

Simscape™ Multibody™

Reference



MATLAB® & SIMULINK®

R2019a



How to Contact MathWorks



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



Phone: 508-647-7000



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

Simscape™ Multibody™ Reference

© COPYRIGHT 2002–2019 by The MathWorks, Inc.

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

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

Trademarks

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

Patents

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

Revision History

March 2012	Online only	New for Version 4.0 (Release R2012a)
September 2012	Online only	Revised for Version 4.1 (Release R2012b)
March 2013	Online only	Revised for Version 4.2 (Release R2013a)
September 2013	Online only	Revised for Version 4.3 (Release R2013b)
March 2014	Online only	Revised for Version 4.4 (Release R2014a)
October 2014	Online only	Revised for Version 4.5 (Release R2014b)
March 2015	Online only	Revised for Version 4.6 (Release R2015a)
September 2015	Online only	Revised for Version 4.7 (Release R2015b)
March 2016	Online only	Revised for Version 4.8 (Release R2016a) (Renamed from <i>SimMechanics™ Reference</i>)
September 2016	Online only	Revised for Version 4.9 (Release R2016b)
March 2017	Online only	Revised for Version 5.0 (Release R2017a)
September 2017	Online only	Revised for Version 5.1 (Release R2017b)
March 2018	Online only	Revised for Version 5.2 (Release R2018a)
September 2018	Online only	Revised for Version 6.0 (Release R2018b)
March 2019	Online only	Revised for Version 6.1 (Release R2019a)

Blocks—Alphabetical List

1

Configuration Parameters

2

Simscape Multibody Pane: General	2-2
Simscape Multibody Pane Overview	2-2
Simscape Multibody Pane: Diagnostics	2-3
Invalid visual properties	2-4
Repeated vertices in a cross-section	2-4
Unconnected frame port	2-5
Unconnected Geometry port	2-5
Redundant block	2-6
Conflicting reference frames	2-7
Rigidly constrained block	2-7
Unsatisfied high priority state targets	2-8
Overspecified targets in kinematic loops	2-9
Simscape Multibody Pane: Explorer	2-10
Open Mechanics Explorer on model update or simulation ...	2-10

Multibody Visualization

3

Functions—Alphabetical List

4

First-Generation Conversion

5

Convert a First-Generation Model	5-2
Frames and Signals	5-2
Rigid Bodies	5-3
Multibody Assembly	5-5
System Dynamics	5-6
Model Visualization	5-7
Simulation and Analysis	5-9
Third-Party Model Import	5-10

Blocks—Alphabetical List

6-DOF Joint

Joint with one spherical and three prismatic primitives

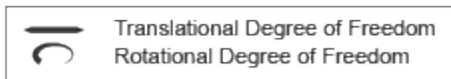
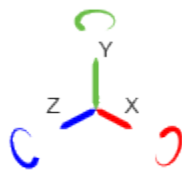


Library

Joints

Description

This block represents a joint with three translational and three rotational degrees of freedom. Three prismatic primitives provide the translational degrees of freedom. One spherical primitive provides the three rotational degrees of freedom.



Joint Degrees of Freedom

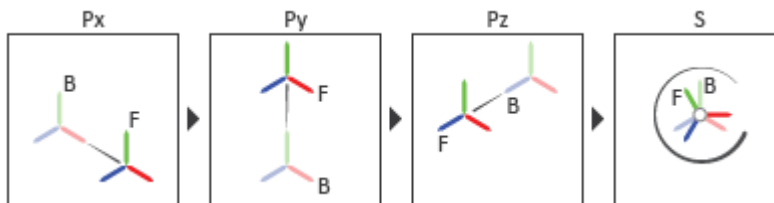
The joint block represents motion between the base and follower frames as a sequence of time-varying transformations. Each joint primitive applies one transformation in this sequence. The transformation translates or rotates the follower frame with respect to the

joint primitive base frame. For all but the first joint primitive, the base frame coincides with the follower frame of the previous joint primitive in the sequence.

At each time step during the simulation, the joint block applies the sequence of time-varying frame transformations in this order:

- 1 Translation:
 - a Along the X axis of the X Prismatic Primitive (Px) base frame.
 - b Along the Y axis of the Y Prismatic Primitive (Py) base frame. This frame is coincident with the X Prismatic Primitive (Px) follower frame.
 - c Along the Z axis of the Z Prismatic Primitive (Pz) base frame. This frame is coincident with the Y Prismatic Primitive (Py) follower frame.
- 2 Rotation:
 - About a general 3-D axis resolved in the base frame. This frame is coincident with the Z Prismatic Primitive (Pz) follower frame.

The figure shows the sequence in which the joint transformations occur at a given simulation time step. The resulting frame of each transformation serves as the base frame for the following transformation. Because 3-D rotation occurs as a single rotation about an arbitrary 3-D axis (as opposed to three separate rotations about the X, Y, Z axes), gimbal lock does not occur.



Joint Transformation Sequence

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the

joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Prismatic Primitive: State Targets

Specify the prismatic primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative position, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative velocity, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is m for position and m/s for velocity.

Prismatic Primitive: Internal Mechanics

Specify the prismatic primitive internal mechanics. Internal mechanics include linear spring forces, accounting for energy storage, and damping forces, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the distance between base and follower frame origins at which the spring force is zero. The default value is 0. Select or enter a physical unit. The default is m.

Spring Stiffness

Enter the linear spring constant. This is the force required to displace the joint primitive by a unit distance. The default is 0. Select or enter a physical unit. The default is N/m.

Damping Coefficient

Enter the linear damping coefficient. This is the force required to maintain a constant joint primitive velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N/(m/s).

Prismatic Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation

speed, reducing the transition region improves accuracy while expanding it improves speed.

Prismatic Primitive: Actuation

Specify actuation options for the prismatic joint primitive. Actuation modes include **Force** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Actuation signals are resolved in the base frame.

Force

Select an actuation force setting. The default setting is **None**.

Actuation Force Setting	Description
None	No actuation force.
Provided by Input	Actuation force from physical signal input. The signal provides the force acting on the follower frame with respect to the base frame along the joint primitive axis. An equal and opposite force acts on the base frame.
Automatically computed	Actuation force from automatic calculation. Simscape Multibody computes and applies the actuation force based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.

Actuation Motion Setting	Description
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Prismatic Primitive: Sensing

Select the variables to sense in the prismatic joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative position of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Velocity

Select this option to sense the relative velocity of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Acceleration

Select this option to sense the relative acceleration of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Actuator Force

Select this option to sense the actuation force acting on the follower frame with respect to the base frame along the joint primitive axis.

Spherical Primitive: State Targets

Specify the desired initial states of the spherical joint primitive and their relative priority levels. States that you can target include position and velocity. Use the priority level to help the assembly algorithm decide which of the state targets in a model to more precisely satisfy should conflicts between them arise.

Even in the absence of state target conflicts, the true initial states may differ from those specified here. Such discrepancies can occur due to kinematic constraints arising from other parts of the model. If a state target cannot be satisfied precisely, it is satisfied

approximately. Discrepancies are noted in Simscape Variable Viewer (**Analysis > Simscape > Variable Viewer**).

Specify Position Target

Check to specify the desired rotation of the follower frame relative to the base frame at the start of simulation.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Select a method to specify the joint primitive state target.

Method	Description
None	Constrain the base and follower frames to share the same orientation.
Aligned Axes	Set frame rotation by aligning two follower frame axes with two base frame axes.
Standard Axis	Specify frame rotation as an angle about a standard axis (x , y , or z).
Arbitrary Axis	Specify frame rotation as an angle about a general $[x, y, z]$ axis.
Rotation Sequence	Specify frame rotation as a sequence of three elementary rotations.

Method	Description
Rotation Matrix	Specify frame rotation as a right-handed orthogonal rotation matrix.

Aligned Axes

Select two pairs of base-follower frame axes.

Parameter	Description
Pair 1	First pair of base-follower frame axes to align.
Pair 2	Second pair of base-follower frame axes to align. Axis choices depend on Pair 1 axis selections.

Standard Axis

Select a standard rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	Standard rotation axis (X, Y, or Z) resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Arbitrary Axis

Select a general 3-D rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	General rotation axis [X Y Z] resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Rotation Sequence

Specify a sequence of three elementary rotations about the selected permutation of x, y, and z axes. These rotation sequences are also known as Euler and Tait-Bryan sequences. The rotations are those of the follower frame relative to the frame selected in the **Rotate About** parameter.

If you set the **Rotate About** parameter to **Follower Frame**, the follower frame rotates about its own axes. These axes change orientation with each successive rotation. If you set the **Rotate About** parameter to **Base Frame**, the follower frame rotates about the fixed base frame axes.

Parameter	Description
Rotation About	Frame whose axes to rotate the follower frame about.
Sequence	Sequence of axes about which to apply the elementary rotations.
Angles	Three-element vector with elementary rotation angles about the axes specified in the Sequence parameter.

Rotation Matrix

Specify the 3×3 transformation matrix of a proper rotation between the base and follower frames. The matrix must be orthogonal and have determinant +1. The default matrix is $[1 \ 0 \ 0; \ 0 \ 1 \ 0; \ 0 \ 0 \ 1]$.

Specify Velocity Target

Check to specify the desired rotational velocity of the follower frame relative to the base frame at the start of simulation.

Value

Enter the relative rotational velocity of the follower frame against the base frame, as projected on the axes of the selected **Resolution Frame** (by default **Follower**). This parameter requires a three-element vector with the $[x \ y \ z]$ components of the resolved relative velocity.

Resolution Frame

Select the frame in which to resolve the components of the velocity target. The resolution frame is not a measurement frame—the specified velocity is always that of the follower frame relative to the base frame. The resolution frame merely

provides an alternate set of axes with respect to which to interpret the relative velocity components. The default setting is `Follower`.

Spherical Primitive: Internal Mechanics

Specify the spherical primitive internal mechanics. This includes linear spring and damping forces, accounting for energy storage and dissipation, respectively. To ignore internal mechanics, keep spring stiffness and damping coefficient values at the default value of 0.

Equilibrium Position

Select a method to specify the spring equilibrium position. The equilibrium position is the rotation angle between base and follower port frames at which the spring torque is zero.

Method	Description
None	Constrain the base and follower frames to share the same orientation.
Aligned Axes	Set frame rotation by aligning two follower frame axes with two base frame axes.
Standard Axis	Specify frame rotation as an angle about a standard axis (x, y, or z).
Arbitrary Axis	Specify frame rotation as an angle about a general [x, y, z] axis.
Rotation Sequence	Specify frame rotation as a sequence of three elementary rotations.
Rotation Matrix	Specify frame rotation as a right-handed orthogonal rotation matrix.

Aligned Axes

Select two pairs of base-follower frame axes.

Parameter	Description
Pair 1	First pair of base-follower frame axes to align.

Parameter	Description
Pair 2	Second pair of base-follower frame axes to align. Axis choices depend on Pair 1 axis selections.

Standard Axis

Select a standard rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	Standard rotation axis (X, Y, or Z) resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Arbitrary Axis

Select a general 3-D rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	General rotation axis [X Y Z] resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Rotation Sequence

Specify a sequence of three elementary rotations about the selected permutation of x, y, and z axes. These rotation sequences are also known as Euler and Tait-Bryan sequences. The rotations are those of the follower frame relative to the frame selected in the **Rotate About** parameter.

If you set the **Rotate About** parameter to **Follower Frame**, the follower frame rotates about its own axes. These axes change orientation with each successive rotation. If you set the **Rotate About** parameter to **Base Frame**, the follower frame rotates about the fixed base frame axes.

Parameter	Description
Rotation About	Frame whose axes to rotate the follower frame about.
Sequence	Sequence of axes about which to apply the elementary rotations.
Angles	Three-element vector with elementary rotation angles about the axes specified in the Sequence parameter.

Rotation Matrix

Specify the 3×3 transformation matrix of a proper rotation between the base and follower frames. The matrix must be orthogonal and have determinant +1. The default matrix is $[1 \ 0 \ 0; \ 0 \ 1 \ 0; \ 0 \ 0 \ 1]$.

Spring Stiffness

Enter the linear spring constant. This is the torque required to displace the joint primitive by a unit angle. The term linear refers to the mathematical form of the spring equation. The default is 0. Select a physical unit. The default is N*m/deg.

Damping Coefficient

Enter the linear damping coefficient. This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames. The default is 0. Select a physical unit. The default is N*m/(deg/s).

Spherical Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Spherical Primitive: Actuation

Specify actuation options for the spherical joint primitive. Actuation modes include **Torque** only. Selecting a torque input adds the corresponding physical signal port to the block. Use this port to specify the actuation torque signal.

Torque

Select a source for the actuation torque. The default setting is **None**.

Actuation Torque Setting	Description
None	Apply no actuation torque.

Actuation Torque Setting	Description
Provided by Input	Apply an actuation torque based on a physical signal. The signal specifies the torque acting on the follower frame with respect to the base frame. An equal and opposite torque acts on the base frame. Selecting this option exposes additional parameters.

Torque (X), Torque (Y), Torque (Z)

Select in order to actuate the spherical joint primitive about each standard Cartesian axis (X, Y, Z) separately. The block exposes the corresponding physical signal ports. Use these ports to specify the actuation torque signals. The signals must be scalar values.

Torque (XYZ)

Select in order to actuate the spherical joint primitive about an arbitrary axis [X Y Z]. The block exposes the corresponding physical signal port. Use this port to specify the actuation torque signal. The signal must be a 3-D vector.

Frame

Select the frame to resolve the actuation torque signal in. The axes of this frame establish the directions of the X, Y, and Z torque components. The default setting is Base.

Spherical Primitive: Sensing

Select the motion variables to sense in the spherical joint primitive. The block adds the corresponding physical signal ports. Use these ports to output the numerical values of the motion variables.

The block measures each motion variable for the follower frame with respect to the base frame. It resolves that variable in the resolution frame that you select from the **Frame** drop-down list.

Motion Variables	Description
Position	Quaternion describing follower frame rotation with respect to base frame. The quaternion coefficients are $\left[\cos\left(\frac{\theta}{2}\right), n_x \sin\left(\frac{\theta}{2}\right), n_y \sin\left(\frac{\theta}{2}\right), n_z \sin\left(\frac{\theta}{2}\right) \right]$. The measurement is the same in all measurement frames.
Velocity (X), Velocity (Y), Velocity (Z)	Angular velocity components about X, Y, and Z axes.
Velocity	3-D angular velocity vector with components about X, Y, and Z axes.
Acceleration (X), Acceleration (Y), Acceleration (Z)	Angular acceleration components about X, Y, and Z axes.
Acceleration	3-D angular acceleration vector with components about X, Y, and Z axes.

Frame

Select the frame to resolve the measurement in. The axes of this frame establish the directions of X, Y, and Z vector components. The default setting is Base.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force

or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame

- F — Follower frame

Actuation Ports

The prismatic joint primitives provide the following actuation ports:

- f_x, f_y, f_z — Actuation forces of the X, Y, and Z prismatic joint primitives
- p_x, p_y, p_z — Desired trajectories of the X, Y, and Z prismatic joint primitives

The spherical joint primitive provides the following actuation ports:

- t — Actuation torque vector $[t_x, t_y, t_z]$ acting on the spherical joint primitive
- t_x, t_y, t_z — X, Y, and Z components of the actuation torque acting on the spherical joint primitive

Sensing Ports

The prismatic primitives provide the following sensing ports:

- p_x, p_y, p_z — Positions of the X, Y, and Z prismatic joint primitives
- v_x, v_y, v_z — Velocities of the X, Y, and Z prismatic joint primitives
- a_x, a_y, a_z — Accelerations of the X, Y, and Z prismatic joint primitives
- f_x, f_y, f_z — Actuation forces acting on the X, Y, and Z prismatic joint primitives

The spherical primitive provides the following sensing ports:

- Q — Orientation of the spherical joint primitive in quaternion form
- w_x, w_y, w_z — X, Y, and Z angular velocity components of the spherical joint primitive
- w — Angular velocity $[w_x, w_y, w_z]$ of the spherical joint primitive
- b_x, b_y, b_z — X, Y, and Z angular acceleration components of the spherical joint primitive
- b — Angular acceleration $[b_x, b_y, b_z]$ of the spherical joint primitive

The following sensing ports provide the composite forces and torques acting on the joint:

- f_c — Constraint force
- t_c — Constraint torque

- ft — Total force
- tt — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Bushing Joint | Prismatic Joint | Spherical Joint

Topics

“Motion Sensing”

“Measurement Frames”

“Actuating and Sensing with Physical Signals”

Introduced in R2012a

Angle Constraint

Fixed angle between two frame Z axes

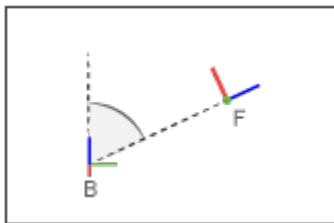


Library

Constraints

Description

This block applies a fixed angle between the Z axes of the base and follower port frames. The frames lose one rotational degree of freedom if the constraint angle is greater than 0° and smaller than 180° . They lose two rotational degrees of freedom if the constraint angle is exactly 0° or 180° —that is, if the frames are parallel or anti-parallel. The figure shows the constraint angle between two frames.



Parameters

Type

Angle constraint type. The default setting is General.

Type	Purpose
Parallel	Align the base and follower frame +Z axes.
Anti-Parallel	Align the base frame +Z axis with the follower frame -Z axis.
Perpendicular	Make the base and follower frame Z axes perpendicular to each other.
General	Hold the specified angle between the Z axes of the base and follower port frames.

Angle

Constraint angle between the base and follower frame Z axes. The angle must lie in the range $0 < \theta < 180$ deg. For an angle of 0 or 180 deg, set **Type** to Parallel or Anti-Parallel instead. The default value is 45 deg.

Constraint Torque Sensing

Select whether to compute and output the distance constraint torque vector and its magnitude. The distance constraint torque is the torque the block must apply in order to maintain the angle you specify between the base and follower port frames.

Direction

Constraint torques act in pairs. As expressed by Newton’s third law of motion, if the base port frame exerts a constraint torque on the follower port frame, then the follower port frame must exert an equal and opposite torque on the base port frame. Select which of the two constraint torques to sense:

- **Follower on Base** — Sense the constraint torque that the follower port frame exerts on the base port frame.
- **Base on Follower** — Sense the constraint torque that the base port frame exerts on the follower port frame.

Resolution Frame

The block expresses the constraint torque vector in terms of its Cartesian vector components. The splitting of a vector into vector components is known as vector resolution. The frame whose axes define the vector component directions is known as

the resolution frame. Select whether to resolve the constraint torque vector in the base or follower port frame.

Torque Vector

Compute and output the Cartesian components of the angle constraint torque vector. The output signal is a three-dimensional vector with components expressed about the X, Y, and Z axes of the resolution frame.

Signed Torque Magnitude

Compute and output the magnitude of the angle constraint torque, including its sign.

Ports

The block provides two frame ports:

- B — Base frame port
- F — Follower frame port

In addition, the block provides two physical signal output ports:

- t — Angle constraint torque vector
- tm — Signed magnitude of the angle constraint torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Bevel Gear Constraint | Common Gear Constraint | Distance Constraint | Point on Curve Constraint | Rack and Pinion Constraint

Introduced in R2012a

Bearing Joint

Joint with one prismatic and three revolute primitives

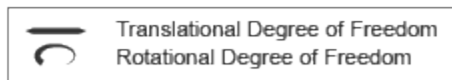
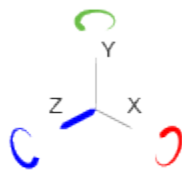


Library

Joints

Description

This block represents a joint with one translational and three rotational degrees of freedom. One prismatic primitive provides the translational degree of freedom. Three revolute primitives provide the three rotational degrees of freedom.



Joint Degrees of Freedom

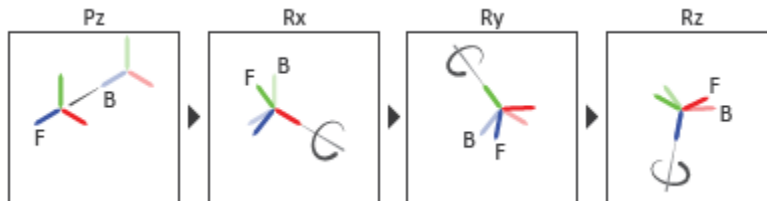
The joint block represents motion between the base and follower frames as a sequence of time-varying transformations. Each joint primitive applies one transformation in this sequence. The transformation translates or rotates the follower frame with respect to the

joint primitive base frame. For all but the first joint primitive, the base frame coincides with the follower frame of the previous joint primitive in the sequence.

At each time step during the simulation, the joint block applies the sequence of time-varying frame transformations in this order:

- 1 Translation:
 - Along the Z axis of the Z Prismatic Primitive (Pz) base frame.
- 2 Rotation:
 - a About the X axis of the X Revolute Primitive (Rx) base frame. This frame is coincident with the Z Prismatic Primitive (Pz) follower frame.
 - b About the Y axis of the Y Revolute Primitive (Ry) base frame. This frame is coincident with the X Revolute Primitive (Rx) follower frame.
 - c About the Z axis of the Z Revolute Primitive (Rz) base frame. This frame is coincident with the Y Revolute Primitive (Ry) follower frame.

The figure shows the sequence in which the joint transformations occur at a given simulation time step. The resulting frame of each transformation serves as the base frame for the following transformation. Because 3-D rotation occurs as a sequence, it is possible for two axes to align, causing to the loss of one rotational degree of freedom. This phenomenon is known as gimbal lock.



Joint Transformation Sequence

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the

joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Prismatic Primitive: State Targets

Specify the prismatic primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative position, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative velocity, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is m for position and m/s for velocity.

Prismatic Primitive: Internal Mechanics

Specify the prismatic primitive internal mechanics. Internal mechanics include linear spring forces, accounting for energy storage, and damping forces, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the distance between base and follower frame origins at which the spring force is zero. The default value is 0. Select or enter a physical unit. The default is m.

Spring Stiffness

Enter the linear spring constant. This is the force required to displace the joint primitive by a unit distance. The default is 0. Select or enter a physical unit. The default is N/m.

Damping Coefficient

Enter the linear damping coefficient. This is the force required to maintain a constant joint primitive velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N/(m/s).

Prismatic Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation

speed, reducing the transition region improves accuracy while expanding it improves speed.

Prismatic Primitive: Actuation

Specify actuation options for the prismatic joint primitive. Actuation modes include **Force** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Actuation signals are resolved in the base frame.

Force

Select an actuation force setting. The default setting is **None**.

Actuation Force Setting	Description
None	No actuation force.
Provided by Input	Actuation force from physical signal input. The signal provides the force acting on the follower frame with respect to the base frame along the joint primitive axis. An equal and opposite force acts on the base frame.
Automatically computed	Actuation force from automatic calculation. Simscape Multibody computes and applies the actuation force based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.

Actuation Motion Setting	Description
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Prismatic Primitive: Sensing

Select the variables to sense in the prismatic joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative position of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Velocity

Select this option to sense the relative velocity of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Acceleration

Select this option to sense the relative acceleration of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Actuator Force

Select this option to sense the actuation force acting on the follower frame with respect to the base frame along the joint primitive axis.

Revolute Primitive: State Targets

Specify the revolute primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative rotation angle, measured about the joint primitive axis, of the follower

frame with respect to the base frame. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative angular velocity, measured about the joint primitive axis, of the follower frame with respect to the base frame. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is deg for position and deg/s for velocity.

Revolute Primitive: Internal Mechanics

Specify the revolute primitive internal mechanics. Internal mechanics include linear spring torques, accounting for energy storage, and linear damping torques, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the rotation angle between base and follower frames at which the spring torque is zero. The default value is 0. Select or enter a physical unit. The default is deg.

Spring Stiffness

Enter the linear spring constant. This is the torque required to rotate the joint primitive by a unit angle. The default is 0. Select or enter a physical unit. The default is N*m/deg.

Damping Coefficient

Enter the linear damping coefficient. This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N*m/(deg/s).

Revolute Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper

forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Revolute Primitive: Actuation

Specify actuation options for the revolute joint primitive. Actuation modes include **Torque** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Input signals are resolved in the base frame.

Torque

Select an actuation torque setting. The default setting is **None**.

Actuation Torque Setting	Description
None	No actuation torque.
Provided by Input	Actuation torque from physical signal input. The signal provides the torque acting on the follower frame with respect to the base frame about the joint primitive axis. An equal and opposite torque acts on the base frame.
Automatically computed	Actuation torque from automatic calculation. Simscape Multibody computes and applies the actuation torque based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Revolute Primitive: Sensing

Select the variables to sense in the revolute joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative rotation angle of the follower frame with respect to the base frame about the joint primitive axis.

Velocity

Select this option to sense the relative angular velocity of the follower frame with respect to the base frame about the joint primitive axis.

Acceleration

Select this option to sense the relative angular acceleration of the follower frame with respect to the base frame about the joint primitive axis.

Actuator Torque

Select this option to sense the actuation torque acting on the follower frame with respect to the base frame about the joint primitive axis.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Actuation Ports

The prismatic joint primitive provides the following actuation ports:

- f_z — Actuation force acting on the Z prismatic joint primitive
- p_z — Desired trajectory of the Z prismatic joint primitive

The revolute joint primitives provide the following actuation ports:

- t_x, t_y, t_z — Actuation torques acting on the X, Y, and Z revolute joint primitives
- q_x, q_y, q_z — Desired rotations of the X, Y, and Z revolute joint primitives

Sensing Ports

The prismatic primitive provides the following sensing ports:

- p_z — Position of the Z prismatic joint primitive
- v_z — Velocity of the Z prismatic joint primitive
- a_z — Acceleration of the Z prismatic joint primitive
- f_z — Actuation force acting on the Z prismatic joint primitive

The revolute primitives provide the following sensing ports:

- q_x, q_y, q_z — Angular positions of the X, Y, and Z revolute joint primitives
- w_x, w_y, w_z — Angular velocities of the X, Y, and Z revolute joint primitives
- b_x, b_y, b_z — Angular accelerations of the X, Y, and Z revolute joint primitives

- t_x , t_y , t_z — Actuation torques acting on the X, Y, and Z revolute joint primitives

The following sensing ports provide the composite forces and torques acting on the joint:

- f_c — Constraint force
- t_c — Constraint torque
- f_t — Total force
- t_t — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Prismatic Joint | Revolute Joint

Topics

“Actuating and Sensing with Physical Signals”

“Motion Sensing”

“Rotational Measurements”

“Translational Measurements”

Introduced in R2012a

Belt-Cable End

Tip of the cord of a pulley system

Library: Simscape / Multibody / Belts and Cables



Description

The Belt-Cable End block represents the tip of a cord by which to anchor, drive, or load a pulley system. This tip serves as an interface between the belt-cable domain specific to pulleys and the frame domain general to all multibody components. A belt-cable port (**E**) identifies the cord whose tip this block defines and its relative placement within a pulley system. A frame port (**R**) identifies the local reference frame of the tip and its placement in the broader multibody model.



The cord is inextensible. Its effective length can change only if a source of a cord is provided—for example, in the form of a spool. If the second terminus of the cord is another tip, the length of the cord is fixed, and so must be the distance around the pulleys from one tip to the other. This distance is monitored during simulation, through calculations based on the placement and geometry of the pulleys. The two measures—the length and the distance—must always agree.

The cord is also always taut. From one of its tips to the nearest pulley, the cord must form a straight line, one tangent to the circumference of the pulley and parallel to its plane of rotation. The reference frame of the tip is placed on this line with its *z*-axis directed along

it and away from the pulley. Constraints imposed on the tip by components outside of the pulley system must act without interfering with this alignment.

Note that the frame and belt-cable ports belong to different physical domains. As a rule, ports connect only to like ports—frame ports to other frame ports, belt-cable ports to other belt-cable ports. The belt-cable domain imposes also the special requirement that each network of belt-cable connection lines contain one (and no more than one) Belt-Cable Properties block. The necessary attributes of the pulley cord are specified through this block.

Ports

Frame

R — Local reference frame

frame

Reference frame by which to connect the tip of the cord to the remainder of a multibody model.

Belt-Cable

E — Cord attachment point

belt-cable

Point at which to terminate the cord of a pulley system.

Extended Capabilities

C/C++ Code Generation

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

See Also

Belt-Cable Properties | Belt-Cable Spool | Pulley

Introduced in R2018a

Belt-Cable Properties

General characteristics of the cord of a pulley system

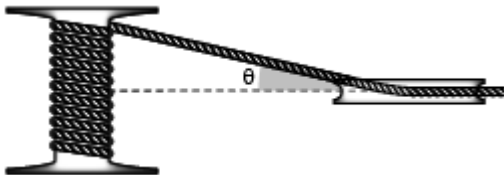
Library: Simscape / Multibody / Belts and Cables



Description

The Belt-Cable Properties block sets the attributes of a pulley cord, among them its geometry and color, as well as its alignment constraints. For the purpose of visualization, the cord is treated as a one-dimensional line with color and opacity. Circular arc segments straddle the pulleys and straight line segments bridge the distances between the arcs.

By default, the cord can enter and exit a drum—pulley or spool—at an angle to its center plane. This angle can vary during simulation—for example, due to translation of the drum on a prismatic joint. While the contact point is always in the center plane of the drum, the drum can move when mounted on a joint. Use this strategy to model, for example, the winding of a cord on a spool and the fleet angle of the same (denoted θ in the figure).

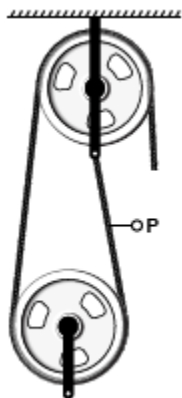


The cord can also be constrained to enter and exit a drum in the center plane of that drum. The assembly may still be three-dimensional, with different pulleys on different planes, but only as long as the contact angles are each zero. Set the **Drum Belt-Cable Alignment** block parameter to **Monitored Planar** to enforce this constraint. (Because the constraint is monitored, the simulation stops with an error if a contact angle should be other than zero.)

Other attributes are set automatically by modeling assumptions and calculations. Among the assumptions are those of a massless cord that is both inextensible and always taut.

Length is determined at the moment of assembly and fixed thereafter. Its value is consistent with a cord that precisely spans the pulley system as arranged in its initial configuration.

That configuration depends on the starting positions of the various joints in a model. These are matched where feasible to the state targets specified in the joint blocks. It is possible to use those targets to set the length of the cord and to guide the placement of its ends—for example to apply an initial rotation to a spool or an initial translation to a load-bearing anchor.



The belt-cable port (**P**) identifies the cord characterized by this block. It matters little where in a belt-cable network the port connects. It is important, however, that each belt-cable network in a model contain one instance of this block. A belt-cable network is distinct from another if there is no belt-cable connection line between the two.

The visualization of the cord is by default on but it can be suppressed in the block. If it is on, you can click any point on a cord to highlight its entire length. Look in the tree view pane for the name of the Belt-Cable Properties block associated with the selected cord—the block name is identified there. Right-click the block name and select **Go To Block** if necessary to update the cord visualization properties.

Ports

Belt-Cable

P — Belt-cable network attachment point

belt-cable

Attachment point for the belt-cable network whose properties this block aims to specify.

Parameters

Drum Belt-Cable Alignment — Type of alignment allowed between the cord and the center plane of a drum

Unrestricted (default) | Monitored Planar

Type of alignment allowed between the cord and the center plane of a drum (whether pulley or spool). Select `Unrestricted` to enable the cord to enter and exit the drum center plane at an angle. The entry and exit angles can differ during simulation. Select `Monitored Planar` to require that the cord always enter and exit a drum in line with its center plane. If at any time the contact angle differs from zero, the simulation then stops with an error.

Type — Means by which to visualize the cord

Pitch Line (default) | None

Means by which to show the cord in a model visualization. The default setting corresponds to a pitch line that arcs around each pulley and spool at the pitch radii specified in their respective blocks. The line segments are circular in the ranges of contact between the cord and pulley or spool; they are straight in the distances between the arcs.

Visual Properties — Parameterizations for color and opacity

Simple (default) | Advanced

Parameterization for specifying visual properties. Select `Simple` to specify color and opacity. Select `Advanced` to add specular highlights, ambient shadows, and self-illumination effects.

Simple: Color — True color as [R,G,B] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three-element vector with values constrained to 0-1

RGB color vector with red (R), green (G), and blue (B) color amounts specified on a 0-1 scale. A color picker provides an alternative interactive means of specifying a color. If you change the **Visual Properties** setting to Advanced, the color specified in this parameter becomes the **Diffuse Color** vector.

Simple: Opacity — Surface opacity as scalar number on 0-1 scale

1.0 (default) | scalar with value constrained to 0-1

Graphic opacity specified on a scale of 0-1. An opacity of 0 corresponds to a completely transparent graphic and an opacity of 1 to a completely opaque graphic.

Advanced: Diffuse Color — True color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three- or four-element vector with values constrained to 0-1

True color under direct white light specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. An optional fourth element specifies the color opacity also on a scale of 0-1. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Specular Color — Highlight color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of specular highlights specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Ambient Color — Shadow color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of shadow areas in diffuse ambient light, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Emissive Color — Self-illumination color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Surface color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Shininess — Highlight sharpness as scalar number on 0-128 scale
75 (default) | scalar with value constrained to 0-128

Sharpness of specular light reflections, specified as a scalar number on a 0-128 scale. Increase the shininess value for smaller but sharper highlights. Decrease the value for larger but smoother highlights.

Extended Capabilities

C/C++ Code Generation

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

See Also

Belt-Cable End | Belt-Cable Spool | Pulley

Introduced in R2018a

Belt-Cable Spool

Source and sink of cord in a pulley system

Library: Simscape / Multibody / Belts and Cables



Description

The Belt-Cable Spool block represents a cylindrical drum on which to wind (and from which to unwind) the cord of a pulley system. The spool marks an end to the cord and the point at which a motor or other power source often pulls on a load. A Belt-Cable End block generally marks the second tip of the cord, to which the load itself is commonly attached. Depending on whether it is winding or unwinding, the spool can behave as an infinite source of cord or as an infinite sink of the same.



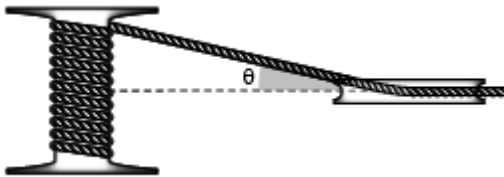
The spool serves as an interface between the belt-cable domain specific to pulley systems and the frame domain general to all other multibody components. The belt-cable port (**A**) identifies the tip of the cord to be wound on the spool and the relative placement of that tip within a pulley system. A frame port (**R**) identifies the reference frame of the spool and its placement in the broader multibody model.

The degrees of freedom of the spool are a function of the joint by which the spool connects to other components. It is common for a revolute joint to provide those degrees of freedom; they reduce in this case to rotation about a single axis (that of the spool). Actuation inputs, specified directly through the joint by means of torque or motion signals, serve to drive the spool and to wind (or unwind) the cord.

The cord enters and exits the drum in tangency with the drum circumference. Consistent with the right-hand rule, the winding is in a counterclockwise direction about the rotation axis of the drum. This axis is by definition the z-axis of the local reference frame (**R**). To reverse the direction of the winding, you must flip the local reference frame so that the z-axis points in the opposite direction—for example, by the application of a frame rotation through a Rigid Transform block.

The surface of the spool (smooth or grooved) is not considered in the model. In addition, the cord is assumed to wrap around the spool in a circle that is of constant radius (that of the *pitch* circle) and coplanar with the transverse cross section of the winch. Changes in spool radius due to winding are ignored.

By default, the cord can enter and exit a spool at an angle to its center plane (θ in the figure). This angle can vary during simulation—for example, due to translation of the spool on a prismatic joint. While the contact point is always in the center plane of the spool, the spool can move when mounted on a joint. The cord can also be constrained to enter and exit the spool in its center plane. Whether this constraint is enforced depends on the settings of the Belt-Cable Properties block.



The inertia of the spool and of the cord wound on it are also ignored. To capture the inertia of a spool of fixed mass, use the Solid or Inertia block. Consider the Solid block if solid geometry is important in the model. To capture the variable mass properties of a cord as it winds on, and unwinds from, the spool, use a block from the Variable Solids library—for example Variable Cylindrical Solid or General Variable Mass.

Ports

Frame

R — Local reference frame

frame

Reference frame for attachment of the spool to the remainder of a multibody model.

Belt-Cable

A — Spool cable attachment point

belt-cable

Point of tangency between the center line of the cord and pitch circle of the spool.

Parameters

Pitch Radius — Distance from the center of the spool to the center line of the cord

10 cm (default) | positive scalar in units of length

Distance from the center of the spool to the running axis of the cord measured in the arc within which contact occurs. In compound pulley systems, the differences in pitch radii often determine the ratio at which speed is reduced or torque is augmented.

Sensing — Selection of kinematic variables to sense

Unchecked (default) | Checked

Selection of kinematic variables to sense. Select a check box to expose a physical signal port for the corresponding variable. The variables available for sensing are:

- **Spool Angle A** — Angle, measured in the xy plane of the reference frame, from the local x -axis to the line between the frame origin and point of contact **A**.

If the point of contact is above the xz -plane (in the $+y$ -region of the reference frame), the angle is positive. If the point of contact is below the xz -plane, the angle is negative. The angle is zero when the point of contact happens to be exactly in the xz -plane.

The angle is not modular. Rather than be constrained to a 360-degree range—snapping back to the beginning of the range after completing a turn—the measured value changes continuously with repeated turns. Every turn that the drum makes adds (or subtracts) 2π to the measurement.

Use port **qpa** for this measurement.

- **Fleet Angle A** — Angle from the xy -plane of the reference frame to the cord at point of contact **A**. The xy -plane is the same as the center plane of the drum.

If the cord approaches the point of contact from above the xy -plane (in the $+z$ region of the reference frame), the angle is positive. If the cord approaches from below, the angle is negative. The angle is zero when the cord approaches the point of contact in the center plane of the drum.

The angle is modular, which is to say that its value is bound—here, between $-\pi/2$ to $+\pi/2$. This range is open. The measured value can vary between $-\pi/2$ and $+\pi/2$, but it cannot hit either limit.

Note that if the **Drum Belt-Cable Alignment** parameter of the Belt-Cable Properties block is set to **Monitored Planar**, the pulley assembly is required to be planar, and the fleet angle is therefore always zero. To model a nonplanar assembly, use the default setting for that parameter: **Unrestricted**.

Use port **qfa** for this measurement.

Extended Capabilities

C/C++ Code Generation

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

See Also

Belt-Cable End | Belt-Cable Properties | Pulley

Introduced in R2018a

Bevel Gear Constraint

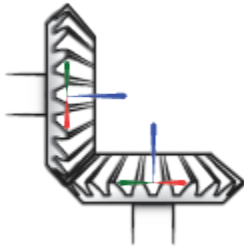
Kinematic constraint between two bevel gear bodies with angled intersecting rotation axes

Library: Simscape / Multibody / Gears and Couplings / Gears



Description

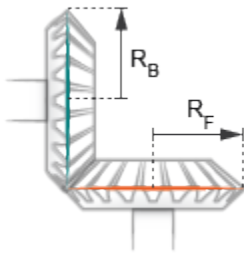
The Bevel Gear Constraint block represents a kinematic constraint between two gear bodies with intersecting rotation axes held at a specified angle. The base and follower frame ports identify the connection frames on the gear bodies. The gear rotation axes coincide with the connection frame z-axes. The gears rotate at a fixed velocity ratio determined by the gear pitch radii.



The block represents only the kinematic constraint characteristic to a bevel gear system. Gear inertia and geometry are solid properties that you must specify using Solid blocks. The gear constraint model is ideal. Backlash and gear losses due to Coulomb and viscous friction between teeth are ignored. You can, however, model viscous friction at joints by specifying damping coefficients in the joint blocks.

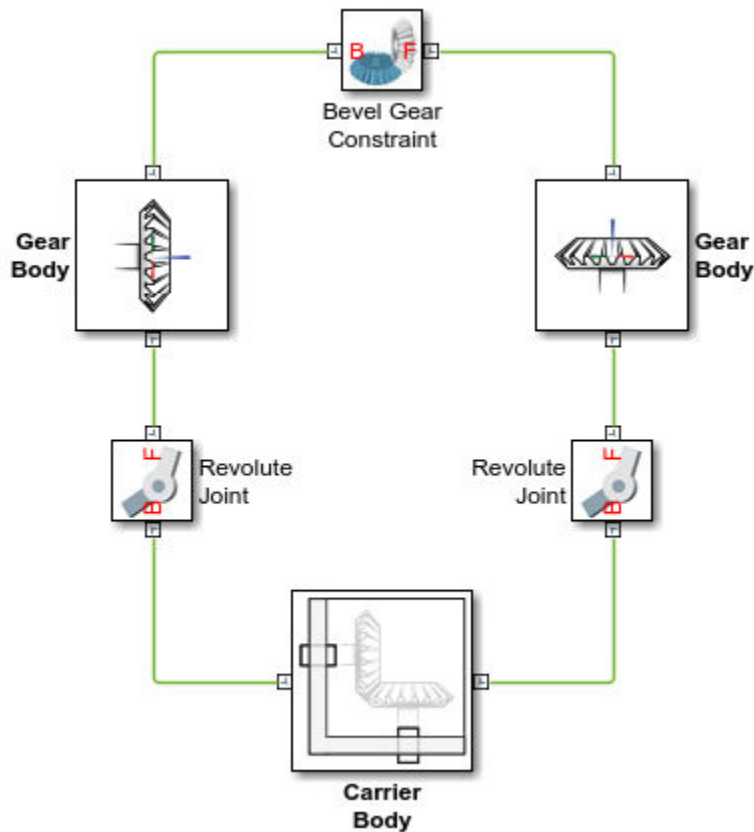
Gear Geometry

The bevel gear constraint is parameterized in terms of the dimensions of the gear pitch circles. The pitch circles are imaginary circles concentric with the gear bodies and tangent to the tooth contact point. The pitch radii, labeled R_B and R_F in the figure, are the outer radii that the gears would have if they were reduced to friction cones in mutual contact.



Gear Assembly

Gear constraints occur in closed kinematic loops. The figure shows the closed-loop topology of a simple bevel gear model. Joint blocks connect the gear bodies to a common fixture or carrier, defining the maximum degrees of freedom between them. A Bevel Gear Constraint block connects the gear bodies, eliminating one degree of freedom and effectively coupling the gear motions.



Assembly Requirements

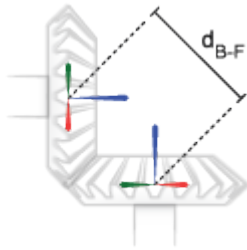
The block imposes special restrictions on the relative positions and orientations of the gear connection frames. The restrictions ensure that the gears assemble only at distances and angles suitable for meshing. The block enforces the restrictions during model assembly, when it first attempts to place the gears in mesh, but relies on the remainder of the model to keep the gears in mesh during simulation.

Position Restrictions

- The distance between the base and follower frame origins must be such that, at the given shaft angle and pitch radii, the gear pitch circles are tangent to each other. This distance, denoted d_{B-F} , follows from the law of cosines:

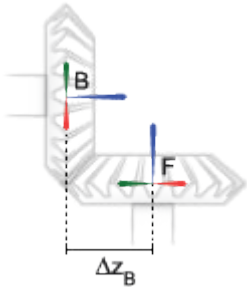
$$d_{B-F} = \sqrt{R_B^2 + R_F^2 - 2R_B R_F \cos(\pi - \theta)}$$

where R_B is the pitch radius of the base gear, R_F is the pitch radius of the follower gear, and θ_{Shaft} is the intersection angle between the rotation axes.



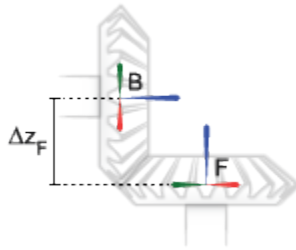
- The distance between the base and follower frame origins along the z-axis of the base frame, denoted Δz_B , must be equal to:

$$\Delta z_B = R_F \cdot \sin(\theta_{\text{Shaft}})$$



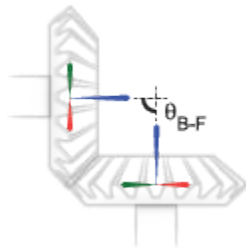
- The distance between the base and follower frame origins along the z-axis of the follower frame, denoted Δz_F , must be equal to:

$$\Delta z_F = R_B \cdot \sin(\theta_{\text{Shaft}})$$



Orientation Restrictions

- The imaginary lines extending from the base and follower z-axes must intersect at the shaft angle set in the block dialog box. The angle is denoted θ_{B-F} in the figure. If the **Shaft Axes** parameter is set to Perpendicular, the angle is 90° .



Ports

Frame

B — Base frame

frame

Connection frame on the base bevel gear

F — Follower frame

frame

Connection frame on the follower bevel gear

Parameters

Base Gear Radius — Radius of the base gear pitch circle

10 cm (default) | positive scalar with units of length

Radius of the base gear pitch circle. The pitch circle is concentric with the gear and tangent to the tooth contact points. The gear radii impact the torque transmission between the base and follower gear bodies.

Follower Gear Radius — Radius of the follower gear pitch circle

10 cm (default) | positive scalar with units of length

Radius of the follower gear pitch circle. The pitch circle is concentric with the gear and tangent to the tooth contact points. The gear radii impact the torque transmission between the base and follower gear bodies.

Shaft Axes — Parameterization for the gear shaft angle

Perpendicular (default) | Arbitrarily Oriented

Parameterization for the intersection angle between the bevel gear shafts. Select Perpendicular to align the gear shafts at a right angle. Select Arbitrarily Oriented to align the gear shafts at any angle from 0 to 180 deg.

Angle Between Shafts — Angle between the base and follower shafts

90 deg (default) | positive scalar

Angle between the imaginary lines extending from the base and follower frame z-axes. The angle must be in the range of 0-180 deg. The actual angle between the base and follower gears, typically set through rigid transforms, joints, and occasionally other constraints, must be the same as that specified here.

Dependencies

This parameter is enabled when the **Shaft Axes** parameter is set to Arbitrarily Oriented.

Extended Capabilities

C/C++ Code Generation

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

See Also

[Bevel Gear Constraint](#) | [Common Gear Constraint](#) | [Rack and Pinion Constraint](#)

Topics

“Bevel Gear”

Introduced in R2013b

Bushing Joint

Joint with three prismatic and three revolute primitives

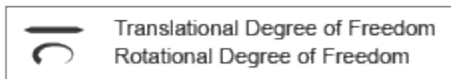
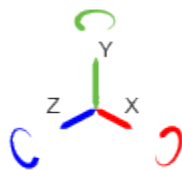


Library

Joints

Description

This block represents a joint with three translational and three rotational degrees of freedom. Three prismatic primitives provide the translational degrees of freedom. Three revolute primitives provide the rotational degrees of freedom.



Joint Degrees of Freedom

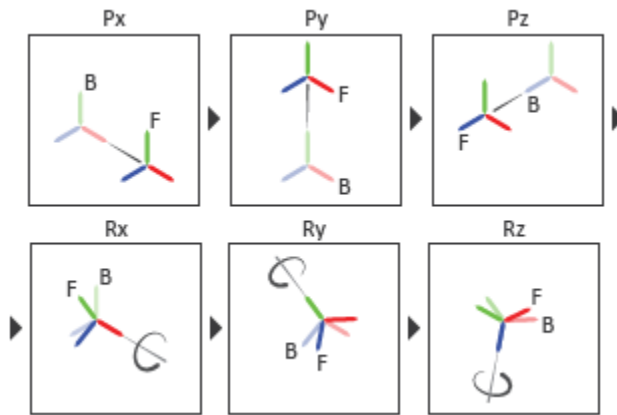
The joint block represents motion between the base and follower frames as a sequence of time-varying transformations. Each joint primitive applies one transformation in this sequence. The transformation translates or rotates the follower frame with respect to the

joint primitive base frame. For all but the first joint primitive, the base frame coincides with the follower frame of the previous joint primitive in the sequence.

At each time step during the simulation, the joint block applies the sequence of time-varying frame transformations in this order:

- 1** Translation:
 - a** Along the X axis of the X Prismatic Primitive (Px) base frame.
 - b** Along the Y axis of the Y Prismatic Primitive (Py) base frame. This frame is coincident with the X Prismatic Primitive (Px) follower frame.
 - c** Along the Z axis of the Z Prismatic Primitive (Pz) base frame. This frame is coincident with the Y Prismatic Primitive (Py) follower frame.
- 2** Rotation:
 - a** About the X axis of the X Revolute Primitive (Rx) base frame. This frame is coincident with the Z Prismatic Primitive (Pz) follower frame.
 - b** About the Y axis of the Y Revolute Primitive (Ry) base frame. This frame is coincident with the X Revolute Primitive (Rx) follower frame.
 - c** About the Z axis of the Z Revolute Primitive (Rz) base frame. This frame is coincident with the Y Revolute Primitive (Ry) follower frame.

The figure shows the sequence in which the joint transformations occur at a given simulation time step. The resulting frame of each transformation serves as the base frame for the following transformation. Because 3-D rotation occurs as a sequence, it is possible for two axes to align, causing to the loss of one rotational degree of freedom. This phenomenon is known as gimbal lock.



Joint Transformation Sequence

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Prismatic Primitive: State Targets

Specify the prismatic primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative position, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative velocity, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is m for position and m/s for velocity.

Prismatic Primitive: Internal Mechanics

Specify the prismatic primitive internal mechanics. Internal mechanics include linear spring forces, accounting for energy storage, and damping forces, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the distance between base and follower frame origins at which the spring force is zero. The default value is 0. Select or enter a physical unit. The default is m.

Spring Stiffness

Enter the linear spring constant. This is the force required to displace the joint primitive by a unit distance. The default is 0. Select or enter a physical unit. The default is N/m.

Damping Coefficient

Enter the linear damping coefficient. This is the force required to maintain a constant joint primitive velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N/(m/s).

Prismatic Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance

along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Prismatic Primitive: Actuation

Specify actuation options for the prismatic joint primitive. Actuation modes include **Force** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Actuation signals are resolved in the base frame.

Force

Select an actuation force setting. The default setting is **None**.

Actuation Force Setting	Description
None	No actuation force.

Actuation Force Setting	Description
Provided by Input	Actuation force from physical signal input. The signal provides the force acting on the follower frame with respect to the base frame along the joint primitive axis. An equal and opposite force acts on the base frame.
Automatically computed	Actuation force from automatic calculation. Simscape Multibody computes and applies the actuation force based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Prismatic Primitive: Sensing

Select the variables to sense in the prismatic joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative position of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Velocity

Select this option to sense the relative velocity of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Acceleration

Select this option to sense the relative acceleration of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Actuator Force

Select this option to sense the actuation force acting on the follower frame with respect to the base frame along the joint primitive axis.

