input argument, and the first specified output value for all input values less than the first specified input argument.

Restricted Parameters

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

- **Interpolation method**
- **Extrapolation method**

All other block parameters are available for modification.

Ports

The block has one physical signal input port and one physical signal output port.

References

[1] D. Kahaner, Cleve Moler, Stephen Nash, *Numerical Methods and Software*, Prentice Hall, 1988

[2] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Wetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1992

See Also

PS Lookup Table (2D)

PS Lookup Table (2D)

Purpose Approximate two-dimensional function using specified lookup method

Library Physical Signals/Lookup Tables

Description

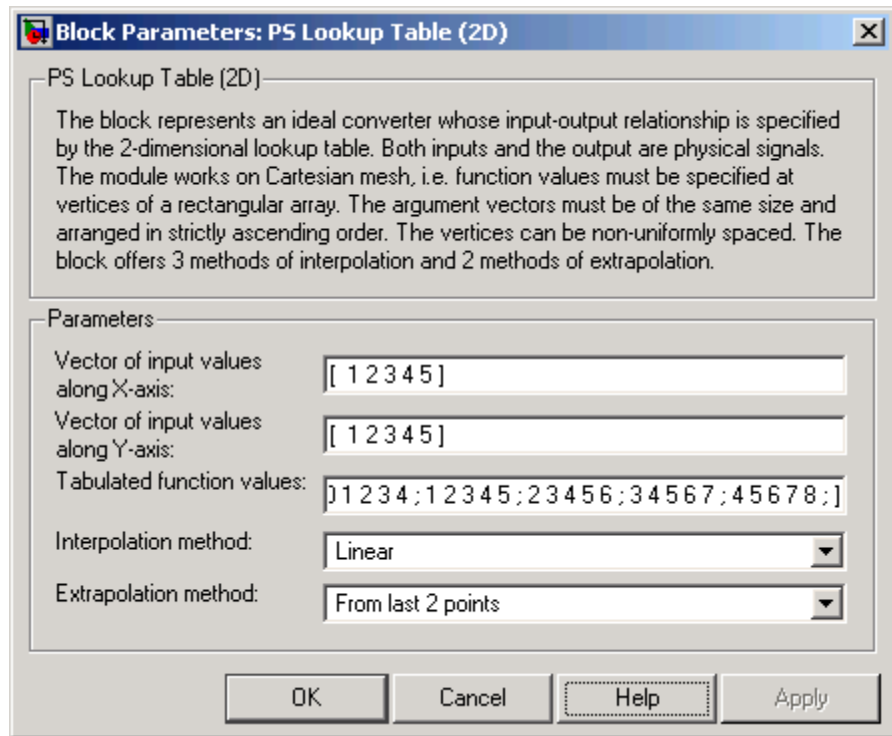


The PS Lookup Table (2D) block computes an approximation to some function $z=f(x,y)$ given the x , y , z data points. The two inputs and the output are physical signals.

The x and y data vectors must be *strictly monotonically increasing* (i.e., the value of the next element in the vector is greater than the value of the preceding element). The matrix size of the tabulated function values must match the dimensions defined by the input vectors.

You define the lookup table by specifying the **Vector of input values along X-axis** parameter as a 1-by- m vector of x data points, the **Vector of input values along Y-axis** parameter as a 1-by- n vector of y data points, and the **Tabulated function values** as an m -by- n matrix of z data points. The block works on Cartesian mesh, i.e., function values must be specified at vertices of a rectangular array. The block generates output based on the input grid lookup using the selected interpolation and extrapolation methods. You have a choice of three interpolation methods and two extrapolation methods.

Dialog Box and Parameters



Vector of input values along X-axis

Specify the vector of input values along the x -axis as a tabulated 1-by- m array. The input values vector must be strictly monotonically increasing. The values can be non-uniformly spaced.

Vector of input values along Y-axis

Specify the vector of input values along the y -axis as a tabulated 1-by- n array. The input values vector must be strictly monotonically increasing. The values can be non-uniformly spaced.

PS Lookup Table (2D)

Tabulated function values

Specify the output values as a tabulated 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.

Interpolation method

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 bilinear interpolation algorithm, which is an extension of linear interpolation for functions in two variables. The method performs linear interpolation first in x -direction and then in y -direction.
- **Cubic** — Uses the bicubic interpolation algorithm described in [1].
- **Spline** — Uses the bicubic spline interpolation algorithm described in [1].

Extrapolation method

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:

- **From last 2 points** — Extrapolates using the linear method (regardless of the interpolation method specified) based on the last two output values at the appropriate grid location, similar to PS Lookup Table (1D) block.
- **From last point** — Uses the last specified output value at the appropriate grid location, similar to PS Lookup Table (1D) block..

Restricted Parameters

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

- **Interpolation method**

- **Extrapolation method**

All other block parameters are available for modification.

Ports

The block has two physical signal input ports and one physical signal output port.

References

[1] W.H.Press, B.P.Flannery, S.A.Teulkolsky, W.T.Wetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1992

See Also

PS Lookup Table (1D)

PS Math Function

Purpose Apply mathematical function to input physical signal

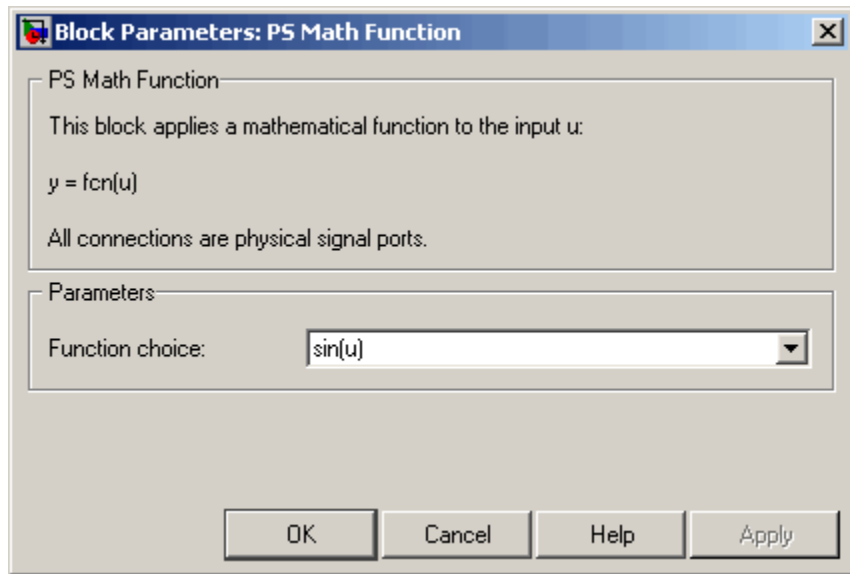
Library Physical Signals/Functions

Description The PS Math Function block applies a mathematical function to 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
$\sin(u)$	Sinus	$\sin(u)$
$\cos(u)$	Cosinus	$\cos(u)$
$\exp(u)$	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
$\text{sqrt}(u)$	Square root	$u^{0.5}$
$1/u$	Reciprocal	$1/u$

Dialog Box and Parameters



Function choice

Select the function to perform. The block output is the result of the operation of the function on the input.

Ports

The block has one physical signal input port and one physical signal output port.

See Also

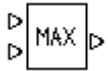
PS Add
PS Divide
PS Gain
PS Product
PS Subtract

PS Max

Purpose Output maximum of two input physical signals

Library Physical Signals/Nonlinear Operators

Description The PS Max block outputs the maximum of its two input physical signals:



$$y = \max(u_1, u_2)$$

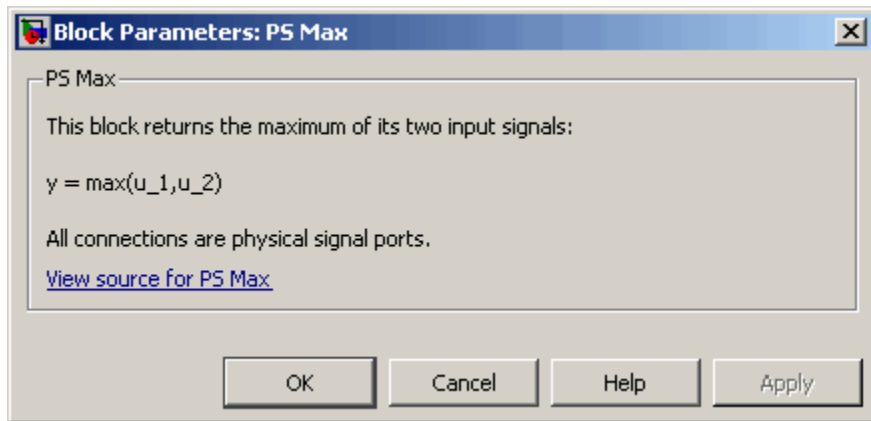
where

u_1 Physical signal at the first input port

u_2 Physical signal at the second input port

y Physical signal at the output port

Dialog Box and Parameters



The PS Max block has no parameters.

Ports

The block has two physical signal input ports and one physical signal output port.

See Also

PS Abs

PS Dead Zone

PS Min

PS Saturation

PS Sign

PS Min

Purpose Output minimum of two input physical signals

Library Physical Signals/Nonlinear Operators

Description The PS Min block outputs the minimum of its two input physical signals:



$$y = \min(u_1, u_2)$$

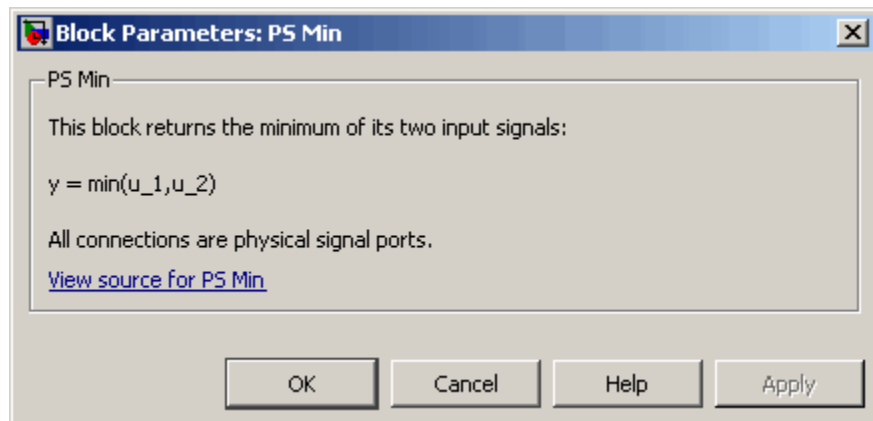
where

u_1 Physical signal at the first input port

u_2 Physical signal at the second input port

y Physical signal at the output port

Dialog Box and Parameters



The PS Min block has no parameters.

Ports The block has two physical signal input ports and one physical signal output port.

See Also PS Abs
PS Dead Zone

PS Max

PS Saturation

PS Sign

PS Product

Purpose Multiply two physical signal inputs

Library Physical Signals/Functions

Description The PS Product block outputs the product of two input physical signals:



$$y = u_1 \cdot u_2$$

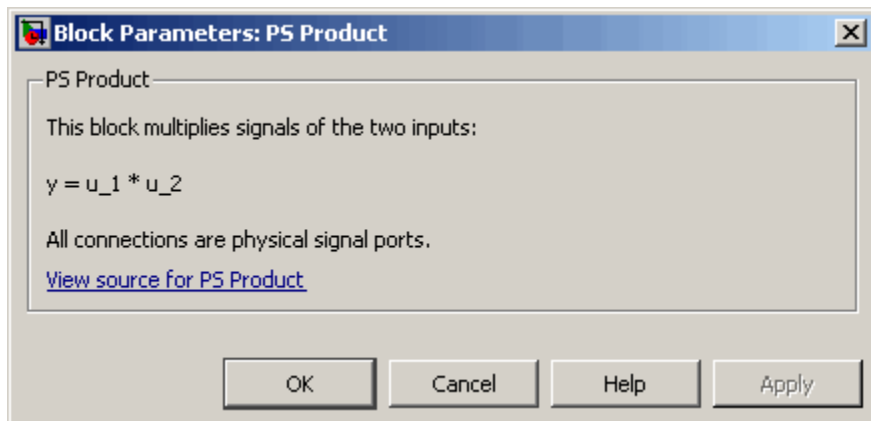
where

u_1 Physical signal at the first input port

u_2 Physical signal at the second input port

y Physical signal at the output port

Dialog Box and Parameters



The PS Product block has no parameters.

Ports The block has two physical signal input ports and one physical signal output port.

See Also PS Add
PS Divide

PS Gain

PS Math Function

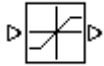
PS Subtract

PS Saturation

Purpose Limit range of physical signal

Library Physical Signals/Nonlinear Operators

Description

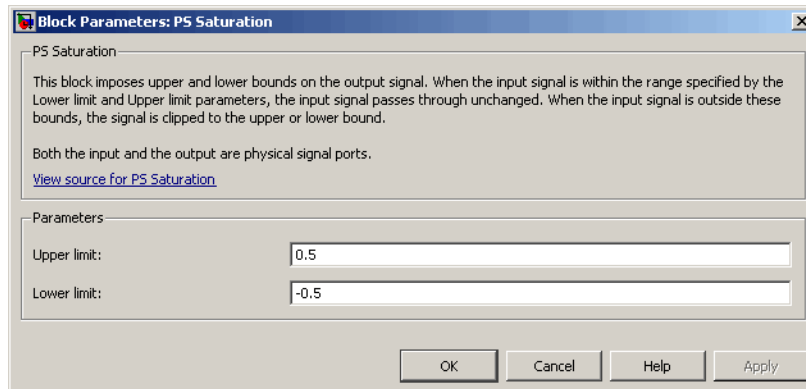


The PS Saturation block imposes upper and lower bounds on a physical signal. When the input signal is within the range specified by the **Lower limit** and **Upper limit** parameters, the input signal passes through unchanged. When the input signal is outside these bounds, the signal is clipped to the 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.

Dialog Box and Parameters



Upper limit

The upper bound on the input signal. When the input signal to the Saturation block is above this value, the output of the block is clipped to this value. The default is 0.5.

Lower limit

The lower bound on the input signal. When the input signal to the Saturation block is below this value, the output of the block is clipped to this value. The default is -0.5.

Ports The block has one physical signal input port and one physical signal output port.

See Also

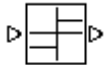
- PS Abs
- PS Dead Zone
- PS Max
- PS Min
- PS Sign

PS Sign

Purpose Output sign of input physical signal

Library Physical Signals/Nonlinear Operators

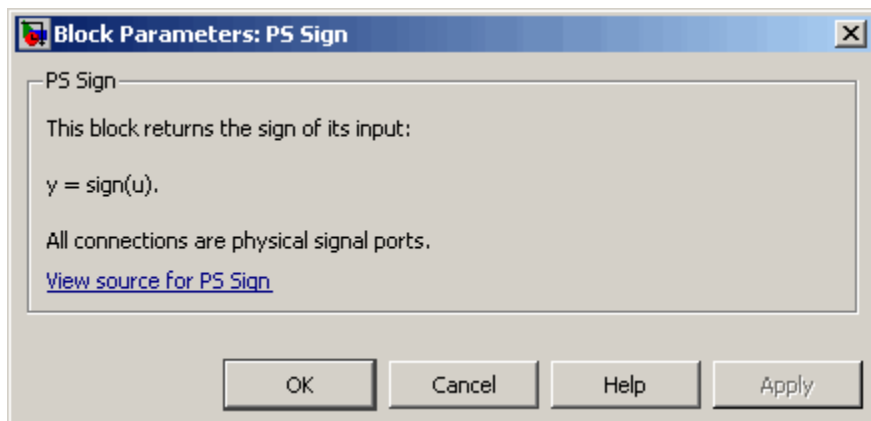
Description The PS Sign block returns the sign of the input physical signal:



- 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.

Dialog Box and Parameters



The PS Sign block has no parameters.

Ports The block has one physical signal input port and one physical signal output port.

See Also PS Abs
PS Dead Zone
PS Max

PS Min

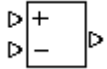
PS Saturation

PS Subtract

Purpose Compute simple subtraction of two input physical signals

Library Physical Signals/Functions

Description The PS Subtract block subtracts one physical signal input from another and outputs the difference:



$$y = u_1 - u_2$$

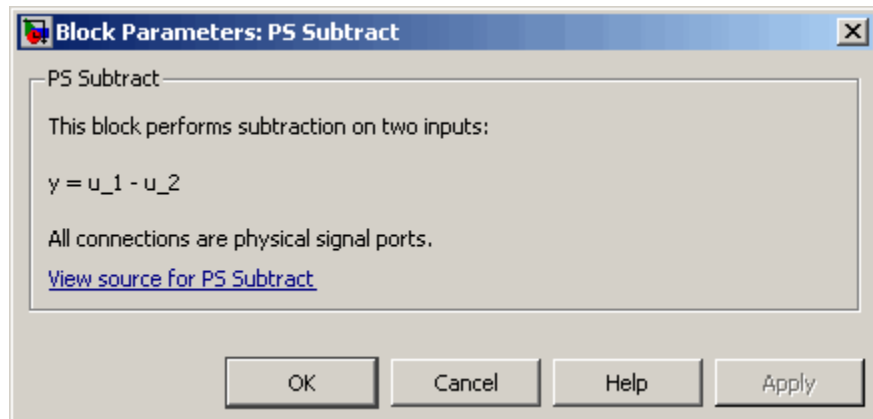
where

u_1 Physical signal at the first input port (marked with the plus sign)

u_2 Physical signal at the second input port (marked with the minus sign)

y Physical signal at the output port

Dialog Box and Parameters



The PS Subtract block has no parameters.

Ports The block has two physical signal input ports and one physical signal output port.

See Also

PS Add

PS Divide

PS Gain

PS Math Function

PS Product

PS-Simulink Converter

Purpose Convert physical signal into Simulink output signal

Library Utilities

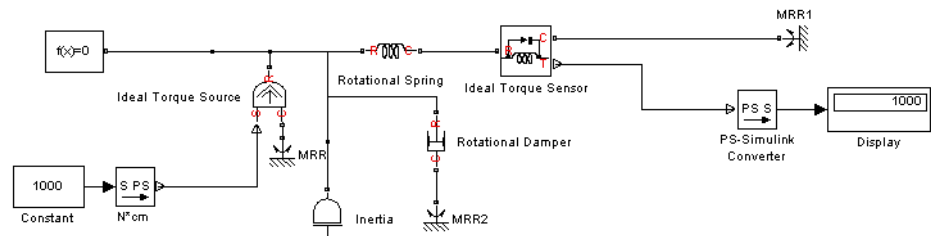
Description



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.

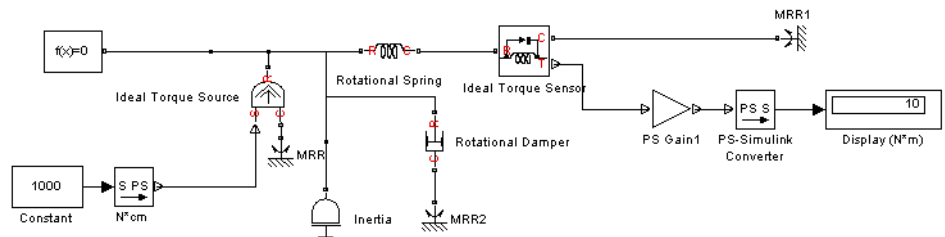
The **Output signal unit** parameter lets you specify the desired units for the output signal. These units must be commensurate with the units of the input physical signal coming into the block. The Simulink output signal is unitless, but if you specify a desired output unit, the block applies a gain 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.

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.



Note Currently, physical units are not propagated through the blocks in the Physical Signals library, such as PS Add, PS Gain, and so on. If your diagram contains a Physical Signals block before a PS-Simulink Converter block, the unit specification in the PS-Simulink Converter block is ignored.

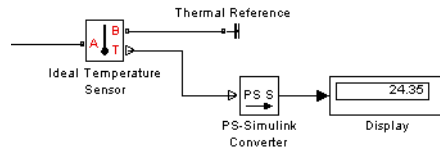
In the following example, the PS-Simulink Converter block is installed after the PS Gain1 block. The display reading will remain the same regardless of the **Output signal unit** parameter setting in the PS-Simulink Converter block.