Revolute Primitive: State Targets

Specify the revolute primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative rotation angle, measured about the joint primitive axis, of the follower frame with respect to the base frame. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative angular velocity, measured about the joint primitive axis, of the follower frame with respect to the base frame. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely

Priority Level	Description
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is deg for position and deg/s for velocity.

Revolute Primitive: Internal Mechanics

Specify the revolute primitive internal mechanics. Internal mechanics include linear spring torques, accounting for energy storage, and linear damping torques, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the rotation angle between base and follower frames at which the spring torque is zero. The default value is 0. Select or enter a physical unit. The default is deg.

Spring Stiffness

Enter the linear spring constant. This is the torque required to rotate the joint primitive by a unit angle. The default is 0. Select or enter a physical unit. The default is N*m/deg.

Damping Coefficient

Enter the linear damping coefficient. This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N*m/(deg/s).

Revolute Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or

bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Revolute Primitive: Actuation

Specify actuation options for the revolute joint primitive. Actuation modes include **Torque** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation

mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Input signals are resolved in the base frame.

Torque

Select an actuation torque setting. The default setting is None.

Actuation Torque Setting	Description
None	No actuation torque.
Provided by Input	Actuation torque from physical signal input. The signal provides the torque acting on the follower frame with respect to the base frame about the joint primitive axis. An equal and opposite torque acts on the base frame.
Automatically computed	Actuation torque from automatic calculation. Simscape Multibody computes and applies the actuation torque based on model dynamics.

Motion

Select an actuation motion setting. The default setting is Automatically Computed.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Revolute Primitive: Sensing

Select the variables to sense in the revolute joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each

quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative rotation angle of the follower frame with respect to the base frame about the joint primitive axis.

Velocity

Select this option to sense the relative angular velocity of the follower frame with respect to the base frame about the joint primitive axis.

Acceleration

Select this option to sense the relative angular acceleration of the follower frame with respect to the base frame about the joint primitive axis.

Actuator Torque

Select this option to sense the actuation torque acting on the follower frame with respect to the base frame about the joint primitive axis.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement.

Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Actuation Ports

The prismatic joint primitives provide the following actuation ports:

- f_x, f_y, f_z — Actuation forces acting on the X, Y, and Z prismatic joint primitives
- p_x, p_y, p_z — Desired trajectories of the X, Y, Z prismatic joint primitives

The revolute joint primitives provide the following actuation ports:

- t_x, t_y, t_z — Actuation torques acting on the X, Y, and Z revolute joint primitives
- q_x, q_y, q_z — Desired rotations of the X, Y, and Z revolute joint primitives

Sensing Ports

The prismatic joint primitives provide the following sensing ports:

- p_x, p_y, p_z — Positions of the X, Y, and Z prismatic joint primitives
- v_x, v_y, v_z — Velocities of the X, Y, and Z prismatic joint primitives
- a_x, a_y, a_z — Accelerations of the X, Y, and Z prismatic joint primitives
- f_x, f_y, f_z — Actuation forces acting on the X, Y, and Z prismatic joint primitives

The revolute joint primitives provide the following sensing ports:

- q_x, q_y, q_z — Angular positions of the X, Y, and Z revolute joint primitives
- w_x, w_y, w_z — Angular velocities of the X, Y, and Z revolute joint primitives
- b_x, b_y, b_z — Angular accelerations of the X, Y, and Z revolute joint primitives
- t_x, t_y, t_z — Actuation torques acting on the X, Y, and Z revolute joint primitives

The following sensing ports provide the composite forces and torques acting on the joint:

- f_c — Constraint force
- t_c — Constraint torque
- f_t — Total force
- t_t — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

6-DOF Joint | Gimbal Joint | Prismatic Joint | Revolute Joint

Topics

“Actuating and Sensing with Physical Signals”

“Motion Sensing”

“Rotational Measurements”

“Translational Measurements”

Introduced in R2012a

Cartesian Joint

Joint with three prismatic primitives

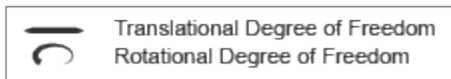
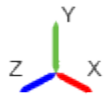


Library

Joints

Description

This block represents a joint with three translational degrees of freedom. Three prismatic primitives provide the three translational degrees of freedom. The base and follower frames remain parallel during simulation.



Joint Degrees of Freedom

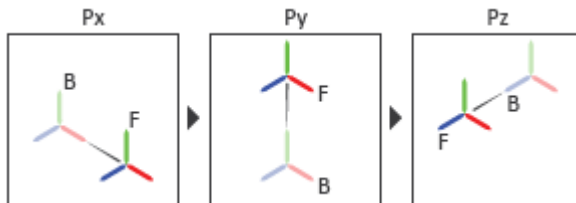
The joint block represents motion between the base and follower frames as a sequence of time-varying transformations. Each joint primitive applies one transformation in this sequence. The transformation translates the follower frame with respect to the joint

primitive base frame. For all but the first joint primitive, the base frame coincides with the follower frame of the previous joint primitive in the sequence.

At each time step during the simulation, the joint block applies the sequence of time-varying frame transformations in this order:

- 1 Translation:
 - a Along the X axis of the X Prismatic Primitive (Px) base frame.
 - b Along the Y axis of the Y Prismatic Primitive (Py) base frame. This frame is coincident with the X Prismatic Primitive (Px) follower frame.
 - c Along the Z axis of the Z Prismatic Primitive (Pz) base frame. This frame is coincident with the Y Prismatic Primitive (Py) follower frame.

The figure shows the sequence in which the joint transformations occur at a given simulation time step. The resulting frame of each transformation serves as the base frame for the following transformation.



Joint Transformation Sequence

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise.

The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Prismatic Primitive: State Targets

Specify the prismatic primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative position, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative velocity, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is m for position and m/s for velocity.

Prismatic Primitive: Internal Mechanics

Specify the prismatic primitive internal mechanics. Internal mechanics include linear spring forces, accounting for energy storage, and damping forces, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the distance between base and follower frame origins at which the spring force is zero. The default value is 0. Select or enter a physical unit. The default is m.

Spring Stiffness

Enter the linear spring constant. This is the force required to displace the joint primitive by a unit distance. The default is 0. Select or enter a physical unit. The default is N/m.

Damping Coefficient

Enter the linear damping coefficient. This is the force required to maintain a constant joint primitive velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N/(m/s).

Prismatic Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper

bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Prismatic Primitive: Actuation

Specify actuation options for the prismatic joint primitive. Actuation modes include **Force** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Actuation signals are resolved in the base frame.

Force

Select an actuation force setting. The default setting is **None**.

Actuation Force Setting	Description
None	No actuation force.
Provided by Input	Actuation force from physical signal input. The signal provides the force acting on the follower frame with respect to the base frame along the joint primitive axis. An equal and opposite force acts on the base frame.
Automatically computed	Actuation force from automatic calculation. Simscape Multibody computes and applies the actuation force based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Prismatic Primitive: Sensing

Select the variables to sense in the prismatic joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative position of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Velocity

Select this option to sense the relative velocity of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Acceleration

Select this option to sense the relative acceleration of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Actuator Force

Select this option to sense the actuation force acting on the follower frame with respect to the base frame along the joint primitive axis.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force

or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame

- F — Follower frame

Actuation Ports

The prismatic joint primitives provide the following actuation ports:

- f_x , f_y , f_z — Actuation forces acting on the X, Y, and Z prismatic joint primitives
- p_x , p_y , p_z — Desired trajectories of the X, Y, Z prismatic joint primitives

Sensing Ports

The prismatic joint primitives provide the following sensing ports:

- p_x , p_y , p_z — Positions of the X, Y, and Z prismatic joint primitives
- v_x , v_y , v_z — Velocities of the X, Y, and Z prismatic joint primitives
- a_x , a_y , a_z — Accelerations of the X, Y, and Z prismatic joint primitives
- f_x , f_y , f_z — Actuation forces acting on the X, Y, and Z prismatic joint primitives

The following sensing ports provide the composite forces and torques acting on the joint:

- f_c — Constraint force
- t_c — Constraint torque
- f_t — Total force
- t_t — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Prismatic Joint | Rectangular Joint

Topics

“Actuating and Sensing with Physical Signals”

“Motion Sensing”

“Translational Measurements”

Introduced in R2012a

Common Gear Constraint

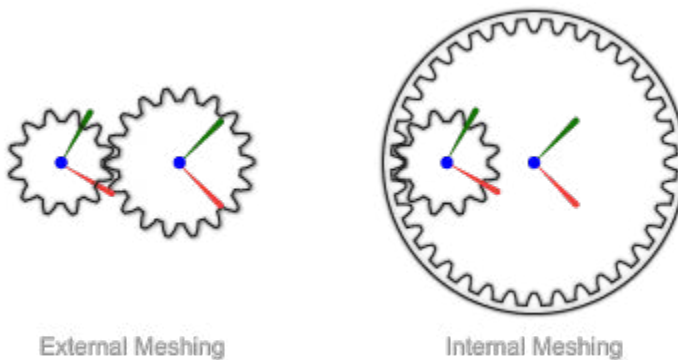
Kinematic constraint between two coplanar spur gear bodies with parallel rotation axes

Library: Simscape / Multibody / Gears and Couplings / Gears



Description

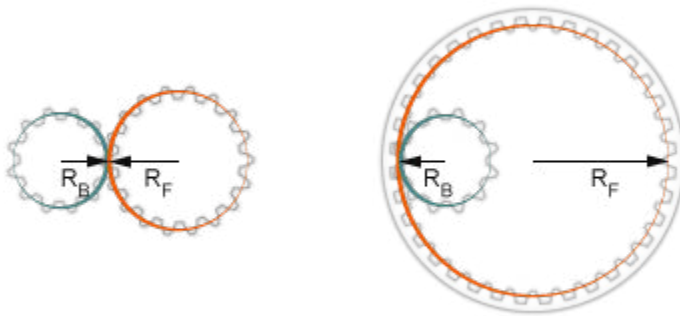
The Common Gear Constraint block represents a kinematic constraint between two coplanar spur gear bodies with parallel rotation axes. The gear meshing can be external to both gears or internal to one of the gears. The base and follower frame ports identify the connection frames on the spur gear bodies. The gear rotation axes coincide with the frame z -axes.



The block represents only the kinematic constraint characteristic to a spur gear system. Gear inertia and geometry are solid properties that you must specify using Solid blocks. The gear constraint model is ideal. Backlash and gear losses due to Coulomb and viscous friction between teeth are ignored. You can, however, model viscous friction at joints by specifying damping coefficients in the joint blocks.

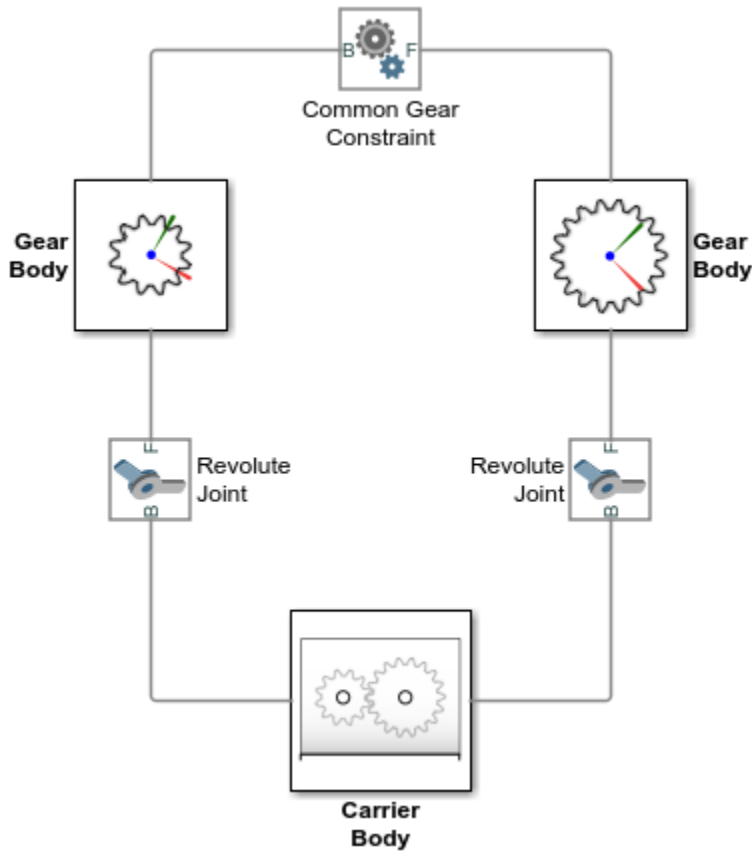
Gear Geometry

The common gear constraint is parameterized in terms of the dimensions of the gear pitch circles. A pitch circle is an imaginary circle concentric with the gear body and tangent to the tooth contact point. The pitch radii, labeled R_B and R_F in the figure, are the radii that the gears would have if they were reduced to friction cylinders in mutual contact.



Gear Assembly

Gear constraints occur in closed kinematic loops. The figure shows the closed-loop topology of a simple common gear model. Joint blocks connect the gear bodies to a common fixture or carrier, defining the maximum degrees of freedom between them. A Common Gear Constraint block connects the gear bodies, eliminating one degree of freedom and effectively coupling the two gear motions.

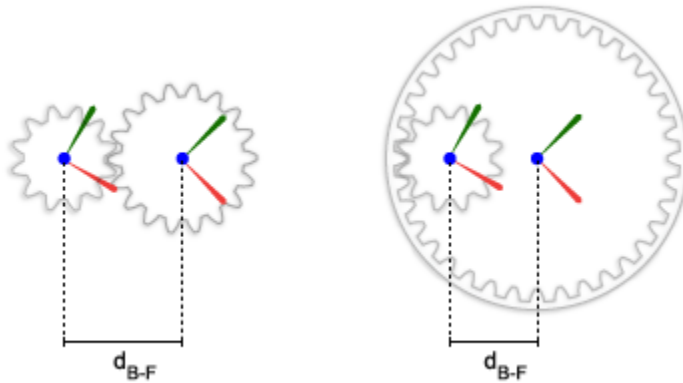


Assembly Requirements

The block imposes special restrictions on the relative positions and orientations of the gear connection frames. The restrictions ensure that the gears assemble only at distances and angles suitable for meshing. The block enforces the restrictions during model assembly, when it first attempts to place the gears in mesh, but relies on the remainder of the model to keep the gears in mesh during simulation.

Position Restrictions

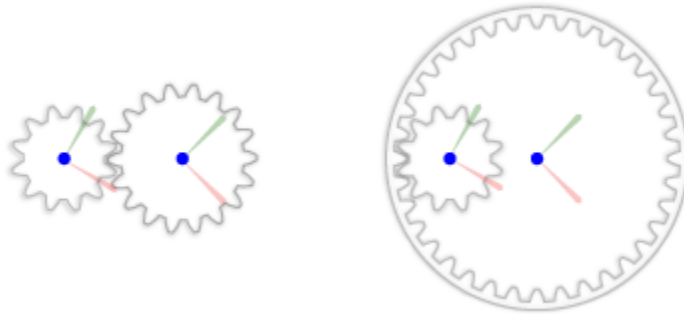
- The distance between the z-axes of the base and follower frame, denoted d_{B-F} in the figure, must equal the distance between the gear centers. This constraint ensures that the rotation axes of the gears are at the proper distance for meshing.



- The follower frame origin must lie on the xy plane of the base frame. This constraint ensures that the pitch circle of one gear is coplanar with the pitch circle of the other.

Orientation Restrictions

- The z-axes of the base and follower frames must point in the same direction. This constraint ensures that the gear rotation axes are parallel to each other. The figure shows the z-axes of the base and follower frames pointing out of the screen.



Ports

Frame

B — Base frame

frame

Connection frame on the base gear body.

F — Follower frame

frame

Connection frame on the follower gear body.

Parameters

Type — Type of meshing between the base and follower gear bodies

External (default) | Internal

Type of meshing between the base and follower gear bodies. Select **External** if both gears have outward-facing teeth. Select **Internal** if one gear has inward-facing teeth. Such a gear is known as a ring gear. The gear with the greater pitch radius serves as the ring gear.

Center Distance — Distance between the base and follower gear centers

20 cm (default) | positive scalar in units of length

Distance between the centers of the base and follower gear bodies. This distance is the sum of the base and follower gear pitch radii.

Dependencies

This parameter is enabled when the **Specification Method** parameter is set to Center Distance and Ratio.

Gear Ratio (Nf/Nb) — Ratio of follower gear teeth to base gear teeth

1.0 (default) | unitless positive scalar

Number of follower gear teeth divided by the number of base gear teeth. The block uses this ratio to determine the speed and torque transmitted between the base and follower gear shafts.

Dependencies

This parameter is enabled when the **Specification Method** parameter is set to Center Distance and Ratio.

Base Gear Radius — Radius of the pitch circle of the base gear body

10 cm (default) | positive scalar in units of length

Radius of the pitch circle of the base gear body. The pitch circle is an imaginary circle concentric with the gear body and tangent to the tooth contact point.

Dependencies

This parameter is enabled when the **Specification Method** parameter is set to Pitch Circle Radii.

Follower Gear Radius — Radius of the pitch circle of the follower gear body

10 cm (default) | positive scalar in units of length

Radius of the pitch circle of the follower gear body. The pitch circle is an imaginary circle concentric with the gear body and tangent to the tooth contact point.

Dependencies

This parameter is enabled when the **Specification Method** parameter is set to Pitch Circle Radii.

Extended Capabilities

C/C++ Code Generation

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

See Also

Bevel Gear Constraint | Rack and Pinion Constraint | Worm and Gear Constraint

Topics

“External Spur Gear”

“Internal Spur Gear”

Introduced in R2013a

Constant Velocity Joint

Joint with two rotational DoFs between shafts constrained to spin with equal velocity



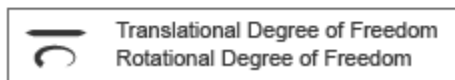
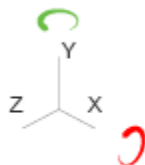
Library

Joints

Description

This block represents a joint with two rotational degrees of freedom constrained to maintain a constant angular velocity about the base and follower Z axes. The base and follower frame origins remain coincident throughout simulation.

The joint applies three rotation transformations between the base and follower frames in the sequence azimuth \rightarrow bend angle \rightarrow -azimuth. Each transformation takes place relative to the intermediate frame resulting from any prior transformations. For example, the bend angle transformation takes place relative to the intermediate frame resulting from the azimuth transformation.



Joint Degrees of Freedom

A set of optional state targets guide assembly for the joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Optional sensing ports output the joint primitive motion through physical signals. Motion variables that you can sense include joint position, velocity, and acceleration. Selecting a variable in the Sensing menu exposes the physical signal port for that variable.

Parameters

Constant Velocity Primitive: State Targets

Specify Position Target

Desired joint primitive position at the start of simulation. This is the relative angular position of the follower frame relative to the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Desired joint velocity at the start of simulation. This is the relative angular velocity of the follower frame relative to the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines

which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Joint primitive angles to specify. Angles include bend and azimuth angles.

Value: **Bend Angle**

Angle between the base and follower frame Z axes. The block applies this angle about the rotated Y axis resulting from the azimuth transformation. At zero bend angle, the follower frame Z axis is coincident with the base frame Z axis.

Value: **Azimuth**

Angle about the base frame Z axis prior to bending. At zero azimuth, the base and follower Z axes are in the XZ plane of the base frame.

Constant Velocity Primitive: Sensing

Select the variables to sense in the constant velocity joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame.

Variable	Description
Bend Angle: Position	Angle between the base and follower frame Z axes
Bend Angle: Velocity	First time derivative of the bend angle.
Bend Angle: Acceleration	Second time derivative of the bend angle.
Azimuth: Position	Angle about the base frame Z axis prior to bending.

Variable	Description
Azimuth: Velocity	First time derivative of the azimuth angle.
Azimuth: Acceleration	Second time derivative of the azimuth angle.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Sensing Ports

The constant velocity joint primitive provides the following sensing ports:

- qb — Bend angle
- wb — First time-derivative of the bend angle
- bb — Second time-derivative of the bend angle
- qa — Azimuth angle
- wa — First time-derivative of the azimuth angle
- ba — Second time-derivative of the azimuth angle

The following sensing ports provide the composite forces and torques acting on the joint:

- fc — Constraint force
- tc — Constraint torque

- ft — Total force
- tt — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Universal Joint

Introduced in R2015a

Cylindrical Joint

Joint with one prismatic and one revolute primitives possessing parallel motion axes

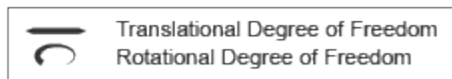
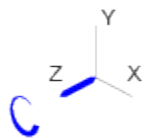


Library

Joints

Description

This block represents a joint with one translational and one rotational degree of freedom. One prismatic primitive provides the translational degree of freedom. One revolute primitive provides the rotational degree of freedom. The translation and rotation axes remain aligned during simulation.



Joint Degrees of Freedom

The joint block represents motion between the base and follower frames as a sequence of time-varying transformations. Each joint primitive applies one transformation in this sequence. The transformation translates or rotates the follower frame with respect to the

joint primitive base frame. For all but the first joint primitive, the base frame coincides with the follower frame of the previous joint primitive in the sequence.

At each time step during the simulation, the joint block applies the sequence of time-varying frame transformations in this order:

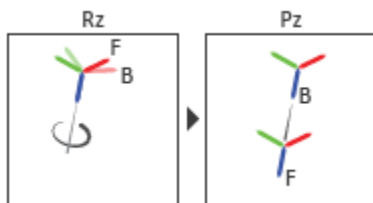
1 Rotation:

- About the Z axis of the Z Revolute Primitive (Rz) base frame.

2 Translation:

- Along the Z axis of the Z Prismatic Primitive (Pz) base frame.

The figure shows the sequence in which the joint transformations occur at a given simulation time step. The resulting frame of each transformation serves as the base frame for the following transformation.



Joint Transformation Sequence

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Revolute Primitive: State Targets

Specify the revolute primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative rotation angle, measured about the joint primitive axis, of the follower frame with respect to the base frame. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative angular velocity, measured about the joint primitive axis, of the follower frame with respect to the base frame. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is deg for position and deg/s for velocity.

Revolute Primitive: Internal Mechanics

Specify the revolute primitive internal mechanics. Internal mechanics include linear spring torques, accounting for energy storage, and linear damping torques, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the rotation angle between base and follower frames at which the spring torque is zero. The default value is 0. Select or enter a physical unit. The default is deg.

Spring Stiffness

Enter the linear spring constant. This is the torque required to rotate the joint primitive by a unit angle. The default is 0. Select or enter a physical unit. The default is N*m/deg.

Damping Coefficient

Enter the linear damping coefficient. This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N*m/(deg/s).

Revolute Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Revolute Primitive: Actuation

Specify actuation options for the revolute joint primitive. Actuation modes include **Torque** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Input signals are resolved in the base frame.

Torque

Select an actuation torque setting. The default setting is None.

Actuation Torque Setting	Description
None	No actuation torque.
Provided by Input	Actuation torque from physical signal input. The signal provides the torque acting on the follower frame with respect to the base frame about the joint primitive axis. An equal and opposite torque acts on the base frame.
Automatically computed	Actuation torque from automatic calculation. Simscape Multibody computes and applies the actuation torque based on model dynamics.

Motion

Select an actuation motion setting. The default setting is Automatically Computed.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Revolute Primitive: Sensing

Select the variables to sense in the revolute joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative rotation angle of the follower frame with respect to the base frame about the joint primitive axis.

Velocity

Select this option to sense the relative angular velocity of the follower frame with respect to the base frame about the joint primitive axis.

Acceleration

Select this option to sense the relative angular acceleration of the follower frame with respect to the base frame about the joint primitive axis.

Actuator Torque

Select this option to sense the actuation torque acting on the follower frame with respect to the base frame about the joint primitive axis.

Prismatic Primitive: State Targets

Specify the prismatic primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative position, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative velocity, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is m for position and m/s for velocity.

Prismatic Primitive: Internal Mechanics

Specify the prismatic primitive internal mechanics. Internal mechanics include linear spring forces, accounting for energy storage, and damping forces, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the distance between base and follower frame origins at which the spring force is zero. The default value is 0. Select or enter a physical unit. The default is m.

Spring Stiffness

Enter the linear spring constant. This is the force required to displace the joint primitive by a unit distance. The default is 0. Select or enter a physical unit. The default is N/m.

Damping Coefficient

Enter the linear damping coefficient. This is the force required to maintain a constant joint primitive velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N/(m/s).

Prismatic Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper

bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Prismatic Primitive: Actuation

Specify actuation options for the prismatic joint primitive. Actuation modes include **Force** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Actuation signals are resolved in the base frame.

Force

Select an actuation force setting. The default setting is **None**.

Actuation Force Setting	Description
None	No actuation force.
Provided by Input	Actuation force from physical signal input. The signal provides the force acting on the follower frame with respect to the base frame along the joint primitive axis. An equal and opposite force acts on the base frame.
Automatically computed	Actuation force from automatic calculation. Simscape Multibody computes and applies the actuation force based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Prismatic Primitive: Sensing

Select the variables to sense in the prismatic joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative position of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Velocity

Select this option to sense the relative velocity of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Acceleration

Select this option to sense the relative acceleration of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Actuator Force

Select this option to sense the actuation force acting on the follower frame with respect to the base frame along the joint primitive axis.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force

or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame

- F — Follower frame

Actuation Ports

The prismatic joint primitive provides the following actuation ports:

- fz — Actuation force acting on the Z prismatic joint primitive
- pz — Desired trajectory of the Z prismatic joint primitive

The revolute joint primitive provides the following actuation ports:

- tz — Actuation torque acting on the Z revolute joint primitive
- qz — Desired rotation of the Z revolute joint primitive

Sensing Ports

The prismatic joint primitive provides the following sensing ports:

- pz — Position of the Z prismatic joint primitive
- vz — Velocity of the Z prismatic joint primitive
- az — Acceleration of the Z prismatic joint primitive
- fz — Actuation force acting on the Z prismatic joint primitive

The revolute joint primitive provides the following sensing ports:

- qz — Angular position of the Z revolute joint primitive
- wz — Angular velocity of the Z revolute joint primitive
- bz — Angular acceleration of the Z revolute joint primitive
- tz — Actuation torque acting on the Z revolute joint primitive

The following sensing ports provide the composite forces and torques acting on the joint:

- fc — Constraint force
- tc — Constraint torque
- ft — Total force
- tt — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Prismatic Joint | Revolute Joint

Topics

“Actuating and Sensing with Physical Signals”

“Motion Sensing”

“Rotational Measurements”

“Translational Measurements”

Introduced in R2012a

Distance Constraint

Fixed distance between two frame origins

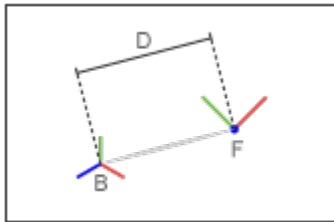


Library

Constraints

Description

This block applies a fixed distance between the origins of the base and follower port frames. The frames lose one translational degree of freedom with respect to each other. The constraint distance between the frame origins, labeled D in the figure, must be greater than zero.



The block provides constraint force sensing in the form of a vector or a signed magnitude. These quantities are contained in physical signals that the block outputs through Simscape PS ports. The constraint force is the force required to maintain the specified distance between the port frame origins.

Parameters

Distance

Constraint distance between the base and follower frame origins. The distance must be greater than zero. For a distance of zero, use a Spherical Joint or Gimbal Joint block instead. The default value is 1 m.

Constraint Force Sensing

Select whether to compute and output the distance constraint force vector and its signed magnitude. The distance constraint force is the force that the block must apply in order to maintain the distance you specify between the base and follower port frames.

Direction

Constraint forces act in pairs. As expressed by Newton's third law of motion, if the base port frame exerts a constraint force on the follower port frame, then the follower port frame must exert an equal and opposite force on the base port frame. Select which of the two constraint forces to sense:

- **Follower on Base** — Sense the constraint force that the follower port frame exerts on the base port frame.
- **Base on Follower** — Sense the constraint force that the base port frame exerts on the follower port frame.

Resolution Frame

The block expresses the constraint force vector in terms of its Cartesian vector components. The splitting of a vector into vector components is known as vector resolution. The frame whose axes define the vector component directions is known as the resolution frame. Select whether to resolve the constraint force vector in the base or follower port frame.

Force Vector

Compute and output the Cartesian components of the distance constraint force vector. The output signal is a three-dimensional vector, $[f_x, f_y, f_z]$.

Signed Force Magnitude

Compute and output the magnitude of the distance constraint force, including its sign.

Ports

The block provides two frame ports:

- B — Base frame port
- F — Follower frame port

In addition, the block provides two physical signal output ports:

- f — Distance constraint force vector
- f_m — Signed magnitude of the distance constraint force

Extended Capabilities

C/C++ Code Generation

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

See Also

[Angle Constraint](#) | [Bevel Gear Constraint](#) | [Common Gear Constraint](#) | [Point on Curve Constraint](#) | [Rack and Pinion Constraint](#)

Introduced in R2012a

External Force and Torque

General force and torque arising outside the modeled system



Library

Forces and Torques

Description

This block represents a general force and torque that an external agency applies on a rigid body frame. The force and torque input can be constant or it can vary with time. The block provides a set of physical signal ports that you use to specify this input. The ports are hidden by default. Selecting an actuation mode exposes the corresponding physical signal port.

Each force and torque input acts on the origin of the follower frame in a direction that depends on the chosen force type and resolution frame. A force/torque vector component acts along/about the associated Cartesian axis, resolved in the chosen resolution frame. For example, the **Force(X)** input acts along the X axis of the resolution frame. A complete force/torque vector acts in the general direction that its components specify, resolved in the chosen resolution frame.

A force input with a positive value acts along the positive direction of the associated axis. A torque input with a positive value acts about the associated axis according to the right-hand rule.

Parameters

Actuation: Force

Select the force inputs to specify and the frame to resolve them in.

Force Resolution Frame

Select the frame to resolve each force signal in. The components of the force vector align with the axes of the resolution frame. The default setting is **Attached Frame**. The table summarizes the resolution frames that you can select.

Resolution Frame	Description
World	Resolve each force component in the World frame of the model.
Attached Frame	Resolve each force component in the follower frame of the External Force and Torque block. The follower frame is the attached frame of the block.

Force Inputs

Select the force inputs to specify. Options include the complete force vector and the separate components of that vector. Selecting a force input exposes the physical signal port associated with that input. Use that port to specify the force input via physical signals. The table summarizes the force inputs that you can select.

Force Input	Description	Input
Force(X) , Force(Y) , Force(Z)	Specify separately the force components acting on the origin of the follower frame along the X, Y, and Z axes of the resolution frame	Scalar

Force Input	Description	Input
Force	Specify the complete force vector [fx fy fz] acting on the origin of the follower frame along the X, Y, and Z axes of the resolution frame	Three-element vector

Actuation: Torque

Select the torque inputs to specify and the frame to resolve them in.

Torque Resolution Frame

Select the frame to resolve each torque signal in. The components of the torque vector align with the axes of the resolution frame. The default setting is **Attached Frame**. The table summarizes the resolution frames that you can select.

Resolution Frame	Description
World	Resolve each torque component in the World frame of the model.
Attached Frame	Resolve each torque component in the follower frame of the External Force and Torque block. The follower frame is the attached frame of the block.

Torque Inputs

Select the torque inputs to specify. Options include the complete torque vector and the separate components of that vector. Selecting a torque input exposes the physical signal port associated with that input. Use that port to specify the torque input via physical signals. The table summarizes the torque inputs that you can select.

Force Input	Description	Input
Torque(X), Torque(Y), Torque(Z)	Specify separately the torque components acting on the origin of the follower frame about the X, Y, and Z axes of the resolution frame	Scalar
Torque	Specify the complete torque vector [fx fy fz] acting on the origin of the follower frame about a general direction in the resolution frame	Three-element vector

Ports

The block contains frame port F, representing the follower frame. Selecting an actuation mode exposes additional physical signal ports. Use the ports to input the selected actuation signals.

Each physical signal port has a unique label. The table identifies the actuation modes that the port labels correspond to.

Port Label	Description
fx, fy, fz	Force vector components acting on the origin of the follower frame along the X, Y, and Z axes, respectively
f	Force vector [fx, fy, fz] acting on the origin of the follower frame along a general direction [X Y Z]
tx, ty, tz	Torque vector components acting on the origin of the follower frame about the X, Y, and Z axes, respectively
t	Torque vector [tx ty tz] acting on the origin of the follower frame about a general direction [X Y Z]

Extended Capabilities

C/C++ Code Generation

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

See Also

Internal Force | Inverse Square Law Force | Spring and Damper Force

Topics

“Actuating and Sensing with Physical Signals”

Introduced in R2012a

File Solid

Solid element with properties derived from external file

Library: Simscape / Multibody / Body Elements



Description

The File Solid block models a solid element with geometry, inertia, color, and reference frame derived from an external file. The file must be of a part model, which is to say that it contains at least solid geometry data. Some formats may provide color and inertia data, though such properties can be specified manually if need be.

Among the supported formats are those native to CATIA (V4, V5, and V6), Creo, Inventor, Unigraphics NX, Solid Edge, SolidWorks, and Parasolid (all CAD applications common in industry and academia). These include CATPART, PRT, IPT, SLDPRT, and X_T (and its binary version, X_B). Other valid formats, not associated with a specific application but common in 3-D modeling, include SAT (often referred to as ACIS), JT, STL, and STEP.

(CAD drawing and assembly files, which do not contain the necessary data for a solid element, cannot be imported to the block.)

Inertia Calculations

For part model files with density data, the block gives the option to (automatically) set the mass, center of mass, and inertia tensor of the solid from calculation. This behavior is enabled by default (through the **Type** and **Based On** parameters under the **Inertia** node, which, in their original states, will read Calculate from Geometry and Density from File).

If the imported file does not contain density data, you must specify it (or, equivalently, mass) for the calculations to be made. Set the **Based On** parameter to Custom Density or Custom Mass to enter the missing data.

Alternatively, if you have the complete mass properties of the imported part—often provided, for CAD models, by the CAD application itself—you can enter them directly as block parameters. Set the inertia **Type** parameter to **Custom** in order to do this.


Note that the frame in which the moments and products of inertia are defined will vary among CAD applications. In this block, the origin of that frame is assumed to be at the center of mass (and its axes parallel to those of the reference frame). This frame is referred to here as the inertia resolution frame. (The center of mass, on the other hand, is defined in the reference frame.) For more information, see “Specifying Custom Inertias”.

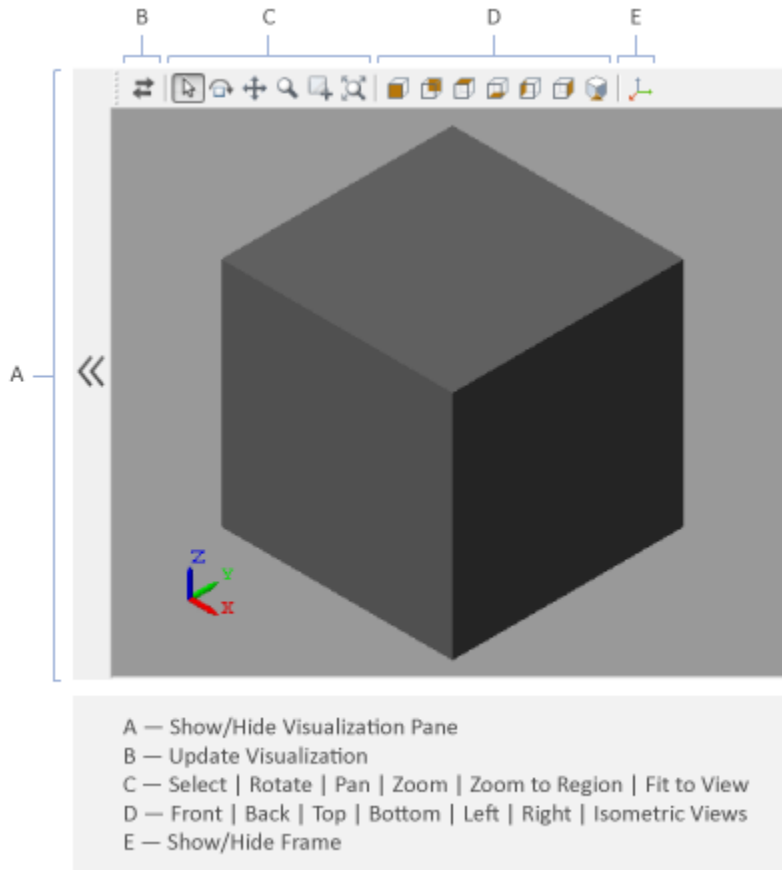
Derived Values

If the mass properties are computed from geometry, you can view their values in the block dialog box. To do so, expand the **Derived Values** node under **Inertia** and click **Update**. (This feature, as it is specified to computed properties, requires that the inertia **Type** setting be **Calculated from Geometry**.) If a geometry or inertia block parameter changes, click the **Update** button once again to display the new mass properties. All values are in SI units of length (m) and mass (kg).

Solid Visualization

The block dialog box contains a collapsible visualization pane. This pane provides instant visual feedback on the solid you are modeling. Use it to find and fix any issues with the shape and color of the solid. You can examine the solid from different perspectives by selecting a standard view or by rotating, panning, and zooming the solid.

Select the Update Visualization button  to view the latest changes to the solid geometry in the visualization pane. Select **Apply** or **OK** to commit your changes to the solid. Closing the block dialog box without first selecting **Apply** or **OK** causes the block to discard those changes.



Solid Visualization Pane

Right-click the visualization pane to access the visualization context-sensitive menu. This menu provides additional options so that you can change the background color, split the visualization pane into multiple tiles, and modify the view convention from the default **+Z up (XY Top)** setting.




Connection Frames

Like most components, the solid connects through frames, of which it has at least one. The default frame, which serves as its reference and is associated with port **R**, gets its

origin and axes from the data in the imported file. (The origin is generally the zero coordinate of the CAD model or, if such technology is used, the 3-D scan, contained in the file.)

For those cases in which the reference frame is ill-placed for connection, or in which multiple connection frames are needed, the block comes with a frame creation tool. Treat this tool as an interactive alternative to the Rigid Transform block (the latter a numerical means to add and translate as well as rotate frames, though one that keeps the frames separate from the solid).

You can create (and edit) frames using geometry features as constraints—placing the frame origin on, and orienting the frame axes along, selected vertices, edges, and faces. You can also use the reference frame origin and its axes, as well as the center of mass and the principal inertia axes, to define the new frames. Each frame adds to the block a new frame port (its label derived from the name given in the frame creation pane).

To create or edit a frame, first expand the **Frames** node in the block dialog box. Click the  button to create a frame or the  button to edit a frame (if one, other than the reference frame, already exists). The frame definitions depend on a mix of geometry and inertia data, so you must have previously imported a part geometry file. If a block parameter changes, you must refresh the visualization pane (by clicking the  button) in order to create or edit a frame.

Frame Definition

A custom frame is fully defined when its origin and axes are too. Of these, the axes require the most care. You must specify two axes, one primary and one secondary. The primary axis defines the plane (that normal to it) on which the other axes must lie. The secondary axis is merely the projection of a selected direction—axis or geometry feature—on that plane.

The remaining (and unspecified) axis is set by requiring that all three be perpendicular and ordered according to the right-hand rule. Naturally, the secondary axis must have a vector component perpendicular to the primary axis. If the two are parallel, the frame is invalid. If the frame is then saved, its orientation is set to that of the reference frame.

To use a geometry feature for the frame origin or axis definitions:

- 1 In the frame creation pane, select the **Based on Geometric Feature** radio button.
- 2 In the solid visualization pane, click a vertex, edge, or face. Zoom in, if necessary, to more precisely select a feature.

- 3 Again in the frame creation pane, click the **Use Selected Feature** button.

MATLAB Variables

It is common in a model to parameterize blocks in terms of MATLAB variables. Instead of a scalar, vector, or string, for example, a block parameter will have in its field the name of a variable. The variable is defined elsewhere, often in a subsystem mask or in the model workspace, sometimes by reference to an external M file.

This approach suits complex models in which multiple blocks must share the same parameter value—a common density, say, or color, if defined as an RGB vector. When the MATLAB variable definition then changes, so do all block parameters that depend on it. Consider using MATLAB variables here if a parameter is likely to be shared by several blocks in a large model.

(For a simple example with Solid blocks parameterized in terms of workspace variables, open the `sm_compound_body` model)

Ports

Frame

R — Reference frame

frame

Frame by which to connect the solid in a model. The frame node to which this port connects—generally another frame port or a frame junction—determines the position and orientation of the solid relative to other components. Add a Rigid Transform block between the port and the node if the frames they represent must be offset from one another.

Parameters

Geometry

From File: File Type — Type of geometry file to import

STEP (default) | STL

Type of geometry file to import. Automatic inertia calculation is available for STEP files only. You must specify all inertial properties explicitly for STL files.

File Name — Name of the part model file to import

custom character vector

Name and extension of the part model file to import. If the file is not on the MATLAB path, the file location must be specified. The file location can be specified as an absolute path, starting from the root directory of the file system—e.g., 'C:/Users/JDoe/Documents/myShape.STEP'. It can also be specified as a relative path, starting from a folder on the MATLAB path—e.g., 'Documents/myShape.STEP'.

Unit Type — Source for solid geometry units

From File (default) | Custom

Source of the solid geometry units. Select **From File** to use the units specified in the imported file. Select **Custom** to specify your own units.

Unit — Length units in which geometry coordinates are specified in the imported file

m (default) | cm | mm | km | in | ft

Length units in which to interpret the vertex coordinates provided in the STL geometry file. Changing the units changes the scale of the imported geometry.

Inertia

Type — Inertia parameterization to use

Calculate from Geometry (default) | Point Mass | Custom

Inertia parameterization to use. Select **Point Mass** to model a concentrated mass with negligible rotational inertia. Select **Custom** to model a distributed mass with the specified moments and products of inertia. The default setting, **Calculate from Geometry**, enables the block to automatically calculate the rotational inertia properties from the solid geometry and either density or mass.

Based on — Parameter to base inertia calculation on

Density from File (default) | Custom Density | Custom Mass

Parameter to use in inertia calculation. The block calculates the inertia tensor from the solid geometry and the parameter selected.

Use the default setting of **Density** from **File** to base the calculations on the density obtained from the imported file. (Note that only some formats can carry density data. Of those that do, only some will actually carry it. Often this data is specified in a CAD application before saving or exporting the part model file.)

Use **Custom Density** to specify a density other than that obtained from the imported file. Use **Custom Mass** to instead specify the total mass of the solid.

Density – Mass per unit volume of material

1000 kg/m³ (default)

Mass per unit volume of material. The mass density can take on a positive or negative value. Specify a negative mass density to model the effects of a void or cavity in a solid body.

Mass – Total mass of the solid element

1 kg (default) | scalar with units of mass

Total mass to attribute to the solid element. This parameter can be positive or negative. Use a negative value to capture the effect of a void or cavity in a compound body (one comprising multiple solids and inertias), being careful to ensure that the mass of the body is on the whole positive.

Custom: Center of Mass – Center-of-mass coordinates

[0 0 0] m (default) | three-element vector with units of length

[x y z] coordinates of the center of mass relative to the block reference frame. The center of mass coincides with the center of gravity in uniform gravitational fields only.

Custom: Moments of Inertia – Diagonal elements of inertia tensor

[1 1 1] kg*m² (default) | three-element vector with units of mass*length²

Three-element vector with the [I_{xx} I_{yy} I_{zz}] moments of inertia specified relative to a frame with origin at the center of mass and axes parallel to the block reference frame. The moments of inertia are the diagonal elements of the inertia tensor

$$\begin{pmatrix} I_{xx} & & \\ & I_{yy} & \\ & & I_{zz} \end{pmatrix},$$

where:

- $I_{xx} = \int_V (y^2 + z^2) dm$
- $I_{yy} = \int_V (x^2 + z^2) dm$
- $I_{zz} = \int_V (x^2 + y^2) dm$

Custom: Products of Inertia – Off-diagonal elements of inertia tensor

[0 0 0] kg*m² (default) | three-element vector with units of mass*length²

Three-element vector with the [I_{yz} I_{zx} I_{xy}] products of inertia specified relative to a frame with origin at the center of mass and axes parallel to the block reference frame. The products of inertia are the off-diagonal elements of the inertia tensor

$$\begin{pmatrix} I_{xy} & I_{zx} \\ I_{xy} & I_{yz} \\ I_{zx} & I_{yz} \end{pmatrix},$$

where:

- $I_{yz} = - \int_V yz dm$
- $I_{zx} = - \int_V zx dm$
- $I_{xy} = - \int_V xy dm$

Derived Values – Display of calculated values of mass properties

button

Display of the calculated values of the solid mass properties—mass, center of mass, moments of inertia, and products of inertia. Click the **Update** button to calculate and display the mass properties of the solid. Click this button following any changes to the block parameters to ensure that the displayed values are still current.

The center of mass is resolved in the local reference frame of the solid. The moments and products of inertia are each resolved in the inertia frame of resolution—a frame whose

axes are parallel to those of the reference frame but whose origin coincides with the solid center of mass.

Dependencies

The option to calculate and display the mass properties is active when the **Inertia > Type** block parameter is set to `Calculate` from `Geometry`.

Graphic

Type — Graphic to use in the visualization of the solid

From `Geometry` (default) | `Marker` | `None`

Choice of graphic to use in the visualization of the solid. The graphic is by default the geometry specified for the solid. Select `Marker` to show instead a simple graphic marker, such as a sphere or cube. Change this parameter to `None` to eliminate this solid altogether from the model visualization.

Marker: Shape — Shape of the marker to assign to the solid

`Sphere` (default) | `Cube` | `Frame`

Shape of the marker by means of which to visualize the solid. The motion of the marker reflects the motion of the solid itself.

Marker: Size — Width of the marker in pixels

10 (default) | scalar with units of pixels

Width of the marker in pixels. This width does not scale with zoom level. Note that the apparent size of the marker depends partly on screen resolution, with higher resolutions packing more pixels per unit length, and therefore producing smaller icons.

Visual Properties — Parameterizations for color and opacity

`Simple` (default) | `Advanced` | `From File`

Parameterization for specifying visual properties. Select `Simple` to specify color and opacity. Select `Advanced` to add specular highlights, ambient shadows, and self-illumination effects. Select `From File` if the imported file has color data and you want to use it in the model.

(Only some file formats allow color data. In those that do, that data is often optional. If your file does not specify color, the solid will take on a gray hue (the default solid color). Select another parameterization to customize color in such cases.)

Simple: Color — True color as [R,G,B] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three-element vector with values constrained to 0-1

RGB color vector with red (R), green (G), and blue (B) color amounts specified on a 0-1 scale. A color picker provides an alternative interactive means of specifying a color. If you change the **Visual Properties** setting to Advanced, the color specified in this parameter becomes the **Diffuse Color** vector.

Simple: Opacity — Surface opacity as scalar number on 0-1 scale

1.0 (default) | scalar with value constrained to 0-1

Graphic opacity specified on a scale of 0-1. An opacity of 0 corresponds to a completely transparent graphic and an opacity of 1 to a completely opaque graphic.

Advanced: Diffuse Color — True color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three- or four-element vector with values constrained to 0-1

True color under direct white light specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. An optional fourth element specifies the color opacity also on a scale of 0-1. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Specular Color — Highlight color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of specular highlights specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Ambient Color — Shadow color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of shadow areas in diffuse ambient light, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Emissive Color — Self-illumination color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Surface color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Shininess — Highlight sharpness as scalar number on 0-128 scale
75 (default) | scalar with value constrained to 0-128




Sharpness of specular light reflections, specified as a scalar number on a 0-128 scale. Increase the shininess value for smaller but sharper highlights. Decrease the value for larger but smoother highlights.

Frames

Show Port R — Show the reference frame port for connection to other blocks
checked (default) | cleared

Clear the check box to hide the reference frame port in the Solid block. Hiding the reference frame port suppresses the frame visualization in Mechanics Explorer. You must expose the reference frame port if the block has no custom frames.

New Frame — Create a custom frame for connection to other blocks
empty (default)

Select the Create button  to define a new frame using the frame-creation interface. Each new frame appears on a row above the **New Frame** parameter. To edit an existing frame, select the Edit button . To delete an existing frame, select the Delete button .

Frame Creation Interface

Frame Name — MATLAB® string used to identify the custom frame
custom character vector

Frame identifier specified as a MATLAB string. This string identifies the frame port in the block diagram and in the tree view pane of Mechanics Explorer. Keep the frame name short to ensure it fits in the block icon width.

Frame Origin — Position of the custom frame origin
At Reference Frame Origin (default) | **At Center of Mass** | **Based on Geometric Feature**

Select the location of the frame origin. Options include:

- **At Reference Frame Origin** — Make the new frame origin coincident with the reference frame origin. This is the default option.
- **At Center of Mass** — Make the new frame origin coincident with the solid center of mass. The reference frame origin is located at the center of mass in symmetrical shapes such as spheres and bricks but not in certain extrusions or revolutions.
- **Based on Geometric Feature** — Place the new frame origin at the center of the selected geometry feature. Valid geometry features include surfaces, lines, and points. You must select a geometry feature from the visualization pane and then select the **Use Selected Feature** button. The name of the selected geometry feature appears in the field below this option.

Frame Axes: Primary Axis — Axis used to constrain the possible directions of the remaining frame axes

Along Reference Frame Axis (default) | **Along Principal Inertia Axis** | **Based on Geometric Feature**

Select the axis of the new frame that you want to set as the primary axis. The primary axis constrains the possible orientations of the remaining two axes. Specify the orientation of the primary axis by selecting from the following options:

- **Along Reference Frame Axis** — Align the primary axis with the selected axis of the reference frame.
- **Along Principal Inertia Axis** — Align the primary axis with the selected principal inertia axis. The principal inertia axes are those about which the products of inertia are zero.
- **Based on Geometric Feature** — Align the primary axis with the vector associated with the selected geometric feature. Valid geometric features include surfaces and lines.

Frame Axes: Secondary Axis — Axis used to constrain the possible directions of the remaining frame axis

Along Reference Frame Axis (default) | **Along Principal Inertia Axis** | **Based on Geometric Feature**

Select the axis of the new frame that you want to set as the secondary axis. The secondary axis is the projection of the selected direction onto the normal plane of the primary axis. Select the direction to project from the following options:

- **Along Reference Frame Axis** — Project the selected reference frame axis onto the normal plane of the primary axis. Align the secondary axis with the projection.

- **Along Principal Inertia Axis** — Project the selected principal inertia axis onto the normal plane of the primary axis. Align the secondary axis with the projection. The principal inertia axes are those about which the products of inertia are zero.
- **Based on Geometric Feature** — Project the vector associated with the selected geometry feature onto the normal plane of the primary axis. Align the secondary axis with the projection. Valid geometry features include surfaces and lines. You must select a geometry feature from the visualization pane and then select the **Use Selected Feature** button.

Extended Capabilities

C/C++ Code Generation

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

See Also

Rigid Transform | Solid | Variable Brick Solid | Variable Cylindrical Solid | Variable Spherical Solid

Topics

“Modeling Bodies”
“Representing Solid Geometry”
“Specifying Custom Inertias”
“Creating Custom Solid Frames”
“Manipulate the Color of a Solid”

Introduced in R2018b

General Flexible Beam

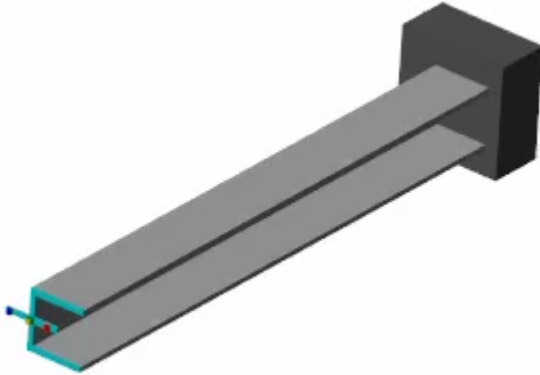
Slender extrusion with elastic properties for deformation

Library: Simscape / Multibody / Body Elements / Flexible Bodies / Beams



Description

The General Flexible Beam block models a body—a slender extrusion of constant cross section—with elastic properties and therefore the ability to deform. The beam can bend, stretch, and twist, as dictated by its boundary and loading conditions.



Cantilever Beam Subjected to Bending and Torsion

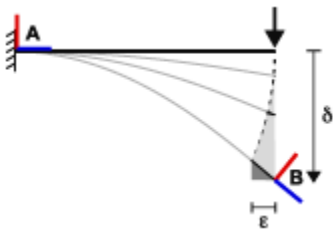
The deformation can vary over time. Variable deformation is superposed on the motion of the body as a whole. In other words, the beam can deform as it translates and rotates in the world frame. The coupling between the two motions is captured, with deformations impacting the overall motion of the body and vice versa.

For an example of a flexible beam, at the MATLAB command prompt, enter `smdoc_flexible_cantilever_channel`. The example uses an asymmetrical cross section for the beam and simulates its various deformations under a point load, an applied moment, or both. The degree to which the beam bends and twists varies with the point of application of the force.

Deformation Models

Bending and axial deformation follow from classical (Euler-Bernoulli) beam theory. The bending can be about any axis in the cross-sectional plane (xy) of the beam. Cross-sectional slices are assumed to be rigid in plane, to stay planar during deformation, and to always be perpendicular to the (generally deformed) centerline of the beam. As the theory is linear, deformation is assumed to be small (relative to the size of the beam).

The figure shows the impact of the linear assumption on the accuracy of the block. (The beam in this example is cantilevered and subjected to a transverse point load. The tip, rather than follow its true (and naturally curved) path, deflects in a straight line. At small deformations (δ), the two practically overlap, but, at large angles, the difference between them becomes evident. That error (ε) is shown in gray.)



Torsion, on the other hand, derives from classical (Saint-Venant) torsion theory. Cross-sectional slices are then rigid in plane but free to warp out of plane. As with bending, the theory is linear and deformation is assumed to be small.

The theories require the beam to be slender, which is to say that its length must far exceed the width of its cross section. The length (in the longitudinal direction) is specified as a block parameter—**Length**. The cross section (in the transverse plane) is specified in the block parameter of the same name. If the beam is stubby (short and thick), the model will simulate, but, as a key assumption has not been satisfied, the simulation results may not accurately reflect reality.

Beam Discretization

How accurate the deformation calculation is depends on how it is discretized. By default, the beam comprises a single beam element. The bending displacement distributions throughout the element are obtained by cubic Hermite interpolation between its ends; the axial displacement and torsional rotation distributions, on the other hand, are obtained by linear interpolation. The distributions become increasingly accurate as the beam is split into elements.

Use the **Number of Elements** block parameter (under the **Discretization** node) to change the discretization of the beam. Experiment with this parameter to obtain a good compromise between simulation accuracy (which may require more beam elements) and simulation speed (fewer beam elements). Use the fewest elements needed to satisfy your accuracy requirements.

Material Properties

The block is parameterized in terms of mass and stiffness parameters. These are as defined in textbooks on the mechanics of materials, with values being available from engineering databases. Here, the material is assumed to be homogeneous and isotropic (with the same properties everywhere and in every direction) as well as linearly elastic.

Damping is assumed to be linear. The damping matrix of the beam is proportional to the stiffness matrix of the same, with a constant of proportionality equal to the **Damping Constant** block parameter. Set this parameter to better capture, for example, the decay in oscillation amplitude that occurs in underdamped beams.

Note that damping can have a significant impact on simulation speed. Experiment with the value of the damping constant to balance the need for simulation accuracy with the need for simulation speed.

Beam Geometry

The beam cross section is specified as a MATLAB matrix of $[x, y]$ coordinates. Each matrix row corresponds to a point, the collection of which connects, in the order given, to form a polyline. The material region is to the left of the polyline (as observed from one point to the next), with empty space being to its right. To ensure that the cross section is closed, the end points of the polyline are automatically connected to each other.

The cross section can take many shapes, including asymmetrical ones, like those in tee, channel, and angle beams. Cross sections with holes, however, are not supported. The

techniques for modeling holes in the Solid block should not be used in this block. Such techniques require a cut through the material, leaving the beam cross section technically open. The beam will then behave in ways uncharacteristic for a closed cross-section with a true hole.

The extrusion itself is formed by sweeping the cross section along the local z -axis. The sweep is symmetric with respect to the xy -plane: one end is at half the beam length in the positive z direction, and the other at the same distance in the negative direction. The cross section is constant in both shape and size from one end to the other.


It is important that the polyline not intersect itself. Polylines with intersecting segments produce invalid cross sections and cause errors in the block. Note also that an excessive number of points can adversely affect the performance of the block, slowing down the calculation of the sectional properties and the rendering of the beam in Mechanics Explorer.

Connection Frames

At the ends of the extrusion are two frames (**A** and **B**) by which to connect the beam. The frames fall on the z -axis of the local reference frame, whose origin sets the $[0, 0]$ cross-section point. The reference frame, labeled **R** in the block visualization pane, serves merely as an internal reference for the beam, and it has no frame port by which to connect.

Beam Visualization

Issues of geometry and color—a beam too short, say, or a color specified wrong—are often easiest to catch by eye. Check the visualization pane regularly for unwanted changes, as these are easiest to troubleshoot (and the block parameters to revert) when they first

occur. Click the  button—at the far left in the visualization toolbar—to update the visualization contents when a parameter changes.

Pan, rotate, roll, and zoom (in, out, or to a region) to better see a region of interest. Select a standard view, such as *front*, *right*, or *top*, to quickly align the camera with the axes of the reference frame. Look for these tools and views in the buttons of the visualization toolbar. (Hover over a button for its name if unsure of what it does.)

Right-click the visualization pane to see more (context-sensitive) options. Use them to change background color, to set the *up* axis, or to split the visualization pane into tiles (for example, to see the beam from different views at once).

Click **Apply** or **OK** to commit any changes to the model. Note that closing the dialog box without clicking either button will cause all changes to be discarded.

Deformation under Gravity

The beam responds to gravity, but only that specified in the Mechanism Configuration block. The force due to a Gravitational Field block is ignored. If the frame network of which this block is a part contains a Gravitational Field block, the beam behaves as though in zero gravity. Using Flexible Beam and Gravitational Field blocks in the same frame network causes Diagnostic Viewer to issue a compilation warning.

Note that modeling gravity with both Mechanism Configuration and Gravitational Field blocks results in a (more severe) compilation error. The simulation can then no longer run until one source of gravity is eliminated—by setting gravity to None in the Mechanism Configuration block, for example, or by disconnecting every Gravitational Field block in the frame network.

Simulation Performance

Flexible beams can negatively impact the simulation speed of a model. The more of them there are, the slower the simulation becomes. For best performance, use the fewest flexible beams possible, avoiding them where deformation is negligible and rigid bodies suffice. Discretize each beam only as finely as you must, and experiment with the damping coefficient to better balance simulation accuracy with simulation speed.

Note that models with flexible beams tend to be numerically stiff. To avoid simulation issues, consider using a stiff solver such as `ode15s` or `ode23t`. Stiff solvers are denoted as such in the Configuration Parameters window. Using appropriate solver tolerances—both relative and absolute—can help to speed up simulation. Adjust these parameters if necessary to improve simulation performance.

Ports

Frame

A — Connection frame

frame

Frame—one of two—by which to connect the beam in a model. In the undeformed configuration, this frame is at half the beam length in the $-z$ direction relative to the origin of the local reference frame.

B — Connection frame

frame

Frame—one of two—by which to connect the beam in a model. In the undeformed configuration, this frame is at half the beam length in the $+z$ direction relative to the origin of the local reference frame.

Parameters

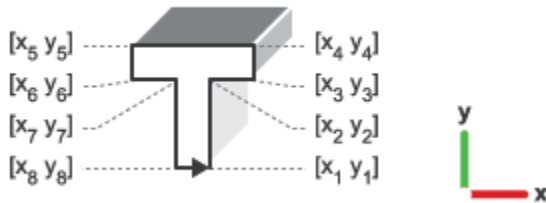
Geometry

Cross-section — Cross-section coordinates specified on the XY plane

[0.5 0.5; -0.5 0.5; -0.5 -0.5; 0.5 -0.5] m (default) | two-column matrix with units of length

Cross section of the beam, specified as an $[x,y]$ coordinate matrix. Each row gives the $[x,y]$ coordinates of a point on the cross-section outline. The points connect in the order given to form a closed polyline. To ensure that the polyline is closed, a line segment is always inserted between the last and first points specified.

The matrix rows must be arranged such that, from one point to the next, the interior of the cross section lies to the left and the exterior to the right. To ensure that the interior and exterior are always well defined, the polyline generated by the coordinate matrix must not intersect itself.



Length — Sweep length of the extrusion

10 m (default) | scalar with units of length

Total length of the beam. The beam is generated by extruding the specified cross section along the z -axis of the local reference frame. The extrusion is symmetric about the xy -plane, with half of the beam being extruded in the negative direction of the z -axis and half in the positive direction.

**Stiffness and Inertia****Density — Mass per unit volume of material**

2700 kg/m³ (default) | positive scalar with units of mass/volume

Mass per unit volume of material—assumed here to be distributed uniformly throughout the beam. The default value corresponds to aluminum.

Specify — Elastic properties in terms of which to parameterize the beam

Young's Modulus and Poisson's Ratio (default) | Young's and Shear Modulus

Elastic properties in terms of which to parameterize the beam. These properties are commonly available from materials databases.

Young's Modulus — Ratio of axial stress to axial strain

70 GPa (default) | positive scalar in units of stress (force/area)

Young's modulus of elasticity for the beam. This parameter is as defined in textbooks on the mechanics of materials (the ratio of axial stress to axial strain). The greater its value, the stronger the resistance to bending and axial deformation. The default value corresponds to aluminum.

Poisson's Ratio — Ratio of transverse to longitudinal strains

0.33 (default) | unitless scalar

Poisson's ratio for the beam. This parameter is as defined in textbooks on the mechanics of materials (the ratio of transverse to longitudinal strain). The value specified must be greater than or equal to 0 and smaller than 0.5. The default value corresponds to aluminum.

Shear Modulus — Ratio of shear stress to engineering shear strain

26 GPa (default) | positive scalar in units of stress (force/area)

Shear modulus (or modulus of rigidity) of the beam. This parameter is as defined in textbooks on the mechanics of materials (the ratio of shear stress to engineering shear strain). The greater its value, the stronger the resistance to torsional deformation. The default value corresponds to aluminum.

Damping Constant — Measure of damping for dynamic deformations

0.001 s (default) | scalar greater than or equal to zero and with units of time

Proportionality constant for damping internal to the beam. The larger this parameter, the faster the decay in beam vibrations when they occur. The product of the damping constant and the stiffness matrix of the beam gives the damping matrix used in the block calculations. This parameter can significantly impact simulation performance. Experiment with its value to balance simulation speed with simulation accuracy.

Derived Values — Calculated values of mass and stiffness sectional properties

button

Calculated values of the mass and stiffness sectional properties of the beam. Click **Update** to see those values—and again if a parameter changes, to refresh those values.

The properties given include **Centroid** and **Shear Center**. The centroid is the point at which an axial force extends (or contracts) the beam without bending. The shear center is that through which a transverse force must pass to bend the beam without twisting.

The stiffness sectional properties are computed as follows:

- **Axial Rigidity:** EA
- **Flexural Rigidity:** $[EI_x, EI_y]$
- **Cross Flexural Rigidity:** EI_{xy}
- **Torsional Rigidity:** GJ

The mass sectional properties are computed as follows:

- **Mass per Unit Length:** ρA
- **Mass Moment of Inertia Density:** $[\rho I_x, \rho I_y]$
- **Mass Product of Inertia Density:** ρI_{xy}
- **Polar Mass Moment of Inertia Density:** ρI_p

The equation parameters include:

- A — Cross-sectional area
- ρ — Density
- E — Young's modulus
- G — Shear modulus
- J — Torsional constant (obtained from the solution of Saint-Venant's warping partial differential equation)

The remaining parameters are the relevant moments of area of the beam. These are calculated about the axes of a centroidal frame—one aligned with the local reference frame but located with its origin at the centroid. The moments of area are:

- I_x, I_y — Centroidal second moments of area:

$$[I_x, I_y] = \left[\int_A y^2 dA, \int_A x^2 dA \right]$$

- I_{xy} — Centroidal product moment of area:

$$I_{xy} = \int_A xy dA$$

- I_p — Centroidal polar moment of area:

$$I_p = I_x + I_y$$

Discretization

Number of Elements — Number of sections into which to divide the beam

1 (default) | positive and unitless integer scalar

Number of elements into which to divide the beam. The more elements there are, the more accurate the computed deflections may be, albeit at the cost of simulation speed.

Graphic

Type — Graphic to use in the visualization of the beam

From Geometry (default) | None

Choice of graphic to use in the visualization of the beam. The graphic is by default the geometry specified for the beam. Change this parameter to None to eliminate this beam altogether from the model visualization.

Visual Properties — Parameterizations for color and opacity

Simple (default) | Advanced

Parameterization for specifying visual properties. Select Simple to specify color and opacity. Select Advanced to add specular highlights, ambient shadows, and self-illumination effects.

Simple: Color — True color as [R,G,B] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three-element vector with values constrained to 0-1

RGB color vector with red (R), green (G), and blue (B) color amounts specified on a 0-1 scale. A color picker provides an alternative interactive means of specifying a color. If you change the **Visual Properties** setting to Advanced, the color specified in this parameter becomes the **Diffuse Color** vector.

Simple: Opacity — Surface opacity as scalar number on 0-1 scale

1.0 (default) | scalar with value constrained to 0-1

Graphic opacity specified on a scale of 0-1. An opacity of 0 corresponds to a completely transparent graphic and an opacity of 1 to a completely opaque graphic.

Advanced: Diffuse Color — True color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three- or four-element vector with values constrained to 0-1

True color under direct white light specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. An optional fourth element specifies the color opacity also on a scale of 0-1. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Specular Color — Highlight color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of specular highlights specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Ambient Color — Shadow color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of shadow areas in diffuse ambient light, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Emissive Color — Self-illumination color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Surface color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Shininess — Highlight sharpness as scalar number on 0-128 scale

75 (default) | scalar with value constrained to 0-128

Sharpness of specular light reflections, specified as a scalar number on a 0-128 scale. Increase the shininess value for smaller but sharper highlights. Decrease the value for larger but smoother highlights.

Extended Capabilities

C/C++ Code Generation

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

See Also

Rigid Transform | Solid | Variable Brick Solid | Variable Cylindrical Solid | Variable Spherical Solid

Topics

“Modeling Bodies”

“Representing Solid Geometry”

“Specifying Custom Inertias”

“Creating Custom Solid Frames”

“Manipulate the Color of a Solid”

Introduced in R2018b

General Variable Mass

Mass element with variable inertial properties

Library: Simscape / Multibody / Body Elements / Variable Mass



Description

The General Variable Mass block adds to the attached frame a mass element with constant or time-varying inertial properties. The mass element can be a point mass without rotational inertia or a custom mass with rotational inertia. The inertial properties include mass, center of mass, moments of inertia, and products of inertia. Each inertial property can be independently configured as constant or time-varying.

The geometry of the mass element is unspecified. A marker or equivalent inertia ellipsoid identifies the mass element in the visualization pane of Mechanics Explorer. An inertia ellipsoid provides a graphical representation of the principal moments of inertia of the mass element. The block includes an option to hide the variable mass element in the Mechanics Explorer visualization window.

Ports

Input

m — Mass

physical signal specified as a scalar with units of mass

Input port for the time-varying mass.

Dependencies

This port is enabled when the **Inertia > Mass** parameter is set to Time-Varying.

com — Center of mass

physical signal specified as a 3-by-1 or 1-by-3 vector with units of length

Input port for the time-varying center-of-mass coordinates. Specify the coordinates in the order [x y z] relative to the block reference frame.

Dependencies

This port is enabled when the **Inertia > Center of Mass** parameter is set to Time-Varying.

I — Inertia

physical signal specified as a 3-by-3 matrix with units of mass \times length²

Input port for the time-varying inertia tensor. Specify the tensor elements in the order [Ixx Ixy Ixz; Iyx Iyy Iyz; Izx Izy Izz]. The elements are defined relative to a frame with origin at the center of mass and axes aligned with the reference frame. See the **Inertia tensor** parameter description for the definitions of the moments and products of inertia.

Dependencies

This port is enabled when the **Inertia > Type** parameter is set to Custom.

Frame

R — Reference frame

frame

Local reference frame of the variable mass element. Connect the port to a frame line or another frame port to define the relative position and orientation of the variable mass.

Parameters

Inertia

Type — Choice of point or distributed mass

Custom (default) | Point Mass

Choice of point or distributed mass. Select **Point Mass** to model a concentrated mass with no rotational inertia. Select **Custom** to model a distributed mass with the specified inertia tensor and center of mass.

Mass — Mass parameterization

Time-Varying (default) | Constant

Choice of fixed or variable mass. Select **Time-Varying** to specify the mass as a variable using physical signal input port **m**. Select **Constant** to specify the mass as a constant parameter.

Mass: Value — Aggregate mass of the mass element

1 kg (default) | scalar with units of mass

Aggregate mass of the mass element. The mass can be a positive or negative value. Specify a negative mass to model the aggregate effect of voids and cavities in a compound body. The mass is constant when this parameter is active.

Dependencies

This parameter is enabled when the **Mass** parameter is set to **Constant**.

Center of Mass — Center-of-mass parameterization

Time-Varying (default) | Constant

Choice of fixed or variable center of mass. Select **Time-Varying** to specify the center of mass as a variable using physical signal input port **com**. Select **Constant** to specify the center of mass as a constant parameter.

Center of Mass: Value — Center-of-mass coordinates

[0 0 0] m (default) | three-element vector with units of length

[x y z] coordinates of the center of mass relative to the origin of the reference frame. The center of mass coincides with the center of gravity in uniform gravitational fields only. The center of mass is constant when this parameter is active.

Dependencies

This parameter is enabled when the **Center of Mass** parameter is set to **Constant**.

Inertia Matrix — Inertia-matrix parameterization

Time-Varying (default) | Constant

Choice of a variable or fixed inertia matrix. Select **Time-Varying** to specify the inertia matrix as a variable using physical signal input port **I**. Select **Constant** to specify the moments and products of inertia separately as constant block parameters.

Inertia Matrix: Moments of Inertia — Diagonal elements of the inertia matrix

[1 1 1] kg * m² (default) | three-element vector with units of mass*length²

Moments of inertia of the variable mass element specified in the order $[I_{xx} \ I_{yy} \ I_{zz}]$. The moments of inertia are defined relative to a frame with origin at the center of mass and with axes parallel to the reference frame axes. The moments of inertia are the diagonal elements of the inertia tensor,

$$\begin{pmatrix} I_{xx} & & \\ & I_{yy} & \\ & & I_{zz} \end{pmatrix},$$

where:

- $I_{xx} = \int_V (y^2 + z^2) dm$
- $I_{yy} = \int_V (x^2 + z^2) dm$
- $I_{zz} = \int_V (x^2 + y^2) dm$

Dependencies

This parameter is enabled when the **Inertia Matrix** parameter is set to Constant.

Inertia Matrix: Products of Inertia — Off-diagonal elements of the inertia matrix

[0 0 0] kg * m² (default) | 3-element array with units of mass * length²

Products of inertia of the variable mass element specified in the order $[I_{yz} \ I_{zx} \ I_{xy}]$. The products of inertia are defined relative to a frame with origin at the center of mass and with axes parallel to the reference frame axes. The products of inertia are the off-diagonal elements of the inertia matrix,

$$\begin{pmatrix} & I_{xy} & I_{zx} \\ I_{xy} & & I_{yz} \\ I_{zx} & I_{yz} & \end{pmatrix},$$

where:

- $I_{yz} = - \int_V yz \, dm$

- $I_{zx} = - \int_V zx \, dm$

- $I_{xy} = - \int_V xy \, dm$

Dependencies

This parameter is enabled when the **Inertia Matrix** parameter is set to Constant.

Graphic

Type — Geometry type to use in model visualizations

Equivalent Inertia Ellipsoid (default) | Marker | None

Visualization setting for this solid. Marker dimensions are fixed and specified in pixel units. Ellipsoid dimensions are variable and depend on the specified mass and inertia tensor. The center of the ellipsoid coincides with the specified center of mass. Marker visualization is active on model update and during simulation. Ellipsoid visualization is active during simulation only.

Marker: Shape — Shape of the graphic marker

Sphere (default) | Cube | Frame

Geometrical shape of the graphic marker. Mechanics Explorer shows the marker using the selected shape.

Marker: Size — Pixel size of the graphic marker

10 (default) | scalar with units of pixels

Size of the marker in units of pixels. The size does not change with zoom level.

Visual Properties — Parameterizations for color and opacity

Simple (default) | Advanced

Parameterization for specifying visual properties. Select **Simple** to specify color and opacity. Select **Advanced** to add specular highlights, ambient shadows, and self-illumination effects.

Simple: Color — True color as [R,G,B] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three-element vector with values constrained to 0-1

RGB color vector with red (R), green (G), and blue (B) color amounts specified on a 0-1 scale. A color picker provides an alternative interactive means of specifying a color. If you change the **Visual Properties** setting to Advanced, the color specified in this parameter becomes the **Diffuse Color** vector.

Simple: Opacity — Surface opacity as scalar number on 0-1 scale

1.0 (default) | scalar with value constrained to 0-1

Graphic opacity specified on a scale of 0-1. An opacity of 0 corresponds to a completely transparent graphic and an opacity of 1 to a completely opaque graphic.

Advanced: Diffuse Color — True color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three- or four-element vector with values constrained to 0-1

True color under direct white light specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. An optional fourth element specifies the color opacity also on a scale of 0-1. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Specular Color — Highlight color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of specular highlights specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Ambient Color — Shadow color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of shadow areas in diffuse ambient light, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Emissive Color — Self-illumination color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Surface color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Shininess — Highlight sharpness as scalar number on 0-128 scale
75 (default) | scalar with value constrained to 0-128

Sharpness of specular light reflections, specified as a scalar number on a 0-128 scale. Increase the shininess value for smaller but sharper highlights. Decrease the value for larger but smoother highlights.

Extended Capabilities

C/C++ Code Generation

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

See Also

Graphic | Inertia | Solid | Spline

Topics

“Specifying Custom Inertias”

“Representing Solid Inertia”

“Manipulate the Color of a Solid”

Introduced in R2016b

Gimbal Joint

Joint with three revolute primitives

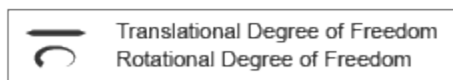
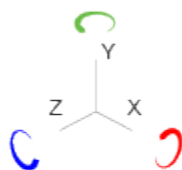


Library

Joints

Description

This block represents a joint with three rotational degrees of freedom. Three revolute primitives provide the three rotational degrees of freedom. The base and follower frame origins remain coincident during simulation.



Joint Degrees of Freedom

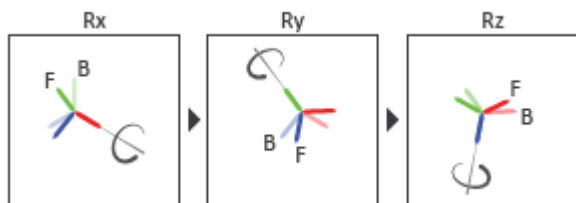
The joint block represents motion between the base and follower frames as a sequence of time-varying transformations. Each joint primitive applies one transformation in this sequence. The transformation translates or rotates the follower frame with respect to the

joint primitive base frame. For all but the first joint primitive, the base frame coincides with the follower frame of the previous joint primitive in the sequence.

At each time step during the simulation, the joint block applies the sequence of time-varying frame transformations in this order:

- 1** Rotation:
 - a** About the X axis of the X Revolute Primitive (Rx) base frame.
 - b** About the Y axis of the Y Revolute Primitive (Ry) base frame. This frame is coincident with the X Revolute Primitive (Rx) follower frame.
 - c** About the Z axis of the Z Revolute Primitive (Rz) base frame. This frame is coincident with the Y Revolute Primitive (Ry) follower frame.

The figure shows the sequence in which the joint transformations occur at a given simulation time step. The resulting frame of each transformation serves as the base frame for the following transformation. Because 3-D rotation occurs as a sequence, it is possible for two axes to align, causing to the loss of one rotational degree of freedom. This phenomenon is known as gimbal lock.



Joint Transformation Sequence

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound,

both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Revolute Primitive: State Targets

Specify the revolute primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative rotation angle, measured about the joint primitive axis, of the follower frame with respect to the base frame. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative angular velocity, measured about the joint primitive axis, of the follower frame with respect to the base frame. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is deg for position and deg/s for velocity.

Revolute Primitive: Internal Mechanics

Specify the revolute primitive internal mechanics. Internal mechanics include linear spring torques, accounting for energy storage, and linear damping torques, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the rotation angle between base and follower frames at which the spring torque is zero. The default value is 0. Select or enter a physical unit. The default is deg.

Spring Stiffness

Enter the linear spring constant. This is the torque required to rotate the joint primitive by a unit angle. The default is 0. Select or enter a physical unit. The default is N*m/deg.

Damping Coefficient

Enter the linear damping coefficient. This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N*m/(deg/s).

Revolute Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper

bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Revolute Primitive: Actuation

Specify actuation options for the revolute joint primitive. Actuation modes include **Torque** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Input signals are resolved in the base frame.

Torque

Select an actuation torque setting. The default setting is None.

Actuation Torque Setting	Description
None	No actuation torque.
Provided by Input	Actuation torque from physical signal input. The signal provides the torque acting on the follower frame with respect to the base frame about the joint primitive axis. An equal and opposite torque acts on the base frame.
Automatically computed	Actuation torque from automatic calculation. Simscape Multibody computes and applies the actuation torque based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Revolute Primitive: Sensing

Select the variables to sense in the revolute joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative rotation angle of the follower frame with respect to the base frame about the joint primitive axis.

Velocity

Select this option to sense the relative angular velocity of the follower frame with respect to the base frame about the joint primitive axis.

Acceleration

Select this option to sense the relative angular acceleration of the follower frame with respect to the base frame about the joint primitive axis.

Actuator Torque

Select this option to sense the actuation torque acting on the follower frame with respect to the base frame about the joint primitive axis.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force

or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame

- F — Follower frame

Actuation Ports

The revolute joint primitives provide the following actuation ports:

- t_x, t_y, t_z — Actuation torques acting on the X, Y, and Z revolute joint primitives
- q_x, q_y, q_z — Desired rotations of the X, Y, and Z revolute joint primitives

Sensing Ports

The revolute joint primitives provide the following sensing ports:

- q_x, q_y, q_z — Angular positions of the X, Y, and Z revolute joint primitives
- w_x, w_y, w_z — Angular velocities of the X, Y, and Z revolute joint primitives
- b_x, b_y, b_z — Angular accelerations of the X, Y, and Z revolute joint primitives
- t_x, t_y, t_z — Actuation torques acting on the X, Y, and Z revolute joint primitives

The following sensing ports provide the composite forces and torques acting on the joint:

- f_c — Constraint force
- t_c — Constraint torque
- f_t — Total force
- t_t — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Bushing Joint | Revolute Joint | Spherical Joint

Topics

“Actuating and Sensing with Physical Signals”

“Motion Sensing”

“Rotational Measurements”

Introduced in R2012a

Graphic

Marker with graphic properties

Library: Simscape / Multibody / Body Elements



Description

The Graphic block adds a simple marker to the attached frame. The marker has a simple geometry, color, and no inertia. You can use this block to highlight a frame of interest in the Mechanics Explorer visualization pane. The graphic marker has no impact on model dynamics.



Graphic Marker Geometries

Ports

Frame

R — Reference frame

frame

Local reference frame of the graphic marker. Connect to a frame line or frame port to define the relative position and orientation of the marker.

Parameters

Shape — Shape of the marker to show during visualization

Sphere (default) | Cube | Frame

Shape of the marker to show in the visualization of the model. The motion of the marker reflects the motion of the frame to which the block is connected.

Size — Width of the marker in pixels

10 (default)

Width of the marker in pixels. This width does not scale with zoom level. Note that the apparent size of the marker depends partly on screen resolution, with higher resolutions packing more pixels per unit length, and therefore producing smaller icons.

Visual Properties — Parameterizations for color and opacity

Simple (default) | Advanced

Parameterization for specifying visual properties. Select **Simple** to specify color and opacity. Select **Advanced** to add specular highlights, ambient shadows, and self-illumination effects.

Simple: Color — True color as [R,G,B] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three-element vector with values constrained to 0-1

RGB color vector with red (R), green (G), and blue (B) color amounts specified on a 0-1 scale. A color picker provides an alternative interactive means of specifying a color. If you change the **Visual Properties** setting to **Advanced**, the color specified in this parameter becomes the **Diffuse Color** vector.

Simple: Opacity — Surface opacity as scalar number on 0-1 scale

1.0 (default) | scalar with value constrained to 0-1

Graphic opacity specified on a scale of 0-1. An opacity of 0 corresponds to a completely transparent graphic and an opacity of 1 to a completely opaque graphic.

Advanced: Diffuse Color — True color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three- or four-element vector with values constrained to 0-1

True color under direct white light specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. An optional fourth element specifies the color opacity also on a scale of 0-1. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Specular Color — Highlight color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of specular highlights specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Ambient Color — Shadow color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of shadow areas in diffuse ambient light, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Emissive Color — Self-illumination color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Surface color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Shininess — Highlight sharpness as scalar number on 0-128 scale

75 (default) | scalar with value constrained to 0-128

Sharpness of specular light reflections, specified as a scalar number on a 0-128 scale. Increase the shininess value for smaller but sharper highlights. Decrease the value for larger but smoother highlights.

Extended Capabilities

C/C++ Code Generation

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

See Also

Inertia | Solid | Spline

Topics

“Visualize Simscape Multibody Frames”

“Manipulate the Color of a Solid”

Introduced in R2012a

Gravitational Field

Field of force due to point mass



Library

Forces and Torques

Description

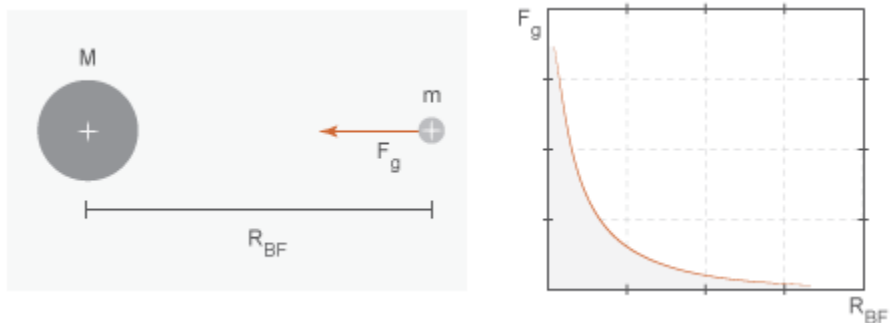
This block represents the gravitational field of a point mass. This field applies a gravitational force at the center of mass of each rigid body. The force magnitude decays with the square distance from the field origin, coincident with the base port frame origin. The force on a rigid body follows from Newton's universal gravitation law:

$$F_g = -G \frac{Mm}{R_{BF}^2},$$

where:

- F_g is the force that the gravitational field exerts on a given rigid body.
- G is the universal gravitational constant, $6.67384 \times 10^{-11} \text{ m}^3\text{kg}^{-1}\text{s}^{-2}$.
- M is the total mass generating the gravitational field.
- m is the total mass of the rigid body the gravitational force acts upon.
- R_{BF} is the distance between the source mass position and the rigid body center of mass.

The figure shows these variables. The plot shows the inverse square dependence between the gravitational force and distance.



The source mass can be positive or negative. Combine multiple instances of this block to model the gravitational effects that positive and negative mass disturbances impose on a stronger gravitational field, such as a reduction in the gravitational pull of a planet due to a concentration of low-density material along a portion of its surface.

This block excludes the gravitational forces that other rigid bodies exert on the field source mass. To include these forces, you can connect Gravitational Field blocks to other rigid bodies in the model. Alternatively, you can use the Inverse Square Law Force block to model the gravitational forces between a single pair of rigid bodies.

The gravitational field is time invariant. To specify a time-varying, spatially uniform field, use the Mechanism Configuration block.

Parameters

Mass

Total mass generating the gravitational field. The resulting gravitational forces are directly proportional to this mass. This mass adds no inertia to the model. The default value for the mass parameter is 1.0 kg.

Ports

Frame port B represents a frame with origin at the point mass responsible for the gravitational field.

Extended Capabilities

C/C++ Code Generation

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

See Also

Inverse Square Law Force | Mechanism Configuration

Topics

“Model Gravity in a Planetary System”

Introduced in R2014a

Inertia

Mass element with fixed inertial properties

Library: Simscape / Multibody / Body Elements



Description

The Inertia block adds to the attached frame a point or distributed mass with fixed inertial properties. The type of mass (point or distributed) depends on the parameterization selected for the block. A selection of **Point Mass** assumes the mass to be concentrated at a point and therefore devoid of rotational inertia. A selection of **Custom** assumes the mass to be distributed in space, allowing it to possess nonzero moments of inertia, products of inertia, and center-of-mass coordinates (the latter against the reference frame of the block).

A choice of marker provides a means to identify the inertia in the model visualization. The visualization opens (by default) in Mechanics Explorer at the start of simulation or upon diagram update. Use the marker to track the motion of the inertia against the remainder of the model. Various marker shapes are available—an inertia icon, a sphere, a cube, a frame—each with configurable dimensions. To eliminate the inertia from the model visualization (while retaining its effects on the model dynamics), the block provides an option of **None**.

Ports

Frame

R — Reference frame

frame

Local reference frame of the inertia element. Connect to a frame line or frame port to define the relative position and orientation of the inertia.

Parameters

Inertia

Type — Inertia parameterization to use

Point Mass (default) | Custom

Inertia parameterization to use. Select **Point Mass** to represent a mass with no rotational inertia. Select **Custom** to represent a distributed mass with rotational inertia.

Mass — Total mass of the solid element

1 kg (default) | scalar with units of mass

Total mass to attribute to the solid element. This parameter can be positive or negative. Use a negative value to capture the effect of a void or cavity in a compound body (one comprising multiple solids and inertias), being careful to ensure that the mass of the body is on the whole positive.

Custom: Center of Mass — Center-of-mass coordinates

[0 0 0] m (default) | three-element vector with units of length

[x y z] coordinates of the center of mass relative to the block reference frame. The center of mass coincides with the center of gravity in uniform gravitational fields only.

Custom: Moments of Inertia — Diagonal elements of inertia tensor

[1 1 1] kg*m² (default) | three-element vector with units of mass*length²

Three-element vector with the [I_{xx} I_{yy} I_{zz}] moments of inertia specified relative to a frame with origin at the center of mass and axes parallel to the block reference frame. The moments of inertia are the diagonal elements of the inertia tensor

$$\begin{pmatrix} I_{xx} \\ I_{yy} \\ I_{zz} \end{pmatrix},$$

where:

- $I_{xx} = \int_V (y^2 + z^2) dm$
- $I_{yy} = \int_V (x^2 + z^2) dm$

- $$I_{zz} = \int_V (x^2 + y^2) dm$$

Custom: Products of Inertia – Off-diagonal elements of inertia tensor

[0 0 0] kg*m² (default) | three-element vector with units of mass*length²

Three-element vector with the [I_{yz} I_{zx} I_{xy}] products of inertia specified relative to a frame with origin at the center of mass and axes parallel to the block reference frame. The products of inertia are the off-diagonal elements of the inertia tensor

$$\begin{pmatrix} I_{xy} & I_{zx} \\ I_{xy} & I_{yz} \\ I_{zx} & I_{yz} \end{pmatrix},$$

where:

- $$I_{yz} = - \int_V yz dm$$

- $$I_{zx} = - \int_V zx dm$$

- $$I_{xy} = - \int_V xy dm$$

Graphic

Type – Graphic to use in the visualization of the inertia

Marker (default) | None

Choice of graphic to use in the visualization of the inertia. The graphic is by default a simple marker, specifically an inertia icon. Other markers are available, these encompassing spheres, cubes, and frames. Change this parameter to **None** to eliminate the inertia altogether from the model visualization.

Marker: Shape – Shape of the marker to assign to the inertia

Inertia Icon (default) | Sphere | Cube | Frame

Shape of the marker by means of which to visualize the inertia. The motion of the marker reflects the motion of the inertia itself.

Marker: Size — Width of the marker in pixels

10 (default) | scalar with units of pixels

Width of the marker in pixels. This width does not scale with zoom level. Note that the apparent size of the marker depends partly on screen resolution, with higher resolutions packing more pixels per unit length, and therefore producing smaller icons.

Visual Properties — Parameterizations for color and opacity

Simple (default) | Advanced

Parameterization for specifying visual properties. Select **Simple** to specify color and opacity. Select **Advanced** to add specular highlights, ambient shadows, and self-illumination effects.

Simple: Color — True color as [R,G,B] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three-element vector with values constrained to 0-1

RGB color vector with red (R), green (G), and blue (B) color amounts specified on a 0-1 scale. A color picker provides an alternative interactive means of specifying a color. If you change the **Visual Properties** setting to **Advanced**, the color specified in this parameter becomes the **Diffuse Color** vector.

Simple: Opacity — Surface opacity as scalar number on 0-1 scale

1.0 (default) | scalar with value constrained to 0-1

Graphic opacity specified on a scale of 0-1. An opacity of 0 corresponds to a completely transparent graphic and an opacity of 1 to a completely opaque graphic.

Advanced: Diffuse Color — True color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three- or four-element vector with values constrained to 0-1

True color under direct white light specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. An optional fourth element specifies the color opacity also on a scale of 0-1. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Specular Color — Highlight color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of specular highlights specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Ambient Color — Shadow color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of shadow areas in diffuse ambient light, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Emissive Color — Self-illumination color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Surface color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Shininess — Highlight sharpness as scalar number on 0-128 scale

75 (default) | scalar with value constrained to 0-128

Sharpness of specular light reflections, specified as a scalar number on a 0-128 scale. Increase the shininess value for smaller but sharper highlights. Decrease the value for larger but smoother highlights.

Extended Capabilities

C/C++ Code Generation

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

See Also

General Variable Mass | Graphic | Solid

Topics

“Representing Solid Inertia”

“Specifying Custom Inertias”

Introduced in R2012a

Internal Force

General force acting reciprocally between two frame origins



Library

Forces and Torques

Description

This block represents a general force pair acting reciprocally between base and follower frame origins. The two forces in the pair have equal magnitude but opposite directions. One force acts on the base frame origin, along the vector connecting follower to base frame origins. The other force acts on the follower frame origin, along the vector connecting base to follower frame origins.

To specify the internal force, the block provides physical signal port **fm**. A positive input signal represents a repulsive force, which pushes base and follower frame origins apart. A negative input signal represents an attractive force, which pulls base and follower frame origins together. If at any time the two frame origins are coincident, the internal force direction becomes undefined and Simscape Multibody might stop with an error.

Ports

This block contains frame ports **B** and **F**, representing base and follower port frames, respectively. A physical signal port, **fm**, provides the means to specify the internal force acting between the two port frames.

Extended Capabilities

C/C++ Code Generation

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

See Also

External Force and Torque | Inverse Square Law Force | Spring and Damper Force

Topics

“Actuating and Sensing with Physical Signals”

Introduced in R2013a

Inverse Square Law Force

Force proportional to the inverse square distance between two frame origins



Library

Forces and Torques

Description

This block represents a force pair that is inversely proportional to the square distance between the base and follower frame origins. The two forces in the pair have equal magnitude but opposite directions. One force acts on the base frame origin, along the vector connecting the follower to base frame origins. The other force acts on the follower frame origin, along the vector connecting base to follower frame origins.

The value of the force depends on a force constant that you specify. A positive force constant represents a repulsive force that pushes the two frame origins apart. A negative force constant represents an attractive force that pulls the two frame origins together.

Parameters

Force Constant

Specify the proportionality constant of the inverse square law force. This constant is a lumped parameter that encodes the dependence of the force magnitude on the inverse square distance between the two frame origins. The default value is 1. Select or specify a physical unit.

Sense Force

Select the check box to sense the signed magnitude of the inverse square law force acting between the two frame origins. The block exposes an additional physical signal

port to output the force signal. The output signal is a scalar value. This value is positive if the force is repulsive; it is negative if the force is attractive.

Ports

The block contains frame ports B and F, representing base and follower frames, respectively.

Selecting **Sense Force** in the block dialog box exposes an additional physical signal port, **fm**.

Extended Capabilities

C/C++ Code Generation

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

See Also

External Force and Torque | Internal Force | Spring and Damper Force

Topics

“Actuating and Sensing with Physical Signals”

Introduced in R2012a

Lead Screw Joint

Joint with coupled rotational and translational degrees of freedom



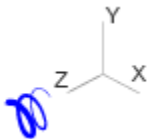
Library

Joints

Description

This block represents a joint with one rotational degree of freedom coupled with one translational degree of freedom. The coupling between the two degrees of freedom ensures that whenever the joint frames rotate relative to each other, they also translate by a commensurate amount and vice-versa. The joint lead determines the translation distance associated with a unit rotation angle while the joint direction determines whether a positive angle results in a positive or negative translation.

During assembly and simulation, the joint aligns the Z axes of its port frames. The common Z axis functions as the rotation and translation axis. Whenever the joint frames rotate, they do so about the common Z axis, and whenever the joint frames translate, they do so along the common Z axis. You can orient the motion axis in a different direction by applying rotation transforms to the joint frames through Rigid Transform blocks.



Joint Degrees of Freedom

A set of optional state targets guide assembly for the joint primitive. Targets include position and velocity. You can specify these based on the relative rotation or translation

between the joint frames. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Each joint primitive has a set of optional sensing ports. These ports provide physical signal outputs that measure joint primitive motion. Variables that you can sense include those describing translational motion, rotational motion, and constraint forces and torques.

Parameters

Lead Screw Primitive

Direction

Handedness of motion between the joint frames. Motion is right-handed if a positive rotation leads to a positive translation and left-handed if a positive rotation leads to a negative translation. The default setting is **Right - Hand**.

Lead

Translation distance between the joint frames due to a unit rotation angle. The larger the lead, the longer the frames must translate before completing a full revolution. The default value is **1.0 mm/rev**.

Lead Screw Primitive: State Targets

Specify the lead screw primitive state targets and their priority levels. A state target is the desired value for one of the joint state variables—position or velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be satisfied.

Specify Position Target

Desired joint primitive position at the start of simulation. This is the relative position, rotational or translational, of the follower frame relative to the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Desired joint velocity at the start of simulation. This is the relative velocity, rotational or translational, of the follower frame relative to the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Based On

Motion type that the state target is based on. Options include **Rotation** and **Translation**. The default setting is **Translation**.

Value

Desired value of the position or velocity state target. The default value is 0.

Lead Screw Primitive: Sensing

Select the variables to sense in the lead screw primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame.

Variable	Description
Rotation: Position	Rotation angle of the follower frame relative to the base frame about the common Z axis. Selecting the check box exposes a physical signal port labeled q.
Rotation: Velocity	Rotational velocity of the follower frame relative to the base frame about the common Z axis. Selecting the check box exposes a physical signal port labeled w.

Variable	Description
Rotation: Acceleration	Rotational acceleration of the follower frame relative to the base frame about the common Z axis. Selecting the check box exposes a physical signal port labeled b.
Translation: Position	Offset distance of the follower frame relative to the base frame along the common Z axis. Selecting the check box exposes a physical signal port labeled p.
Translation: Velocity	Translational velocity of the follower frame relative to the base frame along the common Z axis. Selecting the check box exposes a physical signal port labeled v.
Translation: Acceleration	Translational acceleration of the follower frame relative to the base frame along the common Z axis. Selecting the check box exposes a physical signal port labeled a.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Sensing Ports

The lead screw joint primitive provides the following sensing ports:

- q — Angular position
- w — Angular velocity
- b — Angular acceleration
- p — Linear position
- v — Linear velocity
- a — Linear acceleration

The following sensing ports provide the composite forces and torques acting on the joint:

- f_c — Constraint force
- t_c — Constraint torque
- f_t — Total force
- t_t — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Prismatic Joint | Revolute Joint

Introduced in R2015a

Mechanism Configuration

Mechanism-wide simulation and mechanical parameters



Library

Utilities

Description

This block provides mechanical and simulation parameters to a mechanism, i.e., a self-contained group of interconnected Simscape Multibody blocks. Parameters include gravity and a linearization delta for computing numerical partial derivatives during linearization. These parameters apply only to the target mechanism, i.e., the mechanism that the block connects to.

The Mechanism Configuration block is optional. If you omit it, the gravitational acceleration vector is set to zero. Use only one instance of this block per mechanism, setting uniform gravity to **None** if that mechanism contains one or more Gravitational Field blocks.

Parameters

Uniform Gravity

Type of gravitational acceleration vector in effect at the target mechanism. Options include:

- **None** — Specify zero gravity. The block automatically applies the gravitational acceleration vector $[0\ 0\ 0]$ to the target mechanism. If the mechanism contains one or more Gravitational Field blocks, you must select this option.

- **Constant** — Specify a gravitational acceleration vector that remains constant in space and in time. Selecting this option exposes an additional parameter, **Gravity**. If the target mechanism contains one or more Gravitational Field blocks, you must select **None** instead.
- **Time-Varying** — Specify a gravitational acceleration vector that remains constant in space but varies in time. Selecting this option exposes a physical signal port. Use that port to specify the time-varying gravitational acceleration vector. If the target mechanism contains one or more Gravitational Field blocks, you must select **None** instead.

Gravity

Nominal acceleration vector due to gravity. The block resolves this vector in the mechanism World frame. The default vector is $[0 \ 0 \ -9.80665]$ m/s².

Linearization Delta

Perturbation value for computing numerical partial derivatives during linearization. The default value is 0.001.

Ports

Port	Description
C	Frame port that identifies the target mechanism to which the block parameters apply.
g	Physical signal port through which you specify a time-varying gravity vector.

Extended Capabilities

C/C++ Code Generation

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

See Also

Gravitational Field

Introduced in R2012a

Pin Slot Joint

Joint with one prismatic and one revolute primitives possessing mutually orthogonal motion axes

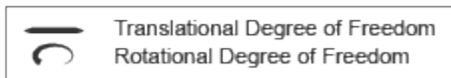
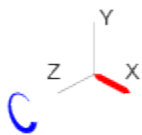


Library

Joints

Description

This block represents a joint with one translational and one rotational degrees of freedom. One prismatic primitive provides the translational degree of freedom. One revolute primitive provides the rotational degree of freedom. Prismatic and revolute axes are mutually orthogonal.



Joint Degrees of Freedom

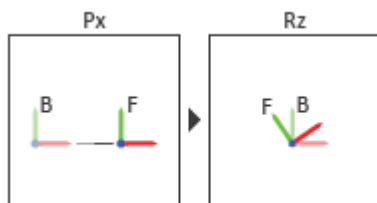
The joint block represents motion between the base and follower frames as a sequence of time-varying transformations. Each joint primitive applies one transformation in this

sequence. The transformation translates or rotates the follower frame with respect to the joint primitive base frame. For all but the first joint primitive, the base frame coincides with the follower frame of the previous joint primitive in the sequence.

At each time step during the simulation, the joint block applies the sequence of time-varying frame transformations in this order:

- 1 Translation:
 - Along the X axis of the X Prismatic Primitive (Px) base frame.
- 2 Rotation:
 - About the Z axis of the Z Revolute Primitive (Rz) base frame. This frame is coincident with the X Prismatic Primitive (Px) follower frame.

The figure shows the sequence in which the joint transformations occur at a given simulation time step. The resulting frame of each transformation serves as the base frame for the following transformation.



Joint Transformation Sequence

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. To enforce the bounds, the joint adds to each a

spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Prismatic Primitive: State Targets

Specify the prismatic primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative position, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative velocity, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is m for position and m/s for velocity.

Prismatic Primitive: Internal Mechanics

Specify the prismatic primitive internal mechanics. Internal mechanics include linear spring forces, accounting for energy storage, and damping forces, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the distance between base and follower frame origins at which the spring force is zero. The default value is 0. Select or enter a physical unit. The default is m.

Spring Stiffness

Enter the linear spring constant. This is the force required to displace the joint primitive by a unit distance. The default is 0. Select or enter a physical unit. The default is N/m.

Damping Coefficient

Enter the linear damping coefficient. This is the force required to maintain a constant joint primitive velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N/(m/s).

Prismatic Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper

bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Prismatic Primitive: Actuation

Specify actuation options for the prismatic joint primitive. Actuation modes include **Force** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Actuation signals are resolved in the base frame.

Force

Select an actuation force setting. The default setting is **None**.

Actuation Force Setting	Description
None	No actuation force.
Provided by Input	Actuation force from physical signal input. The signal provides the force acting on the follower frame with respect to the base frame along the joint primitive axis. An equal and opposite force acts on the base frame.
Automatically computed	Actuation force from automatic calculation. Simscape Multibody computes and applies the actuation force based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Prismatic Primitive: Sensing

Select the variables to sense in the prismatic joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative position of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Velocity

Select this option to sense the relative velocity of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Acceleration

Select this option to sense the relative acceleration of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Actuator Force

Select this option to sense the actuation force acting on the follower frame with respect to the base frame along the joint primitive axis.