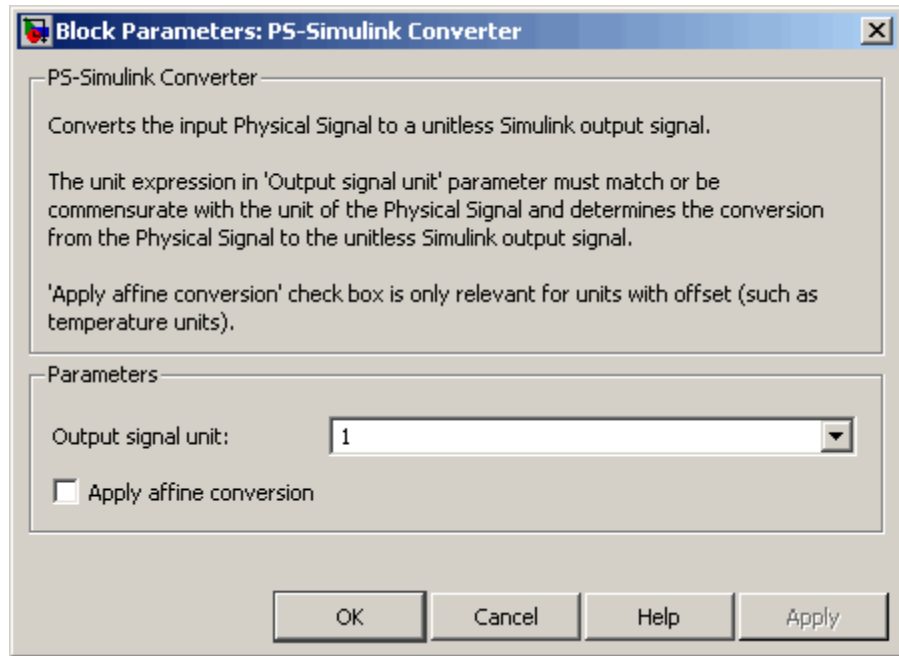
When the output signal is related to thermodynamic variables and contains units of temperature, you must decide whether affine conversion needs to be applied. For more information, see “When to Apply Affine Conversion”. Usually, if the output signal represents a relative temperature, that is, a change 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 C 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.

PS-Simulink Converter



Dialog Box and 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 “Working with Physical Units”. The default value is 1, which means that the unit is not specified. If you do not specify a unit, or if the unit matches the actual units of the input physical signal, no gain is applied.

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

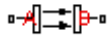
Simulink-PS Converter

Radiative Heat Transfer

Purpose Simulate heat transfer by radiation

Library Thermal Elements

Description



The Radiative Heat Transfer block represents a heat transfer by radiation between two surfaces in such a way that the energy of emitting body is completely absorbed by a receiving body. 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

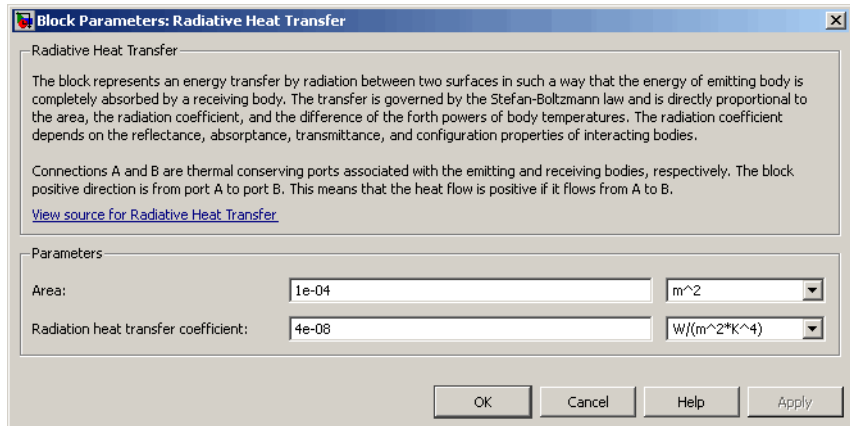
k Radiation heat transfer coefficient

A Surface area

T_A, T_B Temperatures of the bodies

Connections A and B are thermal conserving ports associated with the 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.

Dialog Box and Parameters



Area

Surface area of heat transfer. The default value is 0.0001 m^2 .

Radiation heat transfer coefficient

Heat transfer coefficient according to the Stefan-Boltzmann law. The default value is $4e-8 \text{ W/m}^2/\text{K}^4$.

Ports

The block has the following ports:

A

Thermal conserving port associated with body A.

B

Thermal conserving port associated with body B.

See Also

Conductive Heat Transfer

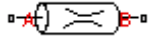
Convective Heat Transfer

Resistive Tube

Purpose Simulate hydraulic pipeline which accounts for friction losses only

Library Hydraulic Elements

Description



The 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]). 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:

$$p = f \frac{(L + L_{eq})}{D_H} \frac{\rho}{2A^2} q \cdot |q|$$

$$f = \begin{cases} K_s / Re & \text{for } Re \leq Re_L \\ f_L + \frac{f_T - f_L}{Re_T - Re_L} (Re - Re_L) & \text{for } Re_L < Re < Re_T \\ \frac{1}{\left(-1.8 \log_{10} \left(\frac{6.9}{Re} + \left(\frac{r/D_H}{3.7} \right)^{1.11} \right) \right)^2} & \text{for } Re \geq Re_T \end{cases}$$

$$Re = \frac{q \cdot D_H}{A \cdot v}$$

where

- ρ 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

Resistive Tube

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 $p = p_A - p_B$.

Basic Assumptions and Limitations

The model is based on the following assumptions:

- Flow is assumed to be fully developed along the pipe length.
- Fluid inertia, fluid compressibility, and wall compliance are not taken into account.

Dialog Box and Parameters

Block Parameters: Resistive Tube

Resistive Tube

This block models hydraulic pipelines with circular and noncircular cross sections and accounts for resistive property only. 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.

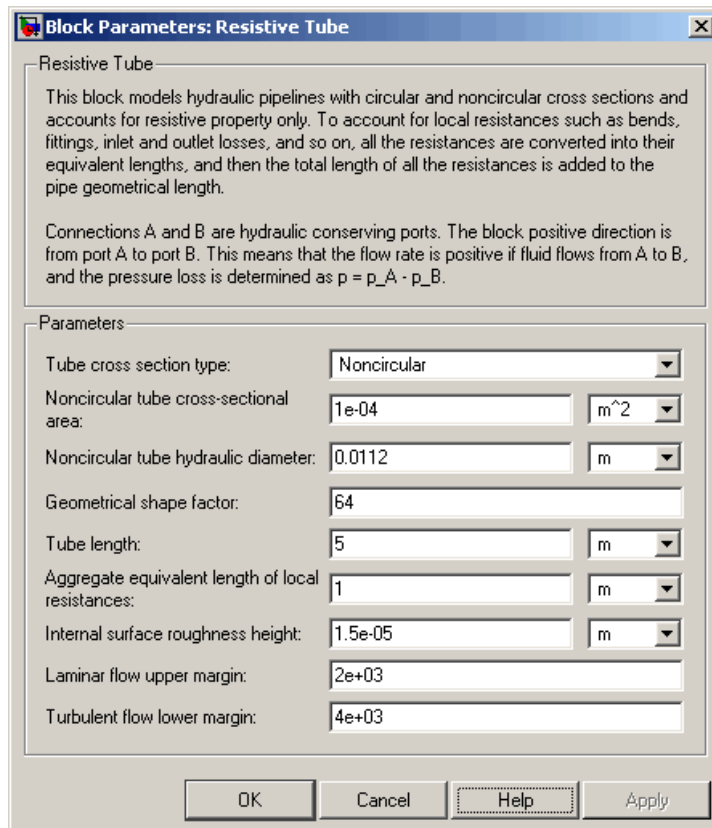
Connections A and B are hydraulic conserving ports. The block positive direction is from port A to port B. This means that the flow rate is positive if fluid flows from A to B, and the pressure loss is determined as $p = p_A - p_B$.

Parameters

Tube cross section type:	Circular
Tube internal diameter:	0.01 m
Geometrical shape factor:	64
Tube length:	5 m
Aggregate equivalent length of local resistances:	1 m
Internal surface roughness height:	1.5e-05 m
Laminar flow upper margin:	2e+03
Turbulent flow lower margin:	4e+03

OK Cancel Help Apply

Resistive Tube



Tube cross section type

The parameter can have one of two values: **Circular** or **Noncircular**. For a circular tube, you need to specify its internal diameter. For a noncircular tube, you need to 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

The parameter is used for computing friction factor at laminar flow and depends of the shape of the tube cross section. For a tube with noncircular cross section, you must 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 (see [1]). 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 $1.5e-5 \text{ m}$, 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.

Resistive Tube

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

Fluid density

The parameter is determined by the type of working fluid selected for the system under design. Use the Hydraulic Fluid block or the Custom Hydraulic Fluid block to specify the fluid properties.

Fluid kinematic viscosity

The parameter is determined by the type of working fluid selected for the system under design. 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

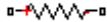
See Also

Linear Hydraulic Resistance

Purpose Simulate 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

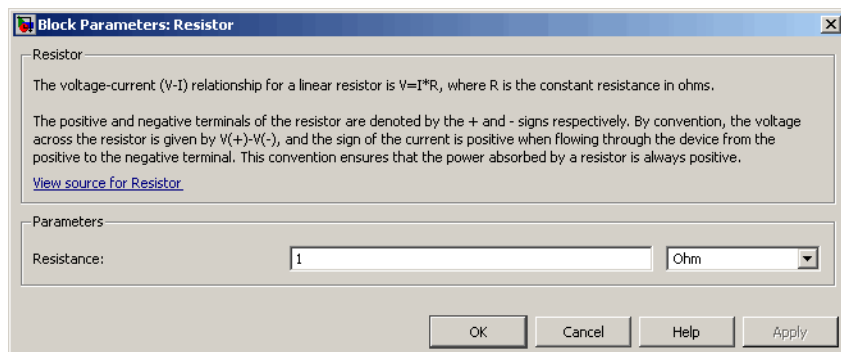
V Voltage

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.

Dialog Box and Parameters



Resistance

Resistance, in ohms. The default value is 1 Ω .

Resistor

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.

See Also

Variable Resistor

Purpose

Simulate 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

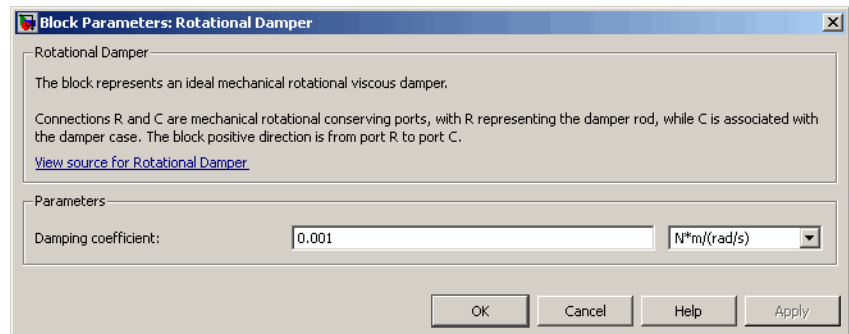
D Damping (viscous friction) coefficient

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.

Dialog Box and Parameters



Damping coefficient

Damping coefficient, defined by viscose friction. The default value is 0.001 N*m/(rad/s).

Rotational Damper

Ports

The block has the following ports:

R

Mechanical rotational conserving port.

C

Mechanical rotational conserving port.

See Also

Rotational Friction

Rotational Hard Stop

Rotational Spring

Rotational Electromechanical Converter

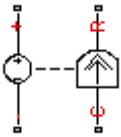
Purpose

Provide 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

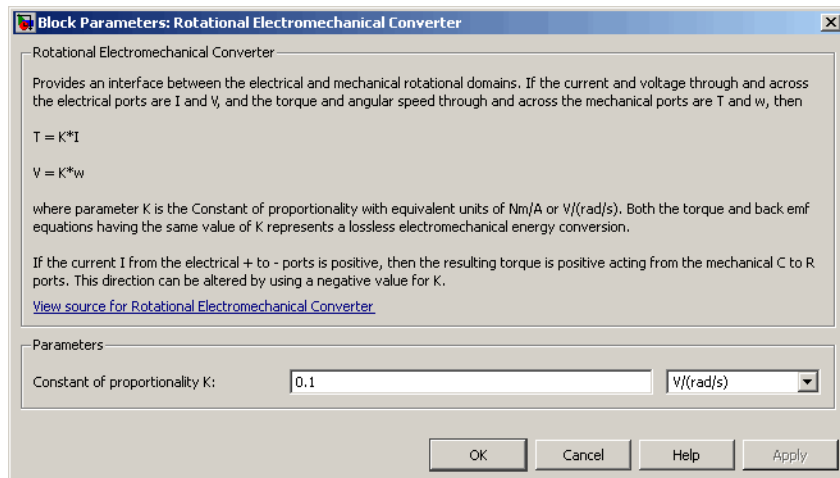
V	Voltage across the electrical ports of the converter
I	Current through the electrical ports of the converter
T	Torque
ω	Angular speed
K	Constant of proportionality

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 .

Rotational Electromechanical Converter

Dialog Box and 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.

See Also

Translational Electromechanical Converter

Purpose

Simulate friction in contact between rotating bodies

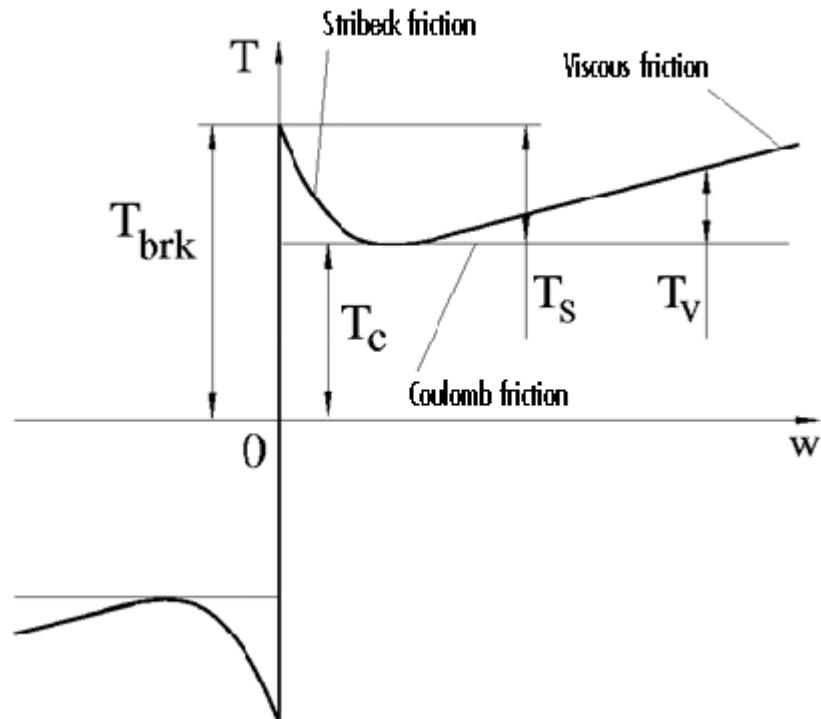
Library

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]). The Coulomb friction, T_c , results in

Rotational Friction

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:

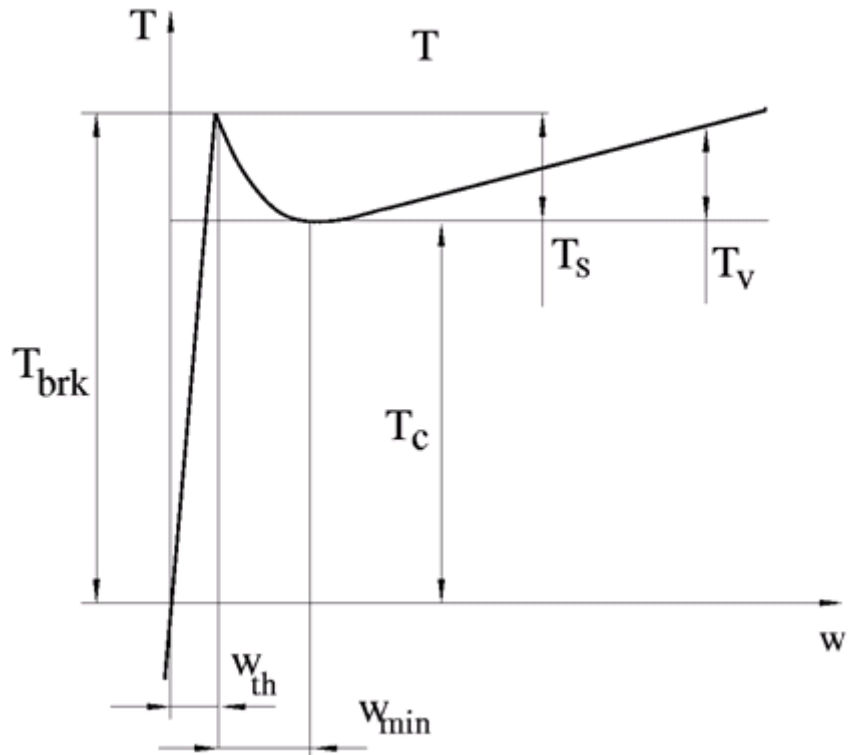
$$T = (T_C + (T_{brk} - T_C) \cdot \exp(-c_v |\omega|)) \text{sign}(\omega) + f \omega$$

$$\omega = \omega_R - \omega_C$$

where

T	Friction torque
T_C	Coulomb friction torque
T_{brk}	Breakaway friction torque
c_v	Coefficient
ω	Relative velocity
ω_R, ω_C	Absolute angular velocities of terminals R and C, respectively
f	Viscous friction coefficient

The approximation above is too idealistic and has a substantial drawback. The characteristic is discontinuous at $\omega = 0$, which creates considerable computational problems. It has been proven that the discontinuous friction model is a nonphysical simplification in the sense that the mechanical contact with distributed mass and compliance cannot exhibit an instantaneous change in torque (see [1]). There are numerous models of friction without discontinuity. The Rotational Friction block implements one of the simplest versions of continuous friction models. The friction torque-relative velocity characteristic of this approximation is shown in the following figure.



The discontinuity is eliminated by introducing a very small, but finite, region in the zero velocity vicinity, within which friction torque is assumed to be linearly proportional to velocity, with the proportionality coefficient T_{brk}/ω_{th} , where ω_{th} is the velocity threshold. It has been proven experimentally that the velocity threshold in the range between 10^{-3} and 10^{-5} rad/s is a good compromise between the accuracy and computational robustness and effectiveness. Notice that friction torque computed with this approximation does not actually stop relative motion when an acting torque drops below breakaway friction level. The bodies will creep relative to each other at a very small velocity proportional to acting torque.

Rotational Friction

As a result of introducing the velocity threshold, the block equations are slightly modified:

- If $|\omega| \geq \omega_{th}$,

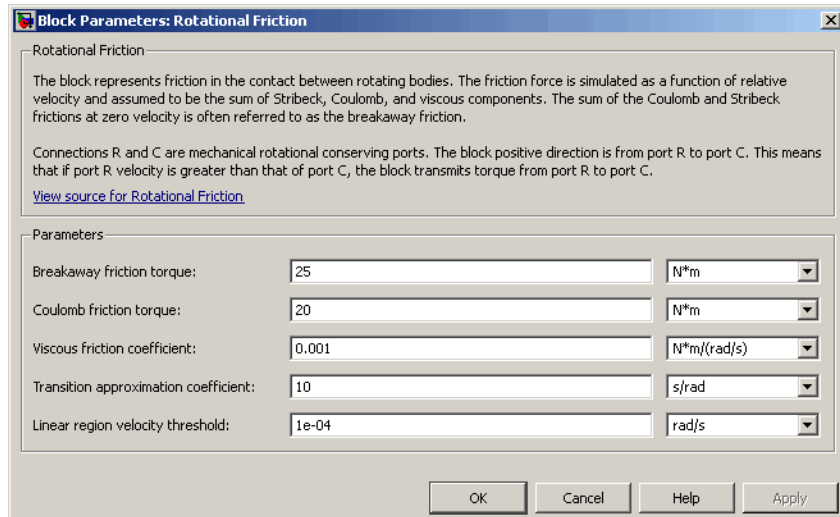
$$T = (T_C + (T_{brk} - T_C) \cdot \exp(-c_v |\omega|)) \text{sign}(\omega) + f\omega$$

- If $|\omega| < \omega_{th}$,

$$T = \omega \frac{(f\omega_{th} + (T_C + (T_{brk} - T_C) \cdot \exp(-c_v \omega_{th})))}{\omega_{th}}$$

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.