Revolute Primitive: State Targets

Specify the revolute primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative rotation angle, measured about the joint primitive axis, of the follower frame with respect to the base frame. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative angular velocity, measured about the joint primitive axis, of the follower

frame with respect to the base frame. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is deg for position and deg/s for velocity.

Revolute Primitive: Internal Mechanics

Specify the revolute primitive internal mechanics. Internal mechanics include linear spring torques, accounting for energy storage, and linear damping torques, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the rotation angle between base and follower frames at which the spring torque is zero. The default value is 0. Select or enter a physical unit. The default is deg.

Spring Stiffness

Enter the linear spring constant. This is the torque required to rotate the joint primitive by a unit angle. The default is 0. Select or enter a physical unit. The default is N*m/deg.

Damping Coefficient

Enter the linear damping coefficient. This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is $\text{N}\cdot\text{m}/(\text{deg}/\text{s})$.

Revolute Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Revolute Primitive: Actuation

Specify actuation options for the revolute joint primitive. Actuation modes include **Torque** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Input signals are resolved in the base frame.

Torque

Select an actuation torque setting. The default setting is **None**.

Actuation Torque Setting	Description
None	No actuation torque.
Provided by Input	Actuation torque from physical signal input. The signal provides the torque acting on the follower frame with respect to the base frame about the joint primitive axis. An equal and opposite torque acts on the base frame.
Automatically computed	Actuation torque from automatic calculation. Simscape Multibody computes and applies the actuation torque based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Revolute Primitive: Sensing

Select the variables to sense in the revolute joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative rotation angle of the follower frame with respect to the base frame about the joint primitive axis.

Velocity

Select this option to sense the relative angular velocity of the follower frame with respect to the base frame about the joint primitive axis.

Acceleration

Select this option to sense the relative angular acceleration of the follower frame with respect to the base frame about the joint primitive axis.

Actuator Torque

Select this option to sense the actuation torque acting on the follower frame with respect to the base frame about the joint primitive axis.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Actuation Ports

The prismatic joint primitive provides the following actuation ports:

- f_x — Actuation force acting on the X prismatic joint primitive
- p_x — Desired trajectory of the X prismatic joint primitive

The revolute joint primitive provides the following actuation ports:

- t_z — Actuation torque acting on the Z revolute joint primitive
- q_z — Desired rotation of the Z revolute joint primitive

Sensing Ports

The prismatic joint primitive provides the following sensing ports:

- p_x — Position of the X prismatic joint primitive
- v_x — Velocity of the X prismatic joint primitive
- a_x — Acceleration of the X prismatic joint primitive
- f_x — Actuation force acting on the X prismatic joint primitive

The revolute joint primitive provides the following sensing ports:

- q_z — Angular position of the Z revolute joint primitive
- w_z — Angular velocity of the Z revolute joint primitive
- b_z — Angular acceleration of the Z revolute joint primitive

- t_z — Actuation torque acting on the Z revolute joint primitive

The following sensing ports provide the composite forces and torques acting on the joint:

- f_c — Constraint force
- t_c — Constraint torque
- f_t — Total force
- t_t — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Cylindrical Joint | Prismatic Joint | Revolute joint

Topics

“Actuating and Sensing with Physical Signals”

“Motion Sensing”

“Rotational Measurements”

“Translational Measurements”

Introduced in R2013a

Planar Joint

Joint with one revolute and two prismatic primitives

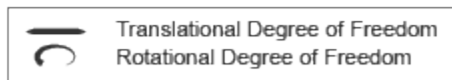
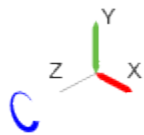


Library

Joints

Description

This block represents a joint with one rotational and two translational degrees of freedom. Two prismatic primitives provide the two translational degrees of freedom. One revolute primitive provides the rotational degree of freedom.



Joint Degrees of Freedom

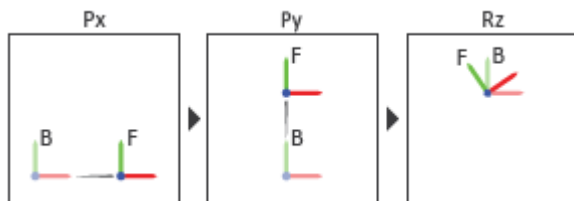
The joint block represents motion between the base and follower frames as a sequence of time-varying transformations. Each joint primitive applies one transformation in this sequence. The transformation translates or rotates the follower frame with respect to the

joint primitive base frame. For all but the first joint primitive, the base frame coincides with the follower frame of the previous joint primitive in the sequence.

At each time step during the simulation, the joint block applies the sequence of time-varying frame transformations in this order:

- 1** Translation:
 - a** Along the X axis of the X Prismatic Primitive (Px) base frame.
 - b** Along the Y axis of the Y Prismatic Primitive (Py) base frame. This frame is coincident with the X Prismatic Primitive (Px) follower frame.
- 2** Rotation:
 - About the Z axis of the Z Revolute Primitive (Rz) base frame. This frame is coincident with the Y Prismatic Primitive (Py) follower frame.

The figure shows the sequence in which the joint transformations occur at a given simulation time step. The resulting frame of each transformation serves as the base frame for the following transformation.



Joint Transformation Sequence

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound,

both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Prismatic Primitive: State Targets

Specify the prismatic primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative position, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative velocity, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is m for position and m/s for velocity.

Prismatic Primitive: Internal Mechanics

Specify the prismatic primitive internal mechanics. Internal mechanics include linear spring forces, accounting for energy storage, and damping forces, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the distance between base and follower frame origins at which the spring force is zero. The default value is 0. Select or enter a physical unit. The default is m.

Spring Stiffness

Enter the linear spring constant. This is the force required to displace the joint primitive by a unit distance. The default is 0. Select or enter a physical unit. The default is N/m.

Damping Coefficient

Enter the linear damping coefficient. This is the force required to maintain a constant joint primitive velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N/(m/s).

Prismatic Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper

bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Prismatic Primitive: Actuation

Specify actuation options for the prismatic joint primitive. Actuation modes include **Force** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Actuation signals are resolved in the base frame.

Force

Select an actuation force setting. The default setting is **None**.

Actuation Force Setting	Description
None	No actuation force.
Provided by Input	Actuation force from physical signal input. The signal provides the force acting on the follower frame with respect to the base frame along the joint primitive axis. An equal and opposite force acts on the base frame.
Automatically computed	Actuation force from automatic calculation. Simscape Multibody computes and applies the actuation force based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Prismatic Primitive: Sensing

Select the variables to sense in the prismatic joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative position of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Velocity

Select this option to sense the relative velocity of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Acceleration

Select this option to sense the relative acceleration of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Actuator Force

Select this option to sense the actuation force acting on the follower frame with respect to the base frame along the joint primitive axis.

Revolute Primitive: State Targets

Specify the revolute primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative rotation angle, measured about the joint primitive axis, of the follower frame with respect to the base frame. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative angular velocity, measured about the joint primitive axis, of the follower

frame with respect to the base frame. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is deg for position and deg/s for velocity.

Revolute Primitive: Internal Mechanics

Specify the revolute primitive internal mechanics. Internal mechanics include linear spring torques, accounting for energy storage, and linear damping torques, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the rotation angle between base and follower frames at which the spring torque is zero. The default value is 0. Select or enter a physical unit. The default is deg.

Spring Stiffness

Enter the linear spring constant. This is the torque required to rotate the joint primitive by a unit angle. The default is 0. Select or enter a physical unit. The default is N*m/deg.

Damping Coefficient

Enter the linear damping coefficient. This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N*m/(deg/s).

Revolute Primitive: Actuation

Specify actuation options for the revolute joint primitive. Actuation modes include **Torque** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Input signals are resolved in the base frame.

Torque

Select an actuation torque setting. The default setting is None.

Actuation Torque Setting	Description
None	No actuation torque.
Provided by Input	Actuation torque from physical signal input. The signal provides the torque acting on the follower frame with respect to the base frame about the joint primitive axis. An equal and opposite torque acts on the base frame.
Automatically computed	Actuation torque from automatic calculation. Simscape Multibody computes and applies the actuation torque based on model dynamics.

Motion

Select an actuation motion setting. The default setting is Automatically Computed.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Revolute Primitive: Sensing

Select the variables to sense in the revolute joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative rotation angle of the follower frame with respect to the base frame about the joint primitive axis.

Velocity

Select this option to sense the relative angular velocity of the follower frame with respect to the base frame about the joint primitive axis.

Acceleration

Select this option to sense the relative angular acceleration of the follower frame with respect to the base frame about the joint primitive axis.

Actuator Torque

Select this option to sense the actuation torque acting on the follower frame with respect to the base frame about the joint primitive axis.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Actuation Ports

The prismatic joint primitives provide the following actuation ports:

- f_x, f_y — Actuation forces acting on the X and Y prismatic joint primitives
- p_x, p_y — Desired trajectories of the X and Y prismatic joint primitives

The revolute joint primitive provides the following actuation ports:

- t_z — Actuation torque acting on the Z revolute joint primitive
- q_z — Desired rotation of the Z revolute joint primitive

Sensing Ports

The prismatic joint primitives provide the following sensing ports:

- p_x, p_y — Positions of the X and Y prismatic joint primitives
- v_x, v_y — Velocities of the X and Y prismatic joint primitives
- a_x, a_y — Accelerations of the X and Y prismatic joint primitives
- f_x, f_y — Actuator forces acting on the X and Y prismatic joint primitives

The revolute joint primitive provides the following sensing ports:

- q_z — Angular position of the Z revolute joint primitive
- w_z — Angular velocity of the Z revolute joint primitive

- bz — Angular acceleration of the Z revolute joint primitive
- tz — Actuation torque acting on the Z revolute joint primitive

The following sensing ports provide the composite forces and torques acting on the joint:

- fc — Constraint force
- tc — Constraint torque
- ft — Total force
- tt — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

[Prismatic Joint](#) | [Rectangular Joint](#) | [Revolute Joint](#)

Topics

[“Actuating and Sensing with Physical Signals”](#)

[“Motion Sensing”](#)

[“Rotational Measurements”](#)

[“Translational Measurements”](#)

Introduced in R2012a

Point On Curve Constraint

Kinematic constraint between a frame origin and a curved path

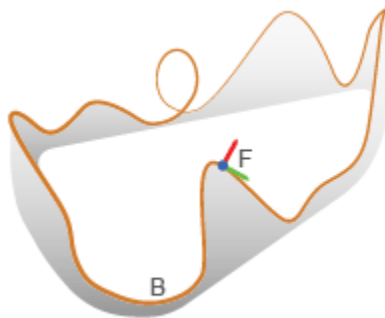


Library

Constraints

Description

This block represents a kinematic constraint between a point and a curve. The constraint allows the follower frame origin to translate only along the curve connected to the base geometry port. The follower frame is free to rotate depending on other constraints in the model. Use this block to model point-on-curve constraints, such as that between a roller coaster and a track or a cam follower and a cam.



Roller Coaster



Cam

Examples of Point-on-Curve Constraints

Specify the constraint curve by connecting a curve block to the base geometry port. As a best practice, always use the curve block as part of a rigid body, for example, by keeping it inside a rigid body subsystem. This enables you to quickly switch, for example, between different cams or roller coaster tracks. Avoid curves with sharp changes in slope, as these can cause simulation issues.

Parameters

Constraint Force Sensing

Direction

Select the force in the constraint action-reaction force pair to sense. You can sense the force that the follower frame exerts on the base curve or vice-versa. The default setting is **Follower on Base**.

Resolution Frame

Select the frame to resolve the constraint force measurement in. You can select the base or follower frame. The default setting is **Base**.

Force Vector

Select the check box to sense the constraint force. The block exposes physical signal output port **f**, which outputs the force measurement as a three-dimensional vector, $[F_x, F_y, F_z]$.

Ports

The block contains two ports:

- **B** — Geometry port associated with the constraint curve (set by connecting the port to a Spline block).
- **F** — Frame port associated with the constraint point (defined as the origin of the frame).

An optional port appears when you select constraint force sensing:

- **f** — Physical signal with the constraint force components $[F_x, F_y, F_z]$

Extended Capabilities

C/C++ Code Generation

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

See Also

[Angle Constraint](#) | [Bevel Gear Constraint](#) | [Common Gear Constraint](#) | [Distance Constraint](#)
| [Rack and Pinion Constraint](#) | [Spline](#)

Introduced in R2015b

Prismatic Joint

Joint with one prismatic primitive

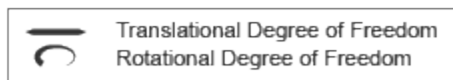
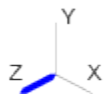


Library

Joints

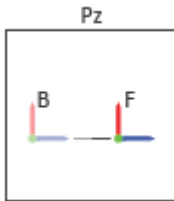
Description

This block represents a joint with one translational degree of freedom. One prismatic primitive provides the translational degree of freedom. The base and follower frames remain parallel during simulation.



Joint Degrees of Freedom

The joint block represents motion between the base and follower frames as a single time-varying transformation. The Z prismatic primitive (Pz) applies this transformation, which causes the follower frame to translate with respect to the base frame along the common Z axis.



Joint Transformation

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Prismatic Primitive: State Targets

Specify the prismatic primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target

must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative position, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative velocity, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is m for position and m/s for velocity.

Prismatic Primitive: Internal Mechanics

Specify the prismatic primitive internal mechanics. Internal mechanics include linear spring forces, accounting for energy storage, and damping forces, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the distance between base and follower frame origins at which the spring force is zero. The default value is 0. Select or enter a physical unit. The default is m.

Spring Stiffness

Enter the linear spring constant. This is the force required to displace the joint primitive by a unit distance. The default is 0. Select or enter a physical unit. The default is N/m.

Damping Coefficient

Enter the linear damping coefficient. This is the force required to maintain a constant joint primitive velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N/(m/s).

Prismatic Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Prismatic Primitive: Actuation

Specify actuation options for the prismatic joint primitive. Actuation modes include **Force** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Actuation signals are resolved in the base frame.

Force

Select an actuation force setting. The default setting is **None**.

Actuation Force Setting	Description
None	No actuation force.
Provided by Input	Actuation force from physical signal input. The signal provides the force acting on the follower frame with respect to the base frame along the joint primitive axis. An equal and opposite force acts on the base frame.

Actuation Force Setting	Description
Automatically computed	Actuation force from automatic calculation. Simscape Multibody computes and applies the actuation force based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Prismatic Primitive: Sensing

Select the variables to sense in the prismatic joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative position of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Velocity

Select this option to sense the relative velocity of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Acceleration

Select this option to sense the relative acceleration of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Actuator Force

Select this option to sense the actuation force acting on the follower frame with respect to the base frame along the joint primitive axis.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Actuation Ports

The prismatic joint primitive provides the following actuation ports:

- f — Actuation force acting on the Z prismatic joint primitive
- p — Desired trajectory of the Z prismatic joint primitive

Sensing Ports

The prismatic joint primitive provides the following sensing ports:

- p — Position of the Z prismatic joint primitive
- v — Velocity of the Z prismatic joint primitive
- a — Acceleration of the Z prismatic joint primitive
- f — Actuation force acting on the Z prismatic joint primitive

The following sensing ports provide the composite forces and torques acting on the joint:

- f_c — Constraint force
- t_c — Constraint torque
- f_t — Total force
- t_t — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Revolute Joint | Spherical Joint

Topics

“Actuating and Sensing with Physical Signals”

“Motion Sensing”

“Translational Measurements”

Introduced in R2012a

Pulley

Wheel wrapped in a cord for the transmission of torque and motion

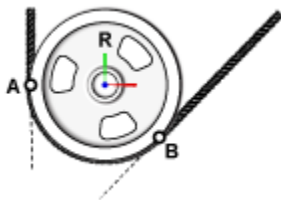
Library: Simscape / Multibody / Belts and Cables



Description

The Pulley block represents a grooved or toothed wheel wrapped in a cord, an arrangement used frequently in the transmission of torque and motion in part for the mechanical advantage that it can provide. The pulley (or sprocket if toothed) is *ideal*: massless and frictionless, with zero slip permitted between its surface and the surrounding cord, itself idealized as taut and inextensible. Use the pulley singly or as part of a compound pulley system such as the block and tackle of a hoist or the timing belt of a car engine.

The pulley has one local reference frame (frame port **R**) and two cord tangency points (belt-cable ports **A** and **B**). The reference frame is placed with its origin at the center of the pulley and its z -axis along the rotation axis of the same. The cord tangency points coincide with the locations at which the cord meets or separates from the pulley. These locations can change during simulation. The belt or cable wraps around the pulley from port **A** to port **B** so as to trace a counterclockwise arc about the z -axis.



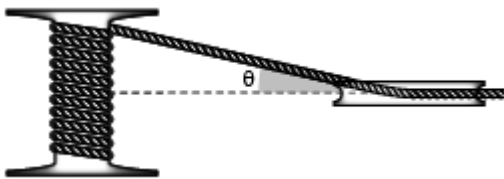
In a closed-loop system of two pulleys—such as a belt drive—the connections of the belt-cable ports determine whether the cord geometry is *crossed* or *open*. As shown in the following schematic, in a system of pulleys in which the z -axes are aligned in parallel, if port **A** of one connects to port **A** of another, then the cord is crossed; if port **A** of one

connects to port **B** of another, then the cord is open. The effect is the same if instead of switching the port connections, one of the frames is flipped so that the z-axes of the pulleys are anti-parallel.



The degrees of freedom of the pulley depend entirely on the joints and constraints (if any) to which it connects. Attaching a pulley to a case by means of a revolute joint imparts to the pulley one rotational degree of freedom relative to the case; one is then free to rotate relative to the other. Fixing the pulley to another pulley, by means of a direct connection, a rigid transform, or a weld joint, constrains the two so that if one rotates, then so must the other.

By default, the cord can enter and exit a pulley at an angle to its center plane (θ in the figure). This angle can vary during simulation—for example, due to translation of the pulley on a prismatic joint. While the contact point is always in the center plane of the pulley, the pulley can move when mounted on a joint. The cord can also be constrained to enter and exit the pulley in its center plane. Whether this constraint is enforced depends on the settings of the Belt-Cable Properties block.



The pulleys must remain at distances that preserve the natural length of the cord. This length is computed from the initial placements of the pulleys and it is fixed: the cord can neither stretch nor slacken during simulation. The length calculations include the arc lengths of the cord about the pulleys. The contact between them is idealized as slipless, with a contact point on the cord always moving at the same instantaneous velocity as its counterpart on the pulley.

Note that the frame and belt-cable ports belong to different multibody domains. As a rule, ports connect only to like ports—frame ports to other frame ports, belt-cable ports to other belt-cable ports. The belt-cable domain has the special requirement that each network or belt-cable connection lines connect to one (and no more than one) Belt-Cable Properties block. It is through that block that the visualization of the cord is configured and the length of the same is (on diagram update) computed.

The visualization of the cord is in the form of a pitch line. The cord meets and separates from the pulley tangent to its circumference. The arc of contact between the cord and the pulley is called the pitch arc. The pitch line of the cord is the sum of the line segments between different pulleys and of their respective pitch arcs. The line segments between the pulleys are shown as rectilinear, consistent with the constraint that no slackening is allowed to take place.

Combine the Pulley block with the Belt-Cable Spool block to retrieve from a winch, and to return to it, additional lengths of cord. An example application is the lowering and raising of the hook block of a tower crane. Use the Belt-Cable End block to define an end point to the cord. The end point contains a frame for connection to a load, fixture, or other part of a multibody model.

Ports

Frame

R — Local reference frame

frame

Reference frame by which to connect the pulley to the frame network of a multibody model.

Belt-Cable

A — Pulley cord tangency point

belt-cable

Point of tangency between the center line of the cord and the pitch circle of the pulley. The cord wraps around the pulley counterclockwise from port **A** to port **B**.

B — Pulley cord tangency point

belt-cable

Point of tangency between the center line of the cord and the pitch circle of the pulley. The cord wraps around the pulley counterclockwise from port **A** to port **B**.

Parameters**Pitch Radius — Distance from the center of the pulley to the center line of the cord**

10 cm (default) | positive scalar in units of length

Distance from the center of the pulley to the centerline of the cord at any point of contact. In compound pulley systems, the differences in pitch radii often determine the ratio at which speed is reduced or torque is augmented.

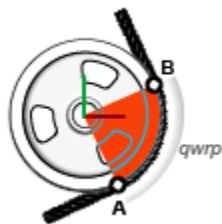
Sensing — Selection of kinematic variables to sense

Unchecked (default) | Checked

Selection of kinematic variables to sense. Select a check box to expose a physical signal port for the corresponding variable. The variables available for sensing are:

- **Wrap Angle** — Angle from the point of contact associated with port **A** to that associated with port **B**. This angle is measured in the center (xy) plane of the pulley. It is always equal to or greater than zero, with its value increasing by 2π for each revolution made counterclockwise about the local z -axis. Use port **qwrp** for this measurement.

The figure shows the wrap angle between the contact points (**A** and **B** of a pulley). The local reference frame indicates the x -axis (horizontal) and the y -axis (vertical) of the pulley.

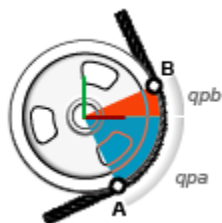


- **Pulley Angle A** — Angle, measured in the xy plane of the reference frame, from the local x -axis to the line between the frame origin and point of contact **A**.

If the point of contact is above the xz -plane (in the $+y$ -region of the reference frame), the angle is positive. If the point of contact is below the xz -plane, the angle is negative. The angle is zero when the point of contact happens to be exactly in the xz -plane.

The angle is not modular. Rather than be constrained to a 360-degree range—snapping back to the beginning of the range after completing a turn—the measured value changes continuously with repeated turns. Every turn that the drum makes adds (or subtracts) 2π to the measurement.

Use port **qpa** for this measurement.



Pulley Angles A and B

- **Pulley Angle B** — Angle, measured in the xy plane of the reference frame, from the local x -axis to the line between the frame origin and point of contact **B**.

If the point of contact is above the xz -plane (in the $+y$ -region of the reference frame), the angle is positive. If the point of contact is below the xz -plane, the angle is negative. The angle is zero when the point of contact happens to be exactly in the xz -plane.

The angle is not modular. Rather than be constrained to a 360-degree range—snapping back to the beginning of the range after completing a turn—the measured value changes continuously with repeated turns. Every turn that the drum makes adds (or subtracts) 2π to the measurement.

Use port **qpb** for this measurement.

- **Fleet Angle A** — Angle from the xy -plane of the reference frame to the cord at point of contact **A**. The xy -plane is the same as the center plane of the drum.

If the cord approaches the point of contact from above the xy -plane (in the $+z$ region of the reference frame), the angle is positive. If the cord approaches from below, the

angle is negative. The angle is zero when the cord approaches the point of contact in the center plane of the drum.

The angle is modular, which is to say that its value is bound—here, between $-\pi/2$ to $+\pi/2$. This range is open. The measured value can vary between $-\pi/2$ and $+\pi/2$, but it cannot hit either limit.

Note that if the **Drum Belt-Cable Alignment** parameter of the Belt-Cable Properties block is set to **Monitored Planar**, the pulley assembly is required to be planar, and the fleet angle is therefore always zero. To model a nonplanar assembly, use the default setting for that parameter: **Unrestricted**.

Use port **qfa** for this measurement.



Fleet Angle A

- **Fleet Angle B** — Angle from the xy -plane of the reference frame to the cord at point of contact **B**. The xy -plane is the same as the center plane of the drum.

If the cord approaches the point of contact from above the xy -plane (in the $+z$ region of the reference frame), the angle is positive. If the cord approaches from below, the angle is negative. The angle is zero when the cord approaches the point of contact in the center plane of the drum.

The angle is modular, which is to say that its value is bound—here, between $-\pi/2$ to $+\pi/2$. This range is open. The measured value can vary between $-\pi/2$ and $+\pi/2$, but it cannot hit either limit.

Note that if the **Drum Belt-Cable Alignment** parameter of the Belt-Cable Properties block is set to **Monitored Planar**, the pulley assembly is required to be planar, and the fleet angle is therefore always zero. To model a nonplanar assembly, use the default setting for that parameter: **Unrestricted**.

Use port **qfb** for this measurement.

Extended Capabilities

C/C++ Code Generation

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

See Also

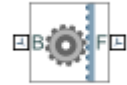
[Belt-Cable End](#) | [Belt-Cable Properties](#) | [Belt-Cable Spool](#)

Introduced in R2018a

Rack and Pinion Constraint

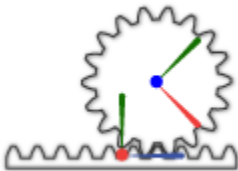
Kinematic constraint between a translating rack body and a rotating pinion body

Library: Simscape / Multibody / Gears and Couplings / Gears



Description

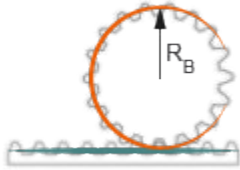
The Rack and Pinion Constraint block represents a kinematic constraint between a translating rack body and a rotating pinion body. The base frame port identifies the connection frame on the pinion body and the follower frame port identifies the connection frame on the rack body. The pinion rotation axis and the rack translation axis coincide with the frame z -axes.



The block represents only the kinematic constraint characteristic to a rack-and-pinion system. Gear inertia and geometry are solid properties that you must specify using Solid blocks. The gear constraint model is ideal. Backlash and gear losses due to Coulomb and viscous friction between teeth are ignored. You can, however, model viscous friction at joints by specifying damping coefficients in the joint blocks.

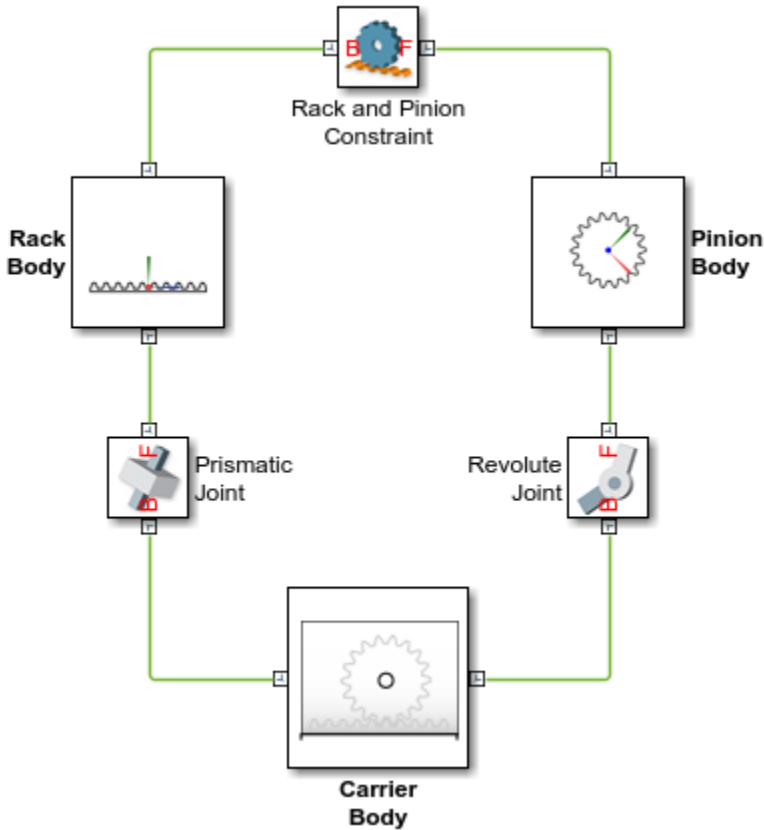
Gear Geometry

The rack-and-pinion constraint is parameterized in terms of the dimensions of the pinion pitch circle. The pitch circle is an imaginary circle concentric with the pinion body and tangent to the tooth contact point. The pitch radius, labeled R_B in the figure, is the radius that the pinion would have if it was reduced to a friction cylinder in contact with a brick approximation of the rack.



Gear Assembly

Gear constraints occur in closed kinematic loops. The figure shows the closed-loop topology of a simple rack-and-pinion model. Joint blocks connect the rack and pinion bodies to a common fixture or carrier, defining the maximum degrees of freedom between them. A Rack and Pinion Constraint block connects the rack and pinion bodies, eliminating one degree of freedom and effectively coupling the rack and pinion motions.



Assembly Requirements

The block imposes special restrictions on the relative positions and orientations of the gear connection frames. The restrictions ensure that the gears assemble only at distances and angles suitable for meshing. The block enforces the restrictions during model assembly, when it first attempts to place the gears in mesh, but relies on the remainder of the model to keep the gears in mesh during simulation.

Position Restrictions

- The distance between the base and follower frame origins along the follower frame y -axis must equal the pinion radius. This constraint ensures that the pitch point of the rack is at the proper distance from the rotation axis of the pinion.
- The follower frame origin must lie on the xy plane of the base frame. This constraint ensures that the pitch point of the rack is coplanar with the pitch circle of the pinion.

Orientation Restrictions

- The x -axis of the follower frame must be perpendicular to the xy plane of the base frame. This constraint ensures that the rack and pinion are coplanar, and therefore that their motion axes are perpendicular to each other.

Ports

Frame

B — Base frame

frame

Connection frame on the pinion body.

F — Follower frame

frame

Connection frame on the rack body.

Parameters

Pinion Radius — Radius of the pitch circle of the pinion body

10 cm (default) | positive scalar in units of length

Radius of the pitch circle of the pinion body. The pitch circle is an imaginary circle concentric with the pinion body and tangent to the tooth contact point.

Extended Capabilities

C/C++ Code Generation

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

See Also

[Bevel Gear Constraint](#) | [Common Gear Constraint](#) | [Worm and Gear Constraint](#)

Topics

“Rack and Pinion”

Introduced in R2013a

Rectangular Joint

Joint with two prismatic primitives

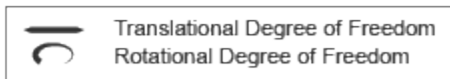
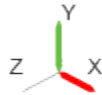


Library

Joints

Description

This block represents a joint with two translational degrees of freedom. Two prismatic primitives provide the two translational degrees of freedom. The base and follower frames remain parallel during simulation.



Joint Degrees of Freedom

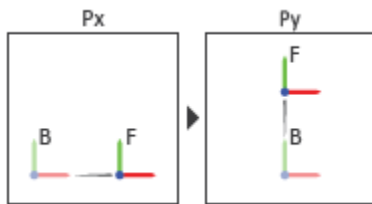
The joint block represents motion between the base and follower frames as a sequence of time-varying transformations. Each joint primitive applies one transformation in this sequence. The transformation translates or rotates the follower frame with respect to the

joint primitive base frame. For all but the first joint primitive, the base frame coincides with the follower frame of the previous joint primitive in the sequence.

At each time step during the simulation, the joint block applies the sequence of time-varying frame transformations in this order:

- 1 Translation:
 - a Along the X axis of the X Prismatic Primitive (Px) base frame.
 - b Along the Y axis of the Y Prismatic Primitive (Py) base frame. This frame is coincident with the X Prismatic Primitive (Px) follower frame.

The figure shows the sequence in which the joint transformations occur at a given simulation time step. The resulting frame of each transformation serves as the base frame for the following transformation.



Joint Transformation Sequence

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Prismatic Primitive: State Targets

Specify the prismatic primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative position, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative velocity, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is m for position and m/s for velocity.

Prismatic Primitive: Internal Mechanics

Specify the prismatic primitive internal mechanics. Internal mechanics include linear spring forces, accounting for energy storage, and damping forces, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the distance between base and follower frame origins at which the spring force is zero. The default value is 0. Select or enter a physical unit. The default is m.

Spring Stiffness

Enter the linear spring constant. This is the force required to displace the joint primitive by a unit distance. The default is 0. Select or enter a physical unit. The default is N/m.

Damping Coefficient

Enter the linear damping coefficient. This is the force required to maintain a constant joint primitive velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N/(m/s).

Prismatic Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Prismatic Primitive: Actuation

Specify actuation options for the prismatic joint primitive. Actuation modes include **Force** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Actuation signals are resolved in the base frame.

Force

Select an actuation force setting. The default setting is None.

Actuation Force Setting	Description
None	No actuation force.
Provided by Input	Actuation force from physical signal input. The signal provides the force acting on the follower frame with respect to the base frame along the joint primitive axis. An equal and opposite force acts on the base frame.
Automatically computed	Actuation force from automatic calculation. Simscape Multibody computes and applies the actuation force based on model dynamics.

Motion

Select an actuation motion setting. The default setting is Automatically Computed.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Prismatic Primitive: Sensing

Select the variables to sense in the prismatic joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative position of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Velocity

Select this option to sense the relative velocity of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Acceleration

Select this option to sense the relative acceleration of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Actuator Force

Select this option to sense the actuation force acting on the follower frame with respect to the base frame along the joint primitive axis.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Actuation Ports

The prismatic joint primitives provide the following actuation ports:

- f_x, f_y — Actuation forces acting on the X and Y prismatic joint primitives
- p_x, p_y — Desired trajectories of the X and Y prismatic joint primitives

Sensing Ports

The prismatic joint primitives provide the following sensing ports:

- px, py — Positions of the X and Y prismatic joint primitives
- vx, vy — Velocities of the X and Y prismatic joint primitives
- ax, ay — Accelerations of the X and Y prismatic joint primitives
- fx, fy — Actuation forces acting on the X and Y prismatic joint primitives

The following sensing ports provide the composite forces and torques acting on the joint:

- fc — Constraint force
- tc — Constraint torque
- ft — Total force
- tt — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Planar Joint | Prismatic Joint

Topics

“Actuating and Sensing with Physical Signals”

“Motion Sensing”

“Translational Measurements”

Introduced in R2012a

Reference Frame

Non-inertial reference frame

Library: Simscape / Multibody / Frames and Transforms



Description

This block represents a reference frame with respect to which you can define other frames. The reference frame is generally non-inertial. It can accelerate with respect to the World frame. This block is optional in a model.

Ports

Frame

R — Reference frame

frame

Local reference frame represented by the block. Connect to a frame line or frame port to define the relative position and orientation of the reference frame.

Extended Capabilities

C/C++ Code Generation

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

See Also

Rigid Transform | World Frame

Topics

“Working with Frames”

“Creating Connection Frames”

“Visualize Simscape Multibody Frames”

Introduced in R2012a

Revolute Joint

Joint with one revolute primitive

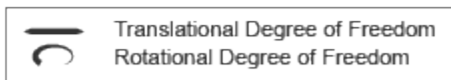
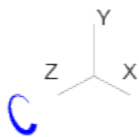


Library

Joints

Description

This block represents a joint with one rotational degree of freedom. One revolute primitive provides the rotational degree of freedom. The base and follower frame origins remain coincident during simulation.



Joint Degrees of Freedom

The joint block represents motion between the base and follower frames as a single time-varying transformation. The Z revolute primitive (Pz) applies this transformation, which causes the follower frame to rotate with respect to the base frame about the common Z axis.



Joint Transformation

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Revolute Primitive: State Targets

Specify the revolute primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target

must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative rotation angle, measured about the joint primitive axis, of the follower frame with respect to the base frame. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative angular velocity, measured about the joint primitive axis, of the follower frame with respect to the base frame. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is deg for position and deg/s for velocity.

Revolute Primitive: Internal Mechanics

Specify the revolute primitive internal mechanics. Internal mechanics include linear spring torques, accounting for energy storage, and linear damping torques, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the rotation angle between base and follower frames at which the spring torque is zero. The default value is 0. Select or enter a physical unit. The default is deg.

Spring Stiffness

Enter the linear spring constant. This is the torque required to rotate the joint primitive by a unit angle. The default is 0. Select or enter a physical unit. The default is N*m/deg.

Damping Coefficient

Enter the linear damping coefficient. This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N*m/(deg/s).

Revolute Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Revolute Primitive: Actuation

Specify actuation options for the revolute joint primitive. Actuation modes include **Torque** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Input signals are resolved in the base frame.

Torque

Select an actuation torque setting. The default setting is **None**.

Actuation Torque Setting	Description
None	No actuation torque.
Provided by Input	Actuation torque from physical signal input. The signal provides the torque acting on the follower frame with respect to the base frame about the joint primitive axis. An equal and opposite torque acts on the base frame.

Actuation Torque Setting	Description
Automatically computed	Actuation torque from automatic calculation. Simscape Multibody computes and applies the actuation torque based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Revolute Primitive: Sensing

Select the variables to sense in the revolute joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative rotation angle of the follower frame with respect to the base frame about the joint primitive axis.

Velocity

Select this option to sense the relative angular velocity of the follower frame with respect to the base frame about the joint primitive axis.

Acceleration

Select this option to sense the relative angular acceleration of the follower frame with respect to the base frame about the joint primitive axis.

Actuator Torque

Select this option to sense the actuation torque acting on the follower frame with respect to the base frame about the joint primitive axis.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Actuation Ports

The revolute joint primitive provides the following actuation ports:

- **t** — Actuation torque acting on the Z revolute joint primitive
- **q** — Desired rotation of the Z revolute joint primitive

Sensing Ports

The revolute joint primitive provides the following sensing ports:

- q — Angular position of the Z revolute joint primitive
- w — Angular velocity of the Z revolute joint primitive
- b — Angular acceleration of the Z revolute joint primitive
- t — Actuation torque acting on the Z revolute joint primitive

The following sensing ports provide the composite forces and torques acting on the joint:

- f_c — Constraint force
- t_c — Constraint torque
- f_t — Total force
- t_t — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Prismatic Joint | Spherical Joint

Topics

“Actuating and Sensing with Physical Signals”

“Motion Sensing”

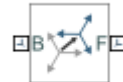
“Rotational Measurements”

Introduced in R2012a

Rigid Transform

Fixed spatial relationship between frames

Library: Simscape / Multibody / Frames and Transforms



Description

This block applies a time-invariant transformation between two frames. The transformation rotates and translates the follower port frame (F) with respect to the base port frame (B). Connecting the frame ports in reverse causes the transformation itself to reverse. The frames remain fixed with respect to each other during simulation, moving only as a single unit. Combine Rigid Transform and Solid blocks to model compound rigid bodies.

Ports

Frame

B — Base frame

frame

Frame with respect to which you specify the transforms.

F — Follower frame

frame

Frame to which you apply the transforms.

Parameters

Rotation: Method – Method for specifying the rotation transform

None (default) | Aligned Axes | Standard Axis | Arbitrary Axis | Rotation Sequence | Rotation Matrix

Method to use to specify the rotation transform between the base and follower frames. The table summarizes the available options.

Method	Description
None	Constrain the base and follower frames to share the same orientation.
Aligned Axes	Set frame rotation by aligning two follower frame axes with two base frame axes.
Standard Axis	Specify frame rotation as an angle about a standard axis (x, y, or z).
Arbitrary Axis	Specify frame rotation as an angle about a general [x, y, z] axis.
Rotation Sequence	Specify frame rotation as a sequence of three elementary rotations.
Rotation Matrix	Specify frame rotation as a right-handed orthogonal rotation matrix.

Aligned Axes

Select two pairs of base-follower frame axes.

Parameter	Description
Pair 1	First pair of base-follower frame axes to align.
Pair 2	Second pair of base-follower frame axes to align. Axis choices depend on Pair 1 axis selections.

Standard Axis

Select a standard rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	Standard rotation axis (X, Y, or Z) resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Arbitrary Axis

Select a general 3-D rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	General rotation axis [X Y Z] resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Rotation Sequence

Specify a sequence of three elementary rotations about the selected permutation of x, y, and z axes. These rotation sequences are also known as Euler and Tait-Bryan sequences. The rotations are those of the follower frame relative to the frame selected in the **Rotate About** parameter.

If you set the **Rotate About** parameter to **Follower Frame**, the follower frame rotates about its own axes. These axes change orientation with each successive rotation. If you set the **Rotate About** parameter to **Base Frame**, the follower frame rotates about the fixed base frame axes.

Parameter	Description
Rotation About	Frame whose axes to rotate the follower frame about.
Sequence	Sequence of axes about which to apply the elementary rotations.

Parameter	Description
Angles	Three-element vector with elementary rotation angles about the axes specified in the Sequence parameter.

Rotation Matrix

Specify the 3×3 transformation matrix of a proper rotation between the base and follower frames. The matrix must be orthogonal and have determinant +1. The default matrix is $[1 \ 0 \ 0; \ 0 \ 1 \ 0; \ 0 \ 0 \ 1]$.

Translation: Method – Method for specifying the translation transform

None (default) | Cartesian | Standard Axis | Cylindrical

Method to use to specify the translation transform between the base and follower frames. The table summarizes the available options.

Method	Description
None	Make base and follower frames coincident. This method requires no parameters.
Cartesian	Specify a 3-D translation in terms of Cartesian coordinates
Standard Axis	Specify a 1-D translation along the X, Y, or Z axis
Cylindrical	Specify a 3-D translation in terms of cylindrical coordinates

Cartesian Axis

Specify the **Offset** of the follower frame with respect to the base frame. This is the 3-D translation vector that brings the base frame into coincidence with the follower frame. Select or enter a physical unit.

Standard Axis

Specify the offset of the follower frame with respect to the base frame along the base frame X, Y, or Z axis. Select or enter a physical unit.

Parameter	Description
Axis	Axis the follower frame translates along

Parameter	Description
Offset	Translation of the follower frame with respect to the base frame along the specified axis

Cylindrical

Specify in cylindrical coordinates the translation that brings the base frame into coincidence with the follower frame. Select or enter a physical unit.

Parameter	Description
Radius	Distance between the origin of the follower frame and the Z axis of the base frame. This is the cylindrical radius coordinate.
Theta	Rotation angle of the line connecting base and follower frame origins with respect to the base frame X axis. This is the cylindrical azimuth coordinate.
Z Offset	Distance between base and follower frame origins along the base frame Z axis. This is the cylindrical length coordinate.

Extended Capabilities

C/C++ Code Generation

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

See Also

Reference Frame | Transform Sensor | Variable Brick Solid | World Frame

Topics

“Working with Frames”

“Creating Connection Frames”
“Visualize Simscape Multibody Frames”

Introduced in R2012a

Solid

Solid element with geometry, inertia, and color


Library: Simscape / Multibody / Body Elements



Description

The Solid block adds to the attached frame a solid element with geometry, inertia, and color. The solid element can be a simple rigid body or part of a compound rigid body—a group of rigidly connected solids, often separated in space through rigid transformations. Combine Solid and Rigid Transform blocks to model a compound rigid body.

Geometry parameters include shape and size. You can choose from a list of preset shapes or import a custom shape from an external file in STL or STEP format. By default, for all but STL-derived shapes, the block automatically computes the mass properties of the solid from the specified geometry and either mass or mass density. You can change this setting in the **Inertia > Type** block parameter.

A reference frame encodes the position and orientation of the solid. In the default configuration, the block provides only the reference frame. A frame-creation interface provides the means to define additional frames based on solid geometry features. You access this interface by selecting the Create button  in the **Frames** expandable area.

Derived Properties


You can view the calculated values of the solid mass properties directly in the block dialog box. Setting the **Inertia > Type** parameter to **Calculate** from **Geometry** causes the block to expose a new node, **Derived Values**. Click the **Update** button provided under this node to calculate the mass properties and display their values in the fields below the button.

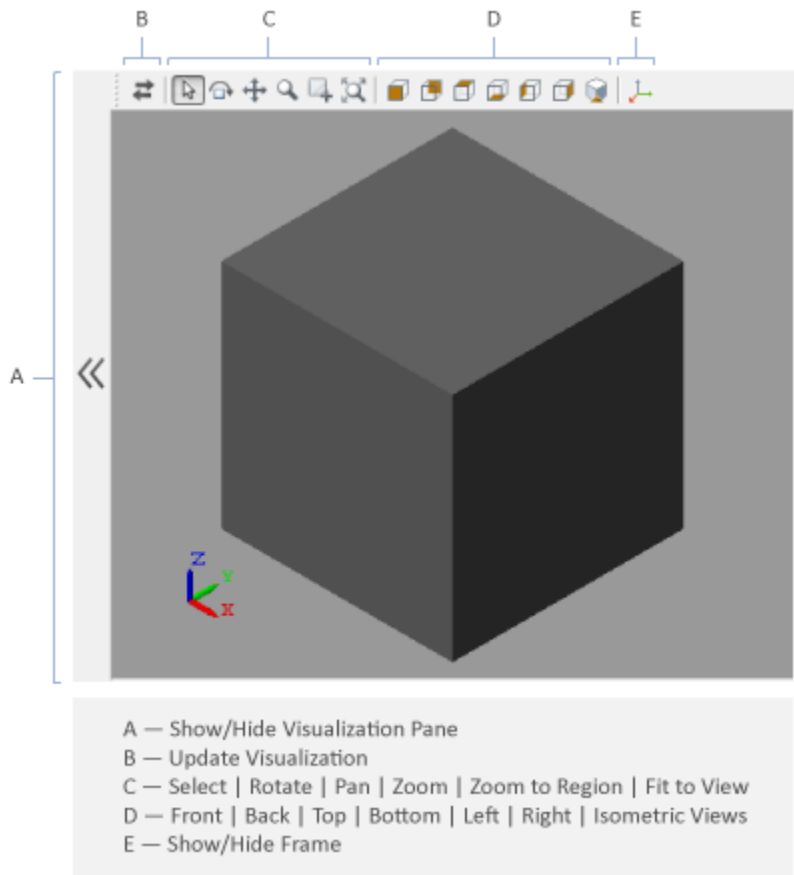
Inertia			
Type	Calculate from Geometry ▾		
Based on	Density ▾		
Density	1000	kg/m ³ ▾	Compile-time ▾
Derived Values			Update
Mass	1000		kg
Center of Mass	[0, 0, 0]		m
Moments of Inertia	[166.667, 166.667, 166.667]		kg*m ²
Products of Inertia	[-0, -0, -0]		kg*m ²

Derived Values Display

Visualization Pane

The block dialog box contains a collapsible visualization pane. This pane provides instant visual feedback on the solid you are modeling. Use it to find and fix any issues with the shape and color of the solid. You can examine the solid from different perspectives by selecting a standard view or by rotating, panning, and zooming the solid.

Select the Update Visualization button  to view the latest changes to the solid geometry in the visualization pane. Select **Apply** or **OK** to commit your changes to the solid. Closing the block dialog box without first selecting **Apply** or **OK** causes the block to discard those changes.



Solid Visualization Pane

Right-click the visualization pane to access the visualization context-sensitive menu. This menu provides additional options so that you can change the background color, split the visualization pane into multiple tiles, and modify the view convention from the default **+Z up (XY Top)** setting.

Ports

Frame

R — Reference frame

frame

Local reference frame of the solid. This frame is fixed with respect to the solid geometry. For the frame placement relative to a specific geometry see “Shape” on page 1-0 . Connect this port to a frame entity—port, line, or junction—to resolve the placement of the reference frame in a model. For more information, see “Working with Frames”.

Parameters

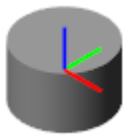
Geometry

Shape — Shape parameterization to use

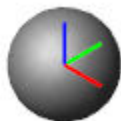
Brick (default) | Cylinder | Sphere | Ellipsoid | Regular Extrusion | General Extrusion | Revolution | From File

Shape parameterization to use. Select a preset shape such as Sphere or use the From File option to import an external STEP or STL geometry.

- **Cylinder** — Cylindrical shape with geometry center coincident with the reference frame origin and symmetry axis coincident with the reference frame z axis.



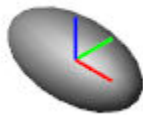
- **Sphere** — Spherical shape with geometry center coincident with the reference frame origin.



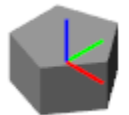
- **Brick** — Prismatic shape with geometry center coincident with the reference frame origin and prismatic surfaces normal to the reference frame x, y, and z axes.



- **Ellipsoid** — Three-dimensional extension of the ellipse with geometry center coincident with the reference frame origin and semi-principal axes coincident with the reference frame x, y, and z axes.



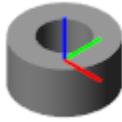
- **Regular Extrusion** — Translational sweep of a regular polygon cross section with geometry center coincident with the reference frame origin and extrusion axis coincident with the reference frame z axis.



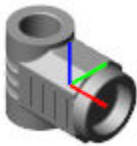
- **General Extrusion** — Translational sweep of a general cross section with geometry center coincident with the [0 0] coordinate on the cross-sectional XY plane and extrusion axis coincident with the reference frame z axis.



- **Revolution** — Rotational sweep of a general cross section with geometry center coincident with the [0 0] coordinate on the cross-sectional XZ plane and revolution axis coincident with the reference frame z axis.



- From File — Imported custom shape with geometry center and orientation as defined in STL or STEP geometry file.

**Cylinder: Radius — Radius of the cylinder**

1 m (default) | scalar with units of length

Distance between the axis of the cylinder and its surface.

**Cylinder: Length — Length of the cylinder**

1 m (default) | scalar with units of length

Distance between the opposing ends of the cylinder.

**Sphere: Radius — Radius of the sphere**

1 m (default) | scalar with units of length

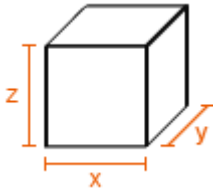
Distance between the center of the sphere and its surface.



Brick: Dimensions — Width, thickness, and height of the brick

[1 1 1] m (default) | scalar with units of length

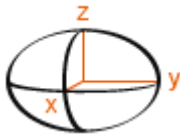
Lengths of the brick sides along the x -, y -, and z -axes of the solid reference frame. These lengths give, in no specific order, the width, thickness, and height of the brick.



Ellipsoid: Radii — Ellipsoid radii along the x , y , and z semiprincipal axes

[1, 1, 1] m (default) | scalar with units of length

Ellipsoid radii along the x , y , and z axes of the solid reference frame. The ellipsoid becomes a sphere if all radii are equal.



Regular Extrusion: Number of Sides — Number of sides of the extrusion cross-section

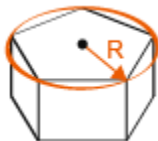
3 (default) | scalar with units of length

Number of sides of the extrusion cross-section. The cross-section is by definition a regular polygon—one whose sides are of equal length. The number specified must be greater than two.

Regular Extrusion: Outer Radius — Radius of the inscribed circle of the extrusion cross-section

1 m (default) | scalar with units of length

Radius of the circle that fully inscribes the extrusion cross-section. The cross-section is by definition a regular polygon—one whose sides are of equal length.



Regular Extrusion: Length — Sweep length of the extrusion

1 m (default) | scalar with units of length

Length by which to sweep the specified extrusion cross-section. The extrusion axis is the z-axis of the solid reference frame. The cross-section is swept by equal amounts in the positive and negative directions.

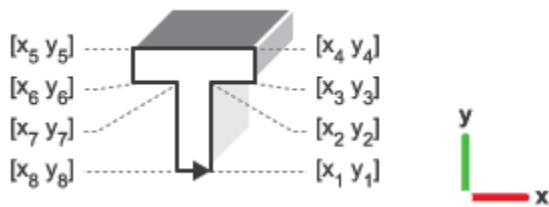


General Extrusion: Cross-section — Cross-section coordinates specified on the XY plane

[1 1; -1 1; -1 -1; 1 -1] (default) | two-column matrix with units of length

Cross-sectional shape specified as an [x,y] coordinate matrix, with each row corresponding to a point on the cross-sectional profile. The coordinates specified must define a closed loop with no self-intersecting segments.

The coordinates must be arranged such that from one point to the next the solid region always lies to the left. The block extrudes the cross-sectional shape specified along the z axis to obtain the extruded solid.



General Extrusion: Length — Sweep length of the extrusion

1 m (default) | scalar with units of length

Length by which to sweep the specified extrusion cross-section. The extrusion axis is the z-axis of the solid reference frame. The cross-section is swept by equal amounts in the positive and negative directions.

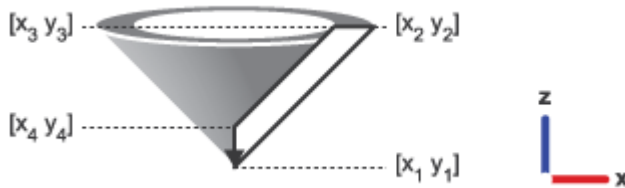


Revolution: Cross-section — Cross-section coordinates specified on the XZ plane

[1 1; 1 -1; 2 -1; 2 1] m (default) | two-column matrix with units of length

Cross-sectional shape specified as an [x,z] coordinate matrix, with each row corresponding to a point on the cross-sectional profile. The coordinates specified must define a closed loop with no self-intersecting segments.

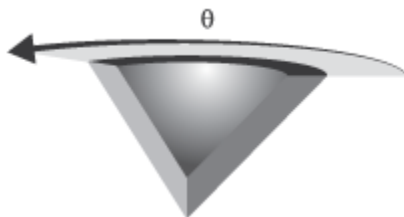
The coordinates must be arranged such that from one point to the next the solid region always lies to the left. The block revolves the cross-sectional shape specified about the reference frame z axis to obtain the revolved solid.



Revolution: Extent of Revolution — Selection of a full or partial revolution

Full (default) | Custom

Type of revolution sweep to use. Use the default setting of Full to revolve the cross-sectional shape by the maximum 360 degrees. Select Custom to revolve the cross-sectional shape by a lesser angle.



Revolution: Revolution Angle — Sweep angle of a partial revolution

180 (default) | positive scalar

Angle of the rotational sweep associated with the revolution.

From File: File Type — Type of geometry file to import

STEP (default) | STL

Type of geometry file to import. Automatic inertia calculation is available for STEP files only. You must specify all inertial properties explicitly for STL files.

From File: File Name — Name of the geometry file to import

custom character vector

Name and extension of the geometry file to import. If the file is not on the MATLAB path, the file location must be specified. The file location can be specified as an absolute path, starting from the root directory of the file system—e.g., 'C:/Users/JDoe/Documents/myShape.STEP'. It can also be specified as a relative path, starting from a folder on the MATLAB path—e.g., 'Documents/myShape.STEP'.

From File: Units — Length units of the numerical data provided in the STL file

m (default) | cm | mm | km | in | ft | yd | mi

Length units in which to interpret the vertex coordinates provided in the STL geometry file. Changing the units changes the scale of the geometry itself.

Inertia

Type — Inertia parameterization to use

Calculate from Geometry (default) | Point Mass | Custom

Inertia parameterization to use. Select **Point Mass** to model a concentrated mass with negligible rotational inertia. Select **Custom** to model a distributed mass with the specified

moments and products of inertia. The default setting, **Calculate from Geometry**, enables the block to automatically calculate the rotational inertia properties from the solid geometry and specified mass or mass density.

Based on — Parameter to base inertia calculation on

Density (default) | Mass

Parameter to use in inertia calculation. The block obtains the inertia tensor from the solid geometry and the parameter selected. Use **Density** if the material properties are known. Use **Mass** if the total solid mass is known.

Density — Mass per unit volume of material

1000 kg/m³ (default)

Mass per unit volume of material. The mass density can take on a positive or negative value. Specify a negative mass density to model the effects of a void or cavity in a solid body.

Mass — Total mass of the solid element

1 kg (default) | scalar with units of mass

Total mass to attribute to the solid element. This parameter can be positive or negative. Use a negative value to capture the effect of a void or cavity in a compound body (one comprising multiple solids and inertias), being careful to ensure that the mass of the body is on the whole positive.

Custom: Center of Mass — Center-of-mass coordinates

[0 0 0] m (default) | three-element vector with units of length

[x y z] coordinates of the center of mass relative to the block reference frame. The center of mass coincides with the center of gravity in uniform gravitational fields only.

Custom: Moments of Inertia — Diagonal elements of inertia tensor

[1 1 1] kg*m² (default) | three-element vector with units of mass*length²

Three-element vector with the [I_{xx} I_{yy} I_{zz}] moments of inertia specified relative to a frame with origin at the center of mass and axes parallel to the block reference frame. The moments of inertia are the diagonal elements of the inertia tensor

$$\begin{pmatrix} I_{xx} \\ I_{yy} \\ I_{zz} \end{pmatrix},$$

where:

- $I_{xx} = \int_V (y^2 + z^2) dm$
- $I_{yy} = \int_V (x^2 + z^2) dm$
- $I_{zz} = \int_V (x^2 + y^2) dm$

Custom: Products of Inertia – Off-diagonal elements of inertia tensor

[0 0 0] kg*m² (default) | three-element vector with units of mass*length²

Three-element vector with the [I_{yz} I_{zx} I_{xy}] products of inertia specified relative to a frame with origin at the center of mass and axes parallel to the block reference frame. The products of inertia are the off-diagonal elements of the inertia tensor

$$\begin{pmatrix} I_{xy} & I_{zx} \\ I_{xy} & I_{yz} \\ I_{zx} & I_{yz} \end{pmatrix},$$

where:

- $I_{yz} = - \int_V yz dm$
- $I_{zx} = - \int_V zx dm$
- $I_{xy} = - \int_V xy dm$

Calculate from Geometry: Derived Values – Display of calculated values of mass properties

button

Display of the calculated values of the solid mass properties—mass, center of mass, moments of inertia, and products of inertia. Click the **Update** button to calculate and display the mass properties of the solid. Click this button following any changes to the block parameters to ensure that the displayed values are still current.

The center of mass is resolved in the local reference frame of the solid. The moments and products of inertia are each resolved in the inertia frame of resolution—a frame whose axes are parallel to those of the reference frame but whose origin coincides with the solid center of mass.

Dependencies

The option to calculate and display the mass properties is active when the **Inertia > Type** block parameter is set to `Calculate` from `Geometry`.

Graphic**Type — Graphic to use in the visualization of the solid**

`From Geometry` (default) | `Marker` | `None`

Choice of graphic to use in the visualization of the solid. The graphic is by default the geometry specified for the solid. Select `Marker` to show instead a simple graphic marker, such as a sphere or cube. Change this parameter to `None` to eliminate this solid altogether from the model visualization.

Marker: Shape — Shape of the marker to assign to the solid

`Sphere` (default) | `Cube` | `Frame`

Shape of the marker by means of which to visualize the solid. The motion of the marker reflects the motion of the solid itself.

Marker: Size — Width of the marker in pixels

`10` (default) | scalar with units of pixels

Width of the marker in pixels. This width does not scale with zoom level. Note that the apparent size of the marker depends partly on screen resolution, with higher resolutions packing more pixels per unit length, and therefore producing smaller icons.

Visual Properties — Parameterizations for color and opacity

`Simple` (default) | `Advanced`

Parameterization for specifying visual properties. Select `Simple` to specify color and opacity. Select `Advanced` to add specular highlights, ambient shadows, and self-illumination effects.

Simple: Color — True color as [R,G,B] vector on 0-1 scale

`[0.5 0.5 0.5]` (default) | three-element vector with values constrained to 0-1

RGB color vector with red (R), green (G), and blue (B) color amounts specified on a 0-1 scale. A color picker provides an alternative interactive means of specifying a color. If you change the **Visual Properties** setting to Advanced, the color specified in this parameter becomes the **Diffuse Color** vector.

Simple: Opacity — Surface opacity as scalar number on 0-1 scale

1.0 (default) | scalar with value constrained to 0-1

Graphic opacity specified on a scale of 0-1. An opacity of 0 corresponds to a completely transparent graphic and an opacity of 1 to a completely opaque graphic.

Advanced: Diffuse Color — True color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three- or four-element vector with values constrained to 0-1

True color under direct white light specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. An optional fourth element specifies the color opacity also on a scale of 0-1. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Specular Color — Highlight color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of specular highlights specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Ambient Color — Shadow color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of shadow areas in diffuse ambient light, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Emissive Color — Self-illumination color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Surface color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Shininess — Highlight sharpness as scalar number on 0-128 scale
75 (default) | scalar with value constrained to 0-128




Sharpness of specular light reflections, specified as a scalar number on a 0-128 scale. Increase the shininess value for smaller but sharper highlights. Decrease the value for larger but smoother highlights.

Frames

Show Port R — Show the reference frame port for connection to other blocks
checked (default) | cleared

Clear the check box to hide the reference frame port in the Solid block. Hiding the reference frame port suppresses the frame visualization in Mechanics Explorer. You must expose the reference frame port if the block has no custom frames.

New Frame — Create a custom frame for connection to other blocks
empty (default)

Select the Create button  to define a new frame using the frame-creation interface. Each new frame appears on a row above the **New Frame** parameter. To edit an existing frame, select the Edit button . To delete an existing frame, select the Delete button .

Frame Creation Interface

Frame Name — MATLAB string used to identify the custom frame
custom character vector

Frame identifier specified as a MATLAB string. This string identifies the frame port in the block diagram and in the tree view pane of Mechanics Explorer. Keep the frame name short to ensure it fits in the block icon width.

Frame Origin — Position of the custom frame origin
At Reference Frame Origin (default) | **At Center of Mass** | **Based on Geometric Feature**

Select the location of the frame origin. Options include:

- **At Reference Frame Origin** — Make the new frame origin coincident with the reference frame origin. This is the default option.

- **At Center of Mass** — Make the new frame origin coincident with the solid center of mass. The reference frame origin is located at the center of mass in symmetrical shapes such as spheres and bricks but not in certain extrusions or revolutions.
- **Based on Geometric Feature** — Place the new frame origin at the center of the selected geometry feature. Valid geometry features include surfaces, lines, and points. You must select a geometry feature from the visualization pane and then select the **Use Selected Feature** button. The name of the selected geometry feature appears in the field below this option.

Frame Axes: Primary Axis — Axis used to constrain the possible directions of the remaining frame axes

Along Reference Frame Axis (default) | **Along Principal Inertia Axis** | **Based on Geometric Feature**

Select the axis of the new frame that you want to set as the primary axis. The primary axis constrains the possible orientations of the remaining two axes. Specify the orientation of the primary axis by selecting from the following options:

- **Along Reference Frame Axis** — Align the primary axis with the selected axis of the reference frame.
- **Along Principal Inertia Axis** — Align the primary axis with the selected principal inertia axis. The principal inertia axes are those about which the products of inertia are zero.
- **Based on Geometric Feature** — Align the primary axis with the vector associated with the selected geometric feature. Valid geometric features include surfaces and lines.

Frame Axes: Secondary Axis — Axis used to constrain the possible directions of the remaining frame axis

Along Reference Frame Axis (default) | **Along Principal Inertia Axis** | **Based on Geometric Feature**

Select the axis of the new frame that you want to set as the secondary axis. The secondary axis is the projection of the selected direction onto the normal plane of the primary axis. Select the direction to project from the following options:

- **Along Reference Frame Axis** — Project the selected reference frame axis onto the normal plane of the primary axis. Align the secondary axis with the projection.
- **Along Principal Inertia Axis** — Project the selected principal inertia axis onto the normal plane of the primary axis. Align the secondary axis with the projection. The principal inertia axes are those about which the products of inertia are zero.

- **Based on Geometric Feature** — Project the vector associated with the selected geometry feature onto the normal plane of the primary axis. Align the secondary axis with the projection. Valid geometry features include surfaces and lines. You must select a geometry feature from the visualization pane and then select the **Use Selected Feature** button.

Extended Capabilities

C/C++ Code Generation

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

See Also

Rigid Transform | Variable Brick Solid | Variable Cylindrical Solid | Variable Spherical Solid

Topics

“Modeling Bodies”

“Representing Solid Geometry”

“Specifying Custom Inertias”

“Creating Custom Solid Frames”

“Manipulate the Color of a Solid”

Introduced in R2012a

Spherical Joint

Joint with one spherical primitive

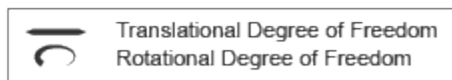
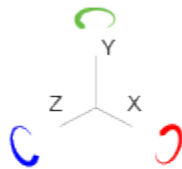


Library

Joints

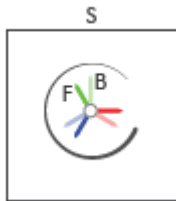
Description

This block represents a joint with three rotational degrees of freedom. One spherical primitive provides the three rotational degrees of freedom. The base and follower frame origins remain coincident during simulation.



Joint Degrees of Freedom

The joint block represents motion between the base and follower frames as a single time-varying transformation. The spherical primitive (S) applies this transformation, which causes the follower frame to rotate with respect to the base frame about an arbitrary 3-D axis. This joint primitive is not susceptible to gimbal lock.



Joint Transformation

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Spherical Primitive: State Targets

Specify the desired initial states of the spherical joint primitive and their relative priority levels. States that you can target include position and velocity. Use the priority level to

help the assembly algorithm decide which of the state targets in a model to more precisely satisfy should conflicts between them arise.

Even in the absence of state target conflicts, the true initial states may differ from those specified here. Such discrepancies can occur due to kinematic constraints arising from other parts of the model. If a state target cannot be satisfied precisely, it is satisfied approximately. Discrepancies are noted in Simscape Variable Viewer (**Analysis > Simscape > Variable Viewer**).

Specify Position Target

Check to specify the desired rotation of the follower frame relative to the base frame at the start of simulation.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Select a method to specify the joint primitive state target.

Method	Description
None	Constrain the base and follower frames to share the same orientation.
Aligned Axes	Set frame rotation by aligning two follower frame axes with two base frame axes.

Method	Description
Standard Axis	Specify frame rotation as an angle about a standard axis (x, y, or z).
Arbitrary Axis	Specify frame rotation as an angle about a general [x, y, z] axis.
Rotation Sequence	Specify frame rotation as a sequence of three elementary rotations.
Rotation Matrix	Specify frame rotation as a right-handed orthogonal rotation matrix.

Aligned Axes

Select two pairs of base-follower frame axes.

Parameter	Description
Pair 1	First pair of base-follower frame axes to align.
Pair 2	Second pair of base-follower frame axes to align. Axis choices depend on Pair 1 axis selections.

Standard Axis

Select a standard rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	Standard rotation axis (X, Y, or Z) resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Arbitrary Axis

Select a general 3-D rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	General rotation axis [X Y Z] resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Rotation Sequence

Specify a sequence of three elementary rotations about the selected permutation of x, y, and z axes. These rotation sequences are also known as Euler and Tait-Bryan sequences. The rotations are those of the follower frame relative to the frame selected in the **Rotate About** parameter.

If you set the **Rotate About** parameter to **Follower Frame**, the follower frame rotates about its own axes. These axes change orientation with each successive rotation. If you set the **Rotate About** parameter to **Base Frame**, the follower frame rotates about the fixed base frame axes.

Parameter	Description
Rotation About	Frame whose axes to rotate the follower frame about.
Sequence	Sequence of axes about which to apply the elementary rotations.
Angles	Three-element vector with elementary rotation angles about the axes specified in the Sequence parameter.

Rotation Matrix

Specify the 3×3 transformation matrix of a proper rotation between the base and follower frames. The matrix must be orthogonal and have determinant +1. The default matrix is $[1 \ 0 \ 0; \ 0 \ 1 \ 0; \ 0 \ 0 \ 1]$.

Specify Velocity Target

Check to specify the desired rotational velocity of the follower frame relative to the base frame at the start of simulation.

Value

Enter the relative rotational velocity of the follower frame against the base frame, as projected on the axes of the selected **Resolution Frame** (by default Follower). This parameter requires a three-element vector with the [x y z] components of the resolved relative velocity.

Resolution Frame

Select the frame in which to resolve the components of the velocity target. The resolution frame is not a measurement frame—the specified velocity is always that of the follower frame relative to the base frame. The resolution frame merely provides an alternate set of axes with respect to which to interpret the relative velocity components. The default setting is Follower.

Spherical Primitive: Internal Mechanics

Specify the spherical primitive internal mechanics. This includes linear spring and damping forces, accounting for energy storage and dissipation, respectively. To ignore internal mechanics, keep spring stiffness and damping coefficient values at the default value of 0.

Equilibrium Position

Select a method to specify the spring equilibrium position. The equilibrium position is the rotation angle between base and follower port frames at which the spring torque is zero.

Method	Description
None	Constrain the base and follower frames to share the same orientation.
Aligned Axes	Set frame rotation by aligning two follower frame axes with two base frame axes.
Standard Axis	Specify frame rotation as an angle about a standard axis (x, y, or z).
Arbitrary Axis	Specify frame rotation as an angle about a general [x, y, z] axis.
Rotation Sequence	Specify frame rotation as a sequence of three elementary rotations.

Method	Description
Rotation Matrix	Specify frame rotation as a right-handed orthogonal rotation matrix.

Aligned Axes

Select two pairs of base-follower frame axes.

Parameter	Description
Pair 1	First pair of base-follower frame axes to align.
Pair 2	Second pair of base-follower frame axes to align. Axis choices depend on Pair 1 axis selections.

Standard Axis

Select a standard rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	Standard rotation axis (X, Y, or Z) resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Arbitrary Axis

Select a general 3-D rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	General rotation axis [X Y Z] resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Rotation Sequence

Specify a sequence of three elementary rotations about the selected permutation of x, y, and z axes. These rotation sequences are also known as Euler and Tait-Bryan sequences. The rotations are those of the follower frame relative to the frame selected in the **Rotate About** parameter.

If you set the **Rotate About** parameter to **Follower Frame**, the follower frame rotates about its own axes. These axes change orientation with each successive rotation. If you set the **Rotate About** parameter to **Base Frame**, the follower frame rotates about the fixed base frame axes.

Parameter	Description
Rotation About	Frame whose axes to rotate the follower frame about.
Sequence	Sequence of axes about which to apply the elementary rotations.
Angles	Three-element vector with elementary rotation angles about the axes specified in the Sequence parameter.

Rotation Matrix

Specify the 3×3 transformation matrix of a proper rotation between the base and follower frames. The matrix must be orthogonal and have determinant +1. The default matrix is $[1 \ 0 \ 0; \ 0 \ 1 \ 0; \ 0 \ 0 \ 1]$.

Spring Stiffness

Enter the linear spring constant. This is the torque required to displace the joint primitive by a unit angle. The term linear refers to the mathematical form of the spring equation. The default is 0. Select a physical unit. The default is N*m/deg.

Damping Coefficient

Enter the linear damping coefficient. This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames. The default is 0. Select a physical unit. The default is N*m/(deg/s).

Spherical Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper

bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Spherical Primitive: Actuation

Specify actuation options for the spherical joint primitive. Actuation modes include **Torque** only. Selecting a torque input adds the corresponding physical signal port to the block. Use this port to specify the actuation torque signal.

Torque

Select a source for the actuation torque. The default setting is **None**.

Actuation Torque Setting	Description
None	Apply no actuation torque.
Provided by Input	Apply an actuation torque based on a physical signal. The signal specifies the torque acting on the follower frame with respect to the base frame. An equal and opposite torque acts on the base frame. Selecting this option exposes additional parameters.

Torque (X), Torque (Y), Torque (Z)

Select in order to actuate the spherical joint primitive about each standard Cartesian axis (X, Y, Z) separately. The block exposes the corresponding physical signal ports. Use these ports to specify the actuation torque signals. The signals must be scalar values.

Torque (XYZ)

Select in order to actuate the spherical joint primitive about an arbitrary axis [X Y Z]. The block exposes the corresponding physical signal port. Use this port to specify the actuation torque signal. The signal must be a 3-D vector.

Frame

Select the frame to resolve the actuation torque signal in. The axes of this frame establish the directions of the X, Y, and Z torque components. The default setting is **Base**.

Spherical Primitive: Sensing

Select the motion variables to sense in the spherical joint primitive. The block adds the corresponding physical signal ports. Use these ports to output the numerical values of the motion variables.

The block measures each motion variable for the follower frame with respect to the base frame. It resolves that variable in the resolution frame that you select from the **Frame** drop-down list.

Motion Variables	Description
Position	Quaternion describing follower frame rotation with respect to base frame. The quaternion coefficients are $\left[\cos\left(\frac{\theta}{2}\right), n_x \sin\left(\frac{\theta}{2}\right), n_y \sin\left(\frac{\theta}{2}\right), n_z \sin\left(\frac{\theta}{2}\right) \right]$. The measurement is the same in all measurement frames.
Velocity (X), Velocity (Y), Velocity (Z)	Angular velocity components about X, Y, and Z axes.
Velocity	3-D angular velocity vector with components about X, Y, and Z axes.
Acceleration (X), Acceleration (Y), Acceleration (Z)	Angular acceleration components about X, Y, and Z axes.
Acceleration	3-D angular acceleration vector with components about X, Y, and Z axes.

Frame

Select the frame to resolve the measurement in. The axes of this frame establish the directions of X, Y, and Z vector components. The default setting is Base.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Actuation Ports

The spherical joint primitive provides the following actuation ports:

- t — Actuation torque vector $[t_x, t_y, t_z]$ acting on the spherical joint primitive
- t_x, t_y, t_z — X, Y, and Z components of the actuation torque acting on the spherical joint primitive

Sensing Ports

The spherical primitive provides the following sensing ports:

- Q — Orientation of the spherical joint primitive in quaternion form
- w_x, w_y, w_z — X, Y, and Z angular velocity components of the spherical joint primitive
- w — Angular velocity $[w_x, w_y, w_z]$ of the spherical joint primitive
- b_x, b_y, b_z — X, Y, and Z angular acceleration components of the spherical joint primitive
- b — Angular acceleration $[b_x, b_y, b_z]$ of the spherical joint primitive

The following sensing ports provide the composite forces and torques acting on the joint:

- f_c — Constraint force
- t_c — Constraint torque
- f_t — Total force
- t_t — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

6-DOF Joint | Bushing Joint | Gimbal Joint | Revolute joint

Topics

“Motion Sensing”

“Measurement Frames”

“Actuating and Sensing with Physical Signals”

Introduced in R2012a

Spline

Cubic interpolating plane curve or space curve

Library: Simscape / Multibody / Curves and Surfaces



Description

This block represents a continuous spline curve based on cubic interpolation between the points specified. The curve can be two-dimensional, such as a planar cam profile, or three-dimensional, such as a roller coaster track. Depending on the end conditions selected, the curve can be either open or closed.



Cam profile — An Example of a 2-D Spline Curve

Whether a spline curve is two- or three-dimensional depends solely on the coordinate matrix dimensions. A two-column matrix specifies a two-dimensional curve in the xy plane. Each row in this matrix provides the $[x, y]$ coordinates of a point. A three-column matrix specifies a three-dimensional curve. Each row in this matrix provides the $[x, y, z]$ coordinates of a point. All coordinates are resolved in the local reference frame of the block.

The spline curve is a piecewise function of third-order polynomial segments connected end-to-end. The curve is built such that adjacent polynomial segments have the same first and second derivatives at the shared end point. If the curve is periodic, an additional curve segment connects the last point specified to the first point. The first and second

derivatives of this segment matches those of the adjacent segments at the shared end point.

Ports

Frame

R — Reference frame

frame

Spline curve reference frame. Connect this frame port to that of another block to resolve the placement of the spline curve in a model.

G — Geometry Specification

geometry

Spline curve geometry. Connect this geometry port to that of a Point On Curve Constraint block to provide that block with a spline curve specification.

Parameters

Interpolation Points — Matrix of points used to generate the spline

6-by-3 matrix associated with a space curve (default) | Two-column matrix for plane curves | Three-column matrix for space curves

Coordinates of the interpolation points specified as an $[x, y]$ matrix for a 2-D curve or $[x, y, z]$ matrix for a 3-D curve. Coordinates are resolved in the reference frame of the block.

If you set the end conditions to **Periodic (Closed)**, the block joins the first and last data points with an additional spline segment. Like all spline segments, the additional segment and its first two derivatives are continuous at the shared point.

Each data point in the coordinate matrix must be unique. If the curve is closed, you must ensure that the first and last data points have different coordinates.

End Conditions — Treatment to apply to the curve end points

Sphere (default) | Cube | Frame

Type of end conditions to use. Periodic end conditions correspond to a closed curve. Natural end conditions correspond to an open curve. The default setting is **Periodic (Closed)**.

Graphic

Type — Solid visualization setting

From Geometry (default) | Marker | None

Visualization setting for this solid. Use the default setting, **From Geometry**, to show the solid geometry. Select **Marker** to show a graphic marker such as a sphere or frame. Select **None** to disable visualization for this solid.

Marker: Shape — Shape of the graphic marker

Sphere (default) | Cube | Frame

Geometrical shape of the graphic marker. Mechanics Explorer shows the marker using the selected shape.

Marker: Size — Pixel size of the graphic marker

10 (default)

Absolute size of the graphic marker in screen pixels. The marker size is invariant with zoom level.

Visual Properties — Parameterizations for color and opacity

Simple (default) | Advanced

Parameterization for specifying visual properties. Select **Simple** to specify color and opacity. Select **Advanced** to add specular highlights, ambient shadows, and self-illumination effects.

Simple: Color — True color as [R,G,B] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three-element vector with values constrained to 0-1

RGB color vector with red (R), green (G), and blue (B) color amounts specified on a 0-1 scale. A color picker provides an alternative interactive means of specifying a color. If you change the **Visual Properties** setting to **Advanced**, the color specified in this parameter becomes the **Diffuse Color** vector.

Simple: Opacity — Surface opacity as scalar number on 0-1 scale

1.0 (default) | scalar with value constrained to 0-1

Graphic opacity specified on a scale of 0-1. An opacity of 0 corresponds to a completely transparent graphic and an opacity of 1 to a completely opaque graphic.

Advanced: Diffuse Color — True color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three- or four-element vector with values constrained to 0-1

True color under direct white light specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. An optional fourth element specifies the color opacity also on a scale of 0-1. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Specular Color — Highlight color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of specular highlights specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Ambient Color — Shadow color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of shadow areas in diffuse ambient light, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Emissive Color — Self-illumination color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Surface color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Shininess — Highlight sharpness as scalar number on 0-128 scale

75 (default) | scalar with value constrained to 0-128

Sharpness of specular light reflections, specified as a scalar number on a 0-128 scale. Increase the shininess value for smaller but sharper highlights. Decrease the value for larger but smoother highlights.

Extended Capabilities

C/C++ Code Generation

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

See Also

Inertia | Solid

Topics

“Visualize Simscape Multibody Frames”

“Manipulate the Color of a Solid”

Introduced in R2015b

Spring and Damper Force

Force proportional to the distance and relative velocity between two frame origins



Library

Forces and Torques

Description

This block represents a linear spring and damper force pair acting reciprocally between base and follower frame origins. The two forces in the pair have equal magnitude but opposite directions. One force acts on the base frame origin, along the vector connecting follower to base frame origins. The other force acts on the follower frame origin, along the vector connecting base to follower frame origins.

The magnitude of the spring force component is proportional to the distance between base and follower frame origins. This distance is the length of the straight line segment connecting the two origins. The magnitude of the damper force component is proportional to the relative velocity of the follower frame origin with respect to the base frame.

Parameters

Natural Length

Enter the equilibrium distance between the base and follower frame origins. This is the distance at which the magnitude of the spring force is zero. The default value is 0. Select or enter a physical unit.

Spring Stiffness

Enter the value of the linear spring constant. The value must be greater than or equal to zero. The default value is zero. Select or enter a physical unit.

Damping Coefficient

Enter the value of the linear damping coefficient. The value must be greater than or equal to zero. The default value is zero. Select or enter a physical unit.

Sense Force

Select to sense the signed magnitude of the spring and damper force acting between the two frame origins. The block exposes an additional physical signal port to output the force signal. The output signal is a scalar value. This value is positive if the force is repulsive; it is negative if the force is attractive.

Ports

The block contains frame ports B and F, representing base and follower frames, respectively.

Selecting the **Sense Force** check box in the block dialog box adds a physical signal port, **fm**.

Extended Capabilities

C/C++ Code Generation

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

See Also

[External Force and Torque](#) | [Internal Force](#) | [Inverse Square Law Force](#)

Topics

[“Actuating and Sensing with Physical Signals”](#)

Introduced in R2012a

Telescoping Joint

Joint with one prismatic and one spherical joint primitive

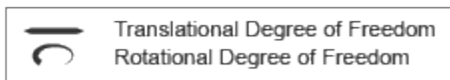
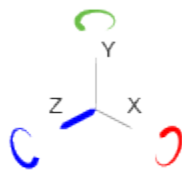


Library

Joints

Description

This block represents a joint with one translational and three rotational degrees of freedom. One prismatic primitive provides the translational degree of freedom. One spherical primitive provides the three rotational degrees of freedom.



Joint Degrees of Freedom

The joint block represents motion between the base and follower frames as a sequence of time-varying transformations. Each joint primitive applies one transformation in this sequence. The transformation translates or rotates the follower frame with respect to the

joint primitive base frame. For all but the first joint primitive, the base frame coincides with the follower frame of the previous joint primitive in the sequence.

At each time step during the simulation, the joint block applies the sequence of time-varying frame transformations in this order:

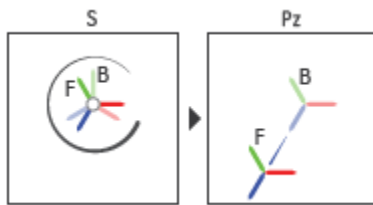
1 Rotation:

- About an arbitrary 3-D axis resolved in the Spherical Primitive (S) base frame.

2 Translation:

- Along the Z axis of the Z Prismatic Primitive (Pz) base frame. This frame is coincident with the Spherical Primitive (S) follower frame.

The figure shows the sequence in which the joint transformations occur at a given simulation time step. The resulting frame of each transformation serves as the base frame for the following transformation. Because 3-D rotation occurs as a single rotation about an arbitrary 3-D axis (as opposed to three separate rotations about the X, Y, Z axes), gimbal lock does not occur.



Joint Transformation Sequence

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound,

both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Spherical Primitive: State Targets

Specify the desired initial states of the spherical joint primitive and their relative priority levels. States that you can target include position and velocity. Use the priority level to help the assembly algorithm decide which of the state targets in a model to more precisely satisfy should conflicts between them arise.

Even in the absence of state target conflicts, the true initial states may differ from those specified here. Such discrepancies can occur due to kinematic constraints arising from other parts of the model. If a state target cannot be satisfied precisely, it is satisfied approximately. Discrepancies are noted in Simscape Variable Viewer (**Analysis > Simscape > Variable Viewer**).

Specify Position Target

Check to specify the desired rotation of the follower frame relative to the base frame at the start of simulation.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Select a method to specify the joint primitive state target.

Method	Description
None	Constrain the base and follower frames to share the same orientation.
Aligned Axes	Set frame rotation by aligning two follower frame axes with two base frame axes.
Standard Axis	Specify frame rotation as an angle about a standard axis (x , y , or z).
Arbitrary Axis	Specify frame rotation as an angle about a general $[x, y, z]$ axis.
Rotation Sequence	Specify frame rotation as a sequence of three elementary rotations.
Rotation Matrix	Specify frame rotation as a right-handed orthogonal rotation matrix.

Aligned Axes

Select two pairs of base-follower frame axes.

Parameter	Description
Pair 1	First pair of base-follower frame axes to align.

Parameter	Description
Pair 2	Second pair of base-follower frame axes to align. Axis choices depend on Pair 1 axis selections.

Standard Axis

Select a standard rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	Standard rotation axis (X, Y, or Z) resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Arbitrary Axis

Select a general 3-D rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	General rotation axis [X Y Z] resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Rotation Sequence

Specify a sequence of three elementary rotations about the selected permutation of x, y, and z axes. These rotation sequences are also known as Euler and Tait-Bryan sequences. The rotations are those of the follower frame relative to the frame selected in the **Rotate About** parameter.

If you set the **Rotate About** parameter to **Follower Frame**, the follower frame rotates about its own axes. These axes change orientation with each successive rotation. If you set the **Rotate About** parameter to **Base Frame**, the follower frame rotates about the fixed base frame axes.

Parameter	Description
Rotation About	Frame whose axes to rotate the follower frame about.
Sequence	Sequence of axes about which to apply the elementary rotations.
Angles	Three-element vector with elementary rotation angles about the axes specified in the Sequence parameter.

Rotation Matrix

Specify the 3×3 transformation matrix of a proper rotation between the base and follower frames. The matrix must be orthogonal and have determinant +1. The default matrix is $[1 \ 0 \ 0; \ 0 \ 1 \ 0; \ 0 \ 0 \ 1]$.

Specify Velocity Target

Check to specify the desired rotational velocity of the follower frame relative to the base frame at the start of simulation.

Value

Enter the relative rotational velocity of the follower frame against the base frame, as projected on the axes of the selected **Resolution Frame** (by default **Follower**). This parameter requires a three-element vector with the $[x \ y \ z]$ components of the resolved relative velocity.

Resolution Frame

Select the frame in which to resolve the components of the velocity target. The resolution frame is not a measurement frame—the specified velocity is always that of the follower frame relative to the base frame. The resolution frame merely provides an alternate set of axes with respect to which to interpret the relative velocity components. The default setting is **Follower**.

Spherical Primitive: Internal Mechanics

Specify the spherical primitive internal mechanics. This includes linear spring and damping forces, accounting for energy storage and dissipation, respectively. To ignore internal mechanics, keep spring stiffness and damping coefficient values at the default value of 0.

Equilibrium Position

Select a method to specify the spring equilibrium position. The equilibrium position is the rotation angle between base and follower port frames at which the spring torque is zero.

Method	Description
None	Constrain the base and follower frames to share the same orientation.
Aligned Axes	Set frame rotation by aligning two follower frame axes with two base frame axes.
Standard Axis	Specify frame rotation as an angle about a standard axis (x , y , or z).
Arbitrary Axis	Specify frame rotation as an angle about a general $[x, y, z]$ axis.
Rotation Sequence	Specify frame rotation as a sequence of three elementary rotations.
Rotation Matrix	Specify frame rotation as a right-handed orthogonal rotation matrix.

Aligned Axes

Select two pairs of base-follower frame axes.

Parameter	Description
Pair 1	First pair of base-follower frame axes to align.
Pair 2	Second pair of base-follower frame axes to align. Axis choices depend on Pair 1 axis selections.

Standard Axis

Select a standard rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	Standard rotation axis (X, Y, or Z) resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Arbitrary Axis

Select a general 3-D rotation axis, resolved in the base frame, and specify the follower frame rotation angle.

Parameter	Description
Axis	General rotation axis [X Y Z] resolved in the base frame.
Angle	Follower frame rotation angle about the rotation axis with respect to the base frame.

Rotation Sequence

Specify a sequence of three elementary rotations about the selected permutation of x, y, and z axes. These rotation sequences are also known as Euler and Tait-Bryan sequences. The rotations are those of the follower frame relative to the frame selected in the **Rotate About** parameter.

If you set the **Rotate About** parameter to **Follower Frame**, the follower frame rotates about its own axes. These axes change orientation with each successive rotation. If you set the **Rotate About** parameter to **Base Frame**, the follower frame rotates about the fixed base frame axes.

Parameter	Description
Rotation About	Frame whose axes to rotate the follower frame about.
Sequence	Sequence of axes about which to apply the elementary rotations.

Parameter	Description
Angles	Three-element vector with elementary rotation angles about the axes specified in the Sequence parameter.

Rotation Matrix

Specify the 3×3 transformation matrix of a proper rotation between the base and follower frames. The matrix must be orthogonal and have determinant +1. The default matrix is [1 0 0; 0 1 0; 0 0 1].

Spring Stiffness

Enter the linear spring constant. This is the torque required to displace the joint primitive by a unit angle. The term linear refers to the mathematical form of the spring equation. The default is 0. Select a physical unit. The default is N*m/deg.

Damping Coefficient

Enter the linear damping coefficient. This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames. The default is 0. Select a physical unit. The default is N*m/(deg/s).

Spherical Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Spherical Primitive: Actuation

Specify actuation options for the spherical joint primitive. Actuation modes include **Torque** only. Selecting a torque input adds the corresponding physical signal port to the block. Use this port to specify the actuation torque signal.

Torque

Select a source for the actuation torque. The default setting is None.

Actuation Torque Setting	Description
None	Apply no actuation torque.

Actuation Torque Setting	Description
Provided by Input	Apply an actuation torque based on a physical signal. The signal specifies the torque acting on the follower frame with respect to the base frame. An equal and opposite torque acts on the base frame. Selecting this option exposes additional parameters.

Torque (X), Torque (Y), Torque (Z)

Select in order to actuate the spherical joint primitive about each standard Cartesian axis (X, Y, Z) separately. The block exposes the corresponding physical signal ports. Use these ports to specify the actuation torque signals. The signals must be scalar values.

Torque (XYZ)

Select in order to actuate the spherical joint primitive about an arbitrary axis [X Y Z]. The block exposes the corresponding physical signal port. Use this port to specify the actuation torque signal. The signal must be a 3-D vector.

Frame

Select the frame to resolve the actuation torque signal in. The axes of this frame establish the directions of the X, Y, and Z torque components. The default setting is Base.

Spherical Primitive: Sensing

Select the motion variables to sense in the spherical joint primitive. The block adds the corresponding physical signal ports. Use these ports to output the numerical values of the motion variables.

The block measures each motion variable for the follower frame with respect to the base frame. It resolves that variable in the resolution frame that you select from the **Frame** drop-down list.

Motion Variables	Description
Position	Quaternion describing follower frame rotation with respect to base frame. The quaternion coefficients are $\left[\cos\left(\frac{\theta}{2}\right), n_x \sin\left(\frac{\theta}{2}\right), n_y \sin\left(\frac{\theta}{2}\right), n_z \sin\left(\frac{\theta}{2}\right) \right]$. The measurement is the same in all measurement frames.
Velocity (X), Velocity (Y), Velocity (Z)	Angular velocity components about X, Y, and Z axes.
Velocity	3-D angular velocity vector with components about X, Y, and Z axes.
Acceleration (X), Acceleration (Y), Acceleration (Z)	Angular acceleration components about X, Y, and Z axes.
Acceleration	3-D angular acceleration vector with components about X, Y, and Z axes.

Frame

Select the frame to resolve the measurement in. The axes of this frame establish the directions of X, Y, and Z vector components. The default setting is Base.

Prismatic Primitive: State Targets

Specify the prismatic primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative position, measured along the joint primitive axis, of the follower frame origin with respect to the base frame origin. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative velocity, measured along the joint primitive axis, of the follower frame

origin with respect to the base frame origin. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is m for position and m/s for velocity.

Prismatic Primitive: Internal Mechanics

Specify the prismatic primitive internal mechanics. Internal mechanics include linear spring forces, accounting for energy storage, and damping forces, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the distance between base and follower frame origins at which the spring force is zero. The default value is 0. Select or enter a physical unit. The default is m.

Spring Stiffness

Enter the linear spring constant. This is the force required to displace the joint primitive by a unit distance. The default is 0. Select or enter a physical unit. The default is N/m.

Damping Coefficient

Enter the linear damping coefficient. This is the force required to maintain a constant joint primitive velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N/(m/s).

Prismatic Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Prismatic Primitive: Actuation

Specify actuation options for the prismatic joint primitive. Actuation modes include **Force** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Actuation signals are resolved in the base frame.

Force

Select an actuation force setting. The default setting is **None**.

Actuation Force Setting	Description
None	No actuation force.
Provided by Input	Actuation force from physical signal input. The signal provides the force acting on the follower frame with respect to the base frame along the joint primitive axis. An equal and opposite force acts on the base frame.
Automatically computed	Actuation force from automatic calculation. Simscape Multibody computes and applies the actuation force based on model dynamics.

Motion

Select an actuation motion setting. The default setting is **Automatically Computed**.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Prismatic Primitive: Sensing

Select the variables to sense in the prismatic joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative position of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Velocity

Select this option to sense the relative velocity of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Acceleration

Select this option to sense the relative acceleration of the follower frame origin with respect to the base frame origin along the joint primitive axis.

Actuator Force

Select this option to sense the actuation force acting on the follower frame with respect to the base frame along the joint primitive axis.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Actuation Ports

The prismatic joint primitive provides the following actuation ports:

- f_z — Actuation force of the Z prismatic joint primitive
- p_z — Desired trajectory of the Z prismatic joint primitive

The spherical joint primitive provides the following actuation ports:

- t — Actuation torque vector $[t_x, t_y, t_z]$ acting on the spherical joint primitive
- t_x, t_y, t_z — X, Y, and Z components of the actuation torque acting on the spherical joint primitive

Sensing Ports

The prismatic primitive provides the following sensing ports:

- p_z — Position of the Z prismatic joint primitive
- v_z — Velocity of the Z prismatic joint primitive
- a_z — Acceleration of the Z prismatic joint primitive
- f_z — Actuation force acting on the Z prismatic joint primitive

The spherical primitive provides the following sensing ports:

- Q — Orientation of the spherical joint primitive in quaternion form
- w_x, w_y, w_z — X, Y, and Z angular velocity components of the spherical joint primitive

- w — Angular velocity [w_x , w_y , w_z] of the spherical joint primitive
- b_x , b_y , b_z — X, Y, and Z angular acceleration components of the spherical joint primitive
- b — Angular acceleration [b_x , b_y , b_z] of the spherical joint primitive

The following sensing ports provide the composite forces and torques acting on the joint:

- f_c — Constraint force
- t_c — Constraint torque
- f_t — Total force
- t_t — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Prismatic Joint | Spherical Joint

Introduced in R2012a

Transform Sensor

Sensor that measures the spatial relationship between two frames



Library

Frames and Transforms

Description

This block represents a sensor that measures the spatial relationship between two frames. Parameters that this sensor measures include rotational and translational position, velocity, and acceleration. The sensor can measure these parameters between any two frames in a model. This block provides the broadest motion sensing capability in Simscape Multibody.

Each measurement provides the value of a parameter for the follower frame with respect to the base frame, resolved in the **Measurement Frame** that you choose. Measurement frames include World as well as rotating and non-rotating base and follower frames.

To output a parameter, the block provides a set of optional physical signal ports. Ports remain hidden until you select the corresponding parameters in the dialog box. Each port outputs a parameter as a time-varying physical signal. By default, measurements are in SI units. If connecting to Simulink® blocks, you can use the PS-Simulink Converter block to select a different physical unit.

Parameters

Measurement Frame

Select a frame in which to resolve the selected spatial measurements. The choice of measurement frame affects the expression of a vector quantity in terms of its X, Y, and

Z components. Some quantities, such as **Angle**, are not affected by the choice of measurement frame. For more information, see “Measurement Frames”. The default is **World**.

Rotation

Select the rotation parameters to sense. These parameters encode the rotation operation required to bring the base frame into coincidence with the follower frame. Rotation observes the right-hand rule: with the rotation axis pointing out of the screen, counterclockwise motion defines positive rotation, while clockwise motion defines negative rotation.

Non-vector quantities require no measurement frame for resolution; these quantities are unaffected by measurement frame choice. Vector quantities, such as **Axis**, are *always* resolved in either base or follower measurement frames; the **World** measurement frame does not apply.

Angle

3-D rotation angle of the follower frame with respect to the base frame. Selecting **Angle** exposes physical signal port **q**.

Axis

Vector components of the normalized rotation axis. The output is a three-element vector with the X, Y, and Z axis components resolved in the measurement frame. Selecting **Axis** exposes physical signal port **axs**.

Quaternion

Unit quaternion that describes the pure rotation of the follower frame with respect to the base frame. The output is a four-element vector with the scalar (S) and vector (V_x, V_y, V_z) quaternion coefficients. The vector provides the coefficients in the order [$S V_x V_y V_z$]. Selecting **Quaternion** exposes physical signal port **Q**.

Transform

Transform matrix that describes the pure rotation of the follower frame with respect to the base frame. The output is a nine-element, 3×3 matrix. Selecting **Transform** exposes physical signal port **R**.

Angular Velocity

Select the angular velocity parameters to sense. The parameters encode the angular velocity of the follower frame with respect to the base frame, resolved in the

measurement frame. Rotation observes the right-hand rule: with the rotation axis pointing out of the screen, counterclockwise motion defines positive rotation, while clockwise motion defines negative rotation.

Omega X/Omega Y/Omega Z

Relative angular velocities about the X, Y, and Z axes of the base frame. Selecting **Omega X**, **Omega Y**, and **Omega Z** exposes physical signal ports **wx**, **wy**, and **wz**.

Quaternion

Unit quaternion that describes the angular velocity of the follower frame with respect to the base frame. The output is a four-element vector with the scalar (S) and vector (V_x, V_y, V_z) quaternion coefficients. The vector provides the coefficients in the order [$S V_x V_y V_z$]. Selecting **Quaternion** exposes physical signal port **Qd**.

Transform

Transform matrix that describes the angular velocity of the follower frame with respect to the base frame. The output is a nine-element, 3×3 matrix. Selecting **Transform** exposes physical signal port **Rd**.

Angular Acceleration

Select the angular acceleration parameters to sense. The parameters encode the angular acceleration of the follower frame with respect to the base frame, resolved in the measurement frame. Rotation observes the right-hand rule: with the rotation axis pointing out of the screen, counterclockwise motion defines positive rotation, while clockwise motion defines negative rotation.

Alpha X/Alpha Y/Alpha Z

Relative angular accelerations about the X, Y, and Z axes of the base frame. Selecting **Alpha X**, **Alpha Y**, and **Alpha Z** exposes physical signal ports **bx**, **by**, and **bz**.

Quaternion

Unit quaternion that describes the angular acceleration of the follower frame with respect to the base frame. The output is a four-element vector with the scalar (S) and vector (V_x, V_y, V_z) quaternion coefficients. The vector provides the coefficients in the order [$S V_x V_y V_z$]. Selecting **Quaternion** exposes physical signal port **Qdd**.

Transform

Transform matrix that describes the angular acceleration of the follower frame with respect to the base frame. The output is a nine-element, 3×3 matrix. Selecting **Transform** exposes physical signal port **Rdd**.

Translation

Select the translation parameters to sense. The parameters encode the translation of the follower frame with respect to the base frame, resolved in the measurement frame.

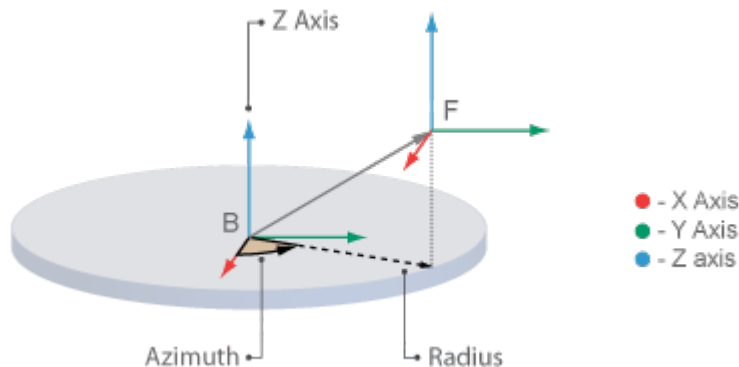
X/Y/Z

Offset vector from the base frame origin to the follower frame origin along the X, Y, and Z axes. Selecting **X**, **Y**, and **Z** exposes physical signal ports **x**, **y**, and **z**.

Radius

Standard radius coordinate found in cylindrical coordinate systems. This radius is the shortest distance from the base frame Z axis to the follower frame origin. The value of the radius is always greater than or equal to zero. Selecting **Radius** exposes physical signal port **rad**.

The figure shows the cylindrical translation parameters **Z**, **Radius**, and **Azimuth**.



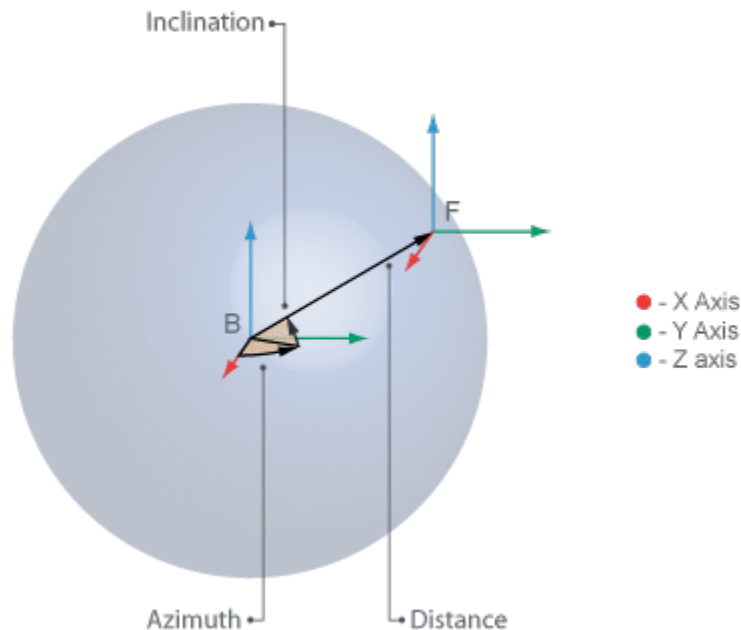
Azimuth

Standard azimuth coordinate found in cylindrical and spherical coordinate systems. The azimuth is the angle from the base frame +X axis to the projection of the ray connecting base to follower frame origins onto the base frame XY plane. The angle measurement observes the right-hand rule. The azimuth falls in the range $[-180^\circ, +180^\circ]$. If base and follower frame origins coincide with each other, the azimuth is undefined. Selecting **Azimuth** exposes sensing port **azm**.

Distance

Standard radius found in spherical coordinate systems. This is the distance from the origin of the base frame to the origin of the follower frame. This distance is always equal to or greater than zero. Selecting **Distance** exposes sensing port **dst**.