Dialog Box and 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.

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 0.001 N*m/(rad/s).

Transition approximation coefficient

The parameter sets the value of coefficient c_v , which is used for the approximation of the transition between the static and the Coulomb frictions. Its value is assigned based on the following considerations: the static friction component reaches approximately 95% of its steady-state value at velocity $3/c_v$, and 98% at velocity $4/c_v$, which makes it possible to develop an approximate relationship $c_v \approx 4/\omega_{min}$, where ω_{min} is the relative velocity at which friction torque has its minimum value. By default, c_v is set to 10 rad/s, which corresponds to a minimum friction at velocity of about 0.4 s/rad.

Linear region velocity threshold

The parameter sets the small vicinity near zero velocity, within which friction torque is considered to be linearly proportional to the relative velocity. The MathWorks recommends that you use values in the range between $1e-5$ and $1e-3$ rad/s. The default value is $1e-4$ rad/s.

Ports

The block has the following ports:

R

Mechanical rotational conserving port.

Rotational Friction

C

Mechanical rotational conserving port.

Examples

The Mechanical Rotational System with Stick-Slip Motion demo (`ssc_rot_system_stick_slip`) 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

See Also

Rotational Damper

Rotational Hard Stop

Rotational Spring

Purpose Simulate double-sided rotational hard stop

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 force 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 hard stop is described with the following equations:

$$T = \begin{cases} K_p \cdot \delta + D_p (\omega_R - \omega_C) & \text{for } \delta \geq g_p \\ 0 & \text{for } g_n < \delta < g_p \\ K_n \cdot \delta + D_n (\omega_R - \omega_C) & \text{for } \delta \leq g_n \end{cases}$$

$$\delta = \varphi_R - \varphi_C$$

$$\omega_R = \frac{d\varphi_R}{dt}$$

$$\omega_C = \frac{d\varphi_C}{dt}$$

where

- T Interaction torque between the slider and the case
- δ Relative angular displacement between the slider and the case

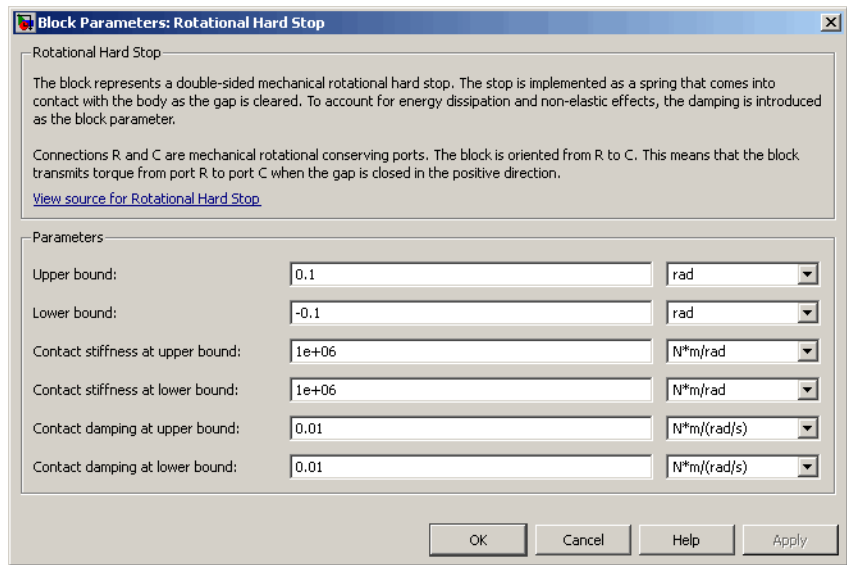
Rotational Hard Stop

g_p	Gap between the slider and the case in positive direction
g_n	Gap between the slider and the case in negative direction
ω_R, ω_C	Absolute angular velocities of terminals R and C, respectively
φ_R, φ_C	Absolute angular displacements of terminals R and C, respectively
K_p	Contact stiffness at positive restriction
K_n	Contact stiffness at negative restriction
D_p	Damping coefficient at positive restriction
D_n	Damping coefficient at negative restriction
t	Time

The equations are derived with respect to the local coordinate system whose axis is directed clockwise from port R to port C. The terms “positive” and “negative” in the variable descriptions refer to this coordinate system, and the gap in negative direction must be specified with negative value. If the local coordinate system is not aligned with the globally assigned positive direction, the gaps interchange their values with respective sign adjustment.

The block is oriented from R to C. This means that the block transmits torque from port R to port C when the gap in positive direction is cleared up.

Dialog Box and Parameters



Upper bound

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. The default value is 0.1 rad.

Lower bound

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. The default value is -0.1 rad.

Contact stiffness at upper bound

The parameter specifies the elastic property of colliding bodies when the slider hits the upper bound. The greater the value of

Rotational Hard Stop

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. The default value is $1e6 \text{ N}^*\text{m}/\text{rad}$.

Contact stiffness at lower bound

The parameter specifies the elastic property of colliding bodies when the slider hits the lower 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. The default value is $1e6 \text{ N}^*\text{m}/\text{rad}$.

Contact damping at upper bound

The parameter specifies dissipating property of colliding bodies when the slider hits the upper bound. At zero damping, the impact is close to an absolutely elastic one. The greater the value of the parameter, the more energy dissipates during an interaction. Keep in mind that damping affects slider motion as long as the slider is in contact with the stop, including the period when slider is pulled back from the contact. For computational efficiency and convergence reasons, The MathWorks recommends that you assign a nonzero value to this parameter. The default value is $0.01 \text{ N}^*\text{m}^*\text{s}/\text{rad}$.

Contact damping at lower bound

The parameter specifies dissipating property of colliding bodies when the slider hits the lower bound. At zero damping, the impact is close to an absolutely elastic one. The greater the value of the parameter, the more energy dissipates during an interaction. Keep in mind that damping affects slider motion as long as the slider is in contact with the stop, including the period when slider is pulled back from the contact. For computational efficiency and convergence reasons, The MathWorks recommends that you assign a nonzero value to this parameter. The default value is $0.01 \text{ N}^*\text{m}^*\text{s}/\text{rad}$.

Ports

The block has the following ports:

- R Mechanical rotational conserving port associated with the slider that travels between stops installed on the case.
- C Mechanical rotational conserving port associated with the case.

See Also

- Rotational Damper
- Rotational Friction
- Rotational Spring

Rotational Hydro-Mechanical Converter

Purpose Simulate ideal hydro-mechanical transducer as building block for rotary actuators

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. Using this block as a basic element, you can build a large variety of rotary actuators by adding application-specific effects, such as fluid compressibility, leakage, friction, hard stops, and so on.

The converter is simulated according to the following equations:

$$q = D(\omega_S - \omega_C) \cdot or$$

$$T = D \cdot p \cdot or$$

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
- F Torque on the shaft
- p Gauge pressure of fluid in the converter chamber
- or Converter orientation with respect to the globally assigned positive direction. If pressure applied at port A generates torque in positive direction, or equals 1. If pressure applied at port B generates torque in positive direction, or equals -1.

Rotational Hydro-Mechanical Converter

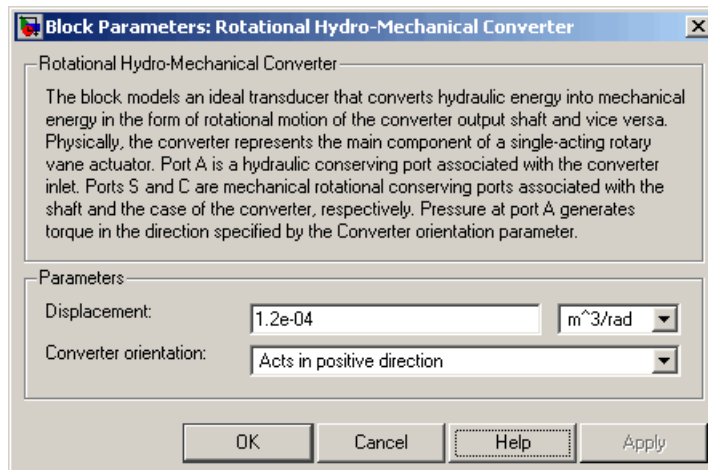
Port A is a hydraulic conserving port associated with the converter inlet. Ports S and C are mechanical rotational conserving ports associated with the shaft and the case of the converter, respectively. Pressure at port A generates torque in the direction specified by the **Converter orientation** parameter.

Basic Assumptions and Limitations

The model is based on the following assumption:

- The block simulates an ideal converter, with only the transduction property considered. No inertia, friction, leakage, or other effects are taken into account.

Dialog Box and Parameters



Displacement

Effective converter displacement. The default value is $1.2 \times 10^{-4} \text{ m}^3/\text{rad}$.

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

Rotational Hydro-Mechanical Converter

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`.

Restricted Parameters

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

- **Converter orientation**

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.

See Also

Translational Hydro-Mechanical Converter

Purpose

Simulate 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:

$$T = K \cdot \varphi$$

$$\varphi = \varphi_{init} + \varphi_R - \varphi_C$$

$$\omega = \frac{d\varphi}{dt}$$

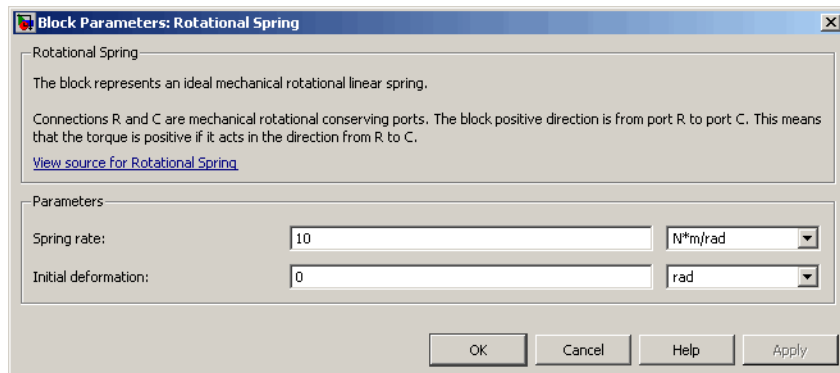
where

- 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.

Rotational Spring

Dialog Box and Parameters



Spring rate

Spring rate. The default value is 10 N*m/rad.

Initial deformation

Spring initial deformation, or offset, in angular units. The deformation is determined as $\varphi = \varphi_{\text{init}} + \varphi_R - \varphi_C$, where φ_{init} is the initial deformation, and φ_R, φ_C are the absolute angular displacements of terminals R and C in the globally assigned positive direction. The spring can be initially compressed ($\varphi_{\text{init}} > 0$) or stretched ($\varphi_{\text{init}} < 0$). 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 "Computing Initial Conditions". The default value is 0.

Ports

The block has the following ports:

- R
Mechanical rotational conserving port.
- C
Mechanical rotational conserving port.

See Also

- Rotational Damper
- Rotational Friction

Rotational Hard Stop

Simulink-PS Converter

Purpose Convert Simulink input signal into physical signal

Library Utilities

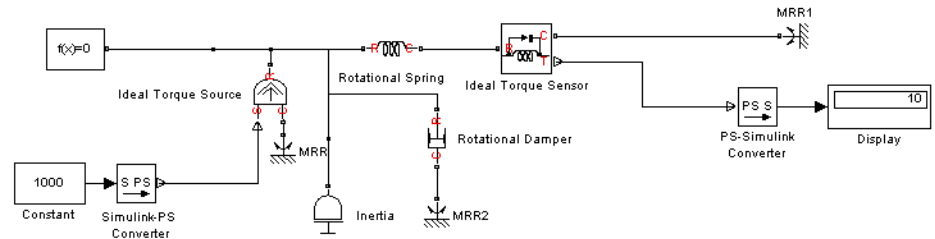
Description



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.

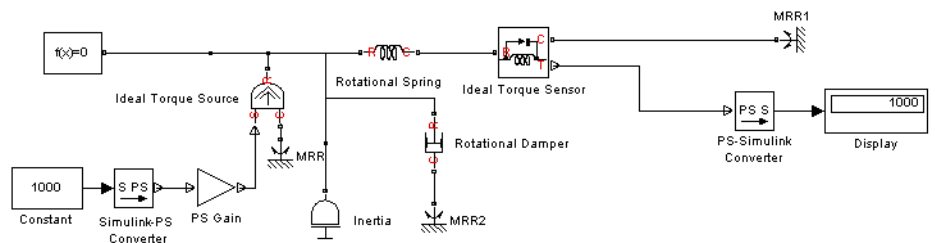
You specify the desired units as the **Input signal unit** parameter. If you leave the block unitless, with the **Input signal unit** parameter set to 1, then the physical signal units are inferred from the destination block. The default destination block units are meter-kilogram-second or MKS (SI). 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 $N\cdot m$, 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 $1000 N\cdot m$. 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 $N\cdot cm$, the unit manager performs the conversion and the Ideal Torque Source block generates torque of $10 N\cdot m$; the torque value in the Display block changes to 10, as shown in the diagram.



Note Currently, physical units are not propagated through the blocks in the Physical Signals library, such as PS Add, PS Gain, and so on. If your diagram contains a Physical Signals block after a Simulink-PS Converter block, the unit specification in the Simulink-PS Converter block does not propagate to the rest of the network.

In the following example, the PS Gain block is installed after the Simulink-PS Converter block. It stops the unit propagation to the rest of the physical network, and the Ideal Torque Source block will generate torque of 1000 N*m regardless of the **Input signal unit** parameter setting in the Simulink-PS Converter block.

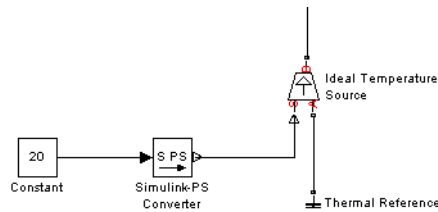


When the input signal is related to thermodynamic variables and contains units of temperature, you must decide whether affine conversion needs to be applied. For more information, see “When to Apply Affine Conversion”. Usually, if the input signal represents

Simulink-PS Converter

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 **C** 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.

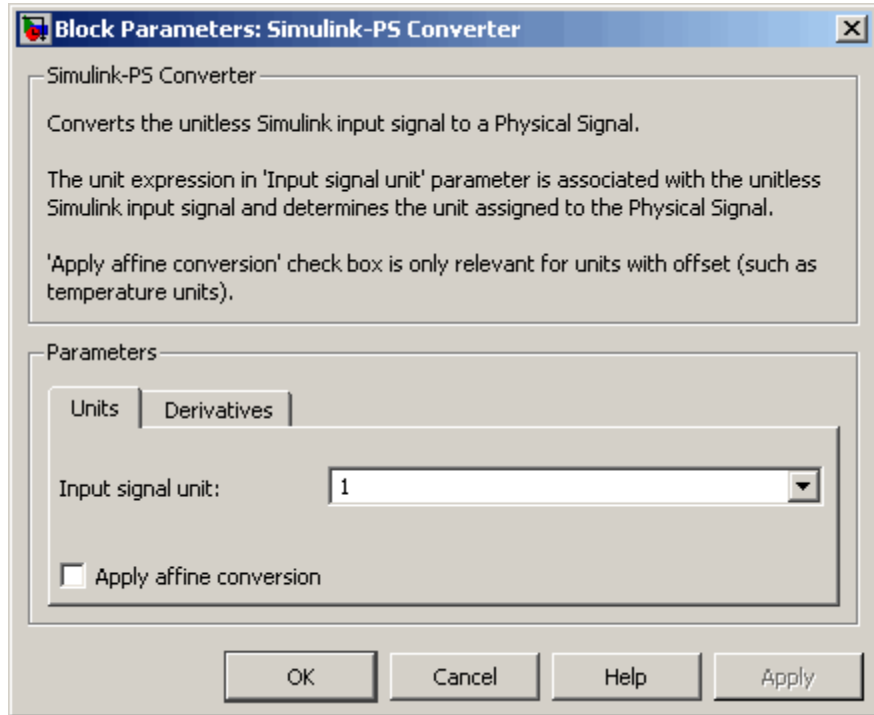


Dialog Box and Parameters

The block dialog box contains two tabs:

- “Units” on page 2-189
- “Derivatives” on page 2-190

Units



Input signal unit

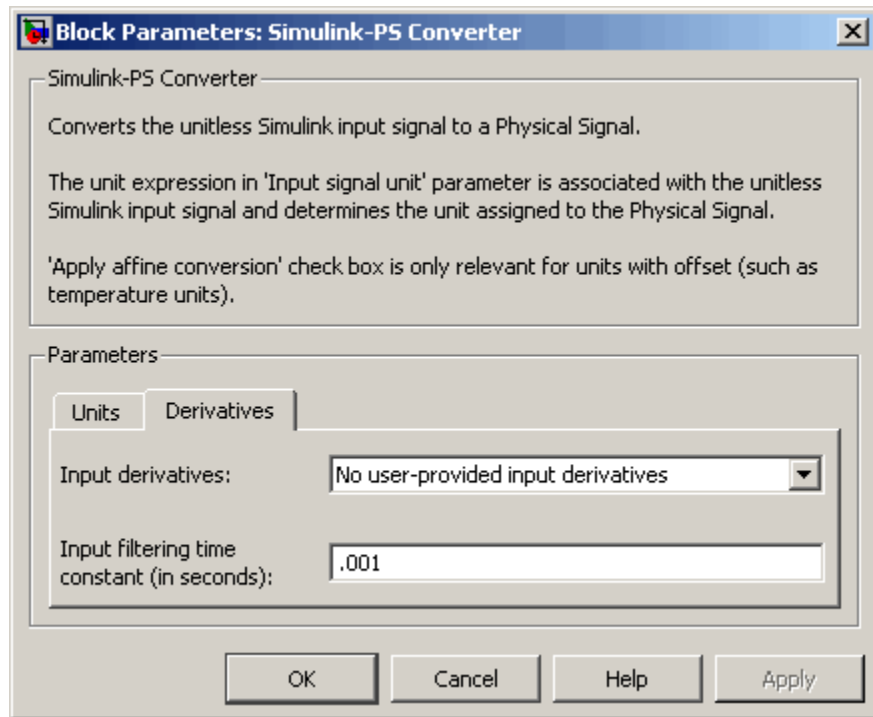
Units to be assigned to the physical signal. These units must be commensurate with the expected default units of the destination block input. You can select a unit from the drop-down list, or type the desired unit name, such as rpm, or a valid expression, such as rad/s. For more information and a list of unit abbreviations, see “Working with Physical Units”. 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.

Simulink-PS Converter

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”.

Derivatives



Input derivatives

This parameter is applicable only when you use an explicit solver for your model. You can select between two ways of providing time derivatives of the input signals:

- No user-input provided derivatives — Provide input derivatives by filtering the input through a low-pass filter.

The derivative of the filtered input is then computed by the simulation engine. This is the default method. If you use it, set the appropriate **Input filtering time constant** parameter value, as described below.

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

- **First derivative of input user-provided** — Provide first derivative of the input signal as an additional input signal to the Simulink-PS Converter block. If you select this option, input filtering is turned off and an additional Simulink input port appears on the Simulink-PS Converter block, to let you connect the signal providing input derivatives.

Input filtering time constant (in seconds)

This parameter is applicable only if the **Input derivatives** parameter is set to **No user-input provided derivatives**. 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 **Input derivatives** parameter has been set to **No user-input provided**

Simulink-PS Converter

derivatives prior to entering Restricted mode, you can change the value of the **Input filtering time constant** parameter.