The figure shows the spherical translation parameters **Azimuth**, **Distance**, and **Inclination**.



Inclination

Standard inclination found in spherical coordinate systems. The inclination is the angle of the ray connecting base to follower frame origins with respect to the projection of this ray onto the base frame XY plane. The angle measurement observes the right-hand rule. The inclination falls in the range $[-90^\circ, +90^\circ]$. If base and follower frame origins coincide with each other, the inclination is undefined. Selecting **Inclination** exposes sensing port **inc**.

Velocity

Select the linear velocity parameters to sense. The parameters encode the linear velocity of the follower frame with respect to the base frame, resolved in the measurement frame. Differentiation of a translation parameter occurs in measurement coordinates, after that parameter is resolved in the chosen measurement frame.

X/Y/Z

Relative linear velocities along the X, Y, and Z axes. Selecting **X**, **Y**, and **Z** exposes physical signal ports **vx**, **vy**, and **vz**.

Radius

Time rate of change of the **Radius** coordinate defined under **Translation**. Selecting **Radius** exposes physical signal port **vrad**.

Azimuth

Time rate of change of the **Azimuth** coordinate defined under **Translation**. Selecting **Azimuth** exposes physical signal port **vazm**.

Distance

Time rate of change of the **Distance** coordinate defined under **Translation**. Selecting **Distance** exposes physical signal port **vdst**.

Inclination

Time rate of change of the **Inclination** coordinate defined under **Translation**. Selecting **Inclination** exposes physical signal port **vinc**.

Acceleration

Select the linear acceleration parameters to sense. The parameters encode the linear acceleration of the follower frame with respect to the base frame, resolved in the measurement frame. Differentiation of a translation parameter occurs in measurement coordinates, after that parameter is resolved in the chosen measurement frame.

X/Y/Z

Relative linear accelerations along the X, Y, and Z axes. Selecting **X**, **Y**, and **Z** exposes physical signal ports **ax**, **ay**, and **az**.

Radius

Second time-derivative of the **Radius** coordinate defined under **Translation**. Selecting **Radius** exposes physical signal port **arad**.

Azimuth

Second time-derivative of the **Azimuth** coordinate defined under **Translation**.
Selecting **Azimuth** exposes physical signal port **aazm**.

Distance

Second time-derivative of the **Distance** coordinate defined under **Translation**.
Selecting **Distance** exposes physical signal port **adst**.

Inclination

Second time-derivative of the **Inclination** coordinate defined under **Translation**.
Selecting **Inclination** exposes physical signal port **ainc**.

Ports

The block contains frame ports B and F, representing base and follower frames, respectively.

Extended Capabilities

C/C++ Code Generation

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

See Also

Rigid Transform

Topics

“Motion Sensing”

“Working with Frames”

Introduced in R2012a

Universal Joint

Joint with two revolute primitives

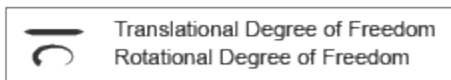
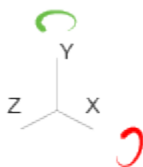


Library

Joints

Description

This block represents a joint with two rotational degrees of freedom. Two revolute primitives provide the two rotational degrees of freedom. The base and follower frame origins remain coincident during simulation.



Joint Degrees of Freedom

The joint block represents motion between the base and follower frames as a sequence of time-varying transformations. Each joint primitive applies one transformation in this sequence. The transformation translates or rotates the follower frame with respect to the

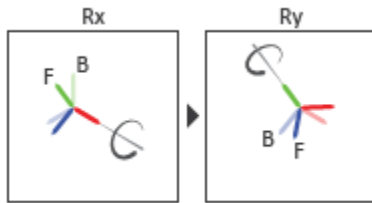
joint primitive base frame. For all but the first joint primitive, the base frame coincides with the follower frame of the previous joint primitive in the sequence.

At each time step during the simulation, the joint block applies the sequence of time-varying frame transformations in this order:

1 Rotation:

- a** About the X axis of the X Revolute Primitive (Rx) base frame.
- b** About the Y axis of the Y Revolute Primitive (Ry) base frame. This frame is coincident with the X Revolute Primitive (Rx) follower frame.

The figure shows the sequence in which the joint transformations occur at a given simulation time step. The resulting frame of each transformation serves as the base frame for the following transformation.



Joint Transformation Sequence

A set of optional state targets guide assembly for each joint primitive. Targets include position and velocity. A priority level sets the relative importance of the state targets. If two targets are incompatible, the priority level determines which of the targets to satisfy.

Internal mechanics parameters account for energy storage and dissipation at each joint primitive. Springs act as energy storage elements, resisting any attempt to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

In all but lead screw and constant velocity primitives, joint limits serve to curb the range of motion between frames. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. To enforce the bounds, the joint adds to each a spring-damper. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the deeper the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Each joint primitive has a set of optional actuation and sensing ports. Actuation ports accept physical signal inputs that drive the joint primitives. These inputs can be forces and torques or a desired joint trajectory. Sensing ports provide physical signal outputs that measure joint primitive motion as well as actuation forces and torques. Actuation modes and sensing types vary with joint primitive.

Parameters

Revolute Primitive: State Targets

Specify the revolute primitive state targets and their priority levels. A state target is the desired value for one of the joint state parameters—position and velocity. The priority level is the relative importance of a state target. It determines how precisely the target must be met. Use the Model Report tool in Mechanics Explorer to check the assembly status for each joint state target.

Specify Position Target

Select this option to specify the desired joint primitive position at time zero. This is the relative rotation angle, measured about the joint primitive axis, of the follower frame with respect to the base frame. The specified target is resolved in the base frame. Selecting this option exposes priority and value fields.

Specify Velocity Target

Select this option to specify the desired joint primitive velocity at time zero. This is the relative angular velocity, measured about the joint primitive axis, of the follower frame with respect to the base frame. It is resolved in the base frame. Selecting this option exposes priority and value fields.

Priority

Select state target priority. This is the importance level assigned to the state target. If all state targets cannot be simultaneously satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. This option applies to both position and velocity state targets.

Priority Level	Description
High (desired)	Satisfy state target precisely
Low (approximate)	Satisfy state target approximately

Note During assembly, high-priority targets behave as exact guides. Low-priority targets behave as rough guides.

Value

Enter the state target numerical value. The default is 0. Select or enter a physical unit. The default is deg for position and deg/s for velocity.

Revolute Primitive: Internal Mechanics

Specify the revolute primitive internal mechanics. Internal mechanics include linear spring torques, accounting for energy storage, and linear damping torques, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0.

Equilibrium Position

Enter the spring equilibrium position. This is the rotation angle between base and follower frames at which the spring torque is zero. The default value is 0. Select or enter a physical unit. The default is deg.

Spring Stiffness

Enter the linear spring constant. This is the torque required to rotate the joint primitive by a unit angle. The default is 0. Select or enter a physical unit. The default is N*m/deg.

Damping Coefficient

Enter the linear damping coefficient. This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames. The default is 0. Select or enter a physical unit. The default is N*m/(deg/s).

Revolute Primitive: Limits

Limit the range of motion of the joint primitive. Joint limits use spring-dampers to resist travel past the bounds of the range. A joint primitive can have a lower bound, an upper bound, both, or, in the default state, neither. The stiffer the spring, the harder the stop, or bounce, if oscillations arise. The stronger the damper, the larger the viscous losses that gradually lessen contact oscillations or, in overdamped primitives, keep them from forming altogether.

Specify Lower Limit

Select to add a lower bound to the range of motion of the joint primitive.

Specify Upper Limit

Select to add an upper bound to the range of motion of the joint primitive.

Value

Location past which to resist joint travel. The location is the offset from base to follower, as measured in the base frame, at which contact begins. It is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

Spring Stiffness

Resistance of the contact spring to displacement past the joint limit. The spring is linear and its stiffness is constant. The larger the value, the harder the stop. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Damping Coefficient

Resistance of the contact damper to motion past the joint limit. The damper is linear and its coefficient is constant. The larger the value, the greater the viscous losses that gradually lessen contact oscillations, if any arise. The proportion of spring to damper forces determines whether the stop is underdamped and prone to oscillations on contact.

Transition Region

Region over which to raise the spring-damper force to its full value. The region is a distance along an axis in prismatic primitives, an angle about an axis in revolute primitives, and an angle between two axes in spherical primitives.

The smaller the region, the sharper the onset of contact and the smaller the time-step required of the solver. In the trade-off between simulation accuracy and simulation speed, reducing the transition region improves accuracy while expanding it improves speed.

Revolute Primitive: Actuation

Specify actuation options for the revolute joint primitive. Actuation modes include **Torque** and **Motion**. Selecting **Provided by Input** from the drop-down list for an actuation mode adds the corresponding physical signal port to the block. Use this port to specify the input signal. Input signals are resolved in the base frame.

Torque

Select an actuation torque setting. The default setting is None.

Actuation Torque Setting	Description
None	No actuation torque.
Provided by Input	Actuation torque from physical signal input. The signal provides the torque acting on the follower frame with respect to the base frame about the joint primitive axis. An equal and opposite torque acts on the base frame.
Automatically computed	Actuation torque from automatic calculation. Simscape Multibody computes and applies the actuation torque based on model dynamics.

Motion

Select an actuation motion setting. The default setting is Automatically Computed.

Actuation Motion Setting	Description
Provided by Input	Joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.
Automatically computed	Joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

Revolute Primitive: Sensing

Select the variables to sense in the revolute joint primitive. Selecting a variable exposes a physical signal port that outputs the measured quantity as a function of time. Each quantity is measured for the follower frame with respect to the base frame. It is resolved in the base frame. You can use the measurement signals for analysis or as input in a control system.

Position

Select this option to sense the relative rotation angle of the follower frame with respect to the base frame about the joint primitive axis.

Velocity

Select this option to sense the relative angular velocity of the follower frame with respect to the base frame about the joint primitive axis.

Acceleration

Select this option to sense the relative angular acceleration of the follower frame with respect to the base frame about the joint primitive axis.

Actuator Torque

Select this option to sense the actuation torque acting on the follower frame with respect to the base frame about the joint primitive axis.

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for specifying actuation inputs and sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Actuation Ports

The revolute joint primitives provide the following actuation ports:

- t_x , t_y — Actuation torques acting on the X and Y revolute joint primitives
- q_x , q_y — Desired rotations of the X and Y revolute joint primitives

Sensing Ports

The revolute joint primitives provide the following sensing ports:

- qx, qy — Angular positions of the X and Y revolute joint primitives
- wx, wy — Angular velocities of the X and Y revolute joint primitives
- bx, by — Angular accelerations of the X and Y revolute joint primitives
- tx, ty — Actuation torques acting on the X and Y revolute joint primitives

The following sensing ports provide the composite forces and torques acting on the joint:

- fc — Constraint force
- tc — Constraint torque
- ft — Total force
- tt — Total torque

See Also

Gimbal Joint | Revolute Joint

Topics

“Actuating and Sensing with Physical Signals”

“Motion Sensing”

“Rotational Measurements”

Introduced in R2012a

Variable Brick Solid

Solid brick with variable mass and size

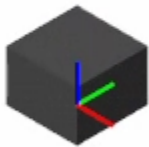
Library: Simscape / Multibody / Body Elements / Variable Mass



Description

The Variable Brick Solid block adds to the attached frame a solid brick with variable mass and size. The mass and side lengths (x , y , and z) of the brick can each be a constant or vary with time. A variable quantity can be specified directly as a physical signal or it can be calculated as a function of the remaining quantities. Only one quantity, either mass or one side length, can be calculated during simulation.

A reference frame encodes the position and orientation of the solid in a model. The frame origin is located at the midpoint of the x - and y -dimensions and at the lower end of the z -dimension. These relationships are preserved during simulation. The z -dimension increases asymmetrically relative to the lower z -plane, along the positive direction of the z -axis.



Variable Brick with z -Dimension Calculated from Mass

Visualization is dynamic. Solid dimensions update continuously as they occur, in the visualization pane of **Mechanics Explorer**. The initial dimensions of the solid depend on the parameters and physical signals that you specify. It is possible for a variable

dimension to begin with a zero value—for example, if it derives from a physical signal whose initial value is zero also.

Density can itself be constant or variable. This quantity is specified as a constant if at least one solid parameter is calculated during simulation. It is calculated as a variable if all solid parameters are explicitly specified, either as (constant) block parameters or as physical signals. As in the case of the Solid block, you can specify a negative density, for example, to model voids in compound bodies.

Ports

Frame

R — Reference frame

frame

Local reference frame of the solid. This frame is fixed with respect to the solid geometry. Its origin is on the *xy* plane, in the geometrical center of the *xy* cross section. Connect this port to a frame entity—port, line, or junction—to resolve the frame placement in a model. For more information, see “Working with Frames”.

Physical Signal Input

lx — x-dimension of the brick

physical signal

Input port for the *x*-dimension of the brick.

ly — y-dimension of the brick

physical signal

Input port for the *y*-dimension of the brick.

lz — z-dimension of the brick

physical signal

Input port for the *z*-dimension of the brick.

m — Brick mass

physical signal

Input port for the mass of the brick.

Physical Signal Output

l_x — x-dimension of the brick

physical signal

Output port for the x-dimension of the brick.

l_y — y-dimension of the brick

physical signal

Output port for the y-dimension of the brick.

l_z — z-dimension of the brick

physical signal

Output port for the z-dimension of the brick.

m — Brick mass

physical signal

Output port for the mass of the brick.

com — Center-of-mass coordinates of the brick

physical signal

Output port for the center of mass of the brick, reported as a three-element vector with Cartesian coordinates resolved in the reference frame of the solid.

I — Inertia matrix of the brick

physical signal

Output port for the inertia matrix of the brick, reported as a nine-element matrix and resolved in the reference frame of the block. The diagonal matrix elements are the moments of inertia. The off-diagonal elements are the products of inertia.

Parameters

Geometry and Inertia

X Length — Parameterization of the x dimension

Constant (default) | Calculate from Mass | Provided by Input

Parameterization of the x dimension of the solid—the length along the x-axis of the local reference frame. Select **Constant** to specify a fixed value as a block parameter. Select **Provided by Input** to specify a variable value as a physical signal input. Use the default setting (**Calculated from Mass**) to obtain this parameter from the specified solid density and remaining dimensions. Selecting **Provided by Input** exposes a new physical signal input port, labeled **lx**, through which to specify the variable value.

X Length: Value — Value of the x dimension

1 m (default) | scalar with units of length

Length of the solid along the x-axis of the local reference frame. The x dimension is constant when this block parameter is active.

Parameter Dependencies

This parameter is active when the **X Length** parameter is set to **Constant**.

Y Length — Parameterization of the y dimension

Constant (default) | Calculate from Mass | Provided by Input

Parameterization of the y dimension of the solid—the length along the y-axis of the local reference frame. Select **Constant** to specify a fixed value as a block parameter. Select **Provided by Input** to specify a variable value as a physical signal input. Use the default setting (**Calculated from Mass**) to obtain this parameter from the specified solid density and remaining dimensions. Selecting **Provided by Input** exposes a new physical signal input port, labeled **ly**, through which to specify the variable value.

Y Length: Value — Value of the y dimension

1 m (default) | scalar with units of length

Length of the solid along the y-axis of the local reference frame. The y dimension is constant when this block parameter is active.

Parameter Dependencies

This parameter is active when the **Y Length** parameter is set to **Constant**.

Z Length — Parameterization of the z dimension

Calculate from Mass (default) | Constant | Provided by Input

Parameterization of the z dimension of the solid—the length along the z-axis of the local reference frame. Select **Constant** to specify a fixed value as a block parameter. Select **Provided by Input** to specify a variable value as a physical signal input. Use the default setting (**Calculated from Mass**) to obtain this parameter from the specified solid density and remaining dimensions. Selecting **Provided by Input** exposes a new physical signal input port, labeled **Iz**, through which to specify the variable value.

Z Length: Value — Value of the z dimension

1 m (default) | scalar with units of length

Length of the solid along the z-axis of the local reference frame. The z dimension is constant when this block parameter is active.

Parameter Dependencies

This parameter is active when the **Z Length** parameter is set to **Constant**.

Mass — Mass parameterization

Provided by Input (default) | Calculate from Geometry

Parameterization of the mass of the solid. Select **Calculate from Geometry** to obtain this parameter from the specified solid density and dimensions. Use the default setting (**Provided by Input**) to specify this parameter directly as a time-variable physical signal. This option exposes a new physical signal input port, labeled **M**, through which to specify the time-variable solid mass.

Mass: Density — Mass per unit volume of material1000 kg/m³ (default) | scalar in units of mass per unit volume

Mass per unit volume of material. The mass density can take on a positive or negative value. Specify a negative mass density to model the effects of a void or cavity in a solid body. The default value, 1000 kg/m³, is characteristic of polymers such as ABS plastic.

Parameter Dependencies

This parameter is active when the **Mass** parameter is set to **Calculate from Geometry**.

Sensing**X Length — Sensing selection for the x dimension**

cleared (default) | checked

Sensing selection for the x dimension of the solid. Check to expose a new physical signal output port, labeled **lx**, through which to output the time-varying value of the x dimension.

Y Length — Sensing selection for the y dimension

cleared (default) | checked

Sensing selection for the y dimension of the solid. Check to expose a new physical signal output port, labeled **ly**, through which to output the time-varying value of the y dimension.

Z Length — Sensing selection for the z dimension

cleared (default) | checked

Sensing selection for the z dimension of the solid. Check to expose a new physical signal output port, labeled **lz**, through which to output the time-varying value of the z dimension.

Mass — Sensing selection for the total mass

cleared (default) | checked

Sensing selection for the total mass of the solid. Check to expose a new physical signal output port, labeled **m**, through which to output the time-varying value of the solid mass.

Center of Mass — Sensing selection for the center-of-mass coordinates

cleared (default) | checked

Sensing selection for the coordinates of the center of mass of the solid. Check to expose a new physical signal output port, labeled **com**, through which to output the time-varying coordinates. The output is a three-element vector with Cartesian coordinates resolved in the reference frame of the solid.

Inertia Matrix — Sensing option for the inertia matrix

cleared (default) | checked

Sensing selection for the inertia matrix of the solid. Check to expose a new physical signal output port, labeled **I**, through which to output the time-varying inertia matrix. The

output is a nine-element matrix with the moments of inertia in the diagonal positions and the products of inertia in the off-diagonal positions. The moments and products of inertia are resolved in the inertia frame of resolution—a frame with axes parallel to those of the reference frame but origin at the center of mass.

Graphic

Type — Solid visualization setting

From Geometry (default) | Marker | None

Visualization setting for this solid. Use the default setting, From Geometry, to show the solid geometry. Select Marker to show a graphic marker such as a sphere or frame. Select None to disable visualization for this solid.

Marker: Shape — Shape of the graphic marker

Sphere (default) | Cube | Frame

Geometrical shape of the graphic marker. Mechanics Explorer shows the marker using the selected shape.

Marker: Size — Pixel size of the graphic marker

10 (default) | scalar with units of pixels

Size of the marker in units of pixels. The size does not change with zoom level.

Visual Properties — Parameterizations for color and opacity

Simple (default) | Advanced

Parameterization for specifying visual properties. Select Simple to specify color and opacity. Select Advanced to add specular highlights, ambient shadows, and self-illumination effects.

Simple: Color — True color as [R,G,B] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three-element vector with values constrained to 0-1

RGB color vector with red (R), green (G), and blue (B) color amounts specified on a 0-1 scale. A color picker provides an alternative interactive means of specifying a color. If you change the **Visual Properties** setting to Advanced, the color specified in this parameter becomes the **Diffuse Color** vector.

Simple: Opacity — Surface opacity as scalar number on 0-1 scale

1.0 (default) | scalar with value constrained to 0-1

Graphic opacity specified on a scale of 0-1. An opacity of 0 corresponds to a completely transparent graphic and an opacity of 1 to a completely opaque graphic.

Advanced: Diffuse Color — True color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three- or four-element vector with values constrained to 0-1

True color under direct white light specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. An optional fourth element specifies the color opacity also on a scale of 0-1. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Specular Color — Highlight color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of specular highlights specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Ambient Color — Shadow color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of shadow areas in diffuse ambient light, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Emissive Color — Self-illumination color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Surface color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Shininess — Highlight sharpness as scalar number on 0-128 scale

75 (default) | scalar with value constrained to 0-128

Sharpness of specular light reflections, specified as a scalar number on a 0-128 scale. Increase the shininess value for smaller but sharper highlights. Decrease the value for larger but smoother highlights.

Extended Capabilities

C/C++ Code Generation

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

See Also

[Rigid Transform](#) | [Solid](#) | [Variable Cylindrical Solid](#) | [Variable Spherical Solid](#)

Topics

[“Modeling Bodies”](#)

[“Representing Solid Geometry”](#)

[“Manipulate the Color of a Solid”](#)

Introduced in R2017b

Variable Cylindrical Solid

Solid cylinder with variable mass and size

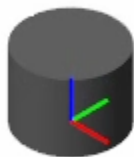
Library: Simscape / Multibody / Body Elements / Variable Mass



Description

The Variable Cylindrical Solid block adds to the attached frame a solid cylinder with variable mass and side. The mass, radius, and length of the cylinder can each be constant or vary with time. A variable quantity can be specified directly as a physical signal or it can be calculated as a function of the remaining quantities. Either the mass or the cylinder dimensions can be calculated during simulation, but not both simultaneously.

A reference frame encodes the position and orientation of the solid in a model. The frame is fixed to the solid with the frame origin at the center of the lower surface (as observed with a Z up view convention). This placement is preserved throughout simulation. Length increases asymmetrically relative to the lower surface, along the positive direction of the z-axis.



Variable Cylinder with Length Calculated from Mass

Visualization is dynamic. Solid dimensions update continuously as they occur, in the visualization pane of **Mechanics Explorer**. The initial dimensions of the solid depend on the parameters and physical signals that you specify. It is possible for a variable

dimension to begin with a zero value—for example, if it derives from a physical signal whose initial value is zero also.

Density can itself be constant or variable. This quantity is specified as a constant if at least one solid parameter is calculated during simulation. It is calculated as a variable if all solid parameters are explicitly specified, either as (constant) block parameters or as physical signals. As in the case of the Solid block, you can specify a negative density, for example, to model voids in compound bodies.

Ports

Frame

R — Reference frame

frame

Local reference frame of the solid. This frame is fixed with respect to the solid geometry. The frame origin is on the xy -plane at the center of the xy cross-section. The z -axis aligns with the longitudinal axis of the cylinder. Connect this port to a frame entity—port, line, or junction—to resolve the frame placement in a model. For more information, see “Working with Frames”.

Physical Signal Input

len — Cylinder length

physical signal

Input port for the length of the cylinder.

r — Radius of the cylinder

physical signal

Input port for the radius of the cylinder.

m — Mass of the cylinder

physical signal

Input port for the mass of the cylinder.

Physical Signal Output

len — Length of the cylinder

physical signal

Output port for the length of the cylinder.

r — Radius of the cylinder

physical signal

Output port for the width of the cylinder.

m — Mass of the cylinder

physical signal

Output port for the mass of the cylinder.

com — Center-of-mass coordinates of the cylinder

physical signal

Output port for the center of mass of the cylinder, reported as a three-element vector with Cartesian coordinates resolved in the reference frame of the solid.

I — Inertia matrix of the cylinder

physical signal

Output port for the inertia matrix of the cylinder, reported as a nine-element matrix, resolved in the reference frame of the block. The diagonal matrix elements are the moments of inertia. The off-diagonal elements are the products of inertia.

Parameters

Geometry and Inertia

Radius — Radius parameterization

Calculate from Mass (default) | Constant | Provided by Input

Parameterization of the radius of the solid. Select **Constant** to specify a fixed value as a block parameter. Select **Provided by Input** to specify a variable value as a physical signal input. Use the default setting (**Calculated from Mass**) to obtain this parameter from the specified solid density and remaining dimensions. Selecting **Provided by**

Input exposes a new physical signal input port, labeled **r**, through which to specify the variable value.

Radius: Value — Value of the radius

1 m (default) | scalar with units of length

Radius of the solid. The longitudinal axis of the solid is aligned with the z-axis of the local reference frame. The z dimension is constant when this block parameter is active.

Parameter Dependencies

This parameter is active when the **Radius** parameter is set to Constant.

Length — Length parameterization

Calculate from Mass (default) | Constant | Provided by Input

Parameterization of the length of the solid. Select **Constant** to specify a fixed value as a block parameter. Select **Provided by Input** to specify a variable value as a physical signal input. Use the default setting (**Calculated from Mass**) to obtain this parameter from the specified solid density and remaining dimensions. Selecting **Provided by Input** exposes a new physical signal input port, labeled **len**, through which to specify the variable value.

Length: Value — Value of the length

1 m (default) | scalar with units of length

Length of the solid. The longitudinal axis of the cylinder is aligned with the z-axis of the local reference frame. The z dimension is constant when this block parameter is active.

Parameter Dependencies

This parameter is active when the **Length** parameter is set to Constant.

Mass — Mass parameterization

Provided by Input (default) | Calculate from Geometry

Parameterization of the mass of the solid. Select **Calculate from Geometry** to obtain this parameter from the specified solid density and dimensions. Use the default setting (**Provided by Input**) to specify this parameter directly as a time-variable physical signal. This option exposes a new physical signal input port, labeled **M**, through which to specify the time-variable solid mass.

Mass: Density — Mass per unit volume of material

1000 kg/m³ (default) | scalar in units of mass per unit volume

Mass per unit volume of material. The mass density can take on a positive or negative value. Specify a negative mass density to model the effects of a void or cavity in a solid body. The default value, 1000 kg/m^3 , is characteristic of polymers such as ABS plastic.

Parameter Dependencies

This parameter is active when the **Mass** parameter is set to Calculate from Geometry.

Sensing**Radius — Sensing selection for the radius**

cleared (default) | checked

Sensing selection for the radius of the solid. Check to expose a new physical signal output port, labeled **r**, through which to output the time-varying value of the radius.

Length — Sensing selection for the length

cleared (default) | checked

Sensing selection for the *z* dimension of the solid. Check to expose a new physical signal output port, labeled **len**, through which to output the time-varying value of the length.

Mass — Sensing selection for the total mass

cleared (default) | checked

Sensing selection for the total mass of the solid. Check to expose a new physical signal output port, labeled **m**, through which to output the time-varying value of the solid mass.

Center of Mass — Sensing selection for the center-of-mass coordinates

cleared (default) | checked

Sensing selection for the coordinates of the center of mass of the solid. Check to expose a new physical signal output port, labeled **com**, through which to output the time-varying coordinates. The output is a three-element vector with Cartesian coordinates resolved in the reference frame of the solid.

Inertia Matrix — Sensing option for the inertia matrix

cleared (default) | checked

Sensing selection for the inertia matrix of the solid. Check to expose a new physical signal output port, labeled **I**, through which to output the time-varying inertia matrix. The output is a nine-element matrix with the moments of inertia in the diagonal positions and

the products of inertia in the off-diagonal positions. The moments and products of inertia are resolve in the inertia frame of resolution—a frame with axes parallel to those of the reference frame but origin at the center of mass.

Graphic

Type — Solid visualization setting

From Geometry (default) | Marker | None

Visualization setting for this solid. Use the default setting, From Geometry, to show the solid geometry. Select Marker to show a graphic marker such as a sphere or frame. Select None to disable visualization for this solid.

Marker: Shape — Shape of the graphic marker

Sphere (default) | Cube | Frame

Geometrical shape of the graphic marker. Mechanics Explorer shows the marker using the selected shape.

Marker: Size — Pixel size of the graphic marker

10 (default) | scalar with units of pixels

Size of the marker in units of pixels. The size does not change with zoom level.

Visual Properties — Parameterizations for color and opacity

Simple (default) | Advanced

Parameterization for specifying visual properties. Select Simple to specify color and opacity. Select Advanced to add specular highlights, ambient shadows, and self-illumination effects.

Simple: Color — True color as [R,G,B] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three-element vector with values constrained to 0-1

RGB color vector with red (R), green (G), and blue (B) color amounts specified on a 0-1 scale. A color picker provides an alternative interactive means of specifying a color. If you change the **Visual Properties** setting to Advanced, the color specified in this parameter becomes the **Diffuse Color** vector.

Simple: Opacity — Surface opacity as scalar number on 0-1 scale

1.0 (default) | scalar with value constrained to 0-1

Graphic opacity specified on a scale of 0-1. An opacity of 0 corresponds to a completely transparent graphic and an opacity of 1 to a completely opaque graphic.

Advanced: Diffuse Color — True color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three- or four-element vector with values constrained to 0-1

True color under direct white light specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. An optional fourth element specifies the color opacity also on a scale of 0-1. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Specular Color — Highlight color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of specular highlights specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Ambient Color — Shadow color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of shadow areas in diffuse ambient light, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Emissive Color — Self-illumination color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Surface color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Shininess — Highlight sharpness as scalar number on 0-128 scale

75 (default) | scalar with value constrained to 0-128

Sharpness of specular light reflections, specified as a scalar number on a 0-128 scale. Increase the shininess value for smaller but sharper highlights. Decrease the value for larger but smoother highlights.

Extended Capabilities

C/C++ Code Generation

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

See Also

[Rigid Transform](#) | [Solid](#) | [Variable Brick Solid](#) | [Variable Spherical Solid](#)

Topics

[“Specifying Custom Inertias”](#)

[“Representing Solid Inertia”](#)

Introduced in R2017b

Variable Spherical Solid

Solid sphere with variable mass and size

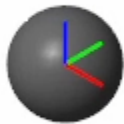
Library: Simscape / Multibody / Body Elements / Variable Mass



Description

The Variable Spherical Solid block adds to the attached frame a solid sphere with variable mass and size. The mass and radius of the sphere can each be constant or vary with time. A variable quantity can be specified directly as a physical signal or it can be calculated as a function of the remaining quantity. Only one quantity, mass or radius, can be calculated during simulation.

A reference frame encodes the position and orientation of the solid relative to other components in a model. The frame is defined relative to the solid geometry so that its origin is located at the center of the sphere. This relationship is preserved during simulation. The radius increases symmetrically in all directions with respect to the frame origin.



Variable Sphere with Radius Calculated from Mass

Visualization is dynamic. Solid dimensions update continuously as they occur, in the visualization pane of **Mechanics Explorer**. The initial dimensions of the solid depend on the parameters and physical signals that you specify. It is possible for a variable

dimension to begin with a zero value—for example, if it derives from a physical signal whose initial value is zero also.

Density can itself be constant or variable. This quantity is specified as a constant if either mass or radius is calculated during simulation. It is calculated as a variable if both mass and radius are explicitly specified, either as (constant) block parameters or as physical signals. As in the case of the Solid block, you can specify a negative density, for example, to model voids in compound bodies.

Ports

Frame

R — Reference frame

frame

Local reference frame of the solid. This frame is fixed with respect to the solid geometry. The frame origin is located at the center of geometry. Connect this port to a frame entity—port, line, or junction—to resolve the frame placement in a model. For more information, see “Working with Frames”.

Physical Signal Input

r — Radius of the sphere

physical signal

Input port for the radius of the sphere.

m — Mass of the sphere

physical signal

Input port for the mass of the sphere.

Physical Signal Output

r — Radius of the sphere

physical signal

Output port for the radius of the sphere.

m — Mass of the sphere

physical signal

Output port for the mass of the sphere.

com — Center-of-mass coordinates of the sphere

physical signal

Output port for the center of mass of the sphere, reported as a three-element vector with Cartesian coordinates resolved in the reference frame of the solid.

I — Inertia matrix of the sphere

physical signal

Output port for the inertia matrix of the sphere, reported as a nine-element matrix resolved in the inertia frame of resolution of the solid—a virtual copy of the reference frame whose origin has been shifted to the center of mass. The axes of the inertia frame of resolution are parallel to the axes of the reference frame. The diagonal elements of the matrix are the moments of inertia and the off-diagonal elements are the products of inertia.

Parameters

Geometry and Inertia

Radius — Radius parameterization

Calculate from Mass (default) | Constant | Provided by Input

Parameterization of the radius of the solid. Select **Constant** to specify a fixed value as a block parameter. Select **Provided by Input** to specify a variable value as a physical signal input. Use the default setting (**Calculated from Mass**) to obtain this parameter from the specified solid density and remaining dimensions. Selecting **Provided by Input** exposes a new physical signal input port, labeled **r**, through which to specify the variable value.

Radius: Value — Value of the radius

1 m (default) | scalar with units of length

Radius of the solid. The longitudinal axis of the solid is aligned with the z-axis of the local reference frame. The z dimension is constant when this block parameter is active.

Parameter Dependencies

This parameter is active when the **Radius** parameter is set to Constant.

Mass — Mass parameterization

Provided by Input (default) | Calculate from Geometry

Parameterization of the mass of the solid. Select **Calculate from Geometry** to obtain this parameter from the specified solid density and dimensions. Use the default setting (**Provided by Input**) to specify this parameter directly as a time-variable physical signal. This option exposes a new physical signal input port, labeled **M**, through which to specify the time-variable solid mass.

Mass: Density — Mass per unit volume of material

1000 kg/m³ (default) | scalar in units of mass per unit volume

Mass per unit volume of material. The mass density can take on a positive or negative value. Specify a negative mass density to model the effects of a void or cavity in a solid body. The default value, 1000 kg/m³, is characteristic of polymers such as ABS plastic.

Parameter Dependencies

This parameter is active when the **Mass** parameter is set to **Calculate from Geometry**.

Sensing**Radius — Sensing selection for the radius**

cleared (default) | checked

Sensing selection for the radius of the solid. Check to expose a new physical signal output port, labeled **r**, through which to output the time-varying value of the radius.

Mass — Sensing selection for the total mass

cleared (default) | checked

Sensing selection for the total mass of the solid. Check to expose a new physical signal output port, labeled **m**, through which to output the time-varying value of the solid mass.

Center of Mass — Sensing selection for the center-of-mass coordinates

cleared (default) | checked

Sensing selection for the coordinates of the center of mass of the solid. Check to expose a new physical signal output port, labeled **com**, through which to output the time-varying

coordinates. The output is a three-element vector with Cartesian coordinates resolved in the reference frame of the solid.

Inertia Matrix — Sensing option for the inertia matrix

cleared (default) | checked

Sensing selection for the inertia matrix of the solid. Check to expose a new physical signal output port, labeled **I**, through which to output the time-varying inertia matrix. The output is a nine-element matrix with the moments of inertia in the diagonal positions and the products of inertia in the off-diagonal positions. The moments and products of inertia are resolve in the inertia frame of resolution—a frame with axes parallel to those of the reference frame but origin at the center of mass.

Graphic**Type — Solid visualization setting**

From Geometry (default) | Marker | None

Visualization setting for this solid. Use the default setting, *From Geometry*, to show the solid geometry. Select *Marker* to show a graphic marker such as a sphere or frame. Select *None* to disable visualization for this solid.

Marker: Shape — Shape of the graphic marker

Sphere (default) | Cube | Frame

Geometrical shape of the graphic marker. Mechanics Explorer shows the marker using the selected shape.

Marker: Size — Pixel size of the graphic marker

10 (default) | scalar with units of pixels

Size of the marker in units of pixels. The size does not change with zoom level.

Visual Properties — Parameterizations for color and opacity

Simple (default) | Advanced

Parameterization for specifying visual properties. Select *Simple* to specify color and opacity. Select *Advanced* to add specular highlights, ambient shadows, and self-illumination effects.

Simple: Color — True color as [R,G,B] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three-element vector with values constrained to 0-1

RGB color vector with red (R), green (G), and blue (B) color amounts specified on a 0-1 scale. A color picker provides an alternative interactive means of specifying a color. If you change the **Visual Properties** setting to Advanced, the color specified in this parameter becomes the **Diffuse Color** vector.

Simple: Opacity — Surface opacity as scalar number on 0-1 scale

1.0 (default) | scalar with value constrained to 0-1

Graphic opacity specified on a scale of 0-1. An opacity of 0 corresponds to a completely transparent graphic and an opacity of 1 to a completely opaque graphic.

Advanced: Diffuse Color — True color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5] (default) | three- or four-element vector with values constrained to 0-1

True color under direct white light specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. An optional fourth element specifies the color opacity also on a scale of 0-1. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Specular Color — Highlight color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of specular highlights specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Ambient Color — Shadow color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Color of shadow areas in diffuse ambient light, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Emissive Color — Self-illumination color as [R,G,B,A] vector on 0-1 scale

[0.5 0.5 0.5 1.0] (default) | three- or four-element vector with values constrained to 0-1

Surface color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector on a 0-1 scale. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Advanced: Shininess — Highlight sharpness as scalar number on 0-128 scale
75 (default) | scalar with value constrained to 0-128

Sharpness of specular light reflections, specified as a scalar number on a 0-128 scale. Increase the shininess value for smaller but sharper highlights. Decrease the value for larger but smoother highlights.

Extended Capabilities

C/C++ Code Generation

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

See Also

Rigid Transform | Solid | Variable Brick Solid | Variable Cylindrical Solid

Topics

“Specifying Custom Inertias”

“Representing Solid Inertia”

Introduced in R2017b

Weld Joint

Joint with zero primitives

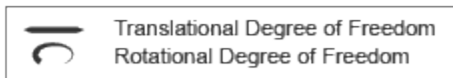
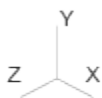


Library

Joints

Description

This block represents a joint with zero degrees of freedom. It contains no joint primitives. Base and follower frames, each connected to a separate rigid body, are coincident for all time. The block dialog box provides sensing options for constraint and total forces and torques.



Joint Degrees of Freedom

Parameters

Composite Force/Torque Sensing

Select the composite forces and torques to sense. Their measurements encompass all joint primitives and are specific to none. They come in two kinds: constraint and total.

Constraint measurements give the resistance against motion on the locked axes of the joint. In prismatic joints, for instance, which forbid translation on the xy plane, that resistance balances all perturbations in the x and y directions. Total measurements give the sum over all forces and torques due to actuation inputs, internal springs and dampers, joint position limits, and the kinematic constraints that limit the degrees of freedom of the joint.

Direction

Vector to sense from the action-reaction pair between the base and follower frames. The pair arises from Newton's third law of motion which, for a joint block, requires that a force or torque on the follower frame accompany an equal and opposite force or torque on the base frame. Indicate whether to sense that exerted by the base frame on the follower frame or that exerted by the follower frame on the base frame.

Resolution Frame

Frame on which to resolve the vector components of a measurement. Frames with different orientations give different vector components for the same measurement. Indicate whether to get those components from the axes of the base frame or from the axes of the follower frame. The choice matters only in joints with rotational degrees of freedom.

Constraint Force

Dynamic variable to measure. Constraint forces counter translation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint force vector through port **fc**.

Constraint Torque

Dynamic variable to measure. Constraint torques counter rotation on the locked axes of the joint while allowing it on the free axes of its primitives. Select to output the constraint torque vector through port **tc**.

Total Force

Dynamic variable to measure. The total force is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total force vector through port **ft**.

Total Torque

Dynamic variable to measure. The total torque is a sum across all joint primitives over all sources—actuation inputs, internal springs and dampers, joint position limits, and kinematic constraints. Select to output the total torque vector through port **tt**.

Ports

This block has two frame ports. It also has optional physical signal ports for sensing dynamical variables such as forces, torques, and motion. You expose an optional port by selecting the sensing check box corresponding to that port.

Frame Ports

- B — Base frame
- F — Follower frame

Sensing Ports

The following sensing ports provide the composite forces and torques acting on the joint:

- fc — Constraint force
- tc — Constraint torque
- ft — Total force
- tt — Total torque

Extended Capabilities

C/C++ Code Generation

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

See Also

Rigid Transform

Introduced in R2012a

World Frame

Inertial reference frame

Library: Simscape / Multibody / Frames and Transforms



Description

This block represents the global reference frame in a model. This frame is inertial and at absolute rest. Rigidly connecting a frame to the World frame makes that frame inertial. Frame axes are orthogonal and arranged according to the right-hand rule.

In a frame network, the World frame is the ultimate reference frame. Directly or indirectly, all other frames are defined with respect to the World frame. If multiple World Frame blocks connect to the same frame network, those blocks identify the same frame. If no World Frame block connects to a frame network, a copy of an existing frame, frozen in its initial position and orientation, serves as the World frame.

Ports

Frame

W — World frame

frame

World frame represented by the block. Connect to another frame to fix the position and orientation of that frame to the world frame.

Extended Capabilities

C/C++ Code Generation

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

See Also

Reference Frame | Rigid Transform

Topics

“Working with Frames”

“Creating Connection Frames”

Introduced in R2012a

Worm and Gear Constraint

Kinematic constraint between worm and gear bodies with perpendicular non-intersecting rotation axes

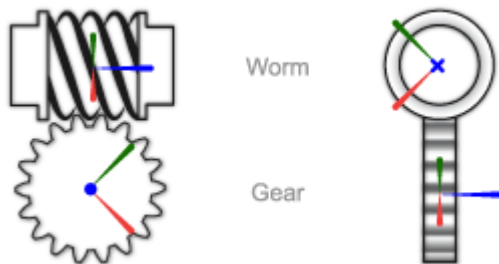
Library: Simscape / Multibody / Gears and Couplings / Gears



Description

The Worm and Gear Constraint block represents a kinematic constraint between worm and gear bodies held at a right angle. The base frame port identifies the connection frame on the worm and the follower frame port identifies the connection frame on the gear. The rotation axes coincide with the connection frame z-axes. The worm and gear rotate at a fixed velocity ratio determined by the gear pitch radii or tooth-thread ratio.

The worm thread direction can follow either right-hand or left-hand conventions. The convention used determines the relative directions of the worm and gear rotational velocities. A right-hand convention causes the worm and gear to rotate in the same direction about the respective z-axes. A left-hand convention causes the worm and gear to rotate in opposite directions instead.

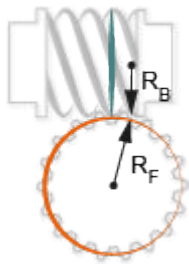


The block represents only the kinematic constraint characteristic to a worm-and-gear system. Gear inertia and geometry are solid properties that you must specify using Solid blocks. The gear constraint model is ideal. Backlash and gear losses due to Coulomb and

viscous friction between teeth are ignored. You can, however, model viscous friction at joints by specifying damping coefficients in the joint blocks.

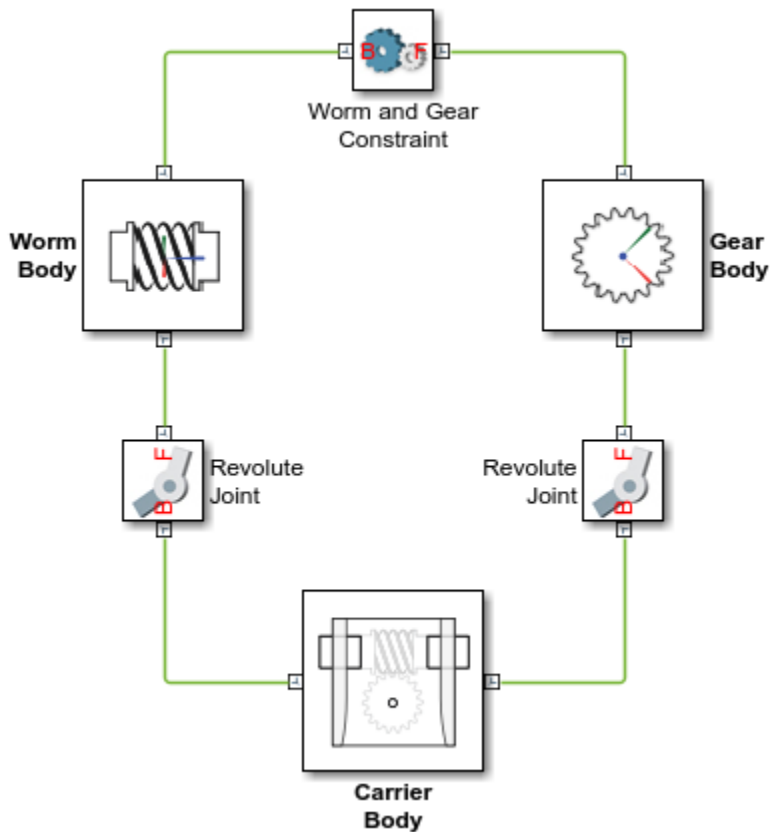
Gear Geometry

The rack-and-pinion constraint is parameterized in terms of the dimensions of the worm and gear pitch circles. The pitch circles are imaginary circles concentric with the worm and gear bodies and tangent to the thread contact point. The pitch radii, labeled R_B and R_F in the figure, are the radii that the worm and gear would have if they were reduced to friction cylinders in mutual contact.



Gear Assembly

Gear constraints occur in closed kinematic loops. The figure shows the closed-loop topology of a simple worm-and-gear model. Joint blocks connect the worm and gear bodies to a common fixture or carrier, defining the maximum degrees of freedom between them. A Worm and Gear Constraint block connects the worm and gear bodies, eliminating one degree of freedom and effectively coupling the worm and gear motions.

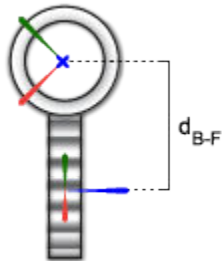


Assembly Requirements

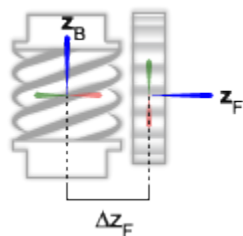
The block imposes special restrictions on the relative positions and orientations of the gear connection frames. The restrictions ensure that the gears assemble only at distances and angles suitable for meshing. The block enforces the restrictions during model assembly, when it first attempts to place the gears in mesh, but relies on the remainder of the model to keep the gears in mesh during simulation.

Position Restrictions

- The distance between the base and follower frame z -axes, denoted d_{B-F} in the figure, must be equal to the distance between the gear centers.

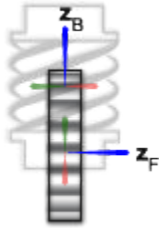


- The translational offset between the base and follower frame origins along the follower frame z -axis, denoted ΔZ_F in the figure, must be zero.

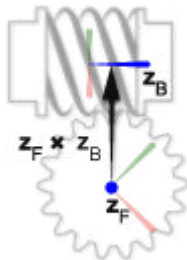


Orientation Restrictions

- The z -axes of the base and follower frames must be perpendicular to each other. The z -axes are shown in blue in the figure.



- The cross product of the follower frame z -axis with the base frame z -axis must be a vector aimed from the follower frame origin to the base frame z -axis. The z -axes and their cross-product vector are shown in the figure. The cross product is defined as $\hat{z}_F \times \hat{z}_B$.



Ports

Frame

B — Base frame

frame

Connection frame on the worm body.

F — Follower frame

frame

Connection frame on the gear body.

Parameters

Worm Direction — Winding direction of the worm thread

Right-Hand (default) | Left-Hand

Winding direction of the worm thread relative to the base frame z-axis. As viewed from the base frame origin, a right-hand thread is one that wraps around the base frame z-axis in a counterclockwise direction. A left-hand thread is one that wraps in a clockwise direction. This parameter determines the relative directions of motion of the worm and gear bodies.

Worm Lead Angle — Angle between the worm thread and rotation plane

10 deg (default) | positive scalar between 0 and 180 in units of angle

Angle between the tangent to the worm thread and the plane perpendicular to the base frame z-axis. The lead angle impacts the gear rotation corresponding to a full worm revolution.

Specification Method — Gear geometry parameterization

Center Distance and Ratio (default) | Pitch Circle Radii

Parameterization for specifying the worm and gear geometries. You can specify the gear dimensions in terms of the distance between the gear centers or the individual gear radii.

Center Distance — Distance between the worm and gear centers

20 cm (default) | positive scalar in units of length

Distance between the worm and gear centers. This distance must equal that enforced by rigid transforms, joints, and any other constraints located between the gear bodies and the common carrier body.

Dependencies

This parameter is enabled when the **Specification Method** parameter is set to Center Distance and Ratio.

Ratio (Ng/Nw) — Ratio of gear teeth to worm threads

1 (default) | positive unitless scalar

Ratio of gear teeth to worm threads, or *starts*. This ratio impacts the torque transmitted between the worm and gear.

Dependencies

This parameter is enabled when the **Specification Method** parameter is set to Center Distance and Ratio.

Worm Radius — Radius of the worm pitch circle

10 cm (default) | positive scalar in units of length

Radius of the worm pitch circle. This is the distance between the worm rotation axis and the tooth-thread contact point. This parameter impacts the torque transmitted between the worm and gear.

Dependencies

This parameter is enabled when the **Specification Method** parameter is set to Pitch Circle Radius.

Gear Radius — Radius of the gear pitch circle

10 cm (default) | positive scalar in units of length

Radius of the gear pitch circle. This is the distance between the gear rotation axis and the tooth-thread contact point. This parameter impacts the torque transmitted between the worm and gear.

Dependencies

This parameter is enabled when the **Specification Method** parameter is set to Pitch Circle Radius.

Extended Capabilities

C/C++ Code Generation

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

See Also

Bevel Gear Constraint | Common Gear Constraint | Rack and Pinion Constraint

Topics

“Worm and Gear”

Introduced in R2016b

Configuration Parameters

- “Simscape Multibody Pane: General” on page 2-2
- “Simscape Multibody Pane: Diagnostics” on page 2-3
- “Simscape Multibody Pane: Explorer” on page 2-10

Simscape Multibody Pane: General

The SimMechanics Second Generation (2G) configuration parameters are arranged into the following sections :

Diagnostics

This section contains configurable diagnostic messages. The messages can be configured to be ignored or to be reported as warnings or errors. Errors will prevent simulation while warnings will allow simulation to proceed. The Mechanics Explorer (if selected) will be opened and visualization shown in all cases.

Explorer

This section contains parameters that configure the Mechanics Explorer.

Simscape Multibody Pane Overview

Configure the mechanical settings for an entire Simscape Multibody model.

Configuration

- This pane appears only if your model contains at least one block from the Simscape product or a product based on the Simscape product, such as the Simscape Multibody product.
- The settings in this pane are saved only if your model contains at least one Simscape Multibody block.

Simscape Multibody Pane: Diagnostics

Evaluation	
Invalid visual properties:	warning ▼
Repeated vertices in a cross-section:	warning ▼
Topology	
Unconnected frame port:	warning ▼
Redundant block:	warning ▼
Conflicting reference frames:	warning ▼
Rigidly constrained block:	error ▼
Assembly	
Unsatisfied high priority state targets:	warning ▼
Overspecified targets in kinematic loops:	error ▼

In this section...