Ports

The block has one or two Simulink input ports, depending on the **Input derivatives** 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

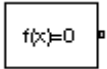
PS-Simulink Converter

Purpose

Represent Physical Networks environment and solver configuration

Library

Utilities

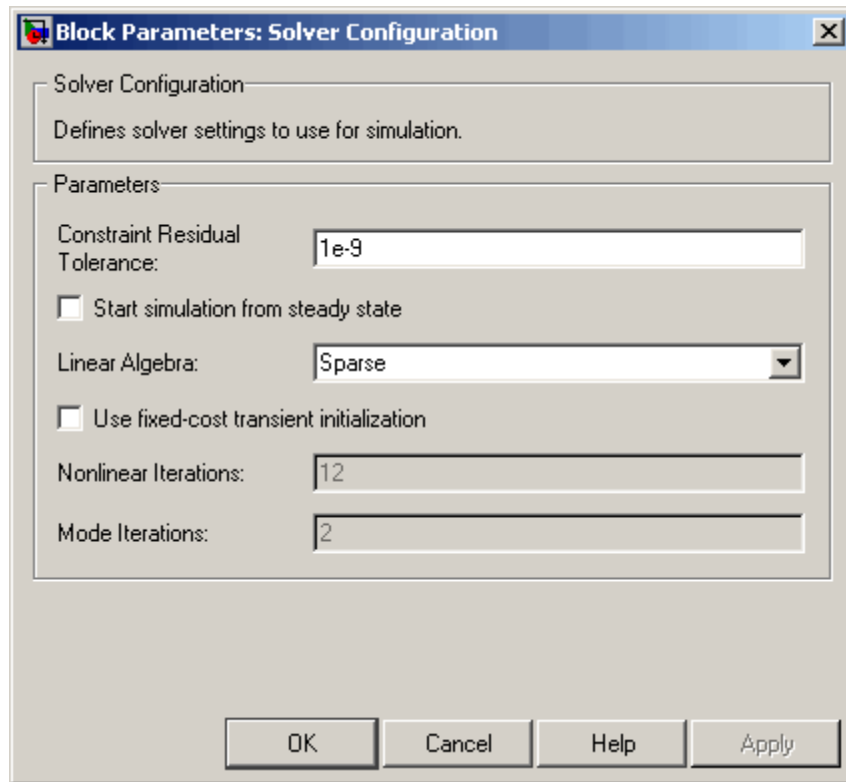
Description

Each physical device represented by a connected Simscape block diagram requires global environment information for simulation. The Solver Configuration block specifies this global information and provides parameters for the solver 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.

Solver Configuration

Dialog Box and Parameters



Constraint Residual 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 hitting the stop, and so on). Decreasing the parameter value (that is, tighter tolerance) results in 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.

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 “Computing Initial Conditions”. Simulation then starts from this steady state.

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.

Linear Algebra

Specifies how the solver treats matrices. The parameter can have one of two values: **Sparse** or **Full**. **Sparse** provides faster results. **Full** is used for code generation. The default value of the parameter is **Sparse**.

Use fixed-cost transient initialization

Lets you perform transient initialization at a fixed computational cost for real-time simulation. If you select this check box, you can specify the maximum number of nonlinear and mode iterations for 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 uses a more robust and time-consuming algorithm, and errors out if it fails to reach convergence at the time of transient initialization.

Nonlinear Iterations

Specify the maximum number of Newton iterations at the time of transient initialization. The **Use fixed-cost transient initialization** check box must be selected. The default number is 12.

Solver Configuration

Mode Iterations

Specify the maximum number of mode iterations at the time of transient initialization. The **Use fixed-cost transient initialization** check box must be selected. The default number is 2.

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.

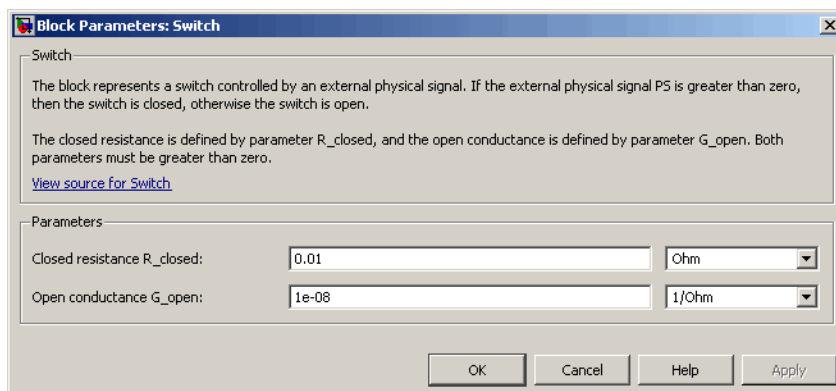
Purpose Simulate switch controlled by external physical signal

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 zero, then the switch is closed, otherwise the switch is open.



Dialog Box and 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 0.01 Ω .

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 1/ Ω .

Ports The block has two electrical conserving ports and one physical signal port PS.

Thermal Mass

Purpose Simulate 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:

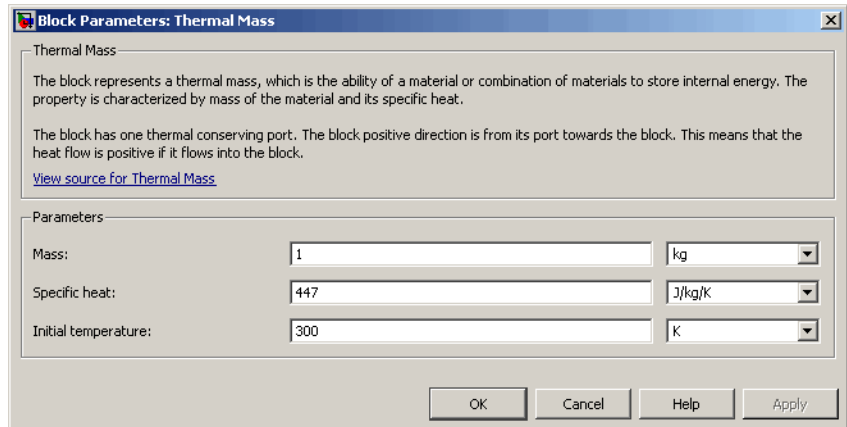
$$Q = c \cdot m \frac{dT}{dt}$$

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.

Dialog Box and Parameters



Mass

Mass. The default value is 1 kg.

Specific heat

Specific heat of the material. The default value is 447 J/kg/K.

Initial temperature

Initial temperature of the mass. 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 "Computing Initial Conditions". The default value is 300 K.

Ports

The block has one thermal conserving port, associated with the mass connection to the system.

See Also

Mass

Thermal Reference

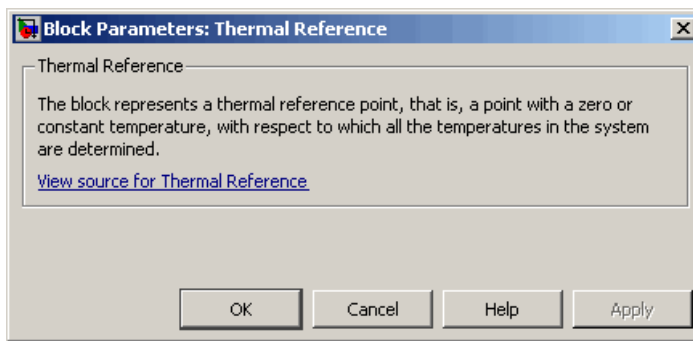
Purpose Simulate reference for thermal ports

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.



Dialog Box and Parameters



The Thermal Reference block has no parameters.

Ports The block has one thermal conserving port.

See Also

- Electrical Reference
- Hydraulic Reference
- Mechanical Rotational Reference
- Mechanical Translational Reference

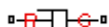
Purpose

Simulate 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

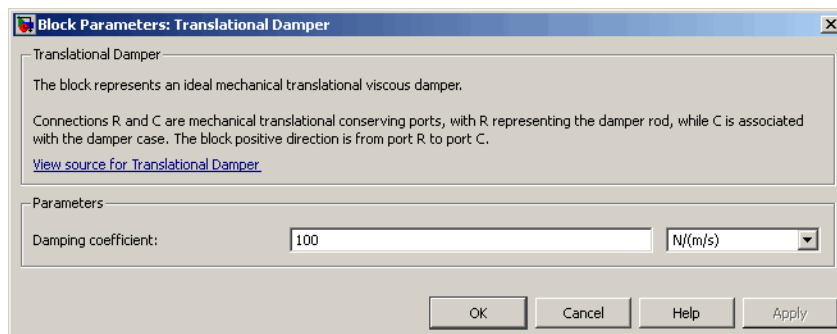
D Damping (viscous friction) coefficient

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.

Dialog Box and Parameters



Damping coefficient

Damping coefficient, defined by viscose friction. The default value is 100 N/(m/s).

Translational Damper

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.

See Also

Translational Friction

Translational Hard Stop

Translational Spring

Translational Electromechanical Converter

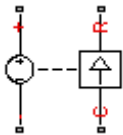
Purpose

Provide 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

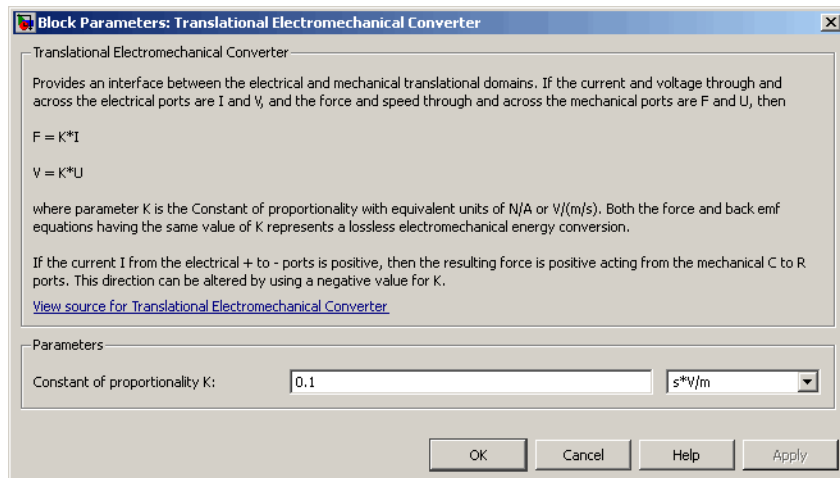
V	Voltage across the electrical ports of the converter
I	Current through the electrical ports of the converter
F	Force
U	Speed
K	Constant of proportionality

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.

Translational Electromechanical Converter

Dialog Box and 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.

See Also

Rotational Electromechanical Converter

Purpose

Simulate friction in contact between moving bodies

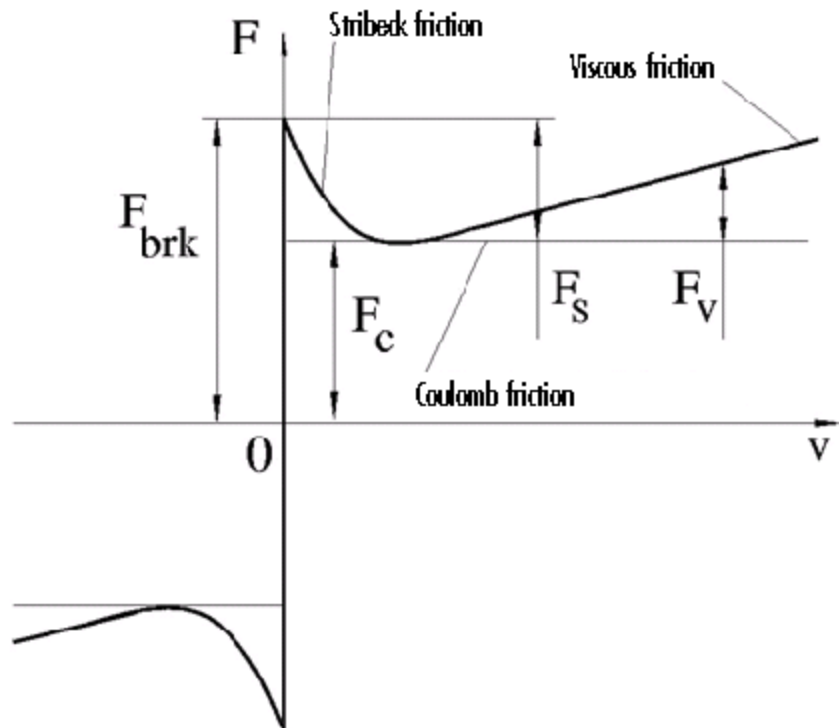
Library

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]). The Coulomb friction, F_C , results in a

Translational Friction

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:

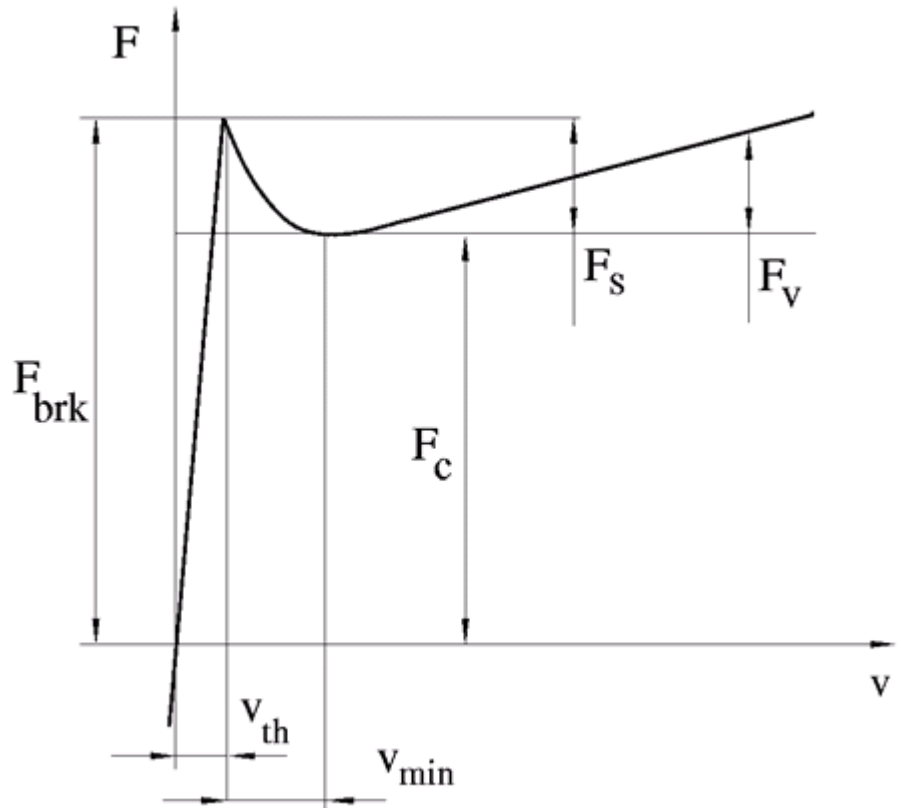
$$F = (F_C + (F_{brk} - F_C) \cdot \exp(-c_v |v|)) \text{sign}(v) + fv$$

$$v = v_R - v_C$$

where

F	Friction force
F_C	Coulomb friction
F_{brk}	Breakaway friction
c_v	Coefficient
v	Relative velocity
v_R, v_C	Absolute velocities of terminals R and C, respectively
f	Viscous friction coefficient

The approximation above is too idealistic and has a substantial drawback. The characteristic is discontinuous at $v = 0$, which creates considerable computational problems. It has been proven that the discontinuous friction model is a nonphysical simplification in the sense that the mechanical contact with distributed mass and compliance cannot exhibit an instantaneous change in force (see [1]). There are numerous models of friction without discontinuity. The Translational Friction block implements one of the simplest versions of continuous friction models. The friction force-relative velocity characteristic of this approximation is shown in the following figure.



The discontinuity is eliminated by introducing a very small, but finite, region in the zero velocity vicinity, within which friction force is assumed to be linearly proportional to velocity, with the proportionality coefficient F_{brk}/v_{th} , where v_{th} is the velocity threshold. It has been proven experimentally that the velocity threshold in the range between 10^{-4} and 10^{-6} m/s is a good compromise between the accuracy and computational robustness and effectiveness. Notice that friction force computed with this approximation does not actually stop relative

Translational Friction

motion when an acting force drops below breakaway friction level. The bodies will creep relative to each other at a very small velocity proportional to acting force.

As a result of introducing the velocity threshold, the block equations are slightly modified:

- If $|v| \geq v_{th}$,

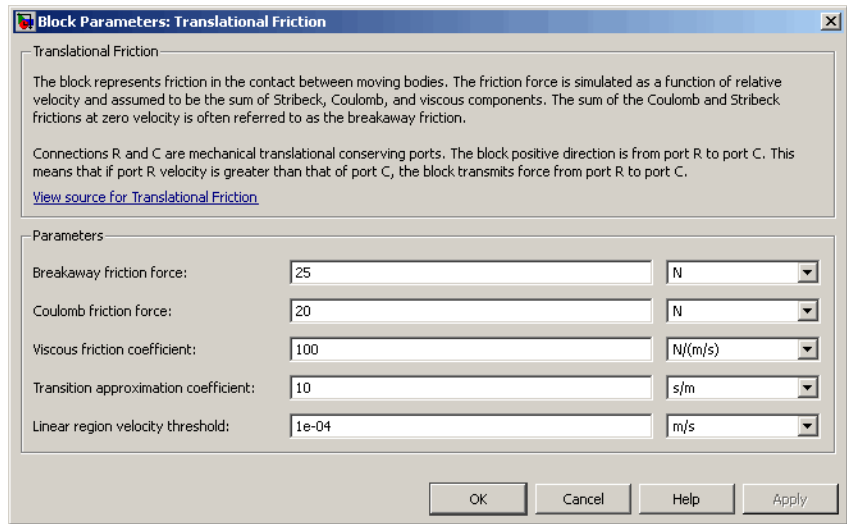
$$F = (F_C + (F_{brk} - F_C) \cdot \exp(-c_v |v|)) \text{sign}(v) + fv$$

- If $|v| < v_{th}$,

$$F = v \frac{(fv_{th} + (F_C + (F_{brk} - F_C) \cdot \exp(-c_v v_{th})))}{v_{th}}$$

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.

Dialog Box and Parameters



Breakaway friction force

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.

Coulomb friction force

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).

Transition approximation coefficient

The parameter sets the value of coefficient c_v , which is used for the approximation of the transition between the static and the Coulomb frictions. Its value is assigned based on the following considerations: the static friction component reaches approximately 95% of its steady-state value at velocity $3/c_v$, and 98% at velocity $4/c_v$, which makes it possible to develop an

Translational Friction

approximate relationship $c_v \approx 4/v_{min}$, where v_{min} is the relative velocity at which friction force has its minimum value. By default, c_v is set to 10 s/m, which corresponds to a minimum friction at velocity of about 0.4 m/s.

Linear region velocity threshold

The parameter sets the small vicinity near zero velocity, within which friction force is considered to be linearly proportional to the relative velocity. The MathWorks recommends that you use values in the range between $1e-6$ and $1e-4$ m/s. The default value is $1e-4$ 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

See Also

Translational Damper

Translational Hard Stop

Translational Spring

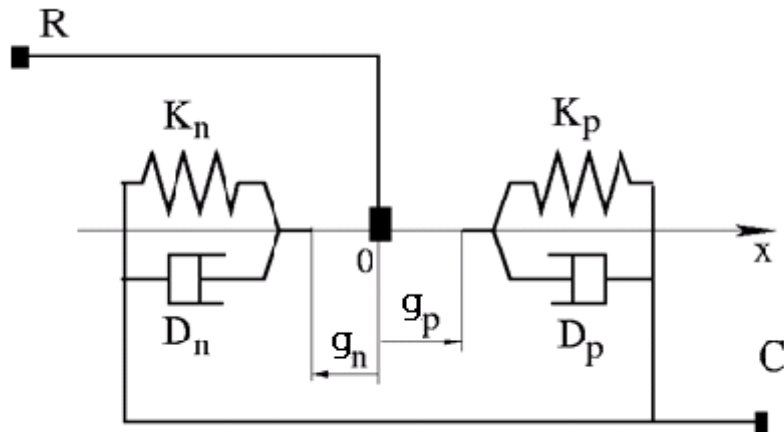
Purpose Simulate double-sided translational hard stop

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. Both ports of the block are of mechanical translational type. 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, the damping is introduced as the block's parameter, thus making it possible to account for energy loss. The following schematic shows the idealization of the mechanical translational hard stop adopted in the block:



The hard stop is described with the following equations:

Translational Hard Stop

$$F = \begin{cases} K_p \cdot \delta + D_p (v_R - v_C) & \text{for } \delta \geq g_p \\ 0 & \text{for } g_n < \delta < g_p \\ K_n \cdot \delta + D_n (v_R - v_C) & \text{for } \delta \leq -g_n \end{cases}$$

$$\delta = x_R - x_C$$

$$v_R = \frac{dx_R}{dt}$$

$$v_C = \frac{dx_C}{dt}$$

where

F	Interaction force between the slider and the case
δ	Relative displacement between the slider and the case
g_p	Gap between the slider and the case in positive direction
g_n	Gap between the slider and the case in negative direction
v_R, v_C	Absolute velocities of terminals R and C, respectively
x_R, x_C	Absolute displacements of terminals R and C, respectively
K_p	Contact stiffness at positive restriction
K_n	Contact stiffness at negative restriction
D_p	Damping coefficient at positive restriction
D_n	Damping coefficient at negative restriction
t	Time

The equations are derived with respect to the local coordinate system whose axis is directed from port R to port C. The terms “positive” and “negative” in the variable descriptions refer to this coordinate system, and the gap in negative direction must be specified with negative value.