“Invalid visual properties” on page 2-4

“Repeated vertices in a cross-section” on page 2-4

“Unconnected frame port” on page 2-5

“Unconnected Geometry port” on page 2-5

“Redundant block” on page 2-6

“Conflicting reference frames” on page 2-7

“Rigidly constrained block” on page 2-7

“Unsatisfied high priority state targets” on page 2-8

“Overspecified targets in kinematic loops” on page 2-9

Invalid visual properties

Select the diagnostic action to take if the application detects an improperly specified color vector.

Settings

Default: warning

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

error

When the application detects this situation, it terminates the simulation and displays an error message.

Command-Line Information

Parameter: SimMechanicsInvalidVisualProperty

Type: string

Value: none | warning | error

Default: warning

Repeated vertices in a cross-section

Select the diagnostic action to take if the application detects repeated vertices in a cross-section.

Settings

Default: warning

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

error

When the application detects this situation, it terminates the simulation and displays an error message.

Command-Line Information

Parameter: SimMechanicsCrossSectionNullEdge

Type: string

Value: none | warning | error

Default: warning

Unconnected frame port

Select the diagnostic action to take if the application detects an unconnected frame port.

Settings

Default: Warning

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

error

When the application detects this situation, it terminates the simulation and displays an error message.

Command-Line Information

Parameter: SimMechanicsUnconnectedFramePorts

Type: string

Value: none | warning | error

Default: warning

Unconnected Geometry port

Select the diagnostic action to take if the application detects an unconnected geometry port.

Settings

Default: Warning

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

error

When the application detects this situation, it terminates the simulation and displays an error message.

Command-Line Information

Parameter: SimMechanicsUnconnectedGeometryPorts

Type: string

Value: none | warning | error

Default: warning

Redundant block

Select the diagnostic action to take if the application detects a redundant block in the model.

Settings

Default: warning

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

error

When the application detects this situation, it terminates the simulation and displays an error message.

Command-Line Information

Parameter: SimMechanicsRedundantBlock

Type: string
Value: none | warning | error
Default: warning

Conflicting reference frames

Select the diagnostic action to take if the application detects conflicting reference frames in the model.

Settings

Default: warning

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

error

When the application detects this situation, it terminates the simulation and displays an error message.

Command-Line Information

Parameter: SimMechanicsConflictingReferenceFrames

Type: string

Value: none | warning | error

Default: warning

Rigidly constrained block

Select the diagnostic action to take if the application detects a rigidly constrained block in the model.

Settings

Default: warning

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

error

When the application detects this situation, it terminates the simulation and displays an error message.

Command-Line Information

Parameter: SimMechanicsRigidlyBoundBlock

Type: string

Value: none | warning | error

Default: error

Unsatisfied high priority state targets

Select the diagnostic action to take if the application detects targets with unsatisfied desired states in the model.

Settings

Default: warning

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

error

When the application detects this situation, it terminates the simulation and displays an error message.

Command-Line Information

Parameter: SimMechanicsUnsatisfiedHighPriorityTargets

Type: string

Value: none | warning | error

Default: warning

Overspecified targets in kinematic loops

Select the diagnostic action to take if the application detects overspecified targets contained in kinematic loops in the model.

Settings

Default: warning

none

The application does not check for this situation.

warning

When the application detects this situation, it displays a warning.

error

When the application detects this situation, it terminates the simulation and displays an error message.

Command-Line Information

Parameter: SimMechanicsJointTargetOverSpecification

Type: string

Value: none | warning | error

Default: error

Simscape Multibody Pane: Explorer

Open Mechanics Explorer on model update or simulation

Open Mechanics Explorer on model update or simulation

Start Mechanics Explorer when model is updated or simulated.

Settings

Default: on

On

Model Explorer starts when model is updated or simulated.

Off

Model Explorer does not start when model is updated or simulated.

Tip

If you clear this check box, you can start Model Explorer by selecting **Desktop > Mechanics Explorers** from the MATLAB Command Window.

Command-Line Information

Parameter: SimMechanicsOpenEditorOnUpdate

Type: string

Value: 'on' | 'off'

Default: 'on'

Multibody Visualization

Mechanics Explorer

Visualize and explore multibody models

Description

Mechanics Explorer is a Simscape Multibody tool based on the Silicon Graphics OpenGL® API that lets you visualize and explore your multibody models. The tool comprises a visualization pane to view the model, a tree view pane to explore the model hierarchy, and a properties pane to examine the individual component parameters.

The visualization pane is interactive and allows you to manipulate the model viewpoint. You can rotate, roll, pan, and zoom the model to more clearly view its components. You can also select from a list of preset viewpoints that includes isometric, front, side, and top views. To access the view manipulation tools, use the Mechanics Explorer **View** menu.

For more information on view manipulation, see “Manipulate the Visualization Viewpoint”.

A **Camera Manager** tool allows you to create, edit, and delete dynamic cameras with moving viewpoints. You can interactively set the camera views at discrete playback times (Keyframes mode) or constrain the camera to coordinate frames in your model (Tracking mode). To open Camera Manager, in the Mechanics Explorer menu bar, select **Tools > Camera Manager**.

For more information on dynamic cameras, see “Create a Dynamic Camera”.

A **Video Creator** tool allows you to configure and create videos from your multibody animations. You can set the video frame rate, frame size, playback speed ratio, and file format. Video Creator captures the model animation as shown in the active visualization tile the moment you click the **Create** button. To open Video Creator, in the Mechanics Explorer menu bar, select **Tools > Video Creator**.

For more information on video creation, see “Create a Model Animation Video”.

Open the Mechanics Explorer App

Update or simulate the model you want to visualize. By default, Mechanics Explorer opens automatically with the corresponding model visualization. The visualization shows

the initial model configuration on model update and a dynamic animation during model simulation. If Mechanics Explorer fails to open, check that automatic model visualization is enabled:

- 1 In the Simulink menu bar, select **Simulation > Model Configuration Parameters**.
- 2 In the Model Configuration Parameters window, select **Simscape Multibody > Explorer**.
- 3 Select the **Open Mechanics Explorer on model update or simulation** check box.

Examples

- “Manipulate the Visualization Viewpoint”
- “Create a Dynamic Camera”
- “Selectively Show and Hide Model Components”
- “Visualize Simscape Multibody Frames”
- “Go to a Block from Mechanics Explorer”
- “Create a Model Animation Video”

See Also

Topics

“Manipulate the Visualization Viewpoint”
“Create a Dynamic Camera”
“Selectively Show and Hide Model Components”
“Visualize Simscape Multibody Frames”
“Go to a Block from Mechanics Explorer”
“Create a Model Animation Video”

Introduced in R2012a

Camera Manager

Create, edit, and delete dynamic cameras



Description

Camera Manager is a **Mechanics Explorer** tool that lets you create, edit, and delete cameras with dynamic viewpoints.

You can constrain the camera trajectories using keyframe and tracking modes. Use the keyframe mode to set the camera viewpoints at specific playback times and apply smooth interpolation between them. Use the tracking mode to fix the camera position and aim to coordinate frames and follow them during playback.

The cameras that you create appear in the cameras list shown in the visualization context-sensitive menu. To select a camera, right-click the visualization pane and select **Camera**. If the visualization pane is split into tiles, you can assign a different camera to each tile. All dynamic cameras use a perspective projection to capture the visualization contents.

Open the Camera Manager App

From the Mechanics Explorer menu bar, select **Tools > Camera Manager**. To open the camera definition pane, click the  button in the **New Camera** field or the  button in an existing camera field. Use the camera definition pane to set the camera mode and trajectory constraints.

Parameters

Camera Name — Name of the camera

MATLAB string

Label used to identify the camera in the main pane of Camera Manager and in the visualization context-sensitive menu.

Mode — Dynamic camera mode

Keyframes (default) | Tracking

Select a mode for defining the camera trajectory:

- **Keyframes** — Set the camera viewpoints at specific playback times. The camera trajectory is the result of smooth interpolation applied between keyframes.
- **Tracking** — Constrain the camera position, aim, and up vector to coordinate frames in the model. The camera trajectory is the result of the constraints applied to the camera.

Keyframes — Set, remove, and navigate keyframes

Buttons

Use the buttons to set, remove, and navigate camera keyframes:

- **Set** — Define a keyframe with the current viewpoint shown in the active visualization tile. Click **Set** for an existing keyframe to modify its definition.
- **Remove** — Remove the currently selected keyframe from the camera trajectory definition. The location of the playback slider identifies the selected keyframe.
- **Previous** and **Next** — Jump to the previous or next defined keyframes.

Before setting keyframes, you must simulate the model and pause playback. The **Keyframes** parameter is active only when the **Mode** parameter is set to Keyframes.

Position — Fix the camera position to a frame origin

Button

Frame origin used to constrain the camera position. During simulation, the camera position follows the trajectory traced by the selected frame origin. To set the camera position:

- 1 In the Mechanics Explorer visualization or tree view panes, select a coordinate frame.
- 2 In Camera Manager, click the **Use Selected Frame** button.

Be sure to select the frame itself and not simply the solid or body it belongs to. The **Position** parameter is active only when the **Mode** parameter is set to Tracking.

Aim — Fix the camera aim to a frame origin or along a frame axis

Button

Frame origin or axis used to constrain the camera orientation. During simulation, the camera aim stays fixed on the selected frame origin or aligned along the selected frame axis. To set the camera aim:

- 1 In the Mechanics Explorer visualization or tree view panes, select a coordinate frame.
- 2 In Camera Manager, click the **Use Selected Frame** button.
- 3 From the adjacent drop-down list, select the frame origin or axis to constrain the camera aim to.

Be sure to select the frame itself and not simply the solid or body it belongs to. The **Aim** parameter is active only when the **Mode** parameter is set to Tracking.

Up Vector — Fix the camera up direction along a frame axis Button

Frame axis used to constrain the camera up direction. During simulation, the up direction stays aligned with the selected axis. To set the camera up direction:

- 1 In the Mechanics Explorer visualization or tree view panes, select a coordinate frame.
- 2 In Camera Manager, click the **Use Selected Frame** button.
- 3 From the adjacent drop-down list, select the frame axis to align the camera up direction with.

Be sure to select the frame itself and not simply the solid or body it belongs to. This parameter is active only when the **Mode** parameter is set to Tracking.

See Also

Topics

“Visualization Cameras”

“Create a Dynamic Camera”

Introduced in R2016a

Video Creator

Configure and create multibody animation videos

Description

Video Creator is a **Mechanics Explorer** tool that lets you configure and create videos of multibody animations. You can modify the video playback speed, frame rate, file format, and frame size. Click **Create** to generate a video with the specified properties. Use the `smwritevideo` function for a programmatic alternative to Video Creator.

Open the Video Creator App

From the Mechanics Explorer menu bar, select **Tools > Video Creator**. You must simulate the model in order to use Video Creator or the programmatic equivalent `smwritevideo` function.

Parameters

Playback Speed Ratio – Video playback speed relative to real time

1.0 (default)

Video playback speed relative to real time, specified as a positive number. The video plays faster than real time at values greater than 1 and slower at values smaller than 1. For example, a ratio of 2 doubles the playback speed while a ratio of 0.5 halves it.

Frame Rate (FPS) – Number of video frames per second

30 (default)

Number of video frames per second of playback time, specified as a positive number. Larger frame rates result in smoother video playback time but also larger file sizes.

Video Format – Video file format

Motion JPEG AVI (default) | Archival | Motion JPEG 2000 | MPEG-4 | Uncompressed AVI

File format to save the video in. The dropdown list provides various formats to select from, including compressed and uncompressed formats.

Frame Size — Video frame width and height

auto (default)

Video frame width and height, specified in pixel units as the two-element row vector [Width Height]. The frame dimensions must be positive integers. For example, the vector [800 400] sets the video frame dimensions to 800 pixels in width and 400 pixels in height. Enter the string `auto` instead to use the current dimensions of the active visualization tile in Mechanics Explorer.

See Also

`smwritevideo`

Introduced in R2016b

Functions—Alphabetical List

addFrameVariables

Package: `simscape.multibody`

Create kinematic variables from select frame pair in `KinematicsSolver` object

Syntax

```
addFrameVariables(ks,groupName,type,base,follower)
addFrameVariables(ks,groupName,type,base,follower,Name,Value)
```

Description

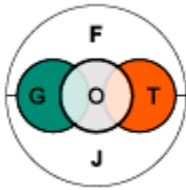
`addFrameVariables(ks,groupName,type,base,follower)` adds to the `KinematicsSolver` object `ks` the variables needed to capture the transforms between a frame pair. The frames can be any pair of interest, one serving as follower, the other as base.

The transforms can be of two types: translations and rotations. The variables are their vector components, bundled to form a new variable group, or to extend one already formed.

The translation variables comprise an x - y - z Cartesian displacement sequence. The rotation variables comprise an x - y - z intrinsic angle sequence. The angles are each about the axes of the rotating follower frame.

The output is an updated table with the frame variables—both new and old—in rows. Each row gives the ID of a variable, the Simulink block paths to the base and follower frames from which it derives, and the unit in which its value is expressed.

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). Joint variables are native to the object and can be assigned from its start as targets, guesses, and outputs. Frame variables must first be created with `addFrameVariables`.



`addFrameVariables(ks, groupName, type, base, follower, Name, Value)` adds the frame variables to the `KinematicsSolver` object `ks` and changes their units to an equivalent measure specified in the Name-Value pair arguments.

Input Arguments

ks — KinematicsSolver object for which to run the analysis

`KinematicsSolver` object

Name of the `KinematicsSolver` object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: `char` | `string`

groupName — Name of group in which to place new variables

string scalar or character array

Name of the group in which to place the new frame variables. Each variable corresponds to a transform component. If adding variables to an existing group, use the name of that group.

The group name is the first of three strings in the IDs of frame variables. The IDs are each structured `group.type.component`.

Example: 'wrist_position'

Data Types: `char` | `string`

type — Type of transform to capture with the frame variables

string scalar or character array

Type of transform which the new frame variables are to capture. Use `Translation` for the linear offsets from base to follower. Use `Rotation` for the angular offsets. The first arrange in *x-y-z* Cartesian sequences, and the second arrange in *x-y-z* angle sequences.

The transform type is the second of three strings in the IDs of frame variables. The IDs are each structured `group.type.component`.

Example: 'translation'

Data Types: `char | string`

base — Simulink path from model root to block port for base frame

string scalar or character array

Simulink path from the root of the model to the frame port from which to obtain the base frame. Frame variables each derive from a frame pair, one serving as base, the other as follower. The base frame is that relative to which the transforms given in the frame variables are specified.

Example: 'sm_import_humanoid_urdf/World/W'

Data Types: `char | string`

follower — Simulink path from model root to block port for follower frame

string scalar or character array

Simulink path from the root of the model to the frame port from which to obtain the follower frame. Frame variables each derive from a frame pair, one serving as base, the other as follower. The follower frame is that relative to which the transforms given in the frame variables are specified.

Example: 'sm_import_humanoid_urdf/left_hand/F'

Data Types: `char | string`

Name-Value Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1, Value1, . . . , NameN, ValueN`.

Example: `addFrameVariables('ks','Translation','sm_four_bar/'World Frame/W','sm_four_bar/Connector Link/Left End Cap/R');`

angleUnit — Unit to use in place of default for angles

string scalar or character array

Unit in which to express the frame variables if of type `Rotation`. The new unit overrides the object default for angles.

Example: 'rad'

Data Types: char | string

lengthUnit — Unit to use in place of default for length

string scalar or character array

Unit in which to express the frame variables if of type `Translation`. The new unit overrides the object default for length.

Example: 'in'

Data Types: char | string

See Also

`KinematicsSolver` | `clearFrameVariables` | `frameVariables` | `jointVariables` | `removeFrameVariables`

Introduced in R2019a

addInitialGuessVariables

Package: `simscape.multibody`

Assign kinematic variables from the `KinematicsSolver` object as guesses

Syntax

```
addInitialGuessVariables(ks,ids)
```

Description

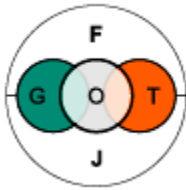
`addInitialGuessVariables(ks,ids)` assigns as guess variables the kinematic variables listed in the `KinematicsSolver` object `ks` under the names given in the `ids` argument. Both joint and frame variables can serve in this role. Those that do bias the solver toward one of equally plausible solutions when several exist. Guess variables are optional but important solver guides in some kinematic problems.

The output is an updated table with the guess variables—both new and old—in rows. Each row gives the ID of a variable, the type and block path of the joint to which it belongs if a joint variable, the base and follower frames from which it spawns if a frame variable, and the unit for its numerical value. The variables rank in the order added.

Most variables can be assigned individually. A few must be assigned in groups—axis components alongside rotation angle in spherical primitives; bend angle alongside azimuth angle in constant-velocity primitives. (A bend angle can be assigned individually but the azimuth angle cannot.)

No attempt is made to satisfy guess variables. They are a starting point in the search for a solution. Use them merely to bias the solver toward a suitable solution when several exist.

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). The same variable can serve as guess and output, but if it serves as target, it cannot double as guess. Assigning a guess variable as target clears it as guess.



Input Arguments

ks — KinematicsSolver object for which to run the analysis

KinematicsSolver object

Name of the KinematicsSolver object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: char | string

ids — Identifiers of kinematic variables to use

character vector or string scalar

Identifiers of the kinematic variables to use. Enter the identifiers as shown in the ID column of the jointVariables, for joint variables, or frameVariables, for frame variables.

Example: 'j1.Rz.q'

Data Types: char | string

See Also

KinematicsSolver | clearInitialGuessVariables | initialGuessVariables | removeInitialGuessVariables

Introduced in R2019a

addOutputVariables

Package: `simscape.multibody`

Assign kinematic variables from the `KinematicsSolver` object as outputs

Syntax

```
addOutputVariables(ks,ids)
```

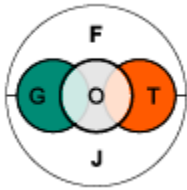
Description

`addOutputVariables(ks,ids)` assigns as output variables the kinematic variables listed in the `KinematicsSolver` object `ks` under the names given in the `ids` argument. Both joint and frame variables can serve as outputs. Those that do are unknowns to solve for and report on during analysis. Their solution is constrained by target variables and biased toward one of equally plausible solutions, when several exist, by guess variables.

The output is an updated table with the output variables—both new and old—in rows. Each row gives the ID of a variable, the type and block path of the joint to which it belongs if a joint variable, the base and follower frames from which it spawns if a frame variable, and the unit for its numerical value. The variables rank in the order added.

Most variables can be assigned individually. A few must be assigned in groups—axis components alongside rotation angle in spherical primitives; bend angle alongside azimuth angle in constant-velocity primitives. (A bend angle can be assigned individually but the azimuth angle cannot.)

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). The same variable can serve as guess and output or as target and output but not as guess and target.



Input Arguments

ks — KinematicsSolver object for which to run the analysis

KinematicsSolver object

Name of the KinematicsSolver object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: char | string

ids — Identifiers of kinematic variables to use

character vector or string scalar

Identifiers of the kinematic variables to use. Enter the identifiers as shown in the ID column of the `jointVariables`, for joint variables, or `frameVariables`, for frame variables.

Example: 'j1.Rz.q'

Data Types: char | string

See Also

`KinematicsSolver` | `clearOutputVariables` | `outputVariables` | `removeOutputVariables`

Introduced in R2019a

addTargetVariables

Package: `simscape.multibody`

Assign kinematic variables from the `KinematicsSolver` object as targets

Syntax

```
addTargetVariables(ks,ids)
```

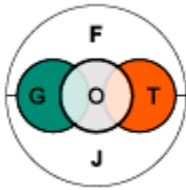
Description

`addTargetVariables(ks,ids)` assigns as target variables the kinematic variables listed in the `KinematicsSolver` object `ks` under the names given in the `ids` argument. Those that do serve as constraints to apply during analysis. Their values must be later defined in order to run the analysis. The solver, when called, searches for a solution compatible with the targeted joint and frame variables.

The output is an updated table with the target variables—both new and old—in rows. Each row gives the ID of a variable, the type and block path of the joint to which it belongs if a joint variable, the base and follower frames from which it spawns if a frame variable, and the unit for its numerical value. The variables rank in the order added.

Most variables can be assigned individually. A few must be assigned in groups—axis components alongside rotation angle in spherical primitives; bend angle alongside azimuth angle in constant-velocity primitives. (A bend angle can be assigned individually but the azimuth angle cannot.)

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). The same variable can serve as target and output, but if it serves as target, it cannot double as guess. Assigning a guess variable as target clears it as guess.



As with position targets in joint blocks in regular Simulink models, target variables should not overconstrain the system they describe. An assembly is overconstrained if a kinematic loop within it has a target for every joint. Kinematic loops such as four-bars and crank-sliders are evident from the model. Others arise when formulating a kinematic problem to solve.

Kinematic chains such as robotic arms and other serial manipulators become kinematic loops when subject to closure constraints. A target for a frame variable between base and end effector frames adds the equivalent of such a constraint. If the closed chain is to be underconstrained, at least one of its joints must stay free of target variables.

Adding a target variable that overconstrains the assembly causes `addTargetVariables` to fail with an error. Consider initial guess variables for the last joint of a kinematic loop, if necessary, to guide the joint into place without overconstraining the assembly. Use the `addInitialGuessVariables` object function to assign a kinematic variable as an initial guess.

Input Arguments

ks — KinematicsSolver object for which to run the analysis

KinematicsSolver object

Name of the KinematicsSolver object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: char | string

ids — Identifiers of kinematic variables to use

character vector or string scalar

Identifiers of the kinematic variables to use. Enter the identifiers as shown in the ID column of the `jointVariables`, for joint variables, or `frameVariables`, for frame variables.

Example: 'j1.Rz.q'

Data Types: `char` | `string`

See Also

`KinematicsSolver` | `clearTargetVariables` | `removeTargetVariables` | `targetVariables`

Introduced in R2019a

clearFrameVariables

Package: simscape.multibody

Drop all frame variables from the KinematicsSolver object

Syntax

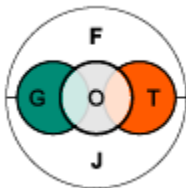
```
clearFrameVariables(ks)
```

Description

`clearFrameVariables(ks)` drops all frame variables from the `KinematicsSolver` object `ks`. Frame variables capture the transforms between any two given frames. Use this object function if none of the frame variables are any longer relevant—for example, before formulating a new kinematic problem for the same multibody model using other frame variables.

Frame and joint variables comprise the whole of kinematic variables in a `KinematicsSolver` object. They can function as targets to constrain the multibody configuration for which to solve the unknowns, as guesses to bias the solution toward one of equally plausible alternatives when several exist, and as outputs—the unknowns in the analysis.

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). Joint variables are native to the object and can be assigned from its start as targets, guesses, and outputs. Frame variables must first be created with `addFrameVariables`.



Input Arguments

ks — KinematicsSolver object for which to run the analysis

`KinematicsSolver` object

Name of the `KinematicsSolver` object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: `'fourBarKS'`

Data Types: `char` | `string`

See Also

`KinematicsSolver` | `addFrameVariables` | `frameVariables` | `removeFrameVariables`

Introduced in R2019a

clearInitialGuessVariables

Package: Simscape.Multibody

Drop all guess variables from the KinematicsSolver object

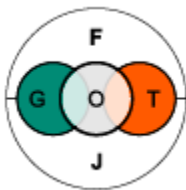
Syntax

```
clearInitialGuessVariables(ks)
```

Description

`clearInitialGuessVariables(ks)` drops all guess variables from the KinematicsSolver object `ks`. Guess variables bias the solver toward one of equally plausible solutions when several exist. Use this function if none of the guess variables are any longer relevant—for example, before formulating a new kinematic problem for the same multibody model using other guess variables. Guess variables are optional but important solver guides in some kinematic problems.

The figure shows the variables of a general KinematicsSolver object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). The same variable can serve as guess and output, but if it serves as target, it cannot double as guess. Assigning a guess variable as target clears it as guess.



Input Arguments

ks — KinematicsSolver object for which to run the analysis
KinematicsSolver object

Name of the `KinematicsSolver` object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: `char` | `string`

See Also

`KinematicsSolver` | `addInitialGuessVariables` | `initialGuessVariables` | `removeInitialGuessVariables`

Introduced in R2019a

clearOutputVariables

Package: simscape.multibody

Drop all output variables from the KinematicsSolver object

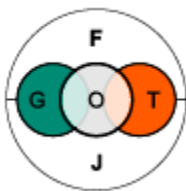
Syntax

```
clearOutputVariables(ks)
```

Description

`clearOutputVariables(ks)` drops all output variables from the `KinematicsSolver` object `ks`. Output variables are the unknowns to solve for and report on during analysis. Use this function if none of the output variables are any longer relevant—for example, before formulating a new kinematic problem for the same multibody model using other output variables.

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). The same variable can serve as guess and output or as target and output but not as guess and target.



Input Arguments

ks — `KinematicsSolver` object for which to run the analysis
`KinematicsSolver` object

Name of the `KinematicsSolver` object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: `char` | `string`

See Also

`KinematicsSolver` | `addOutputVariables` | `outputVariables` | `removeOutputVariables`

Introduced in R2019a

clearTargetVariables

Package: simscape.multibody

Drop all target variables from the KinematicsSolver object

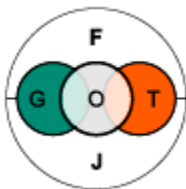
Syntax

```
clearTargetVariables(ks)
```

Description

`clearTargetVariables(ks)` drops all target variables from the `KinematicsSolver` object `ks`. Target variables guide joints and bodies into place for analysis. Use this function if none of the target variables are any longer relevant—for example, to formulate a different kinematic problem to solve.

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). The same variable can serve as target and output, but if it serves as target, it cannot double as guess. Assigning a guess variable as target clears it as guess.



Input Arguments

ks — `KinematicsSolver` object for which to run the analysis
`KinematicsSolver` object

Name of the `KinematicsSolver` object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: `char` | `string`

See Also

`KinematicsSolver` | `addTargetVariables` | `clearTargetVariables` | `removeTargetVariables` | `targetVariables`

Introduced in R2019a

frameVariables

Package: `simscape.multibody`

List all kinematic variables associated with frame pairs

Syntax

```
frameVariables(ks)
```

Description

`frameVariables(ks)` lists the kinematic variables currently defined in the `KinematicsSolver` object `ks` to capture the transforms between frame pairs. The frame pairs are specified when creating the variables. They can be from anywhere in the model and often comprise frames not directly connected by joints.

MATLAB outputs a table with the frame variables in rows. Each row gives the ID of a variable, the base frame against which its transform is defined, the follower frame which the transform describes, and the unit for its numerical value.

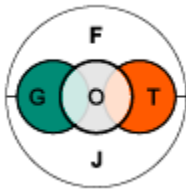
`KinematicsSolver` objects start without frame variables. The table is initially empty and it remains so until populated with variables created from frame pairs. Use the `addFrameVariables` object function to create those variables. Use `removeFrameVariables` to drop variables no longer relevant for analysis, and `clearFrameVariables` to drop all frame variables in one call.

The same pair of frames can give rise different variables. These are generally bundled into a group specific to the pair. Groups are sorted by name, with variables in each group being ranked by transform type and component. The variables are correspondingly named `groupName.transformType.transformComponent`.

The group name is the string given to the `addFrameVariables` object function at the time the variables were created. The transform type is either translation or rotation. The transform component is `x`, `y`, or `z`. A group named `Hand` with the translation components between frames would have for its frame variables `Hand.Translation.X`, `Hand.Translation.Y`, and `Hand.Translation.Z`.

Frame variables can be assigned as targets, guesses, and outputs during analysis. As targets, they help to specify the multibody configuration for which to solve the unknown variables. As guesses, they serve to bias the solver toward one of equally plausible solutions—for example, to ensure that the elbow of a humanoid robot does not hyperextend. As outputs, they add to the list of unknowns to determine.

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). Joint variables are native to the object and can be assigned from its start as targets, guesses, and outputs. Frame variables must first be created with `addFrameVariables`.



Input Arguments

ks — KinematicsSolver object for which to run the analysis

`KinematicsSolver` object

Name of the `KinematicsSolver` object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: `char` | `string`

See Also

`KinematicsSolver` | `addFrameVariables` | `clearFrameVariables` | `jointVariables` | `removeFrameVariables`

Introduced in R2019a

generateCode

Package: `simscape.multibody`

Generate C code to run kinematic analysis on `KinematicsSolver` object

Syntax

```
generateCode(ks)
```

Description

`generateCode(ks)` generates C code to run a kinematic analysis on the `KinematicsSolver` object `ks`. The code is specific to the object and its kinematic variables—its targets, guesses, and outputs—at the time of code generation. A MATLAB function accompanies the outputs and serves to build the code for simulation in Simulink or for deployment outside MATLAB.

The code is stored in C files with a standard H file for global declarations. The files are stored in a folder created in the MATLAB working directory and named after the model from which the `KinematicsSolver` object derives. The name of the folder is `<modelname>_codegen_kinematics`. The accompanying MATLAB function is stored in an M file placed in the MATLAB working directory under the name `<modelname>_solveKinematics.m`.

The MATLAB function is not to be called directly. Call the function from a MATLAB Function block to build the code for simulation. The block builds the code automatically once the simulation starts. Alternatively, pass the name of the MATLAB function as an argument to the `codegen` function of Simulink Coder to build the code for deployment. For example:

```
generateCode(ks);  
codegen -config:mex <MODELNAME>_solveKinematics
```

A Simulink Coder license is required.

Input Arguments

ks — KinematicsSolver object for which to run the analysis

`KinematicsSolver` object

Name of the `KinematicsSolver` object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: `char` | `string`

See Also

`KinematicsSolver` | `solve`

Introduced in R2019a

initialGuessVariables

Package: simscape.multibody

List all kinematic variables assigned as initial guesses

Syntax

```
initialGuessVariables(ks)
```

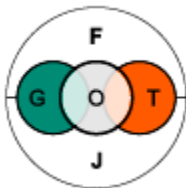
Description

`initialGuessVariables(ks)` lists the kinematic variables in the `KinematicsSolver` object `ks` currently assigned as guesses. Both joint and frame variables can serve in this role. Those that do bias the solver toward one of equally plausible solutions when several exist. Guess variables are optional but important solver guides in some kinematic problems.

The output is a table with the guess variables in rows. Each row gives the ID of a variable, the type and block path of the joint to which it belongs if a joint variable, the base and follower frames from which it derives if a frame variable, and the unit for its numerical value. The variables rank in the order added.

No attempt is made to satisfy guess variables. They are a starting point in the search for a solution. Use them merely to bias the solver toward a suitable solution when several exist.

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). The same variable can serve as guess and output, but if it serves as target, it cannot double as guess. Assigning a guess variable as target clears it as guess.



Input Arguments

ks — KinematicsSolver object for which to run the analysis

KinematicsSolver object

Name of the KinematicsSolver object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: char | string

See Also

KinematicsSolver | addInitialGuessVariables |
clearInitialGuessVariables | removeInitialGuessVariables

Introduced in R2019a

jointVariables

Package: `simscape.multibody`

List all kinematic variables associated with joints

Syntax

```
jointVariables(ks)
```

Description

`jointVariables(ks)` lists the kinematic variables native to the `KinematicsSolver` object `ks` and available to capture the displacements of joints.

MATLAB outputs a table with the joint variables in rows. Each row gives the ID of a variable, the type of joint it belongs to, the path from the root of the model to the corresponding joint block in the Simulink model, and the unit for its numerical value. The table starts out fully populated with all the joint variables identified in the model at the time the `KinematicsSolver` object is constructed.

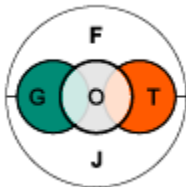
A model can have many joint variables. These are sorted by joint, joint primitive, and primitive component. The variables are correspondingly named according to the scheme `jointName.primitiveType.primitiveComponent`. A gimbal joint, which has three revolute primitives (Rx, Ry, and Rz), each with a rotational component (q), allows for three joint variables. With `j1` for joint name, the variable identifiers become `j1.Rx.q`, `j1.Ry.q`, and `j1.Rz.q`.

Joint primitives can also be prismatic (Px, Py, and Pz), spherical (S), constant-velocity (CV), or lead-screw (LSz). Primitive components can be translations (p), rotations (q), and axis components (ax). Spherical primitives are the only to allow rotation about a general 3-D axis, and so only they have axis components. These are denoted `ax_x`, `ax_y`, and `ax_z`. Other joint primitives have one component each—save for the constant-velocity primitive, which has an azimuth rotation component (`q_a`) and a bend rotation component (`q_b`).

Joint variables can play different roles in the analysis. They can serve as targets, guesses, or outputs. As targets, they help to specify the multibody configuration for which to solve

the output variables. As guesses, they help to bias the solver toward a desired solution—for example, to ensure that the elbow of a humanoid robot bends and extends only in the natural range of 0-145 degrees. As outputs, they add to the unknowns to be determined by the solver.

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). Joint variables are native to the object and can be assigned from its start as targets, guesses, and outputs. Frame variables must first be created with `addFrameVariables`.



Input Arguments

ks — KinematicsSolver object for which to run the analysis

`KinematicsSolver` object

Name of the `KinematicsSolver` object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: `char` | `string`

See Also

`KinematicsSolver` | `frameVariables`

Introduced in R2019a

KinematicsSolver

Kinematic representation of multibody model to analyze

Description

The `KinematicsSolver` object contains a kinematic representation of a multibody model around which to formulate, and for which to solve, a kinematic problem. The analysis can be the standard inverse or forward kinematics or it can be another type. The distinction rests solely on the variables known and sought, and of these many arrangements are possible.

A new object contains only some of the kinematic variables likely to feature in the kinematic analysis. These it inherits from the model. It also has no indication of which variables to compute and which to assume for its arguments. The task of preparing the analysis is one of adding variables to the object and assigning them to suitable roles. Variables derive either from joints or from frame pairs:

- **Joint variables**

The displacements of joint primitives. They are translations for prismatic primitives and rotations for revolute and spherical primitives. Spherical rotations bundle with auxiliary variables to express also the vector components of the 3-D rotation axis.

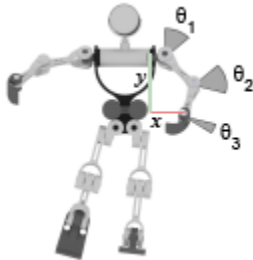
The `KinematicsSolver` object contains all joint variables compatible with the underlying model from its start. Those variables are a snapshot of the model as it is when the object is created. You can neither add nor drop them.

- **Frame variables**

The transforms between select frame pairs. The pairs are often from bodies not connected by joints, the world sometimes in the place of a body. The transforms can be translations or rotations from one frame (the base) to the second (the follower).

The `KinematicsSolver` object starts without frame variables. Those that it gains result from calls to `addFrameVariables`. You can drop them if they turn obsolete. Use `removeFrameVariables` to remove a few at a time and `clearFrameVariables` to remove them all in one call.

Consider a robotic arm with revolute joints for its shoulder, elbow, and wrist. The revolute angles each suit a joint variable. The position of the hand relative to the shoulder, two bodies which do not normally connect through joints, require a group of frame variables—one each for the x , y , and z position offsets between the bodies, or, more precisely, between frames local to them.



The source of a variable does not determine its role in the analysis. A joint variable can be an input argument or an output argument. So can a frame variable. The `KinematicsSolver` object recognises two input types: targets to aim for and guesses to start the analysis from, the latter to bias the solution when alternatives exist.

Targets and outputs feature in a typical analysis. Guesses are less common but valuable in problems with multiple solutions, as a means to bias the solver toward one thought suitable. In a robotic arm, a guess for the elbow helps to keep its rotation angle in the natural range of 0 - 145 degrees during analysis. The `KinematicsSolver` object provides methods for each role:

- **Target variables**

Use the `addTargetVariables` object function to assign joint and frame variables as targets. Use `removeTargetVariables` to drop one or a few from the `KinematicsSolver` object and `clearTargetVariables` to drop them all in one call.

Targets work with a caveat: conflicts with constraints in the model, some posed by still other targets, may keep them all from being precisely satisfied. Targets guide but do not force joints and bodies into place for analysis. The unknowns are solved for whatever multibody configuration the targets manage to produce.

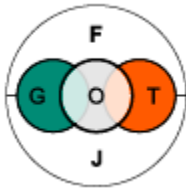
- **Output variables**

Use the `addOutputVariables` object function to assign joint and frame variables as outputs. Use `removeOutputVariables` to drop one or a few from the `KinematicsSolver` object and `clearOutputVariables` to drop them all in one call.

- **Guess variables**

Use the `addInitialGuessVariables` object function to assign joint and frame variables as guesses. Use `removeInitialGuessVariables` to drop one or a few from the `KinematicsSolver` object and `clearInitialGuessVariables` to drop them all in one call.

See the figure for a summary of the variables and their roles in a `KinematicsSolver` object. An object can have joint variables (**J**) and frame variables (**F**). These can serve as targets (**T**) to guide assembly during analysis, as guesses (**G**) to bias the solution toward a suitable configuration, and as outputs (**O**), the unknowns to solve for and report on. Variables assigned as targets cannot double as guesses but they each can double as output.



The `KinematicsSolver` object is ready for analysis once the necessary kinematic variables are in place and their roles are set. For inverse kinematics of the robotic arm, for example, it suffices to create three frame variables—one each for a translation component between hand and shoulder frames. Joint frames are native to the object and available from the start. Assigning the frame variables as targets and the joint variables as outputs readies the object for analysis.

To run the analysis, the object provides a solver. Call it using the `solve` object function. The function takes the numerical values of the targets and guesses as input to calculate the outputs. Often, if the specified targets are within reach, these can be satisfied. Such an outcome is not always possible. A target may be out of the reach of the solver, for example.

To warn of such issues, the solver returns not only the actual values achieved for the variables assigned as targets, but also a series of flags. One gives the overall assembly status. If any target is missed or constraint violated, or if the solution leaves the assembly

in a singular configuration and so with reduced mobility, it will reflect there. The remainder appear in a vector indicating which targets were satisfied and which were not.

Creation

Syntax

```
ks = simscape.multibody.KinematicsSolver(modelName)
ks = simscape.multibody.KinematicsSolver(modelName,Name,Value)
```

Description

`ks = simscape.multibody.KinematicsSolver(modelName)` creates a `KinematicsSolver` object for the model named in `mdl`. The object contains a representation of the model suitable for kinematic analysis. The representation is a snapshot of the model as it is when the object is created. Subsequent changes to the model do not carry over to the object. Create a new object, if necessary to capture those changes.

The model must contain a Simscape Multibody network. It must also be loaded to memory—for example by use of the `load_system` command. If blocks are parameterized in terms of MATLAB variables, those variables must be numerically defined in the model workspace or in the MATLAB workspace. Position targets and actuation inputs for joint blocks are ignored. Block parameters set to Run-Time are evaluated when creating the object and cannot be modified afterward.

A `KinematicsSolver` object is a handle object. A variable created from it contains not a copy of the object but a reference to it. The variable acts as a pointer or *handle*. Modifying a handle modifies also the object and all of its remaining handles. Copying a `KinematicsSolver` object and adding a frame variable to the copy, for example, adds that frame variable to the object and so also to any other handles it might have.

Note Do not save a `KinematicsSolver` object to a MAT file. Important detail about the underlying model is lost during the save. The object becomes unusable once reloaded into MATLAB.

`ks = Simscape.Multibody.KinematicsSolver(modelName, Name, Value)` creates a `KinematicsSolver` object with new defaults for units of angle or length. Use the Name-Value pair arguments `DefaultAngleUnit` and `DefaultLengthUnit` to specify the new defaults.

Properties

ModelName — Name of multibody model

string scalar or character vector

This property is read-only.

Name of the model from which the object derives.

Example: 'sm_four_bar'

Data Types: char | string

DefaultAngleUnit — Default unit for rotation

string scalar or character vector

This property is read-only.

Unit of angle for new kinematic variables. The default is `deg`. A change in default units affects only variables added after the change. Older variables remain in their original units.

Example: 'rad'

Data Types: char | string

DefaultLengthUnit — Default unit for translation

cell array of string scalars | cell array of character vectors

This property is read-only.

Unit of length for new kinematic variables. The default is `m`. A change in default units affects only variables added after the change. Older variables remain in their original units.

Example: 'in'

Data Types: char | string

Object Functions

Listing Variables

frameVariables	List all kinematic variables associated with frame pairs
initialGuessVariables	List all kinematic variables assigned as initial guesses
jointVariables	List all kinematic variables associated with joints
outputVariables	List all kinematic variables assigned as outputs
targetVariables	List kinematic variables assigned as targets

Adding Variables

addFrameVariables	Create kinematic variables from select frame pair in KinematicsSolver object
addInitialGuessVariables	Assign kinematic variables from the KinematicsSolver object as guesses
addOutputVariables	Assign kinematic variables from the KinematicsSolver object as outputs
addTargetVariables	Assign kinematic variables from the KinematicsSolver object as targets

Removing Variables

clearFrameVariables	Drop all frame variables from the KinematicsSolver object
clearInitialGuessVariables	Drop all guess variables from the KinematicsSolver object
clearOutputVariables	Drop all output variables from the KinematicsSolver object
clearTargetVariables	Drop all target variables from the KinematicsSolver object

Clearing Variables

removeFrameVariables	Drop select frame variables from the KinematicsSolver object
removeInitialGuessVariables	Drop select guess variables from the KinematicsSolver object
removeOutputVariables	Drop select output variables from the KinematicsSolver object
removeTargetVariables	Drop select target variables from the KinematicsSolver object

Configuring and Solving

<code>generateCode</code>	Generate C code to run kinematic analysis on KinematicsSolver object
<code>setVariableUnit</code>	Change physical unit of kinematic variable
<code>solve</code>	Run kinematic analysis for KinematicsSolver object

Examples

Run Forward Kinematics on Humanoid Robot Arm

Run forward kinematics on a model of a humanoid robot arm. The model is named `sm_import_humanoid_urdf`, and it is part of the Simscape Multibody installation. For the analysis, specify the rotations of the wrist, elbow, and shoulder joints. Solve for the translation and rotation of the hand in the world frame.

- 1 Load the humanoid robot model into memory and create KinematicsSolver object for its current state.

```
sys = 'sm_import_humanoid_urdf';
load_system(sys);
ks = simscape.multibody.KinematicsSolver(sys);
```

The object contains a kinematic representation of the model and a list of all the joint variables that it contains. Enter `ks.JointVariables` or `jointVariables(ks)` at the MATLAB command prompt to list those variables when necessary. Open the model for use as reference by entering its name at the MATLAB command prompt.

- 2 Add to `ks` a group of frame variables for the hand relative to the world. Specify the **F** frame of the `left_hand` subsystem as follower and the world frame as base. Name the frame variable group `Hand`.

```
base = 'sm_import_humanoid_urdf/World/W';
follower = 'sm_import_humanoid_urdf/left_hand/F';
ks.addFrameVariables('Hand', 'translation', base, follower);
ks.addFrameVariables('Hand', 'rotation', base, follower);
```

The object gains its first six frame variables—three for the x , y , and z translation components and three for the x , y , and z rotation components, each from the world frame to the **F** frame of the hand. Enter `ks.frameVariables` at the MATLAB command prompt to list those variables when necessary.

The paths in `base` and `follower` are the full block paths from the root of the model to the port of the block associated with the port desired. That port is **W** in the World

Frame block for the base frame and **F** of the `left_hand` subsystem block for the follower.

- 3 Assign as targets the joint variables for the left wrist, elbow, and shoulder.

```
ks.addTargetVariables(ks.jointVariables.ID([2,6:8]));
```

The joint variables are referenced by position in the ID column of the joint variables table. Use `jointVariables` to determine which joint variables to specify. Entries 2, 6, 7, and 8 correspond respectively to the elbow, shoulder frontal, shoulder sagittal, and wrist joints.

- 4 Assign as outputs all the frame variables in the Hand group.

```
ks.addOutputVariables(ks.frameVariables.ID);
```

Not all frame variables need feature in the analysis. Use the `frameVariables` function to list all frame variables and determine which to specify when necessary.

- 5 Solve the forward kinematics problem given the elbow, shoulder frontal, shoulder sagittal, and wrist joint angles of 30, 45, 45, and 15 degrees.

```
targets = [30,45,45,15];  
outputVec = ks.solve(targets)
```

```
outputVec =
```

```
    0.2196  
    0.0584  
   -0.0983  
  135.0000  
    0.0027  
    0
```

The `solve` function returns the values of the output variables. The values are sorted in the order of the variables—translation components first, rotation components second. The units are the defaults of `m`, for translation components, and `deg`, for rotation components.

- 6 Clear all target and output variables to ready the `ks` object for another analysis.

```
ks.clearTargetVariables;  
ks.clearOutputVariables;
```

Use the closely related `removeTargetVariables` and `removeOutputVariables` functions to drop just a few target and output variables from the object, when necessary. Use `removeFrameVariables` and `clearFrameVariables` if some or all frame variables are no longer necessary for analysis.

Run Inverse Kinematics on Humanoid Robot Arm

- 1 Load the humanoid robot model into memory and create KinematicsSolver object for its current state.

```
sys = 'sm_import_humanoid_urdf';
load_system(sys);
ks = simscape.multibody.KinematicsSolver(sys);
```

The object contains a kinematic representation of the model and a list of all the joint variables that it contains. Enter `ks.JointVariables` or `jointVariables(ks)` at the MATLAB command prompt to list those variables when necessary. Open the model for use as reference by entering its name at the MATLAB command prompt.

- 2 Add to `ks` a group of frame variables for the hand relative to the world. Specify the **F** frame of the `left_hand` subsystem as follower and the world frame as base. Name the frame variable group `Hand`.

```
base = 'sm_import_humanoid_urdf/World/W';
follower = 'sm_import_humanoid_urdf/left_hand/F';
ks.addFrameVariables('Hand','translation',base,follower);
```

The object gains its first three frame variables, one each for the x , y , and z translation components from the world frame to the **F** frame of the hand. Enter `ks.frameVariables` at the MATLAB command prompt to list those variables when necessary.

The paths in `base` and `follower` are the full block paths from the root of the model to the port of the block associated with the port desired. That port is **W** in the World Frame block for the base frame and **F** of the `left_hand` subsystem block for the follower.

- 3 Assign as targets the frame variables in the `Hand` group.

```
ks.addTargetVariables(ks.frameVariables.ID);
```

Not all frame variables need feature in the analysis. Use the `frameVariables` function to list all frame variables and determine which to specify when necessary.

- 4 Assign as outputs the joint variables for the left wrist, elbow, and shoulder.

```
ks.addOutputVariables(ks.jointVariables.ID([2,6:8]));
```

The joint variables are referenced by position in the ID column of the joint variables table. Use `jointVariables` to determine which joint variables to specify. Entries 2,

6, 7, and 8 correspond respectively to the elbow, shoulder frontal, shoulder sagittal, and wrist joints.

- 5 Solve the inverse kinematics problem given the hand frame translation components of $[16, -12, 0]$ cm or, in the default units of length, $[0.16, -0.12, 0]$ m.

```
targets = [0.16, -0.12, 0];  
outputVec = ks.solve(targets)
```

The `solve` function returns the values of the output variables—the rotation angles of the elbow, shoulder frontal, shoulder sagittal, and wrist joints, each in the default units of deg.

```
outputVec =
```

```
    52.8406  
   -108.3936  
     7.0457  

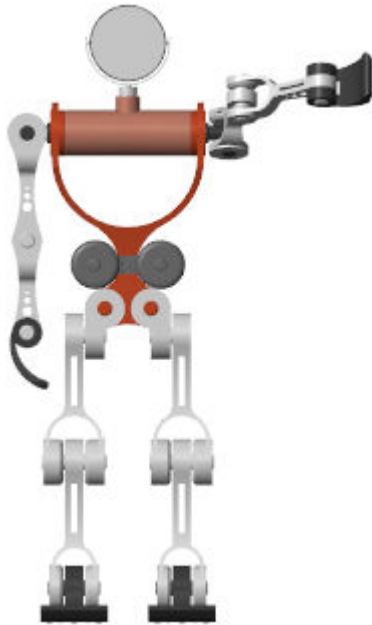
```

- 6 Specify the output values as joint state targets in the `sm_import_humanoid_urdf` model and update the block diagram to visualize the solution in Mechanics Explorer and determine if it is reasonable.

```
for i = 1:length(outputVec)  
    path = ks.outputVariables.JointVariableInfo.BlockPath(i);  
    angle = num2str(outputVec(i));  
    set_param(path, 'PositionTargetSpecify', 'on', ...  
              'PositionTargetValue', angle);  

```

Select **Simulink > Update Diagram** in the Simulink menu bar to update the diagram.



The hand is in the right place, but the elbow has an unnatural bend outside of its normal range of motion. The solution is merely one of several and a better one is possible by specifying the elbow rotation as a guess variable.

- 7** Set the elbow joint variable—the only that needs guidance—as a guess variable and run the analysis once again for an elbow rotation guess of -50 deg.

```
ks.addInitialGuessVariables(ks.jointVariables.ID(2));
guesses = -50;
outputVec = ks.solve(targets,guesses);
```

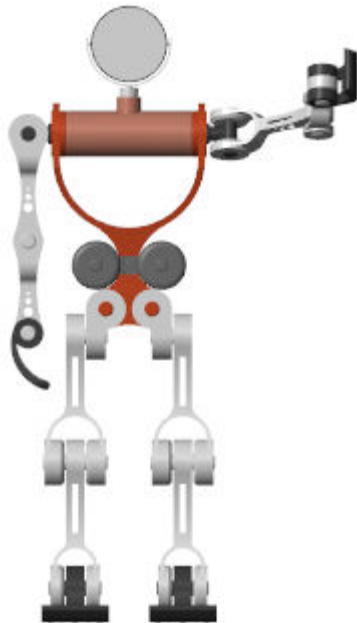
The `solve` function returns a new solution for the joint angles.

```
outputVec =
```

```
-52.8406
-108.3936
 55.5089
 43.6655
```

- 8** Specify the new outputs as joint state targets in the `sm_import_humanoid_urdf` model and update the block diagram to visualize the results.

```
for i = 1:length(outputVec)
    path = ks.outputVariables.JointVariableInfo.BlockPath(i);
    angle = num2str(outputVec(i));
    set_param(path, 'PositionTargetSpecify', 'on',...
        'PositionTargetValue', angle);
end
```



See Also

Introduced in R2019a

outputVariables

Package: simscape.multibody

List all kinematic variables assigned as outputs

Syntax

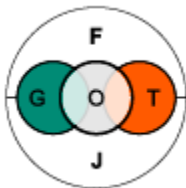
`outputVariables(ks)`

Description

`outputVariables(ks)` lists the kinematic variables in the `KinematicsSolver` object `ks` so far assigned as outputs. Both joint and frame variables can serve in this role. Those that do are unknowns to solve for and report on during analysis. Their solution is constrained by target variables and biased toward one of equally plausible solutions, when several exist, by guess variables.