If the local coordinate system is not aligned with the globally assigned positive direction, the gaps interchange their values with respective sign adjustment.

The block is oriented from R to C. This means that the block transmits force from port R to port C when the gap in positive direction is cleared up.

Dialog Box and Parameters

Block Parameters: Translational Hard Stop

Translational Hard Stop

The block represents a double-sided mechanical translational hard stop that restricts motion of a body between upper and lower bounds. The stop is implemented as a spring that comes into contact with the slider as the gap is cleared. To account for energy dissipation and non-elastic effects, the damping is introduced as the block parameter, thus making it possible to account for energy loss.

Connections R and C are mechanical translational conserving ports. 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.

[View source for Translational Hard Stop](#)

Parameters

Upper bound:	<input type="text" value="0.1"/>	<input type="text" value="m"/>
Lower bound:	<input type="text" value="-0.1"/>	<input type="text" value="m"/>
Contact stiffness at upper bound:	<input type="text" value="1e+06"/>	<input type="text" value="N/m"/>
Contact stiffness at lower bound:	<input type="text" value="1e+06"/>	<input type="text" value="N/m"/>
Contact damping at upper bound:	<input type="text" value="150"/>	<input type="text" value="N/(m/s)"/>
Contact damping at lower bound:	<input type="text" value="150"/>	<input type="text" value="N/(m/s)"/>

OK Cancel Help Apply

Upper bound

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. The default value is 0.005 m.

Translational Hard Stop

Lower bound

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. The default value is -0.005 m.

Contact stiffness at upper bound

The 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. The default value is 10^6 N/m.

Contact stiffness at lower bound

The parameter specifies the elastic property of colliding bodies when the slider hits the lower 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. The default value is 10^6 N/m.

Contact damping at upper bound

The parameter specifies dissipating property of colliding bodies when the slider hits the upper bound. At zero damping, the impact is close to an absolutely elastic one. The greater the value of the parameter, the more energy dissipates during an interaction. Keep in mind that damping affects slider motion as long as the slider is in contact with the stop, including the period when slider is pulled back from the contact. For computational efficiency and convergence reasons, The MathWorks recommends that you assign a nonzero value to this parameter. The default value is 150 N*s/m.

Contact damping at lower bound

The parameter specifies dissipating property of colliding bodies when the slider hits the lower bound. At zero damping, the impact

is close to an absolutely elastic one. The greater the value of the parameter, the more energy dissipates during an interaction. Keep in mind that damping affects slider motion as long as the slider is in contact with the stop, including the period when slider is pulled back from the contact. For computational efficiency and convergence reasons, The MathWorks recommends that you assign a nonzero value to this parameter. The default value is 150 N*s/m.

Ports

The block has the following ports:

R

Mechanical translational conserving port associated with the slider that travels between stops installed on the case.

C

Mechanical translational conserving port associated with the case.

Examples

The Mechanical System with Translational Hard Stop demo (`ssc_mechanical_system_translational_hardstop`) illustrates the use of the Translational Hard Stop block in mechanical systems. Two masses are interacting through a hard stop. The mass on the left is driven by an ideal velocity source. Plotting the displacement of the second mass against the displacement of the first mass produces a typical hysteresis curve.

See Also

Translational Damper

Translational Friction

Translational Spring

Translational Hydro-Mechanical Converter

Purpose Simulate single chamber of hydraulic cylinder as building block for various cylinder models

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. Using this block as a basic element, you can build a large variety of hydraulic cylinder models by adding application-specific effects, such as fluid compressibility, leakage, friction, hard stops, and so on.



The converter is simulated according to the following equations:

$$q = A(v_R - v_C) \cdot or$$

$$F = A \cdot p \cdot or$$

where

- | | |
|-------|---|
| q | Flow rate due to fluid compressibility |
| 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 |
| or | Converter orientation with respect to the globally assigned positive direction. If pressure applied at port A exerts force in positive direction, or equals 1. If pressure applied at port A exerts force in negative direction, or equals -1 . |

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.

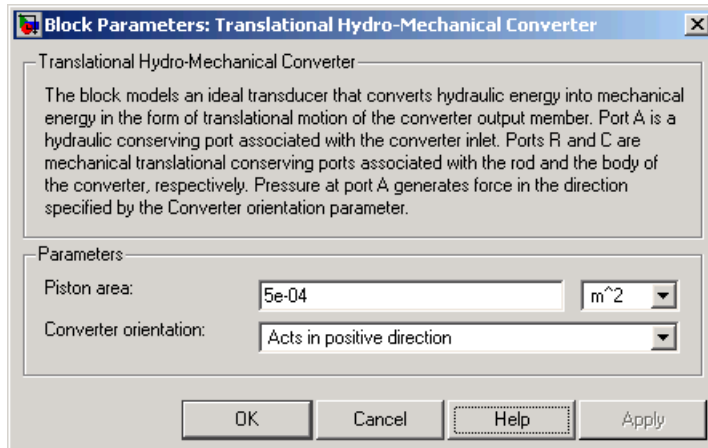
Translational Hydro-Mechanical Converter

Basic Assumptions and Limitations

The model is based on the following assumption:

- The block simulates an ideal converter, with only the transduction property considered. No inertia, friction, leakage, or other effects are taken into account.

Dialog Box and 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`.

Restricted Parameters

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

Translational Hydro-Mechanical Converter

- **Converter orientation**

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.

See Also

Rotational Hydro-Mechanical Converter

Purpose

Simulate 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 (spring offset)

x_R, x_C Absolute displacements of terminals R and C, respectively

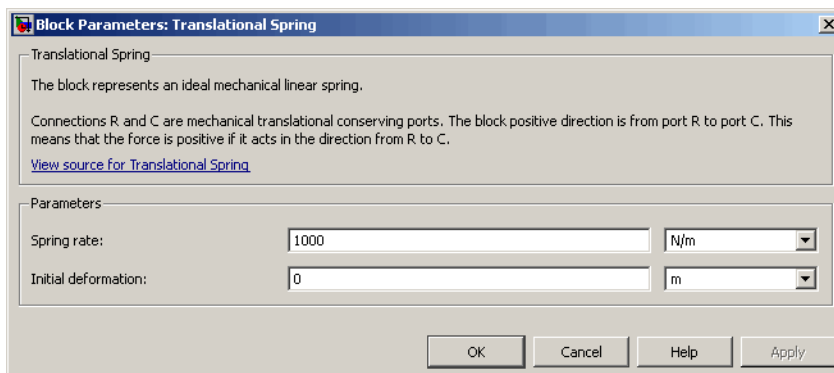
v Relative velocity

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.

Translational Spring

Dialog Box and Parameters



Spring rate

Spring rate. The default value is 1000 N/m.

Initial deformation

Spring initial deformation, or offset, in length units. The deformation is determined as $x = x_{init} + x_R - x_C$, where x_{init} is the initial deformation, and x_R , x_C are the absolute displacements of terminals R and C in the globally assigned positive direction. The spring can be initially compressed ($x_{init} > 0$) or stretched ($x_{init} < 0$). 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 "Computing Initial Conditions". The default value is 0.

Ports

The block has the following ports:

- R
Mechanical translational conserving port.
- C
Mechanical translational conserving port.

See Also

- Translational Damper
- Translational Friction

Translational Hard Stop

Two-Way Connection

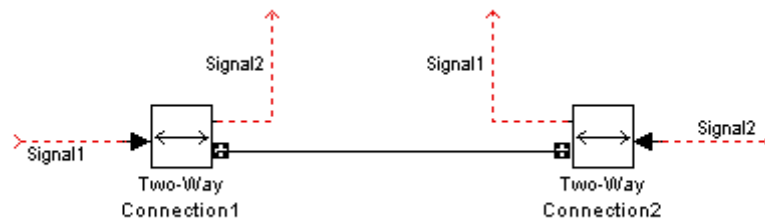
Purpose Create two-way connector port for subsystem


Library Utilities

Description



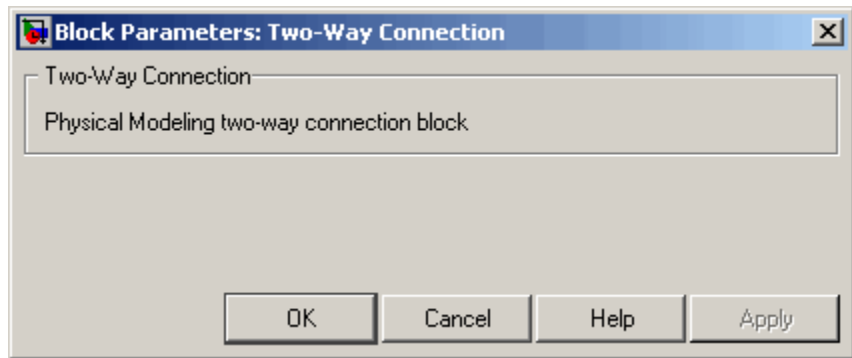
The Two-Way Connection block has a two-way connector port, which transports Simulink signals both ways. You connect this port to another two-way connector port. The schematic below illustrates how the two-way connection works. It carries the signal **Signal1** from the input port of the first Two-Way Connection block to the output port of the second Two-Way Connection block, and at the same time carries the signal **Signal2** from the input port of the second Two-Way Connection block to the output port of the first Two-Way Connection block.



The Two-Way Connection block supports invariant model architecture for top-down or bottom-up design. It lets you build subsystems as Simulink models, based on signals, and then connect them as if they are physical systems. Place the Two-Way Connection blocks inside the subsystem and connect them to the Connection Port blocks. Then the ports on the subsystem boundary appear as two-way connector ports .

Note Two-way connection blocks cannot be connected across nonvirtual subsystems.

Dialog Box and Parameters



The block has no parameters.

Ports

The block has a Simulink input port, a Simulink output port, and a two-way connector port.

See Also

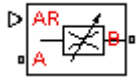
In the Using Simulink documentation, see “Working with Block Masks”.

Variable Area Orifice

Purpose Simulate hydraulic variable orifice created by cylindrical spool and sleeve

Library Hydraulic Elements

Description



The Variable Area 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 orifice area value is $1e-12 \text{ m}^2$. 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:

$$q = \begin{cases} C_D \cdot A \sqrt{\frac{2}{\rho} |p| \cdot \text{sign}(p)} & \text{for } Re \geq Re_{cr} \\ 2C_{DL} \cdot A \frac{D_H}{v \cdot \rho} p & \text{for } Re < Re_{cr} \end{cases}$$

$$p = p_A - p_B$$

$$Re = \frac{q \cdot D_H}{A \cdot v}$$

$$C_{DL} = \left(\frac{C_D}{\sqrt{Re_{cr}}} \right)^2$$

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

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, provided through the signal port
D_H	Orifice hydraulic diameter
ρ	Fluid density
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 differential is determined as $p = p_A - p_B$.

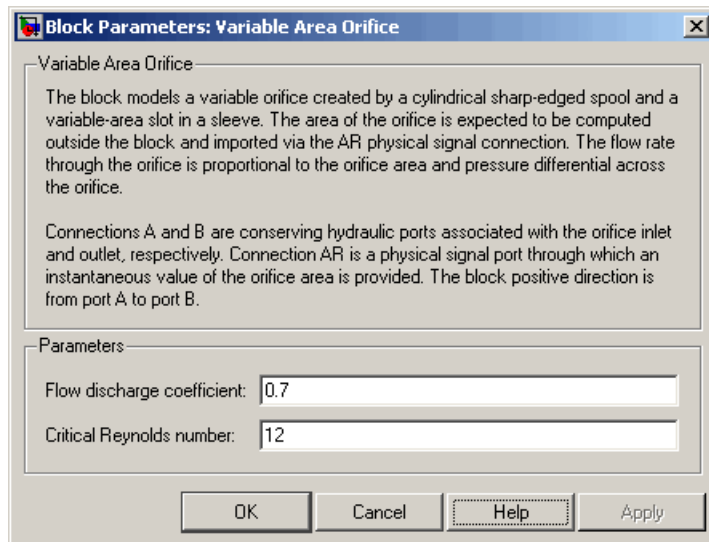
Basic Assumptions and Limitations

The model is based on the following assumptions:

- Fluid inertia is not taken into account.
- The transition between laminar and turbulent regimes is assumed to be sharp and taking place exactly at $Re=Re_{cr}$.

Variable Area Orifice

Dialog Box and 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.

Critical Reynolds number

The maximum Reynolds number for laminar flow. The transition from laminar to turbulent regime is supposed to take place when the Reynolds number reaches this value. The value of the parameter depends on orifice geometrical profile, and the recommendations on the parameter value can be found in hydraulic textbooks. The default value is 12, which corresponds to a round orifice in thin material with sharp edges.

Global Parameters

Fluid density

The parameter is determined by the type of working fluid selected for the system under design. Use the Custom Hydraulic Fluid

block, or the Hydraulic Fluid block available with SimHydraulics block libraries, to specify the fluid properties.

Fluid kinematic viscosity

The parameter is determined by the type of working fluid selected for the system under design. Use the Custom Hydraulic Fluid block, or the Hydraulic Fluid block available with SimHydraulics block libraries, 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.

See Also

Constant Area Orifice

Variable Chamber

Purpose Simulate hydraulic capacity of variable volume with compressible fluid

Library Hydraulic Elements

Description The Variable 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 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:

$$q = \frac{V_0 + V}{E} \cdot \frac{dp}{dt}$$
$$E = E_l \frac{1 + \alpha \left(\frac{p_a}{p_a + p} \right)^{1/n}}{1 + \alpha \frac{p_a^{1/n}}{n \cdot (p_a + p)^{n+1}} E_l}$$

where

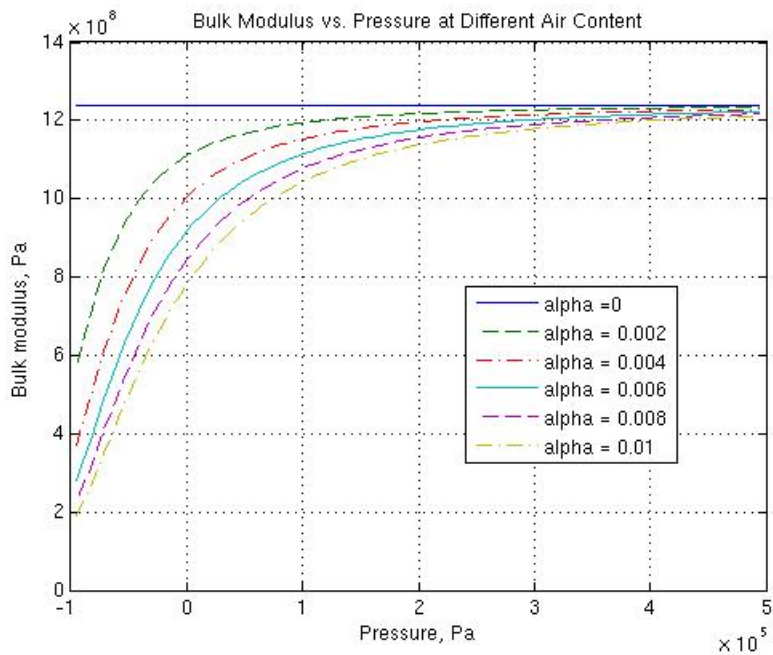
- q Flow rate due to fluid compressibility
- V_0 Initial volume of fluid in the chamber
- V Chamber volume change, provided through port V
- E Fluid bulk modulus

E_l	Pure liquid bulk modulus
p	Gauge pressure of fluid in the chamber
p_a	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 at $p \rightarrow p_a$, thus considerably slowing down further pressure change.

At high pressure, $p \gg p_a$, a small amount of nondissolved gas has practically no effect on the system behavior.

Variable 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.

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.

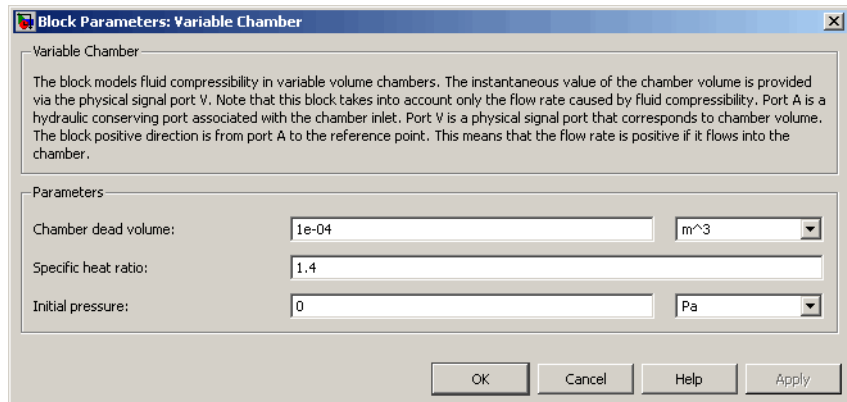
The block positive direction is from port A to the reference point. This means that the flow rate is positive if it flows into the chamber.

Basic Assumptions and Limitations

The model is based on the following assumptions:

- Fluid density remains constant.
- Chamber volume can not be less than the dead volume.
- Fluid fills the entire chamber volume.

Dialog Box and 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.

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. For more information, see "Computing Initial Conditions". The default value is 0.

Restricted Parameters

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

Variable Chamber

- **Chamber orientation**

All other block parameters are available for modification.

Global Parameters

Fluid bulk modulus

The parameter is determined by the type of working fluid selected for the system under design. Use the Hydraulic Fluid block or the Custom Hydraulic Fluid block to specify the fluid properties.

Nondissolved gas ratio

Nondissolved gas relative content determined as a ratio of gas volume to the liquid volume. The parameter is determined by the type of working fluid selected for the system under design. 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.

See Also

Constant Volume Chamber

Piston Chamber

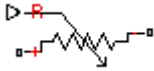
Purpose

Simulate 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

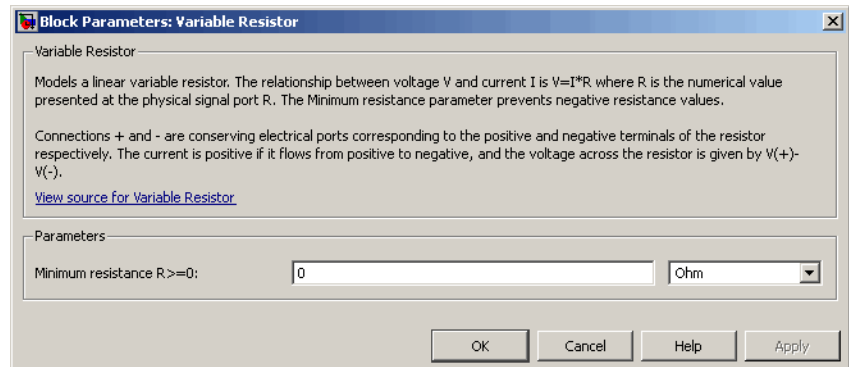
V Voltage

I Current

R Resistance, i.e., the value presented at the control port

Connections + and – are conserving electrical ports corresponding to the positive and negative terminals of the resistor, respectively. P 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, $V(+)-V(-)$.

Dialog Box and Parameters



Variable Resistor

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.

P

Physical signal input port that provides the resistance value.

See Also

Resistor

Purpose Simulate hydraulic capacity of variable volume with compressible fluid

Library None (kept for compatibility purposes only)

Description



Note 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. It is recommended that you replace this block with the Piston Chamber block.

The Variable Volume Chamber block models fluid compressibility in variable volume chambers, such as hydraulic cylinder cavities. 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 Volume 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:

$$q = \frac{V_0 + A \cdot x \cdot or}{E} \cdot \frac{dp}{dt}$$

Variable Volume Chamber

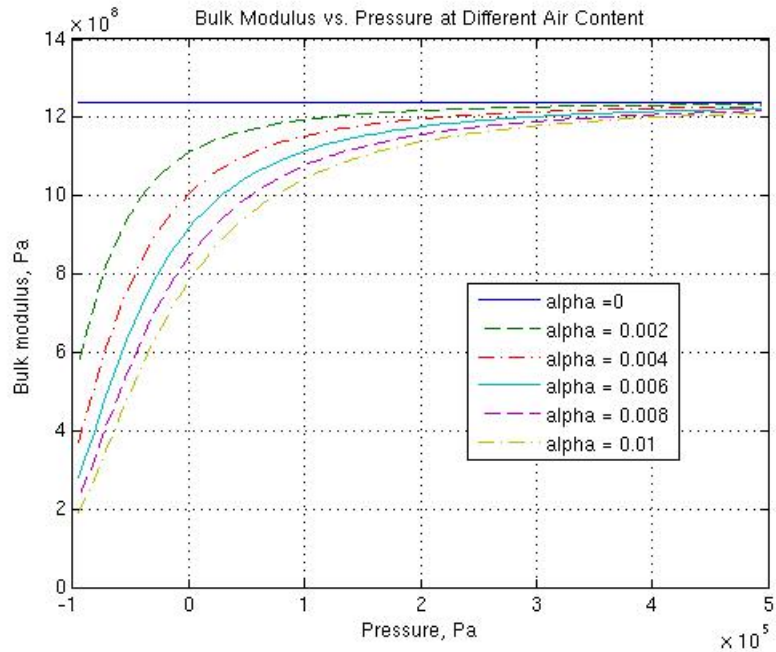
$$E = E_l \frac{1 + \alpha \left(\frac{p_a}{p_a + p} \right)^{1/n}}{1 + \alpha \frac{p_a^{1/n}}{n \cdot (p_a + p)^{\frac{n+1}{n}}} E_l}$$

where

- q Flow rate due to fluid compressibility
- V_0 Initial volume of fluid in the chamber
- A Effective piston area
- 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_a 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

at $p \rightarrow p_a$, thus considerably slowing down further pressure change. At high pressure, $p \gg p_a$, a small amount of nondissolved 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 it is known that cavitation is unlikely in the system under design, you can set the relative gas content in the fluid properties to zero, thus increasing the speed of computations.

Variable Volume Chamber

Basic Assumptions and Limitations

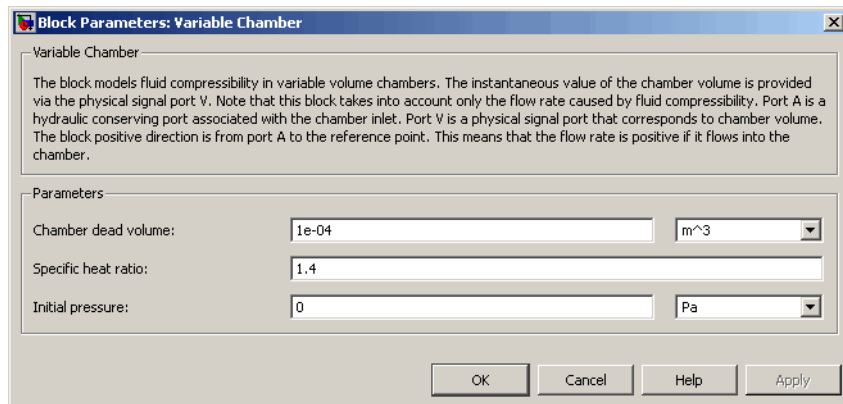
Dialog Box and Parameters

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.

The model is based on the following assumptions:

- Fluid density remains constant.
- Fluid fills the entire chamber volume.



Piston area

Effective piston area. The default value is $5e-4 \text{ m}^2$.

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 `Decreases at positive`. The default value is `Increases at positive`.

Chamber dead volume

Volume of fluid in the chamber at initial 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. For more information, see "Computing Initial Conditions". The default value is 0.

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

Fluid bulk modulus

The parameter is determined by the type of working fluid selected for the system under design. Use the Hydraulic Fluid block or the Custom Hydraulic Fluid block to specify the fluid properties.

Nondissolved gas ratio

Nondissolved gas relative content determined as a ratio of gas volume to the liquid volume. The parameter is determined by the type of working fluid selected for the system under design. 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.

Variable Volume Chamber

P

Physical signal port that controls piston displacement.

See Also

Constant Volume Chamber

Piston Chamber

Translational Hydro-Mechanical Converter

Variable Chamber

Voltage-Controlled Current Source

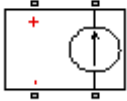
Purpose

Simulate 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

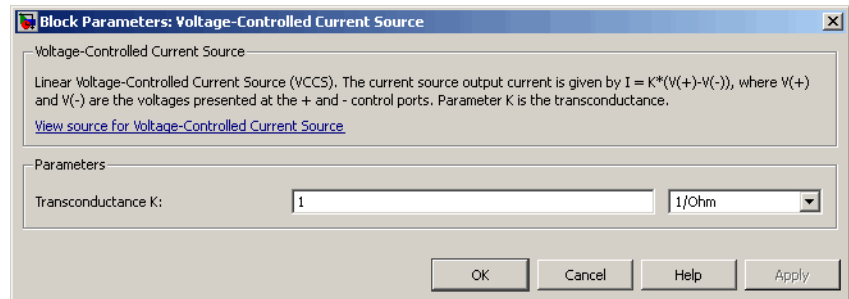
I Current

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.

Dialog Box and 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.

Voltage-Controlled Current Source

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.

See Also

Current-Controlled Current Source

Current-Controlled Voltage Source

Voltage-Controlled Voltage Source

Voltage-Controlled Voltage Source

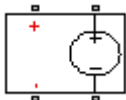
Purpose

Simulate 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

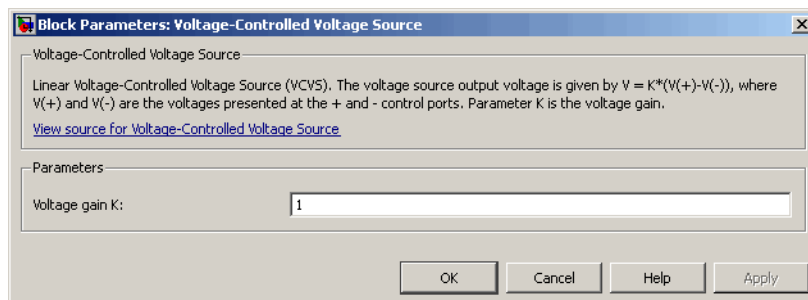
V Output voltage

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.

Dialog Box and 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.

Voltage-Controlled Voltage Source

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.

See Also

Current-Controlled Current Source

Current-Controlled Voltage Source

Voltage-Controlled Current Source

Purpose Simulate voltage sensor in electrical systems

Library Electrical Sensors

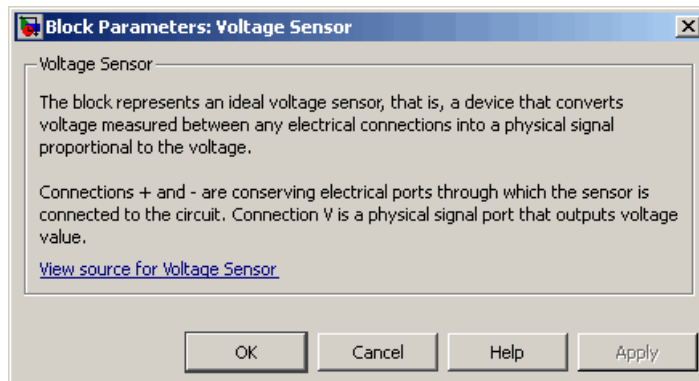
Description



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.

Dialog Box and Parameters



The block has no parameters.

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.

Voltage Sensor

V
Physical signal output port for voltage.

See Also Current Sensor

Purpose

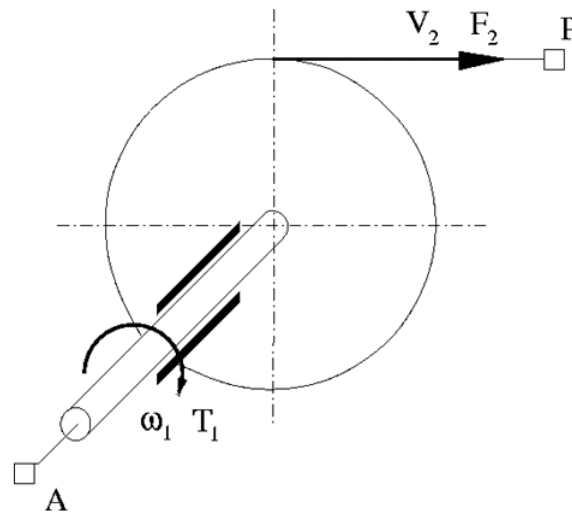
Simulate wheel and axle mechanism in mechanical systems

Library

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:

$$T = r \cdot F \cdot \text{or}$$

$$v = r \cdot \omega \cdot \text{or}$$

Wheel and Axle

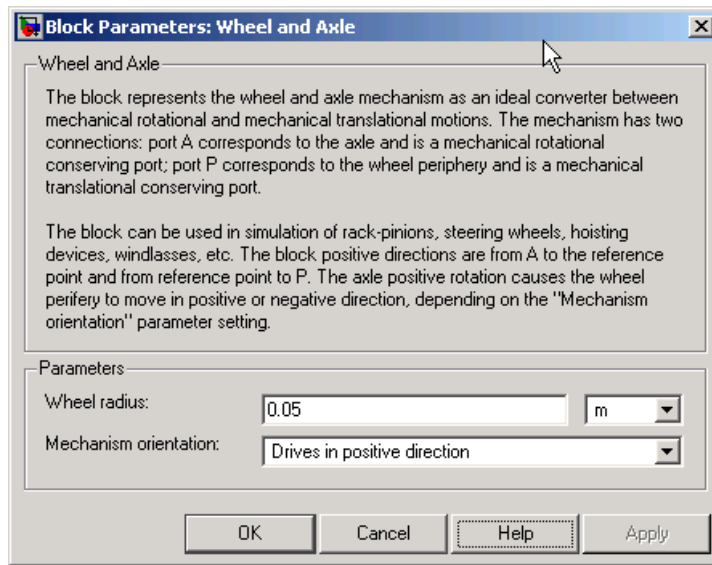
where

T	Torque on the axle
F	Force on the wheel periphery
ω	Angular velocity
v	Linear velocity on the wheel periphery
r	Wheel radius
or	Mechanism orientation indicator. The variable assumes +1 value if axle rotation in the globally assigned positive direction is converted into translational motion in positive direction, and -1 if positive rotation results in translational motion in negative direction.

The block can be used 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.

Dialog Box and Parameters



Wheel radius

Radius of the wheel. The default value is 0.05 m.

Mechanism orientation

The parameter can be set to one of two options: **Drives in positive direction** or **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. The default value is **Drives in positive direction**.

Restricted Parameters

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

Wheel and Axle

- **Mechanism orientation**

All other block parameters are available for modification.

Ports

The block has the following ports:

A

Mechanical rotational conserving port associated with the axle.

P

Mechanical translational conserving port associated with the wheel periphery.

Examples

The Simple Mechanical System demo (`ssc_simple_mechanical_system`) illustrates the use of the Wheel and Axle block in mechanical systems.

See Also

Gear Box

Command Reference

<code>pm_adddimension</code>	Add new dimension to unit registry
<code>pm_addunit</code>	Add new unit to unit registry
<code>pm_getdimensions</code>	Get information about all dimensions in unit registry
<code>pm_getunits</code>	Get information about all units in unit registry
<code>ssc_build</code>	Build custom library from collection of Simscape files
<code>ssc_clean</code>	Clean all derived files generated by library build process
<code>ssc_mirror</code>	Create protected mirror of library of Simscape files
<code>ssc_new</code>	Create new Simscape model populated by required and commonly-used blocks
<code>ssc_protect</code>	Generate Simscape protected files from source files
<code>ssc_reserved</code>	List all reserved Simscape language words

pm_adddimension

Purpose Add new dimension to unit registry

Syntax `pm_adddimension(dimension, unitname)`

Description `pm_adddimension(dimension, unitname)` adds a new dimension named `dimension` with a fundamental unit, `unitname`. `dimension` may be any string. `unitname` must be a valid unit name, that is, it must begin with a letter and contain only letters and numbers.

Examples Add a new unit dimension, `length`, with a fundamental unit of meter, `m`:

```
pm_adddimension('length', 'm');
```

See Also `pm_addunit`, `pm_getdimensions`, `pm_getunits`

Purpose 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');
```

pm_addunit

Add a new unit Fahrenheit, Fh, in terms of Celsius:

```
pm_addunit('Fh', [5/9 -32*5/9], 'C');
```

See Also

pm_adddimension, pm_getdimensions, pm_getunits

Purpose 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

pm_getunits

Purpose 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 =
```

```
'm'  
'kg'  
's'  
'c'  
'K'  
'mol'  
'cm'  
'mm'  
'km'  
'um'  
'C'  
'Fh'  
'R'  
'in'  
'ft'  
'mi'  
'yd'  
'l'  
'gal'  
'g'  
'mg'
```


'lbm'
'oz'
'slug'
'N'
'lbf'
'dyn'
'lb'
'min'
'hr'
'rad'
'deg'
'rev'
'mph'
'fpm'
'fps'
'rpm'
'Hz'
'kHz'
'MHz'
'GHz'
'J'
'Btu'
'eV'
'W'
'HP'
'V'
'A'
'F'
'H'
'Ohm'
'S'
'Wb'
'mV'
'kV'
'pA'
'nA'
'uA'

pm_getunits

'mA'
'kA'
'pF'
'nF'
'uF'
'uH'
'mH'
'kOhm'
'MOhm'
'GOhm'
'nS'
'uS'
'mS'
'Pa'
'bar'
'psi'
'atm'
'lpm'
'gpm'
'Poise'
'cP'
'reyn'
'St'
'cSt'
'Newt'

See Also

pm_adddimension, pm_addunit, pm_getdimensions

Purpose Build custom library from collection of Simscape files

Syntax `ssc_build package`

Description `ssc_build package` generates a custom Simscape library file, named `package_lib.mdl`, containing 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 out of date, `ssc_build` will use the protected files to build the library, but you will get a warning.

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_build` must omit the + character.

The package must be located in a directory on the MATLAB path. The `package_lib.mdl` is automatically placed in the package parent directory. For more information, see “Adding Custom Block Libraries Generated from Simscape Component Files”.

If you run the `ssc_build` command from inside the package directory structure, you can omit the argument.

Examples For example, your top-level package directory, where you store your Simscape files, is named `+SimscapeCustomBlocks`. To generate a custom block library, at the MATLAB Command prompt, type:

```
ssc_build SimscapeCustomBlocks;
```

This command generates a file called `SimscapeCustomBlocks_lib.mdl` in the parent directory of the top-level package (that is, in the same directory that contains your `+SimscapeCustomBlocks` package).

ssc_build

See Also

ssc_clean

ssc_mirror

ssc_protect

Purpose 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.

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`

ssc_mirror

Purpose 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.mdl*, 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.mdl* 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.mdl in the C:\Work\deploy directory.

See Also

ssc_clean

ssc_mirror

ssc_protect

Purpose Create new Simscape model populated by required and commonly-used blocks

Syntax

```
ssc_new
ssc_new('modelName')
ssc_new('modelName','domain')
ssc_new('modelName','domain','solver')
```

Description `ssc_new` creates a new Simscape model, with required and commonly-used blocks already on the model canvas, and opens the Simscape library. By default, it uses the Simulink default new model name `untitled` and the recommended solver `ode15s`.

`ssc_new('modelName')` creates a new Simscape model with the specified name.

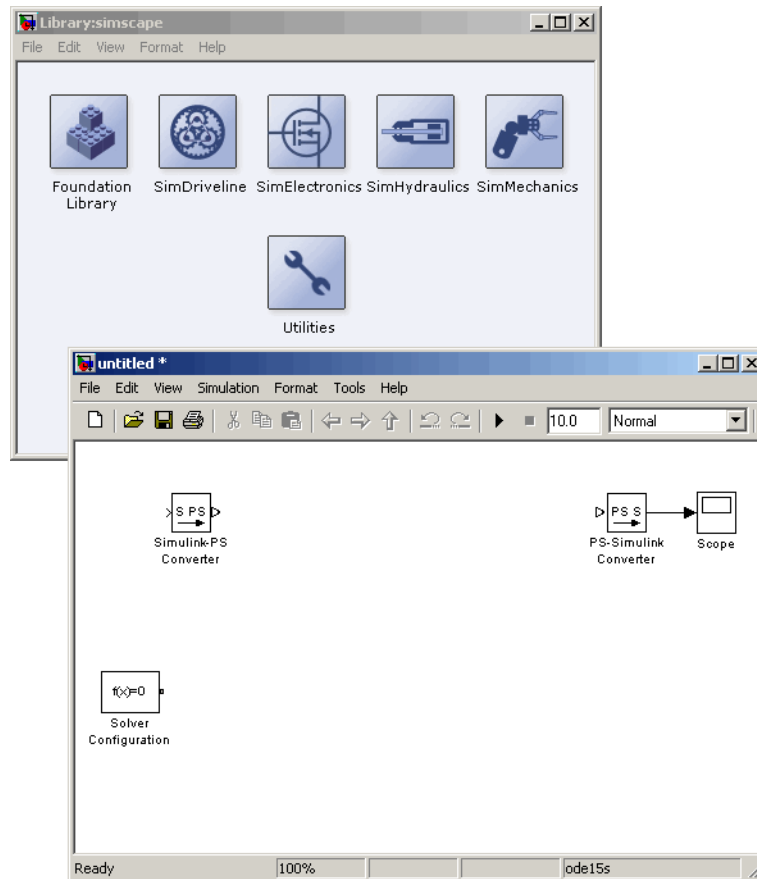
`ssc_new('modelName','domain')` creates a new Simscape model with the specified name, and with domain-specific reference block added to the model canvas. Valid domains types are `'electrical'`, `'hydraulic'`, `'rotational'`, `'translational'`, and `'thermal'`. You can use a cell array of domain types to add more than one type of reference block.

`ssc_new('modelName','domain','solver')` creates a new Simscape model with the specified name and domain type, and with the specified solver type. Recommended solver types for Simscape models are `'ode15s'`, `'ode23t'`, and `'ode14x'`. You can use other Simulink solvers, but, depending on the particular model, they may be less suitable. For more information, see “Working with Solvers”.

Examples To create a generic Simscape model, type:

```
ssc_new
```

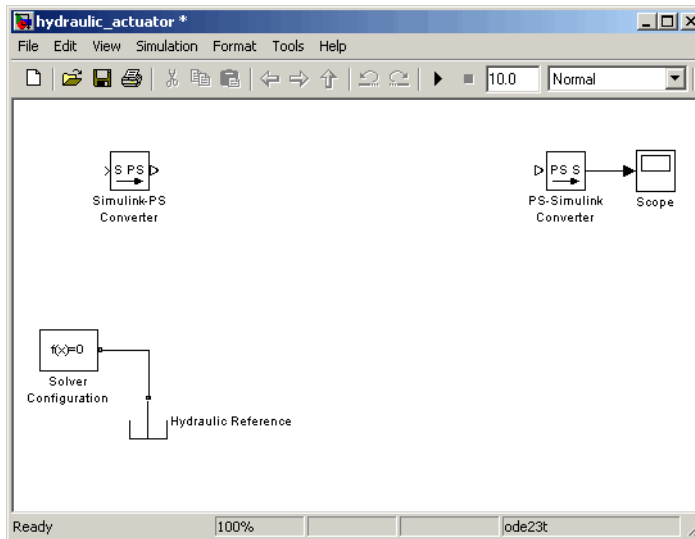
The software opens the main Simscape library and creates a new untitled model, which contains a Solver Configuration block with the default solver set to `ode15s`, a Simulink-PS Converter block, and a PS-Simulink Converter block connected to a Scope block.



To create a hydraulic model, called `hydraulic_actuator.mdl` and using the `ode23t` solver, type:

```
ssc_new('hydraulic_actuator', 'hydraulic', 'ode23t')
```

The software opens the main Simscape library and 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.

See Also

“Creating a New Simscape Model”

Purpose

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). 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 -i
```

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

ssc_reserved

Purpose List all reserved Simscape language words

Syntax `ssc_reserved`
`words = ssc_reserved`

Description Simscape language has certain words, in addition to its keywords, that cannot be used as model or member names. This list may change from release to release, as limitations are removed. Use the `ssc_reserved` command to see the current list of reserved words.

`ssc_reserved` returns a list of reserved Simscape language words.

`words = ssc_reserved` returns a list of reserved words in `words` as a cell array of strings.

Note `ssc_reserved` does not list the Simscape language keywords.

Examples List the currently reserved words, other than keywords, that cannot be used as Simscape language model or member names:

```
ssc_reserved

ans =

    'across_variable'
    'build'
    'description'
    'descriptor'
    'element'
    'input'
    'interface_input'
    'interface_node'
    'interface_output'
    'item_type'
    'local_variable'
    'name'
```

```
'node'  
'output'  
'parameter'  
'setup'  
'signal'  
'source'  
'terminal'  
'through_variable'  
'variable'
```

You cannot use any of these words as model names (domain or component) or member names (parameter, variable, and so on).

ssc_reserved

Language Reference

across	Establish relationship between component variables and nodes
component	Component model keywords
der	Return time derivative of operand
domain	Domain model keywords
equations	Define component equations
inputs	Define component inputs, that is, Physical Signal input ports of block
nodes	Define component nodes, that is, conserving ports of block
outputs	Define component outputs, that is, Physical Signal output ports of block
parameters	Specify component parameters
setup	Prepare component for simulation
through	Establish relationship between component variables and nodes
time	Access global simulation time
value	Convert variable or parameter to unitless value with specified unit conversion
variables	Define domain or component variables

Purpose Establish relationship between component variables and nodes

Syntax `across(variable1, node1.variableA, node2.variableB)`

Description `across(variable1, node1.variableA, node2.variableB)` establishes the following relationship between the three arguments: `variable1` is assigned the value `(node1.variableA - node2.variableB)`. All arguments are variables. The first one is not associated with a node. The second and third must be associated with a node.

The following rules apply:

- All arguments must have consistent units.
- The second and third arguments do not need to be associated with the same domain. For example, one may be associated with a one-phase electrical domain, and the other with a 3-phase electrical.
- Either the second or the third argument may be replaced with `[]` to indicate the reference node.

Examples If a component declaration section contains two electrical nodes, `p` and `n`, and a variable `v = { 0, 'V' }`; specifying voltage, you can establish the following relationship in the setup section:

```
across( v, p.v, n.v );
```

This defines voltage `v` as an Across variable from node `p` to node `n`.

See Also `through`

Purpose Component model keywords

Syntax component
nodes
inputs
outputs
parameters
variables
function setup
equations

Description `component` begins the component model class definition, which is terminated by an `end` keyword. Only blank lines and comments can precede `component`. You must place a component model class definition in a file of the same name with a file name extension of `.ssc`.

See “Basic Simscape Grammar” in the *Simscape Language Guide* for more information on component model definition syntax.

A component file consists of a declaration section, with one or more member declaration blocks, followed by setup and equation sections.

The declarations section may contain any of the following member declaration blocks.

`nodes` begins a nodes declaration block, which is terminated by an `end` keyword. This block contains declarations for all the component nodes, which correspond to the conserving ports of a Simscape block generated from the component file. Each node is defined by assignment to an existing domain. See “Declaring Component Nodes” in the *Simscape Language Guide* for more information.

`inputs` begins an inputs declaration block, which is terminated by an `end` keyword. This block contains declarations for all the inputs, which correspond to the input Physical Signal ports of a Simscape block generated from the component file. Each input is defined as a value with unit. See “Declaring Component Inputs and Outputs” in the *Simscape Language Guide* for more information.

`outputs` begins an outputs declaration block, which is terminated by an `end` keyword. This block contains declarations for all the outputs, which correspond to the output Physical Signal ports of a Simscape block generated from the component file. Each output is defined as a value with unit. See “Declaring Component Inputs and Outputs” in the *Simscape Language Guide* for more information.

`parameters` begins a component parameters definition block, which is terminated by an `end` keyword. This block contains declarations for component parameters. Parameters will appear in the block dialog box when the component file is brought into a block model. Each parameter is defined as a value with unit. See “Declaring Component Parameters” in the *Simscape Language Guide* for more information.

`variables` begins a variables declaration block, which is terminated by an `end` keyword. This block contains declarations for all the variables associated with the component. Variables are internal to the component; they will not appear in a block dialog box when the component file is brought into a block model.

Variables can be defined either by assignment to an existing domain variable or as a value with unit. See “Declaring Component Variables” in the *Simscape Language Guide* for more information.

`function setup` begins the setup section, which is terminated by an `end` keyword. This section relates inputs, outputs, and variables to one another by using `across` and `through` functions. It can also be used for validating parameters, computing derived parameters, and setting initial conditions. See “Defining Component Setup” in the *Simscape Language Guide* for more information.

`equations` begins the equation section, which is terminated by an `end` keyword. This section contains the equations that define how the component works. See “Defining Component Equations” in the *Simscape Language Guide* for more information.

Table of Attributes

For component model attributes, as well as declaration member attributes, see “Attribute Lists” in the *Simscape Language Guide*.

Examples

This file, named `spring.ssc`, defines a rotational spring.

```
component spring
  nodes
    r = foundation.mechanical.rotational.rotational;
    c = foundation.mechanical.rotational.rotational;
  end
  parameters
    k = { 10, 'N*m/rad' };
  end
  variables
    theta = { 0, 'rad' };
    t = { 0, 'N*m' };
    w = { 0, 'rad/s' };
  end
  function setup
    if k < 0
      error( 'Spring rate must be greater than zero' );
    end
    through( t, r.t, c.t );
    across( w, r.w, c.w );
  end
  equations
    t == k * theta;
    w == theta.der;
  end
end
```

See Also

domain

der

Purpose Return time derivative of operand

Syntax `der(x)`
`x.der`

Description The equations function may contain `der` operator, which returns the time derivative of its operand:

$$\text{der}(x) = x.\text{der} = \dot{x} = \frac{dx}{dt}$$

`der` operator takes any numerical expression as its argument:

- `der` applied to expressions that are continuous returns their time derivative
- `der` applied to `time` argument returns 1
- `der` applied to expressions that are parametric or constant returns 0
- `der` applied to countable operands returns 0. For example, `der(a<b)` returns 0 even if `a` and `b` are variables.

The return unit of `der` is the unit of its operand divided by seconds.

The following restrictions apply:

- You cannot form nonlinear expressions of the output from `der`. For example, `der(x)*der(x)` would produce an error because this is no longer a linearly implicit system.
- Higher order derivatives are not allowed. For example, `der(der(x))` would produce an error.
- For a component to compile, the number of differential equations should equal the number of differential variables.

Examples This example shows implementation for a simple dynamic system:

$$\dot{x} = 1 - x$$

The Simscape file looks as follows:

```
component MyDynamicSystem
  variables
    x = 0;
  end
  equations
    x.der == (1 - x)*{ 1, '1/s' }; % x' = 1 - x
  end
end
```

The reason you need to multiply by { 1, '1/s' } is that (1-x) is unitless, while the left-hand side (x.der) has the units of 1/s. Both sides of the equation statement must have the same units.

See Also

equations

domain

Purpose

Domain model keywords

Syntax

```
domain
variables
variables(Balancing = true)
parameters
```

Description

`domain` begins the domain model class definition, which is terminated by an `end` keyword. Only blank lines and comments can precede `domain`. You must place a domain model class definition in a file of the same name with a file name extension of `.ssc`.

See “Basic Simscape Grammar” in the *Simscape Language Guide* for more information on domain model definition syntax.

`variables` begins an Across variables declaration block, which is terminated by an `end` keyword. This block contains declarations for all the Across variables associated with the domain. A domain model class definition can contain multiple Across variables, combined in a single `variables` block. This block is required.

`variables(Balancing = true)` begins a Through variables declaration block, which is terminated by an `end` keyword. This block contains declarations for all the Through variables associated with the domain. A domain model class definition can contain multiple Through variables, combined in a single `through` block. This block is required.

Each variable is defined as a value with unit. See “Declaring Through and Across Variables for a Domain” in the *Simscape Language Guide* for more information.

`parameters` begins a domain parameters declaration block, which is terminated by an `end` keyword. This block contains declarations for domain parameters. These parameters are associated with the domain and can be propagated through the network to all components connected to the domain. This block is optional.

See “Propagation of Domain Parameters” in the *Simscape Language Guide* for more information.

Table of Attributes

For declaration member attributes, see “Attribute Lists”.

Examples

This file, named `rotational.ssc`, declares a mechanical rotational domain, with angular velocity as an Across variable and torque as a Through variable.

```
domain rotational
% Define the mechanical rotational domain
% in terms of across and through variables

variables
  w = { 1 , 'rad/s' }; % angular velocity
end

variables(Balancing = true)
  t = { 1 , 'N*m' }; % torque
end

end
```

This file, named `t_hyd.ssc`, declares a hydraulic domain, with pressure as an Across variable, flow rate as a Through variable, and an associated domain parameter, fluid temperature.

```
domain t_hyd
variables
  p = { 1e6, 'Pa' }; % pressure
end
variables(Balancing = true)
  q = { 1e-3, 'm^3/s' }; % flow rate
end
parameters
  t = { 303, 'K' }; % fluid temperature
end
end
```

domain

See Also

component

Purpose Define component equations

Syntax

```

equations
  EquationExpression1 == EquationExpression2;
end
equations
  if Expression
    EquationStatement [ EquationStatement ]
  [ elseif Expression
    EquationStatement [ EquationStatement ] ]
  else
    EquationStatement [ EquationStatement ]
  end
end

```

Description equations begins the equation section in a component file; this section is terminated by an end keyword. It is executed throughout the simulation. The purpose of the equation section is to establish the mathematical relationships among a component's variables, parameters, inputs, outputs, time and the time derivatives of each of these entities. All members declared in the component are available by their name in the equation section.

The following syntax defines a simple equation.

```

equations
EquationExpression1 == EquationExpression2;
end

```

The statement `EquationExpression1 == EquationExpression2` is an equation statement. It specifies continuous mathematical equality between two objects of class `EquationExpression`. An `EquationExpression` is any valid MATLAB expression that does not use any of the relational operators: `==`, `<`, `>`, `<=`, `>=`, `~=`, `&&`, `||`. `EquationExpression` may be constructed from any of the identifiers defined in the model declaration.

equations

The equation section may contain multiple equation statements. You can also specify conditional equations by using `if` statements as follows:

```
equations
if Expression
EquationStatement [ EquationStatement ]
[ elseif Expression
EquationStatement [ EquationStatement ] ]
else
EquationStatement [ EquationStatement ]
end
end
```

Note The total number of equation statements, their dimensionality, and their order must be the same for every branch of the `if-elseif-else` statement.

The following rules apply to the equation section:

- `Expression` is any valid MATLAB expression. It may be formed with the following operators:
 - Arithmetic
 - Relational
 - Logical
 - Primitive Math
 - Indexing
 - Concatenation
- `EquationExpression` is any valid MATLAB expression, except one formed by relational or logical operators. It may be formed with the following operators:

- Arithmetic
- Primitive Math
- Indexing
- Concatenation
- In the equation section, neither `Expression` nor `EquationExpression` may be formed with the following operators:
 - Matrix Inversion
 - MATLAB functions not listed in Supported Functions on page 4-13
- The colon operator may take only constants or end as its operands.
- All members of the component are accessible in the equation section, but none are writable.

The following MATLAB functions can be used in the equation section. The table contains additional restrictions that pertain only to the equation section. It also indicates whether a function is discontinuous. If the function is discontinuous, it introduces a zero-crossing when used with one or more continuous operands.

Supported Functions

Name	Restrictions	Discontinuous
plus		
uplus		
minus		
uminus		
mtimes		
times		
mpower		
power		

Supported Functions (Continued)

Name	Restrictions	Discontinuous
mldivide	Nonmatrix denominator	
mrdivide	Nonmatrix denominator	
ldivide		
rdivide		
eq	Do not use with continuous variables	Yes
ne	Do not use with continuous variables	Yes
lt		Yes
gt		Yes
le		Yes
ge		Yes
and		Yes
or		Yes
sin		
cos		
tan		
asin		
acos		
atan		
atan2		
log		

Supported Functions (Continued)

Name	Restrictions	Discontinuous
log10		
sinh		
cosh		
tanh		
exp		
sqrt		
abs		Yes
logical		Yes

Examples

For a component where x and y are declared as 1x1 variables, specify an equation of the form $y = x^2$:

```
equations
  y == x^2;
end
```

For the same component, specify the following piecewise equation:

$$y = \begin{cases} x & \text{for } -1 \leq x \leq 1 \\ x^2 & \text{otherwise} \end{cases}$$

This equation, written in the Simscape language, would look like:

```
equations
  if x >= -1 && x <= 1
    y == x;
  else
    y == x^2;
  end
end
```

equations

See Also

der

time

Purpose Define component inputs, that is, Physical Signal input ports of block

Syntax

```
inputs
    in1 = { value , 'unit' };
end
inputs
    in1 = { value , 'unit' }; % label:location
end
```

Description `inputs` begins a component inputs definition block, which is terminated by an `end` keyword. This block contains declarations for component inputs. Inputs will appear as Physical Signal input ports in the block diagram when the component file is brought into a Simscape model. Each input is defined as a value with unit. Specifying an optional comment lets you control the port label and location in the block icon. The following syntax defines a component input, `in1`, as a value with unit. `value` is the initial value. `unit` is a valid unit string, defined in the unit registry.

```
inputs
in1 = { value , 'unit' };
end
```

You can specify the input port label and location, the way you want it to appear in the block diagram, as a comment:

```
inputs
in1 = { value , 'unit' }; % label:location
end
```

where `label` is a string corresponding to the input port name in the block diagram, `location` is one of the following strings: `left`, `right`, `top`, `bottom`.

Examples The following example declares an input port `s`, with a default value of 1 Pa, specifying the control port of a hydraulic pressure source. In the

inputs

block diagram, this port will be named **Pressure** and will be located on the top side of the block icon.

```
inputs
    s = { 1 'Pa' };    % Pressure:top
end
```

See Also

nodes

outputs

Purpose

Define component nodes, that is, conserving ports of block

Syntax

```
nodes
    a = package_name.domain_name;
end
nodes
    a = package_name.domain_name; % label:location
end
```

Description

`nodes` begins a nodes declaration block, which is terminated by an `end` keyword. This block contains declarations for all the component nodes, which correspond to the conserving ports of a Simscape block generated from the component file. Each node is defined by assignment to an existing domain. See “Declaring Component Nodes” in the *Simscape Language Guide* for more information.

The following syntax defines a node, `a`, by associating it with a domain, `domain_name`. `package_name` is the full path to the domain, starting with the top package directory. For more information on packaging your Simscape files, see “Adding Custom Block Libraries Generated from Simscape Component Files” in the *Simscape Language Guide*.

```
nodes
a = package_name.domain_name;
end
```

You can specify the port label and location, the way you want it to appear in the block diagram, as a comment:

```
nodes
a = package_name.domain_name; % label:location
end
```

where `label` is a string corresponding to the port name in the block diagram, `location` is one of the following strings: `left`, `right`, `top`, `bottom`.

Examples

The following example uses the syntax for the Simscape Foundation mechanical rotational domain:

```
nodes
    r = foundation.mechanical.rotational.rotational;
end
```

The name of the top-level package directory is `+foundation`. It contains a subpackage `+mechanical`, with a subpackage `+rotational`, which in turn contains the domain file `rotational.ssc`.

If you want to use your own customized rotational domain called `rotational.ssc` and located at the top level of your custom package directory `+MechanicalElements`, the syntax would be:

```
nodes
    r = MechanicalElements.rotational;
end
```

The following example declares an electrical node using the syntax for the Simscape Foundation electrical domain. In the block diagram, this port will be labelled `+` and will be located on the top side of the block icon.

```
nodes
    p = foundation.electrical.electrical; % +:top
end
```

See Also

```
inputs
outputs
```

Purpose Define component outputs, that is, Physical Signal output ports of block

Syntax

```
outputs
    out1 = { value , 'unit' };
end
outputs
    out1 = { value , 'unit' }; % label:location
end
```

Description `outputs` begins a component outputs definition block, which is terminated by an `end` keyword. This block contains declarations for component outputs. Outputs will appear as Physical Signal output ports in the block diagram when the component file is brought into a Simscape model. Each output is defined as a value with unit. Specifying an optional comment lets you control the port label and location in the block icon.

The following syntax defines a component output, `out1`, as a value with unit. `value` is the initial value. `unit` is a valid unit string, defined in the unit registry.

```
outputs
out1 = { value , 'unit' };
end
```

You can specify the output port label and location, the way you want it to appear in the block diagram, as a comment:

```
outputs
out1 = { value , 'unit' }; % label:location
end
```

where `label` is a string corresponding to the input port name in the block diagram, `location` is one of the following strings: `left`, `right`, `top`, `bottom`.

outputs

Examples

The following example declares an output port `p`, with a default value of 1 Pa, specifying the output port of a hydraulic pressure sensor. In the block diagram, this port will be named **Pressure** and will be located on the bottom side of the block icon.

```
outputs
    p = { 1 'Pa' };    % Pressure:bottom
end
```

See Also

inputs
nodes

Purpose

Specify component parameters

Syntax

```
parameters
  comp_par1 = { value , 'unit' };
end
parameters
  comp_par1 = { value , 'unit' }; % Parameter name
end
```

Description

Component parameters let you specify adjustable parameters for the Simscape block generated from the component file. Parameters will appear in the block dialog box and can be modified when building and simulating a model.

`parameters` begins a component parameters definition block, which is terminated by an `end` keyword. This block contains declarations for component parameters. Parameters will appear in the block dialog box when the component file is brought into a Simscape model. Each parameter is defined as a value with unit. Specifying an optional comment lets you control the parameter name in the block dialog box.

The following syntax defines a component parameter, `comp_par1`, as a value with unit. `value` is the initial value. `unit` is a valid unit string, defined in the unit registry.

```
parameters
comp_par1 = { value , 'unit' };
end
```

To declare a unitless parameter, you can either use the same syntax:

```
par1 = { value , '1' };
```

or omit the unit and use this syntax:

```
par1 = value;
```

Internally, however, this parameter will be treated as a two-member value-unit array `{ value , '1' }`.

parameters

You can specify the parameter name, the way you want it to appear in the block dialog box, as a comment:

```
parameters
comp_par1 = { value , 'unit' }; % Parameter name
end
```

Examples

The following example declares parameter `k`, with a default value of 10 N*m/rad, specifying the spring rate of a rotational spring. In the block dialog box, this parameter will be named **Spring rate**.

```
parameters
    k = { 10 'N*m/rad' }; % Spring rate
end
```

See Also

`variables`

Purpose Prepare component for simulation

Syntax

```
function setup
    [...]
end
```

Description

```
function setup
    [...]
end
```

The body of the `setup` function can contain assignment statements, `if` and `error` statements, and `across` and `through` functions. The `setup` function is executed once for each component instance during model compilation. It takes no arguments and returns no arguments.

Use the `setup` function for the following purposes:

- Validating parameters
- Computing derived parameters
- Setting initial conditions
- Relating inputs, outputs, and variables to one another by using `across` and `through` functions

The following rules apply:

- The `setup` function is executed as regular MATLAB code.
- All members declared in the component are available by their name.
- All members (such as variables, parameters) that are externally writable are writable within `setup`. See “Member Summary” for more information.
- Local MATLAB variables may be introduced in the `setup` function. They are scoped only to the `setup` function.

The following restrictions apply:

- Command syntax is not supported in the `setup` function. You must use the function syntax. For more information, see “Command vs. Function Syntax” in the *MATLAB Programming Fundamentals* documentation.
- Persistent and global variables are not supported. For more information, see “Types of Variables” in the *MATLAB Programming Fundamentals* documentation.
- MATLAB system commands using the `!` operator are not supported.
- `try-end` and `try-catch-end` constructs are not supported.
- Passing declaration members to external MATLAB functions, for example, `my_function(param1)`, is not supported. You can, however, pass member values to external functions, for example, `my_function(param1.value)`.

Examples

The following `setup` function checks the value of a parameter `MyParam`, declared in the declaration section of a component file. It defines a maximum allowed value for this parameter, `MaxValue`, and if `MyParam` is greater than `MaxValue`, overrides it with `MaxValue` and issues a warning.

```
function setup
    MaxValue = {1, 'm' };
    if MyParam > MaxValue
        warning( 'MyParam is greater than MaxValue, overriding with MaxValue' );
        MyParam = MaxValue;
    end
end
```

See Also

`across`
`through`

Purpose	Establish relationship between component variables and nodes
Syntax	<code>through(variableI, node1.variableA, node2.variableB)</code>
Description	<p><code>through(variableI, node1.variableA, node2.variableB)</code> establishes the following relationship between the three arguments: for each <code>variableI</code>, <code>node1.variableA</code> is assigned the value <code>sum(variableI)</code> and <code>node2.variableB</code> is assigned the value <code>sum(-variableI)</code>. All arguments are variables. The first one is not associated with a node. The second and third must be associated with a node.</p> <p>The following rules apply:</p> <ul style="list-style-type: none">• All arguments must have consistent units.• The second and third arguments do not need to be associated with the same domain. For example, one may be associated with a one-phase electrical domain, and the other with a 3-phase electrical.• Either the second or the third argument may be replaced with <code>[]</code> to indicate the reference node.
Examples	<p>For example, if a component declaration section contains two electrical nodes, <code>p</code> and <code>n</code>, and a variable <code>i = { 0, 'A' }</code>; specifying current, you can establish the following relationship in the setup section:</p> <pre>through(i, p.i, n.i);</pre> <p>This defines current <code>i</code> as a Through variable from node <code>p</code> to node <code>n</code>.</p>
See Also	<code>across</code>

time

Purpose Access global simulation time

Syntax time

Description You can access global simulation time from the equation section of a Simscape file using the `time` function.

`time` returns the simulation time in seconds.

Examples The following example illustrates $y = \sin(\omega t)$:

```
component
  parameters
    w = { 1, '1/s' } % omega
  end
  outputs
    y = 0;
  end
  equations
    y == sin( w * time );
  end
end
```

See Also equations

Purpose	Convert variable or parameter to unitless value with specified unit conversion
Syntax	<code>value(a, 'unit')</code> <code>value(a, 'unit', 'type')</code>
Description	<p><code>value(a, 'unit')</code> returns a unitless numerical value, converting <code>a</code> into units <code>unit</code>. <code>a</code> is a variable or parameter, specified as a value with unit, and <code>unit</code> is a unit defined in the unit registry. <code>unit</code> must be commensurate with the units of <code>a</code>.</p> <p><code>value(a, 'unit', 'type')</code> performs either linear or affine conversion of temperature units and returns a unitless numerical value, converting <code>a</code> into units <code>unit</code>. <code>type</code> specifies the conversion type and can be one of two strings: <code>linear</code> or <code>affine</code>. If the type is not specified when converting temperature units, it is assumed to be <code>affine</code>.</p> <p>Use this function in the setup and equation sections of a Simscape file to convert a variable or parameter into a scalar value.</p>
Examples	<p>If <code>a = { 10, 'cm' }</code>, then <code>value(a, 'm')</code> returns 0.1.</p> <p>If <code>a = { 10, 'C' }</code>, then <code>value(a, 'K', 'linear')</code> returns 10.</p> <p>If <code>a = { 10, 'C' }</code>, then <code>value(a, 'K', 'affine')</code> returns 283.15. <code>value(a, 'K')</code> also returns 283.15.</p> <p>If <code>a = { 10, 'cm' }</code>, then <code>value(a, 's')</code> issues an error because the units are not commensurate.</p>

variables

Purpose Define domain or component variables

Syntax

```
variables
  comp_var1 = { value , 'unit' };
end
variables
  domain_across_var1 = { value , 'unit' };
end
variables(Balancing = true)
  domain_through_var1 = { value , 'unit' };
end
```

Description `variables` begins a variables declaration block, which is terminated by an `end` keyword. In a component file, this block contains declarations for all the variables associated with the component. In a domain file, this block contains declarations for all the Across variables associated with the domain. Additionally, domain files must have a separate variables declaration block, with the `Balancing` attribute set to `true`, which contains declarations for all the Through variables associated with the domain.

In a component file, the following syntax defines an Across, Through, or internal variable, `comp_var1`, as a value with unit. `value` is the initial value. `unit` is a valid unit string, defined in the unit registry.

```
variables
comp_var1 = { value , 'unit' };
end
```

In a domain file, the following syntax defines an Across variable, `domain_across1`, as a value with unit. `value` is the initial value. `unit` is a valid unit string, defined in the unit registry.

```
variables
domain_across_var1 = { value , 'unit' };
end
```

In a domain file, the following syntax defines a Through variable, `domain_through1`, as a value with unit. `value` is the initial value. `unit` is a valid unit string, defined in the unit registry.

```
variables(Balancing = true)
domain_through_var1 = { value , 'unit' };
end
```

Examples

The following example initializes the variable `w` (angular velocity) as 0 rad/s:

```
variables
  w = { 0, 'rad/s' };
end
```

The following example initializes the domain Through variable `t` (torque) as 1 N*m:

```
variables(Balancing = true)
  t = { 1, 'N*m' };
end
```

See Also

“Declaring Component Variables”

“Declaring Through and Across Variables for a Domain”

variables

Simscape Foundation Domains

- “Domain Types and Directory Structure” on page 5-2
- “Electrical Domain” on page 5-4
- “Hydraulic Domain” on page 5-5
- “Mechanical Rotational Domain” on page 5-7
- “Mechanical Translational Domain” on page 5-8
- “Thermal Domain” on page 5-9

Domain Types and Directory Structure

Simscape software comes with the following Foundation domains:

- “Electrical Domain” on page 5-4
- “Hydraulic Domain” on page 5-5
- “Mechanical Rotational Domain” on page 5-7
- “Mechanical Translational Domain” on page 5-8
- “Thermal Domain” on page 5-9

Simscape Foundation libraries are organized in a package containing domain and component Simscape files. The name of the top-level package directory is `+foundation`, and the package consists of subpackages containing domain files, structured as follows:

```
- +foundation
|-- +electrical
| |-- electrical.ssc
| |-- ...
|-- +hydraulic
| |-- hydraulic.ssc
| |-- ...
|-- +mechanical
| |-- +rotational
| | |-- +rotational.ssc
| | |-- ...
| |-- +translational
| | |-- +translational.ssc
| | |-- ...
|-- +thermal
| |-- thermal.ssc
| |-- ...
```

To use a Foundation domain in a component declaration, refer to the domain name using the full path, starting with the top package directory. The following example uses the syntax for the Simscape Foundation mechanical rotational domain:

```
r = foundation.mechanical.rotational.rotational;
```

The name of the top-level package directory is `+foundation`. It contains a subpackage `+mechanical`, with a subpackage `+rotational`, which in turn contains the domain file `rotational.ssc`.

The following sections describe each Foundation domain.

Electrical Domain

The electrical domain declaration is shown below.

```
domain electrical
% Electrical Domain

% Copyright 2005-2008 The MathWorks, Inc.

parameters
    Temperature = { 300.15 , 'K' }; % Circuit temperature
    GMIN        = { 1e-12 , '1/Ohm' }; % Minimum conductance, GMIN
end

variables
    v = { 0 , 'V' };
end

variables(Balancing = true)
    i = { 0 , 'A' };
end

end
```

It contains the following variables and parameters:

- Across variable v (voltage), in volts
- Through variable i (current), in amperes
- Parameter *Temperature*, specifying the circuit temperature
- Parameter *GMIN*, specifying minimum conductance

To refer to this domain in your custom component declarations, use the following syntax:

```
foundation.electrical.electrical
```

Hydraulic Domain

The hydraulic domain declaration is shown below.

```
domain hydraulic
% Hydraulic Domain

% Copyright 2005-2008 The MathWorks, Inc.

parameters
    density      = { 850    , 'kg/m^3' }; % Fluid density
    viscosity_kin = { 18e-6 , 'm^2/s' }; % Kinematic viscosity
    bulk         = { 0.8e9 , 'Pa'   }; % Bulk modulus at atm. pressure and no gas
    alpha        = { 0.005 , '1'    }; % Relative amount of trapped air
end

variables
    p = { 0 , 'Pa' };
end

variables(Balancing = true)
    q = { 0 , 'm^3/s' };
end

end
```

It contains the following variables and parameters:

- Across variable p (pressure), in Pa
- Through variable q (flow rate), in m^3/s
- Parameter *density*, specifying the default fluid density
- Parameter *viscosity_kin*, specifying the default kinematic viscosity
- Parameter *bulk*, specifying the default fluid bulk modulus at atmospheric pressure and no gas
- Parameter *alpha*, specifying the default relative amount of trapped air in the fluid

To refer to this domain in your custom component declarations, use the following syntax:

```
foundation.hydraulic.hydraulic
```

Mechanical Rotational Domain

The mechanical rotational domain declaration is shown below.

```
domain rotational
% Mechanical Rotational Domain

% Copyright 2005-2008 The MathWorks, Inc.

variables
  w = { 0 , 'rad/s' };
end

variables(Balancing = true)
  t = { 0 , 'N*m' };
end

end
```

It contains the following variables:

- Across variable w (angular velocity), in rad/s
- Through variable t (torque), in N*m

To refer to this domain in your custom component declarations, use the following syntax:

```
foundation.mechanical.rotational.rotational
```

Mechanical Translational Domain

The mechanical translational domain declaration is shown below.

```
domain translational
% Mechanical Translational Domain

% Copyright 2005-2008 The MathWorks, Inc.

variables
  v = { 0 , 'm/s' };
end

variables(Balancing = true)
  f = { 0 , 'N' };
end

end
```

It contains the following variables:

- Across variable v (velocity), in m/s
- Through variable f (force), in N

To refer to this domain in your custom component declarations, use the following syntax:

```
foundation.mechanical.translational.translational
```


Thermal Domain

The thermal domain declaration is shown below.

```
domain thermal
% Thermal domain

% Copyright 2005-2008 The MathWorks, Inc.

variables
    T = { 0 , 'K' };
end

variables(Balancing = true)
    Q = { 0 , 'J/s' };
end

end
```

It contains the following variables:

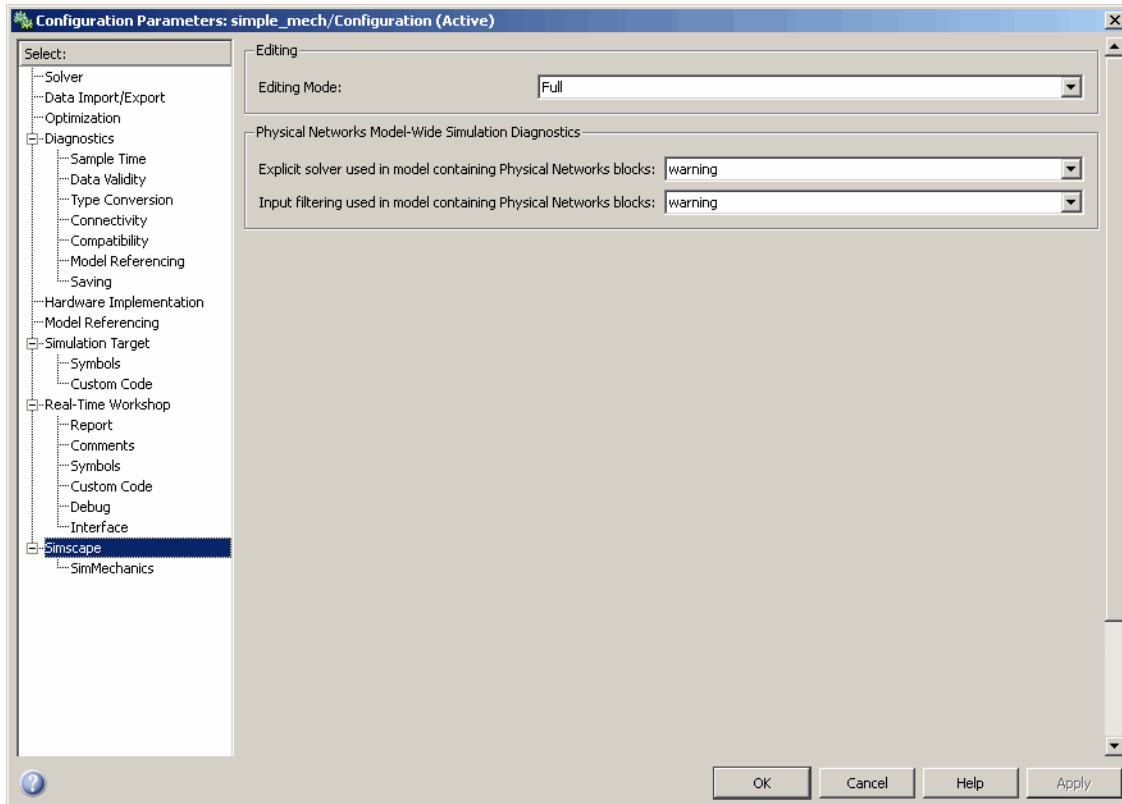
- Across variable T (temperature), in kelvin
- Through variable Q (heat flow), in J/s

To refer to this domain in your custom component declarations, use the following syntax:

```
foundation.thermal.thermal
```


Configuration Parameters

Simscape Pane: General



In this section...

“Simscape Pane Overview” on page 6-3

“Editing Mode” on page 6-4

“Explicit solver used in model containing Physical Networks blocks” on page 6-6

“Input filtering used in model containing Physical Networks blocks” on page 6-8

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 vertical products in Restricted mode, without checking out vertical product licenses, as long as the products are installed on your machine. Simscape vertical products include SimDriveline™, SimElectronics™, SimHydraulics, and SimMechanics™. Use this functionality to perform multidomain physical modeling and simulation while minimizing the number of required licenses.

Note The Simscape Editing Mode functionality requires the ability to share licenses among a group of users. Therefore, it is available only with concurrent licenses, which is the only license type at The MathWorks™ where a license is not tied to a particular workstation. The Simscape Editing Mode functionality is not supported for other license types.

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.

Configuration

This pane appears only if your model contains a block from the Simscape libraries (including Simscape vertical products).

See Also

- About the Simscape Editing Mode
- Working with Restricted and Full Modes
- Selecting a Solver
- Input filtering

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 vertical product licenses for the blocks present in the model.

When you switch from Restricted to Full mode, the license manager checks whether the required vertical product licenses are available and checks them out. If some of the vertical 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 vertical product licenses.

When you switch from Full to Restricted mode, all the vertical 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: string

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: string

Value: 'warning' | 'error' | 'none'

Default: 'warning'

See Also

Selecting a Solver

Input filtering 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 input filtering.

Settings

Default: warning

warning

Makes the system issue a warning upon simulation if the model uses input filtering, because input filtering can appreciably change the input signal and drastically affect simulation results if the time constant is too large. The warning contains a list of Simulink-PS Converter blocks that use input filtering.

error

Makes the system issue an error upon simulation if the model uses input filtering.

If you select this option and use an explicit solver, you have to provide first derivative of the input signal as an additional input signal to each Simulink-PS Converter block.

none

Turns off issuing a warning or error upon simulation when the model uses input filtering.

Command-Line Information

Parameter: InputDerivativeDiagnosticOptions

Type: string

Value: 'warning' | 'error' | 'none'

Default: 'warning'

See Also

Input filtering

across variables

Variables that are measured with a gauge connected in parallel to an element.

conserving ports

Bidirectional hydraulic or mechanical ports that represent physical connections and relate physical variables based on the Physical Network approach.

globally assigned positive direction

Direction considered positive for a model diagram.

nonrestricted parameters

Parameters that are available for modification when you open a model in Restricted mode. Usually, these are the block parameters with plain numerical values, such as **Chamber volume** or **Wheel radius**. Information on restricted and nonrestricted parameters is listed in block reference pages.

physical connections

Bidirectional connections between the blocks that mimic physical connections between elements.

physical signal ports

Unidirectional ports (inports and outports) transferring signals that use an internal Simscape engine for computations.

restricted parameters

Parameters that are not available for modification when you open a model in Restricted mode. You have to be in Full mode to modify them. Usually, these are the block parameterization options, such as **Chamber specification** or **Mechanism orientation**. Information on restricted and nonrestricted parameters is listed in block reference pages.

through variables

Variables that are measured with a gauge connected in series to an element.

vertical products

Products in the Physical Modeling family that use Simscape platform and, as a result, share common functionality such as physical units management, editing modes, and so on.

A

AC Current Source block 2-2

AC Voltage Source block 2-4

C

Capacitor block 2-6

commands

 pm_adddimension 3-2

 pm_addunit 3-3

 pm_getdimensions 3-5

 pm_getunits 3-6

 ssc_build 3-9

 ssc_clean 3-11

 ssc_mirror 3-12

 ssc_new 3-14

 ssc_protect 3-17

 ssc_reserved 3-20

Conductive Heat Transfer block 2-9

configuration parameters

 Simscape pane 6-3

 Editing Mode 6-4

 Explicit solver used in model containing

 Physical Networks blocks 6-6

 Input filtering used in model containing

 Physical Networks blocks 6-8

Connection Port block 2-11

Constant Area Orifice block 2-13

Constant Volume Chamber block 2-17

Controlled Current Source block 2-28

Controlled Voltage Source block 2-29

Convective Heat Transfer block 2-30

Current Sensor block 2-36

Current-Controlled Current Source block 2-32

Current-Controlled Voltage Source block 2-34

Custom Hydraulic Fluid block 2-38

D

DC Current Source block 2-40

DC Voltage Source block 2-41

Diode block 2-43

E

Electrical Reference block 2-45

F

Fluid Inertia block 2-46

G

Gear Box block 2-48

Gyrator block 2-50

H

Hydraulic Reference block 2-52

I

Ideal Angular Velocity Source block 2-53

Ideal Force Sensor block 2-55

Ideal Force Source block 2-57

Ideal Heat Flow Sensor block 2-59

Ideal Heat Flow Source block 2-61

Ideal Hydraulic Flow Rate Sensor block 2-63

Ideal Hydraulic Flow Rate Source block 2-65

Ideal Hydraulic Pressure Sensor block 2-67

Ideal Hydraulic Pressure Source block 2-69

Ideal Rotational Motion Sensor block 2-71

Ideal Temperature Sensor block 2-73

Ideal Temperature Source block 2-75

Ideal Torque Sensor block 2-77

Ideal Torque Source block 2-79

Ideal Transformer block 2-81

Ideal Translational Motion Sensor block 2-83

Ideal Translational Velocity Source block 2-85

Inductor block 2-87

Inertia block 2-90

L

Lever block 2-92
Linear Hydraulic Resistance block 2-96

M

Mass block 2-98
Mechanical Rotational Reference block 2-100
Mechanical Translational Reference block 2-101
Mutual Inductor block 2-102

O

Op-Amp block 2-105

P

Piston Chamber block 2-106
pm_adddimension command 3-2
pm_addunit command 3-3
pm_getdimensions command 3-5
pm_getunits command 3-6
PS Abs block 2-112
PS Add block 2-114
PS Ceil block 2-116
PS Constant block 2-117
PS Dead Zone block 2-118
PS Divide block 2-120
PS Fix block 2-122
PS Floor block 2-124
PS Gain block 2-125
PS Integrator block 2-126
PS Lookup Table (1D) block 2-128
PS Lookup Table (2D) block 2-132
PS Math Function block 2-136
PS Max block 2-138
PS Min block 2-140
PS Product block 2-142
PS Saturation block 2-144
PS Sign block 2-146

PS Subtract block 2-148
PS-Simulink Converter block 2-150

R

Radiative Heat Transfer block 2-154
Resistive Tube block 2-156
Resistor block 2-163
Rotational Damper block 2-165
Rotational Electromechanical Converter
block 2-167
Rotational Friction block 2-169
Rotational Hard Stop block 2-175
Rotational Hydro-Mechanical Converter
block 2-180
Rotational Spring block 2-183

S

Simulink-PS Converter block 2-186
Solver Configuration block 2-193
ssc_build command 3-9
ssc_clean command 3-11
ssc_mirror command 3-12
ssc_new command 3-14
ssc_protect command 3-17
ssc_reserved command 3-20
Switch block 2-197

T

terminology Glossary-1
Thermal Mass block 2-198
Thermal Reference block 2-200
Translational Damper block 2-201
Translational Electromechanical Converter
block 2-203
Translational Friction block 2-205
Translational Hard Stop block 2-211
Translational Hydro-Mechanical Converter
block 2-216

Translational Spring block 2-219
Two-Way Connection block 2-222

V

Variable Area Orifice block 2-224
Variable Chamber block 2-228
Variable Resistor block 2-233

Variable Volume Chamber block 2-235
Voltage Sensor block 2-245
Voltage-Controlled Current Source block 2-241
Voltage-Controlled Voltage Source block 2-243

W

Wheel and Axle block 2-247