The output is a table with the output variables in rows. Each row gives the ID of a variable, the type and block path of the joint to which it belongs if a joint variable, the base and follower frames from which it derives if a frame variable, and the unit for its numerical value. The variables rank in the order added.

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). The same variable can serve as guess and output or as target and output but not as guess and target.



Input Arguments

ks — KinematicsSolver object for which to run the analysis

`KinematicsSolver` object

Name of the `KinematicsSolver` object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: `char` | `string`

See Also

`KinematicsSolver` | `addOutputVariables` | `clearOutputVariables` | `removeOutputVariables`

Introduced in R2019a

removeFrameVariables

Package: simscape.multibody

Drop select frame variables from the KinematicsSolver object

Syntax

```
removeFrameVariables(ks,ids)
```

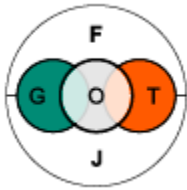
Description

`removeFrameVariables(ks,ids)` drops from the `KinematicsSolver` object `ks` the frame variables named in `ids`. Frame variables capture the transforms between any two given frames. Use this function to remove just a subset of frame variables if they become obsolete. Variables of the same type and in the same group must be removed together.

The output is an updated table with the frame variables—those that remain—in rows. Each row gives the ID of a variable, the base frame against which its transform is defined, the follower frame which the transform describes, and the unit for its numerical value.

Frame and joint variables comprise the whole of kinematic variables in a `KinematicsSolver` object. They can function as targets to constrain the multibody configuration for which to solve the unknowns, as guesses to bias the solution toward one of equally plausible alternatives when several exist, and as outputs—the unknowns in the analysis.

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). Joint variables are native to the object and can be assigned from its start as targets, guesses, and outputs. Frame variables must first be created with `addFrameVariables`.



Input Arguments

ks — KinematicsSolver object for which to run the analysis

KinematicsSolver object

Name of the KinematicsSolver object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: char | string

ids — Identifiers of kinematic variables to use

character vector or string scalar

Identifiers of the kinematic variables to use. Enter the identifiers as shown in the ID column of the `jointVariables`, for joint variables, or `frameVariables`, for frame variables.

Example: 'j1.Rz.q'

Data Types: char | string

See Also

`KinematicsSolver` | `addFrameVariables` | `clearFrameVariables` | `frameVariables`

Introduced in R2019a

removeInitialGuessVariables

Package: simscape.multibody

Drop select guess variables from the KinematicsSolver object

Syntax

```
removeInitialGuessVariables(ks,ids)
```

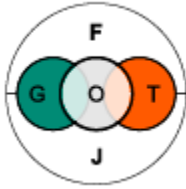
Description

`removeInitialGuessVariables(ks,ids)` drops from the `KinematicsSolver` object `ks` the guess variables named in `ids`. Guess variables provide a starting point for the solution of a kinematic problem and serve to bias the solver toward one of equally plausible alternatives when several exist. Use this function to remove just one or a few guess variables if they become obsolete.

The output is an updated table with the guess variables—those that remain—in rows. Each row gives the ID of a variable, the type and block path of the joint to which it belongs if a joint variable, the base and follower frames from which it spawns if a frame variable, and the unit for its numerical value. The variables rank in the order added.

Most variables can be assigned individually. A few must be assigned in groups—axis components alongside rotation angle in spherical primitives; bend angle alongside azimuth angle in constant-velocity primitives. (A bend angle can be assigned individually but the azimuth angle cannot.)

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). The same variable can serve as guess and output, but if it serves as target, it cannot double as guess. Assigning a guess variable as target clears it as guess.



Input Arguments

ks — KinematicsSolver object for which to run the analysis

KinematicsSolver object

Name of the KinematicsSolver object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: char | string

ids — Identifiers of kinematic variables to use

character vector or string scalar

Identifiers of the kinematic variables to use. Enter the identifiers as shown in the ID column of the jointVariables, for joint variables, or frameVariables, for frame variables.

Example: 'j1.Rz.q'

Data Types: char | string

See Also

KinematicsSolver | addInitialGuessVariables |
clearInitialGuessVariables | initialGuessVariables

Introduced in R2019a

removeOutputVariables

Package: `simscape.multibody`

Drop select output variables from the `KinematicsSolver` object

Syntax

```
removeOutputVariables(ks,ids)
```

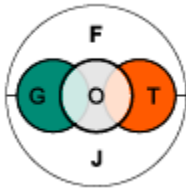
Description

`removeOutputVariables(ks,ids)` drops from the `KinematicsSolver` object `ks` the output variables named in `ids`. Output variables are the unknowns to solve for and report on during analysis. Use this function to remove just one or a few output variables if they become obsolete.

The output is an updated table with the output variables—those that remain—in rows. Each row gives the ID of a variable, the type and block path of the joint to which it belongs if a joint variable, the base and follower frames from which it spawns if a frame variable, and the unit for its numerical value. The variables rank in the order added.

Most variables can be assigned individually. A few must be assigned in groups—axis components alongside rotation angle in spherical primitives; bend angle alongside azimuth angle in constant-velocity primitives. (A bend angle can be assigned individually but the azimuth angle cannot.)

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). The same variable can serve as guess and output or as target and output but not as guess and target.



Input Arguments

ks — KinematicsSolver object for which to run the analysis

KinematicsSolver object

Name of the KinematicsSolver object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: char | string

ids — Identifiers of kinematic variables to use

character vector or string scalar

Identifiers of the kinematic variables to use. Enter the identifiers as shown in the ID column of the `jointVariables`, for joint variables, or `frameVariables`, for frame variables.

Example: 'j1.Rz.q'

Data Types: char | string

See Also

`KinematicsSolver` | `addOutputVariables` | `clearOutputVariables` | `outputVariables`

Introduced in R2019a

removeTargetVariables

Package: simscape.multibody

Drop select target variables from the KinematicsSolver object

Syntax

```
removeTargetVariables(ks,ids)
```

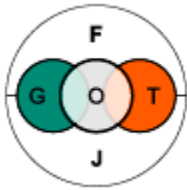
Description

`removeTargetVariables(ks,ids)` drops from the `KinematicsSolver` object `ks` the target variables named in `ids`. Target variables serve to guide joints and bodies into place for analysis. Use this function to remove just one or a few target variables if they become obsolete.

The output is an updated table with the target variables—those that remain—in rows. Each row gives the ID of a variable, the type and block path of the joint to which it belongs if a joint variable, the base and follower frames from which it spawns if a frame variable, and the unit for its numerical value. The variables rank in the order added.

Most variables can be assigned individually. A few must be assigned in groups—axis components alongside rotation angle in spherical primitives; bend angle alongside azimuth angle in constant-velocity primitives. (A bend angle can be assigned individually but the azimuth angle cannot.)

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). The same variable can serve as target and output, but if it serves as target, it cannot double as guess. Assigning a guess variable as target clears it as guess.



Input Arguments

ks — KinematicsSolver object for which to run the analysis

KinematicsSolver object

Name of the KinematicsSolver object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: char | string

ids — Identifiers of kinematic variables to use

character vector or string scalar

Identifiers of the kinematic variables to use. Enter the identifiers as shown in the ID column of the `jointVariables`, for joint variables, or `frameVariables`, for frame variables.

Example: 'j1.Rz.q'

Data Types: char | string

See Also

`KinematicsSolver` | `addTargetVariables` | `clearTargetVariables` | `targetVariables`

Introduced in R2019a

setVariableUnit

Package: simscape.multibody

Change physical unit of kinematic variable

Syntax

```
removeOutputVariables(ks,id,unit)
```

Description

`removeOutputVariables(ks,id,unit)` changes the physical unit of the kinematic variable `id` in the `KinematicsSolver` object `ks` to the measure given in `unit`. That measure must be a valid unit, and the unit must be appropriate for the variable—a length for translation variables and an angle for rotation variables. Rotation axis components, used in spherical joint primitives, must remain unitless.

The new unit applies to every instance of the specified variable: if the variable appears in several variable groups, the unit takes effect in each of the groups.

Input Arguments

ks — KinematicsSolver object for which to run the analysis

`KinematicsSolver` object

Name of the `KinematicsSolver` object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: 'fourBarKS'

Data Types: char | string

id — Identifier for kinematic variable whose units to change

character vector or string scalar

Identifier for the kinematic variable whose units to change. Enter the identifier as shown in the ID column of the `jointVariables` table, for joint variables, or `frameVariables` table for frame variables.

Example: 'j1.Rz.q'

Data Types: `char` | `string`

unit — New unit for specified kinematic variable

`string` scalar or character vector

New unit for the variable `id` of the object `ks`. The unit must be valid and appropriate for the type of variable. Translational variables must be in units of length and rotational variables must be in units of angle.

Example: 'ft'

Data Types: `char` | `string`

See Also

`KinematicsSolver`

Introduced in R2019a

sm_lib

Open the Simscape Multibody block library

Syntax

```
sm_lib
```

Description

sm_lib opens the Simscape Multibody block library. Use this function to access Simscape Multibody blocks without having to wait for the Simulink and Simscape libraries to load.

Examples

Open the Simscape Multibody Block Library

Open the block library from the MATLAB command prompt

```
sm_lib
```

The Simscape Multibody block library opens in a new window.

See Also

smnew

Topics

“Start a Model from a Template”

Introduced in R2012a

smexportonshape

Export a CAD assembly model from Onshape cloud software

Syntax

```
multibodyDescriptionFile = smexportonshape(assemblyURL)
multibodyDescriptionFile = smexportonshape(assemblyURL,Name,Value)
```

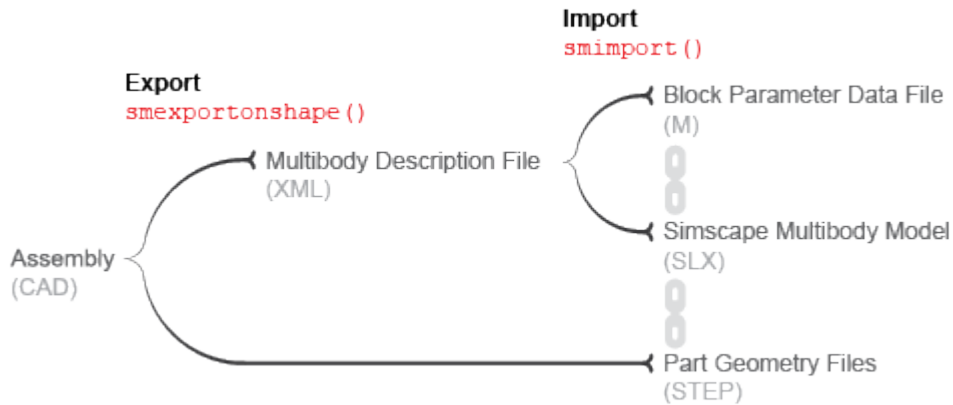
Description

`multibodyDescriptionFile = smexportonshape(assemblyURL)` generates the files that you need in order to import an Onshape® assembly model into the Simscape Multibody environment.

The `assemblyURL` argument is the web address of the Onshape assembly model to export. To obtain the web address, open the Onshape model, select the assembly tab, and copy the URL shown on your web browser.

The generated files include an XML multibody description file and a set of STEP files. The XML file identifies the bodies that comprise the model and defines their kinematic relationships. The STEP files provide the 3-D geometries of the bodies. By default, all files are stored in the current MATLAB folder.

The `multibodyDescriptionFile` output is the name of the XML multibody description file. You must use the `smimport` function with this name as an argument in order to import the Onshape model into the Simscape Multibody environment. The figure shows the export and import stages of the Onshape CAD translation workflow. The Simscape Multibody model and M data file are the product of the import stage.



Onshape CAD Translation Workflow

You must have an active Onshape account. The first time you use this function, you must give the Simscape Multibody Exporter access privileges to your Onshape account. The function uses these privileges strictly to access and export your Onshape models. Onshape software grants the function access via Javascript tokens that keep your login credentials and any user information secure and visible only to you.

To obtain the access tokens for your account, Simscape Multibody software requires you to log in to your Onshape account once per MATLAB session. A secure Onshape log-in page opens automatically on the first use of the `smexportonshape` function of a MATLAB session.

You can revoke the access privileges granted to the Simscape Multibody Exporter at any time. You must, however, restore those privileges if you want to export additional Onshape models. If you revoke the access privileges, then on your next use of `smexportonshape` an Onshape web page opens prompting you to accept or reject a request to restore those privileges.

```
multibodyDescriptionFile = smexportonshape(assemblyURL, Name, Value)
```

adds a name-value pair argument to specify the folder in which to save the XML and STEP files for the model.

Examples

Export a Humanoid Robot Model

Export an Onshape model of a humanoid robot assembly into the current MATLAB folder using the `smexportonshape` function. Then, import the generated model files into the Simscape Multibody environment using the `smimport` function.

- 1 Store the URL of the Onshape model in a MATLAB variable named `url`. The URL must always correspond to the Onshape assembly tab that you want to export.

```
url = 'https://cad.onshape.com/documents/5817806f96eae5105bfa5085/w/15ab3bfb58cacf
```

- 2 Export the humanoid robot model using the `smexportonshape` function. Store the name of the generated multibody description file in a variable named `xmlFile`. You may be prompted to log in to your Onshape account.

```
xmlFile = smexportonshape(url);
```

- 3 Import the model into the Simscape Multibody environment using the `smimport` function. Simscape Multibody software recreates the Onshape model as a block diagram.

```
smimport(xmlFile);
```

- 4 Update the block diagram. Mechanics Explorer opens with a static visualization of the model in its initial configuration—one matching the pose of the Onshape model at the time of export.



Note that the vertical axis of the robot (+Y) differs from the default vertical axis used in the Mechanics Explorer visualization pane (+Z). To orient the robot vertically, select **View > View convention > Y Up (XY Front)**. Select a standard view from the **View > Standard Views** menu to activate the new view convention.



Export a Humanoid Robot Model to a Specific Folder

Export an Onshape model of a humanoid robot assembly into a specific folder using the `smexportonshape` function.

- 1 Store the URL of the Onshape model in a MATLAB variable named `url` and the folder in which to save the model in a variable named `folder`. You must create the folder shown or replace that folder with one to which you have write privileges.

```
url = 'https://cad.onshape.com/documents/5817806f96eae5105bfa5085/w/15ab3bfb58cacf  
folder = 'C:\Documents\Export'
```

- 2 Export the humanoid robot model using the `smexportonshape` function. Use the `FolderPath` name-value pair argument to specify the export folder.

```
xmlFile = smexportonshape(url, 'FolderPath', folder);
```

Import the model into the Simscape Multibody environment as before using the `smimport` function. Update the diagram to visualize the imported model in Mechanics Explorer.

Input Arguments

assemblyURL — Web address of the Onshape assembly model to export

custom string or character vector

Web address of the Onshape assembly model to export. The function uses this address to access the assembly model and export it in a format compatible with Simscape Multibody software.

To obtain the URL, open the Onshape assembly model, select the assembly tab, and copy the URL from the web browser. The assembly model need not belong to your Onshape account if it is shared with you or made public.

Example: `https://cad.onshape.com/documents/3e07ba43d290f9b924933ce8/w/eb80497ae2e1a3af0c4ce16d/e/f7903984700a200643fb6141`

Data Types: `char` | `string`

Name-Value Arguments

Specify optional comma-separated pairs of `Name`, `Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1, Value1, . . . , NameN, ValueN`.

Example: `xmlFile = smexportonshape('https://cad.onshape.com/documents/5817806f96eae5105bfa5085/w/15ab3bfb58cacbf427d77ff3/e/181493813f84966648a8db1b', 'FolderPath', 'C:\Documents\Export');`

folderPath — Destination folder for exported files

custom string or character vector

Path of the folder in which to save the XML and STEP files generated during model export. The path can be absolute or relative. You must have write privileges to the folder in order to save the files there.

Example: `'C:\Documents\Models'`

Data Types: `char` | `string`

Output Arguments

multibodyDescriptionFile — Name of the XML multibody description file generated during export

character vector

Name of the XML multibody description file generated during Onshape CAD export. The name is derived from the OnShape assembly name. You use this name as an argument in the `smimport` function to import the model into the Simscape Multibody environment.

Data Types: char

See Also

`smimport`

Introduced in R2017a

smimport

Import a CAD, URDF, or Robotics System Toolbox model

Syntax

```
[H,dataFileName] = smimport(modelSource)
[H,dataFileName] = smimport(modelSource,Name,Value)
```

Description

`[H,dataFileName] = smimport(modelSource)` creates a Simscape Multibody model from a CAD, URDF, or Robotics System Toolbox model.

`modelSource` is the name of the file or object containing the model or, for CAD import, an intermediate representation of it. CAD models must be in XML files, URDF models in URDF files, and Robotics System Toolbox models in `RigidBodyTree` objects. XML files must conform to the Simscape Multibody XML schema and URDF files must conform to the URDF specification. A Robotics System Toolbox license is required to create `RigidBodyTree` objects.

`H` is the model handle and `dataFileName` is the name of the supporting file that, in imported CAD models, stores the numeric values of block parameters—in a structure array populated with MATLAB variables referenced in the blocks. The data file provides a mechanism to update the imported model if the CAD model changes. Models imported from URDF files or `RigidBodyTree` objects do not rely on data files for block parameters.

XML files can come from different sources. The `smexportonshape` function converts Onshape CAD models into XML files for import. The Simscape Multibody Link plug-in does the same for Autodesk Inventor®, PTC®, and SolidWorks® CAD models. The plug-in is free to download. For other CAD applications, and for multibody modeling tools of other types, the Simscape Multibody XML schema makes it possible to create a custom model export app.

CAD, URDF, and `RigidBodyTree` models all share the same components. These are (i) rigid bodies, also known as parts in CAD models and links in URDF models, and (ii)

kinematic constraints, packaged, in some cases, as joints. Rigid bodies import as Simulink subsystems with Solid and Rigid Transform blocks—the elemental components from which the attributes of shape, inertia, color, and placement derive. Constraints map into joint, gear, and other constraint blocks.

URDF models contain `<link>` elements which in turn contain `<joint>` elements. Likewise, `RigidBodyTree` objects contain `RigidBody` objects which in turn contain `Joint` objects. This hierarchy changes with import into Simscape Multibody. Joints become siblings to rigid bodies and feature not inside rigid body subsystems but alongside them. Joint limits are ignored but home positions persist as position targets specified in the appropriate joint blocks.

`[H,dataFileName] = smimport(modelSource,Name,Value)` creates a Simscape Multibody model from a CAD, URDF, or Robotics System Toolbox model with custom name or regenerates the data file of a previously imported CAD model. Most name-value pair arguments apply only to CAD models. Use `ImportMode` to regenerate parameter data files and `PriorDataFile` to catch inadvertent changes to the model, such as the removal of a part or a change in its name.

Input Arguments

modelSource — Name of model file or object to import

custom string or character vector

Name of the model to import. Use the name of the model file for CAD and URDF models and the name of the `rigidBodyTree` object for Robotics System Toolbox models. Include extension and path for model files. CAD model files must be in XML format and URDF model files in URDF format. XML files must conform to the Simscape Multibody XML schema. If file extension is missing, the model is assumed to be in XML format. If file path is missing, the file is assumed to be on the MATLAB path.

Example: `'robotto.xml'`

Data Types: `char` | `string`

Name-Value Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes.

You can specify several name and value pair arguments in any order as `Name1, Value1, . . . , NameN, ValueN`.

Example: `smimport('sm_robot', 'ModelName', 'robotto', 'DataFileName', 'robottos_data_file');`

ModelSimplification — Model topology simplification mode

`none` (default) | `bringJointsToTop` | `groupRigidBodies`

Model topology simplification mode to use during CAD import. Set `ModelSimplification` to:

- `bringJointsToTop` to group rigidly connected parts into subsystems and promote joints to the top level in the model hierarchy.
- `groupRigidBodies` to group rigidly connected parts into subsystems but leave joints in their original places in the model hierarchy.
- `None` to import the model as is, without simplification.

Joints brought to the top level of a model are renamed using generic names based on the joint type—for example, `Revolute_Joint1`. Subsystems of rigidly connected components that have been grouped together are given generic names based on the string `RigidSubsystem`—for example, `RigidSubsystem1`. This argument applies only to CAD import.

Example: `'bringJointsToTop'`

Data Types: `char` | `string`

ImportMode — Choice of model import or data file update function modes

`modelAndDataFile` (default) | `dataFile`

Option to generate a new model or update existing model data. Set `ImportMode` to `modelAndDataFile` to generate a new model and data file. Set `ImportMode` to `dataFile` to generate a new data file for a previously imported model. The function does not update the block diagram itself. If you do not specify `ImportMode`, the function runs in `modelAndDataFile` mode. This argument applies only to CAD import.

Example: `'dataFile'`

Data Types: `char` | `string`

ModelName — Name of the multibody model to generate

custom string or character vector

Name of the Simscape Multibody model to generate. The model is saved in SLX format. This argument is not valid when `ImportMode` is set to `dataFile`. If you do not specify `ModelName`, the model file is named after the multibody description file. If the multibody description file name is inconsistent with MATLAB naming rules, a slightly modified version is used instead.

Example: 'robotto'

Data Types: char | string

DataFileName — Name of the parameter data file to generate

custom string or character vector

Name of the supporting parameter data file. The data file is an M file with the block parameter values referenced in the imported Simscape Multibody model. If you do not specify `DataFileName`, the data file is named after the multibody description file. If the multibody description file name is inconsistent with MATLAB naming rules, a modified version is used instead. This argument applies only to CAD import.

Example: 'robottos_new_data'

Data Types: char | string

PriorDataFile — Name of the last used parameter data file

custom string or character vector

Name of the last parameter data file associated with a previously imported model. The prior data file helps to identify changes requiring special attention, such as new physical units, added and deleted components, and model topology changes. This argument is valid only when `ImportMode` is set to `dataFile`. This argument applies only to CAD import.

Example: 'robottos_original_data'

Data Types: char | string

VariableName — Name of the MATLAB structure provided in the parameter data file

custom string or character vector

Name of the MATLAB data structure provided in the parameter data file. This structure contains the numerical values of all block parameters in the Simscape Multibody model. If you specify neither `PriorDataFile` nor `VariableName`, the data structure is named `smiData`. If you specify `PriorDataFile` but not `VariableName`, the data structure name is derived from the prior data file. This argument applies only to CAD import.

Example: 'robottosData'

Data Types: char | string

Output Arguments

H — Model handle

double

Model handle returned as a double. Use the model handle to get or set model parameters, for example, using the `get_param` and `set_param` functions.

Data Types: double

dataFileName — Name of the parameter data file

character vector

Name of the parameter data file. The data file is an M file with the block parameter values referenced in the imported Simscape Multibody model. This output applies only to CAD import.

Data Types: char

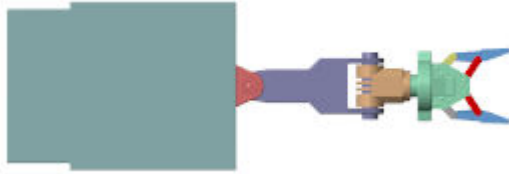
Examples

Import a CAD Model with Default Name

Import a CAD model of a robotic arm. The model has been exported in XML format using Simscape Multibody Link. The XML file is named `sm_robot.xml` and it is part of your Simscape Multibody installation.

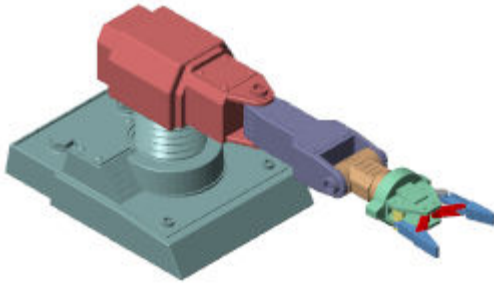
- 1 Import the model and store it in memory as `Untitled`. You can later change the name. As the model is in XML format, you can omit the extension in its name.

```
smimport('sm_robot');
```
- 2 Update the diagram to assemble the model and visualize it in Mechanics Explorer. You can update the diagram from the Simulink menu bar by selecting **Simulation > Update Diagram**.



CAD models often assume the y-axis as the vertical axis while Simscape Multibody models assume the z-axis. As a result, the imported model appears sideways on the screen. Use the **View convention** drop-down list in Mechanics Explorer to correct for the difference in convention.

- 3 In the Mechanics Explorer tool strip, set the **View Convention** parameter to **Y up (XY Front)** and select a standard viewpoint, such as **Isometric**, shown below.



Selecting a standard view activates the new view convention. The model rotates to assume its orientation in the original CAD model.

- 4 Simulate the model. The robot arm lacks a control system and swings erratically under the pull of gravity. Build on the model by adding control systems to simulate useful tasks. Add internal mechanics to joints to dampen their motions and specify joint targets to change the initial arm pose.

Import a CAD Model with Custom Name

Import the CAD model of the robotic arm and save it in the active folder with the name `robotto`. Name the parameter data file `robottos_data_file`.

```
smimport('sm_robot', 'ModelName', 'robotto', ...
        'DataFileName', 'robottos_data_file');
```

Update an Imported CAD Model

Regenerate the data file for the imported CAD model of the robotic arm. To avoid overwriting the original data file, name the new file `robottos_new_data_file`.

```
smimport('sm_robot','ImportMode','dataFile','DataFileName',...  
'robottos_new_data_file','PriorDataFile','robottos_data_file');
```

Point the imported model to the new data file and reinitialize the model workspace.

```
hws = get_param(bdroot,'modelworkspace');  
hws.DataSource = 'MATLAB File';  
hws.FileName = 'robotFileData';  
hws.reload;
```

To do the same using Model Explorer, select the model and, in the **Model Workspace** pane, update the **File Name** parameter. Click **Reinitialize from Source** button to apply the change.

Import a URDF Model

Import a URDF model of a humanoid robot. The model is named `sm_humanoid.urdf` and it is part of your Simscape Multibody installation.

- 1 Import the model and store it in memory as `Untitled`. You can later change the name. As the model is in URDF format, the file extension is required.

```
smimport('sm_humanoid.urdf');
```

- 2 Update the diagram to visualize the model in its initial configuration using Mechanics Explorer. You can update the diagram from the Simulink menu bar by selecting **Simulation > Update Diagram**.



- 3 Simulate the model. As with the robot arm, the humanoid robot lacks a control system and swings erratically under the pull of gravity. The shoulder line serves as the root body in the URDF model, and so it is fixed to the world frame after import.

Try modifying the model—for example, by removing the rigid connection between the shoulder line and the world frame and by adding control subsystems at the various joints. See the Humanoid Robot featured example for a version of the model with basic motion controls. You can open the example by entering `sm_import_humanoid_urdf` at the MATLAB command prompt.



Import a RigidBodyTree Object

Import a RigidBodyTree object for an LBR iiwa serial manipulator (manufactured by KUKA Robotics). The model features in the Robotics System Toolbox installation as a URDF model named `iiwa14.urdf`.

- 1 Convert the URDF model into a RigidBodyTree object.

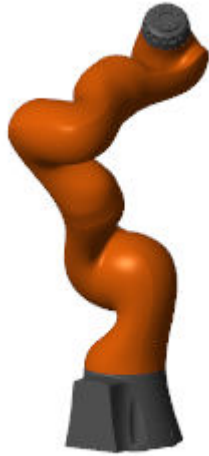
```
iiwaRBT = importrobot('iiwa14.urdf');
```

A Robotics System Toolbox license is required to run `importrobot`.

- 2 Import the `iiwaRST` object into Simscape Multibody.

```
iiwaSM = smimport(iiwaRBT);
```

- 3 Update the diagram to visualize the model in its initial configuration using Mechanics Explorer. You can update the diagram from the Simulink menu bar by selecting **Simulation > Update Diagram**.



- 4 Simulate the model. As with the robot arm, the `iiwa14` model lacks a control system and swings erratically under the pull of gravity. Build on the model by adding control systems to simulate useful tasks. Add internal mechanics to joints to dampen their motions and specify joint targets to change the initial arm pose.

See Also

`smexportonshape`

Introduced in R2012b

smnew

Open Simscape Multibody model template and block library

Syntax

```
smnew  
smnew(modelName)  
smnew(modelName, solverType)
```

Description

smnew creates a model from the Simscape Multibody template. The template includes several commonly used blocks and an automatic variable-step solver selection. Simscape data logging is enabled by default, with the data history limited to 10,000 data points.

smnew(modelName) adds an option to name the model built from the template.

smnew(modelName, solverType) adds an option to specify the Simulink solver to use with the model.

Examples

Create a Simscape Multibody Model

Create a model from the Simscape Multibody template at the MATLAB command prompt:

```
smnew
```

The model name is untitled and the solver type is auto.

Create a Simscape Multibody Model with the Specified Name

Create a model named robotto from the Simscape Multibody template:

```
smnew('robotto')
```

The solver type is auto.

Create a Simscape Multibody Model with the Specified Name and Solver Type

Create a model named `robotto` with the Simulink solver type set to `ode15s` from the Simscape Multibody template:

```
smnew('robotto', 'ode15s')
```

Input Arguments

modelName — Name of the model to create from the template

untitled (default) | String or character vector with the model name

Name of the model to create from the template. The name must conform to the MATLAB naming rules. Do not include the file path in the model name. If the specified character vector is invalid, the model is named `untitled`.

Example: `'robotto'`

Data Types: `char` | `string`

solverType — Simulink solver to use for simulation

auto (default) | String or character vector with the solver name

Solver to use for simulation. The solver type must be a valid Simulink solver, such as `ode45`, or `ode15s`. For best performance, consider using a variable-step solver unless you have a specific need for fixed-step simulation.

Example: `'ode15s'`

Data Types: `char` | `string`

See Also

`sm_lib`

Topics

“Start a Model from a Template”

Introduced in R2012a

smwritevideo

Configure and create multibody animation videos

Syntax

```
smwritevideo(modelIdentifier,videoName)
smwritevideo(modelIdentifier,videoName,Name,Value)
```

Description

`smwritevideo(modelIdentifier,videoName)` creates an animated video from the visualization results of a multibody simulation. `modelIdentifier` is the source model name or handle. `videoName` is the generated video file name and path. You can open the video file with any compatible media player.

The video properties are those specified in the **Video Creator** interface the moment you run the function. If the Video Creator parameters are in their default settings, the video properties are set to those defaults.

Before running `smwritevideo`, you must simulate the model. In addition, the model visualization results must open in a Mechanics Explorer window. If you have previously disabled model visualization, reenable it before continuing. To do this, see “Enable Mechanics Explorer”.

If the model visualization pane is split into tiles, the function captures only the active tile. A colored outline identifies the active tile. Select the desired tile before creating a video.

`smwritevideo(modelIdentifier,videoName,Name,Value)` adds options for specifying the video properties. Use the `Name,Value` pair arguments to change the video file format, frame refresh rate, frame width and height, and playback speed ratio. Unused arguments are set to the latest settings specified in the **Video Creator** tool.

Examples

Create Video of Flapping Wing Model

Create a video named `flapping_wing_video` from the simulation results of the `sm_cam_flapping_wing` featured example. Use the video settings currently specified in the Video Creator tool.

- 1 Open the flapping wing featured example.

```
sm_cam_flapping_wing
```

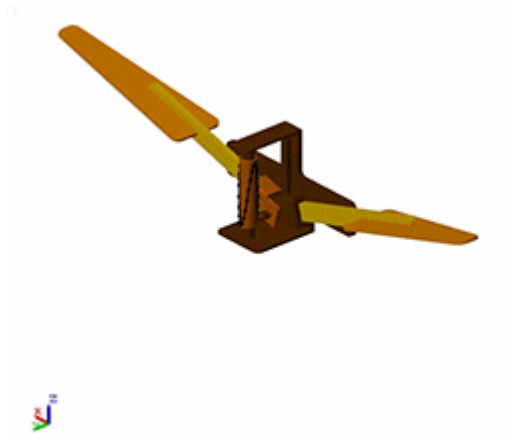
- 2 Simulate the model.

```
sim('sm_cam_flapping_wing')
```

- 3 Create a video of the simulation results.

```
smwritevideo('sm_cam_flapping_wing','flapping_wing_video');
```

The function saves the video as `flapping_wing_video` in the current MATLAB folder. The video file format is that specified in the Video Creator tool. Open the video using your media player of choice.



Create Video of Double Wishbone Suspension Model

Create a video named `wishbone_suspension_video` from the simulation results of the `sm_double_wishbone_suspension` featured example. Change the video settings as shown in the table.

Property	Argument	Setting
Playback Speed Ratio	PlaybackSpeedRatio	2.0
Frame Rate (FPS)	FrameRate	60
Video Format	VideoFormat	uncompressed avi

- 1 Open the wishbone suspension featured example.

```
sm_double_wishbone_suspension
```

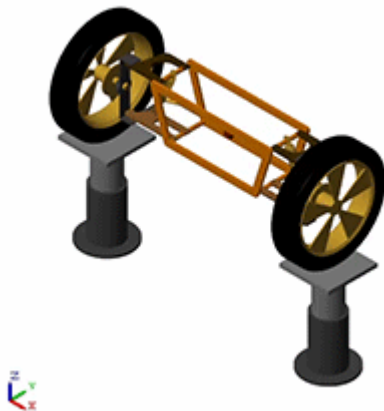
- 2 Simulate the model.

```
sim('sm_double_wishbone_suspension')
```

- 3 Create a video of the simulation results.

```
smwritevideo('sm_double_wishbone_suspension', 'wishbone_suspension_video', ...
'PlaybackSpeedRatio', 2.0, 'FrameRate', 60, 'VideoFormat', 'uncompressed avi');
```

The function saves the video as `wishbone_suspension_video.avi` in the current MATLAB folder. Open the video using your media player of choice. The video plays at twice the original speed seen in Mechanics Explorer.



Input Arguments

modelIdentifier — Name or handle of the source model

Character vector with the model name or handle

Name or handle of the source model, specified as a MATLAB string. You must simulate the specified model before using this function. The model visualization window must be open in order for the function to create a video.

Example: 'sm_cam_flapping_wing'

Data Types: string

videoName — Name and path of the video file

Character vector with the desired video file name

Name and full or relative path of the video file, specified as a string. In the absence of a file path, the function saves the video file in the current MATLAB folder. The file format is determined from the video settings specified using the Video Creator tool or the VideoFormat Name, Value pair argument.

Example: 'flapping_wing_video'

Data Types: string

Name-Value Pair Arguments

Specify optional comma-separated pairs of Name, Value arguments. Name is the argument name and Value is the corresponding value. Name must appear inside quotes. You can specify several name and value pair arguments in any order as Name1, Value1, . . . , NameN, ValueN.

Example: 'PlaybackSpeedRatio', 2.0

PlaybackSpeedRatio — Playback speed relative to real time

1.0 (default)

Video playback speed relative to real time, specified as a positive scalar. Increase this factor for faster playback speeds. For example, a value of 2.0 doubles the video playback speed relative to the base playback speed used in Mechanics Explorer.

Data Types: single | double | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64

FrameRate — Number of video frames per second of playback time

30 (default)

Number of video frames per second of playback time, specified as a positive scalar. Increase this factor for smoother playback but larger video files. Small numbers may lead to choppy videos.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

VideoFormat — Video file format

`motion jpeg avi (default)` | `archival` | `motion jpeg 2000` | `mpeg-4` | `uncompressed avi`

File format to save the video in, specified as a string. Select from a list of compressed and uncompressed formats with varying quality levels and storage space requirements. Use the default format of `uncompressed jpeg avi` if file size is a concern. Use `uncompressed avi` if top video quality is a priority. The `mpeg-4` format is not supported in Linux systems.

Data Types: `string`

FrameSize — Video frame width and height

`auto (default)` | `custom width and height`

Width (W) and height (H) of the video contents, specified in pixel units as the two-element row vector [W H]. The vector elements must be positive integers. Use the default setting of `auto` to obtain the video dimensions from the Mechanics Explorer visualization pane size.

Example: `[800 800]`

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

See Also

Video Creator

Introduced in R2016b

solve

Package: `simscape.multibody`

Run kinematic analysis for `KinematicsSolver` object

Syntax

```
[outputs,statusFlag,targetFlags,targets] = solve(ks,targets,  
initialGuesses)
```

Description

`[outputs,statusFlag,targetFlags,targets] = solve(ks,targets,initialGuesses)` solves, or attempts to solve, the kinematic problem posed in the `KinematicsSolver` object `ks`. The unknowns are the variables assigned as outputs in `ks`. Their solution hinges on the initial position constraints of the model and on the position targets of the object. When multiple solutions exist, position guesses bias the solver toward one over the others.

Input Arguments

ks — KinematicsSolver object for which to run the analysis

`KinematicsSolver` object

Name of the `KinematicsSolver` object for which to run the analysis. The object is a kinematic representation of the model from which it derives. It contains the variables upon which the analysis depends.

Example: `'fourBarKS'`

Data Types: `char` | `string`

targets — Desired values of target variables

vector of doubles

Desired values of the target variables of ks. Specify the values in the order of the variables as shown in the `targetVariables` table. The values are interpreted in the units listed in the table. If no target variables exist, enter an empty vector. If neither target nor initial guess variables exist, enter an empty vector or omit the argument altogether.

Example: '[0 45 30]'

Data Types: double

initialGuesses — Values of initial guess variables

vector of doubles

Values of the initial guess variables of ks. Specify the values in the order of the variables as shown in the `initialGuessVariables` table. The values are interpreted in the units listed in that table. If no initial guess variables exist, enter an empty vector or omit the argument altogether.

Example: '[10 25]'

Data Types: double

Output Arguments

outputs — Computed values of output variables

vector of doubles

Computed values of the output variables. The variables show in the order of their ranking in the `outputVariables` table. They are each in the units listed there.

The solution may not satisfy all position targets or even all model constraints. Check the `statusFlag` argument for an overview of the issues encountered in the solution.

Data Types: double

statusFlag — Flag with overall status of solution

scalar

Flag with the overall status of the analysis results. A positive flag means that all target variables and model constraints have been satisfied. A negative flag means that one or more have not. See the `targetFlags` argument to check which of the targets the solver may have missed. See the `targets` argument to see the actual values reached for each.

- **2:**

All model constraints and target variables are satisfied.

- **1:**

All model constraints and target variables are satisfied. A joint or belt-cable network may be in a singular configuration, however—due to gimbal lock in the former or a zero belt-cable length between pulleys in the latter.

- **-1:**

All model constraints are satisfied, but one or more target variables are not.

- **-2:**

All model constraints are satisfied, but one or more target variables are not. At least one joint or belt-cable network is in a singular configuration—due to gimbal lock in the former or a zero belt-cable length between pulleys in the latter.

- **-3:**

One or more model constraints cannot be satisfied. The solution is invalid. The output variables are NaN.

Data Types: double

targetFlags — Logical flags with status of each target variable

logical vector

Logical flags with the status of each target variable. A logical 1 indicates that a target has been satisfied. A logical 0 indicates that it has been missed. The flags show in the order given in the `targetVariables` table of the object. The vector is empty in kinematic problems without target variables.

Data Types: double

targets — Computed values of target variables

numerical vector

Computed values of the target variables of `ks`. These are the same target variables specified in the input arguments. The variables show in the order of their ranking in the `targetVariables` table. They are each in the units listed there.

Data Types: double

See Also

KinematicsSolver

Introduced in R2019a

targetVariables

Package: `simscape.multibody`

List kinematic variables assigned as targets

Syntax

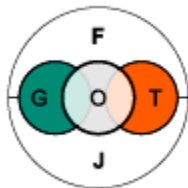
`targetVariables(ks)`

Description

`targetVariables(ks)` lists the kinematic variables in the `KinematicsSolver` object `ks` currently assigned as targets. Both joint and frame variables can serve in this role. Those that do serve as constraints to apply during analysis. Their values must be later defined in order to run the analysis. The solver, when called, searches for a solution compatible with the targeted joint and frame variables.

The output is a table with the target variables in rows. Each row gives the ID of a variable, the type and block path of the joint to which it belongs if a joint variable, the base and follower frames from which it spawns if a frame variable, and the unit for its numerical value. The variables rank in the order added.

The figure shows the variables of a general `KinematicsSolver` object. Target (**T**), guess (**G**), and output (**O**) variables can be joint variables (**J**) or frame variables (**F**). The same variable can serve as target and output, but if it serves as target, it cannot double as guess. Assigning a guess variable as target clears it as guess.



Input Arguments

ks — Kinematic representation of multibody model to analyze

`KinematicsSolver` object

Kinematic representation of the multibody model on which to run the kinematic analysis. The representation is a `KinematicsSolver` object. It has joint variables and, when complete, frame variables, both playing roles as target variables, guess variables, or output variables.

Example: `'sm_four_bar'`

Data Types: `char` | `string`

See Also

`KinematicsSolver` | `addTargetVariables` | `clearTargetVariables` | `removeTargetVariables`

Introduced in R2019a

First-Generation Conversion

Convert a First-Generation Model

The Simscape Multibody environment spans two product generations. The first builds on a modeling paradigm introduced at the inception of the product and actively developed up to software version R2011b. The second builds on a revamped modeling paradigm introduced in software version R2012a and actively developed to this day. The terms *first-generation* and *second-generation* are used here to distinguish between the two.

Second-generation technology is more than a simple alternative to its first-generation counterpart. It is its intended replacement. For this reason, it is recommended that all new models that you create comprise only second-generation blocks. In addition, in order to benefit from the latest software features, and to ensure continued support for those that you currently depend on, it is recommended that any first-generation models you may still have be converted to second-generation models.

The following sections summarize the key similarities and differences between the two software generations. The vast majority of first-generation features have second-generation analogues. The sparse few that do not, massless joint connectors and velocity drivers among them, have relatively simple workarounds. Those workarounds are discussed in some detail in those cases that might not at first sight seem obvious.

Frames and Signals

The constructs known as coordinate systems in first-generation models are in second-generation models referred to as frames. Regardless of their names, they serve an important role in your modeling workflow: they encode in their origins and axes all the position and orientation data in a model. You connect bodies through frames, constrain their motions through frames, even apply forces and sense motion through frames.

In a manner similar to first-generation blocks, second-generation blocks use frame ports to identify their respective frames. The connection rules for these ports remain much the same. A direct connection line between frame ports establishes an identity relationship between them, making the corresponding frames coincident in space. Placing a Rigid Transform block between any two frames enables you to offset those frames in space—by applying a translational offset, a rotational offset, or a mixed translational-rotational offset between them.

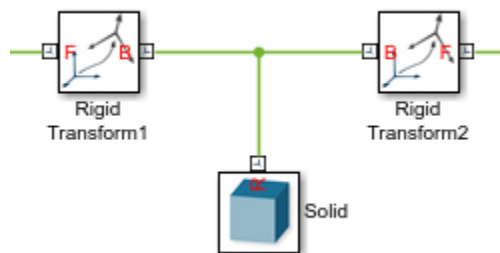
The input and output ports of blocks are now compatible only with Simscape physical signals. In first-generation blocks, the same types of ports were compatible only with Simulink signals. The switch to physical signal inputs and outputs allows for tighter

integration with other Simscape physical domains. You can now directly connect the output of a Simscape Fluids subsystem, for example, to the input of a Simscape Multibody block.

Simulink signals remain a practical means of specifying model inputs and of analyzing model outputs. You can still interface Simscape Multibody blocks with Simulink blocks. You must, however, convert between physical signals and Simulink signals. You do this using the appropriate converter block—PS-Simulink Converter or Simulink-PS Converter.

Rigid Bodies

Bodies are the source of all geometries and nearly all inertias in a model. In a first-generation model, a Body block with inertia, geometry, and frames—originally referred to as coordinate systems, or CSs—represents each body. In the second-generation environment, the Solid block replaces the Body block. Rigid Transform blocks often complement the Solid block, these serving to add frames suited for connection, e.g., to joints and constraints. The figure shows an example.



The table provides a comparison of features relevant to the task of modeling bodies. A green icon denotes a feature that is maximally supported. A yellow icon denotes a feature that is partially supported. A red icon denotes a feature that is not at all supported.

Feature	First Generation	Second Generation
Native Solid Geometries	●	●
Imported solid geometries	●	●
Manual inertia inputs	●	●

Feature	First Generation	Second Generation
Derived inertia parameters	●	●
Variable inertia parameters	●	●
Body visualization	●	●
Interactive frame creation	●	●

Geometry, Inertia, and Frames

The Solid block provides a selection of native geometries parameterized in terms of key dimensions, such as Sphere (parameterized in terms of radius) and Cylinder (parameterized in terms of radius and length). These geometries replace the more primitive convex hulls and inertia ellipsoids used in the Body block. You can also import detailed geometries in STL format (both generations) and STEP format (second generation only).

The Solid block provides also other features not found in the Body block. You can specify the mass properties of the solid explicitly or, for ease of modeling, you can have them derived from density and geometry. You can track the current state of the solid and the placement of its frames using a visualization pane embedded in the block dialog box. And you can add new frames to the solid and place them using geometry features such as vertices, edges, and planes as guides.

A second-generation library of variable-mass blocks enables you to model bodies whose dimensions and mass properties can evolved over time. These blocks, which include General Variable Mass, Variable Brick Solid, Variable Cylindrical Solid, and Variable Spherical Solid, replace the first-generation functionality provided by the combination of the Body block with the Variable Mass & Inertia Actuator block.

A Note on Frame Definitions

Unlike the coordinate systems of first-generation models, the frames of second-generation models are always locally defined. The position and orientation of a frame provided by a Rigid Transform block is always specified relative to a local frame (the *base* frame of the block). This approach is in contrast to that taken in first-generation models. There, the coordinate systems of a Body block are variously defined relative to local coordinate systems (base or follower) or to the world coordinate system.

The requirement that all frames in a model be locally defined ensures that all bodies in that model are reusable. In a second-generation model, you can generally connect a body elsewhere in the model, or even in a different model, without first having to redefine its connection frames. You can in principle create a custom library of body blocks and use them in your models without worrying about which frame a particular connection frame is defined against.

Multibody Assembly

Joints remain the primary means of connecting bodies in a model. Specialized kinematic constraints, such as those characteristic of gears, provide a means to recreate motions not possible through joints alone. Most joint blocks in the first-generation library have second-generation counterpart, and these are built on the same concept of joint primitive found also in first-generation joint blocks. Most constraint blocks have second-generation counterparts also, with the exception of Velocity Driver, although the functionality of this block is easily reproduced with joint blocks.

Disassembled joint blocks and massless connector blocks are among the joint blocks no longer provided in the second-generation library. Disassembled joints are those few whose rotational axes were freely and automatically aligned during the model assembly stage. Such joints provided a convenience in models with loop topologies, where the placement of one joint is fixed the placement of the others upon closure of the loop. Massless connector blocks are pairs of joints connected to each other without a mass element in between. You can approximate a massless joint connector in a second-generation model using other available blocks.

The table provides a comparison of features relevant to the task of assembling bodies with joints and constraints. A green icon denotes a feature that is maximally supported. A yellow icon denotes a feature that is partially supported. A red icon denotes a feature that is not at all supported.

Feature	First Generation	Second Generation
Angle and Distance Constraints	●	●
Point-on-curve constraint	●	●
Gear constraints	●	●

Feature	First Generation	Second Generation
Velocity constraints	●	●
Massless joint connectors	●	●
Disassembled joints	●	●
Joint initial state targets	●	●

For more information, see:

- “Modeling Joint Connections”
- “Modeling Gear Constraints”

Reproducing Massless Joint Connectors

You can approximate a massless joint connector by means of rigid transforms and Solid or Inertia blocks. To do this, connect the joints you wish to use in the massless connector—for example, two Revolute Joint blocks—with a frame connection line. Then, add a Rigid Transform block and use it to specify the translational offset between the joints. Finally, anywhere between the joint blocks, add a Solid or Inertia block and set its mass to a very small value. The inertia of this block ensures that finite torques cannot produce infinite accelerations. The small value of the mass ensures that its effect on the model dynamics is negligible—and that the connector behaves as approximately massless.

System Dynamics

The forces and torques specified in a model determine to a great extent the dynamics exhibited during simulation. You can specify those forces and torques directly: as actuation inputs to joints and the use of specialized forces and torques such as those provided by External Force and Torque and Inverse Square Law Force blocks. You can also specify those forces and torques indirectly: in terms of the joint motions that they must, on the aggregate, produce, and by the introduction of a gravitational acceleration constant (a mere proxy for the gravitational force itself).

Both generations support the use of joint force and torque actuation as well as of joint motion actuation, with some notable differences. Whereas in a first-generation model the actuation inputs derive from separate Joint Actuator blocks, in a second-generation model they are specified directly through the joint blocks themselves. The actuation inputs are

in the form of Simulink signals in a first-generation model, but in the form of Simscape physical signals in a second-generation model.

The table provides a comparison of features relevant to the tasks of applying and sensing forces, torques, and motion variables in a model. A green icon denotes a feature that is maximally supported. A yellow icon denotes a feature that is partially supported. A red icon denotes a feature that is not at all supported.

Feature	First Generation	Second Generation
Constant uniform gravity	●	●
Variable uniform gravity	●	●
Gravitational fields	●	●
Force and torque actuation	●	●
Force and torque sensing	●	●
Motion actuation	●	●
Motion sensing	●	●
Joint springs and dampers	●	●

Reproducing the Effects of Velocity Drivers

In a second-generation model, all motion inputs are specified directly through joints—the components from which bodies derive their degrees of freedom. If you want to constrain the relative velocity of a body, you must configure the appropriate joint to accept motion signals as actuation inputs. The motion signals must by definition provide the time-variable position of the body. As such, if your known variable is velocity, you must first integrate velocity to obtain the final position input.

Model Visualization

















Visualization provides a means to explore and analyze—in a qualitative sense—the results of a multibody simulation. In a second-generation model, visualization is handled by **Mechanics Explorer**, the replacement to the visualization utility of the first-generation environment. By default, Mechanics Explorer opens automatically when you first update or simulate a model. You can manipulate the visualization contents using controls familiar

from the first-generation utility—such as *pan*, *rotate*, *zoom*—and others new in Mechanics Explorer, such as *roll*.

The visualization is static on model update and dynamic during simulation. You can replay a dynamic visualization, more aptly referred to as an *animation*, without having to simulate the model again—a requirement of first-generation models. You can selectively hide bodies, for example, to more clearly visualize others, and navigate to the block corresponding to a selected component. So that you can share the results of your simulations, a **Video Creator** tool enables you to record animations and save them in formats such as MPEG-4 and uncompressed AVI.

Mechanics Explorer supports also dynamic visualization cameras—those that can move during the course of simulation. You can constrain the camera trajectories via keyframes, each a point in time at which you specify the desired camera position and orientation. You can also constrain the camera trajectory by attaching it to and aiming it at frames that you select. Dynamic cameras are useful when visualizing models of moving vehicles, such as that shown in the featured example *Configuring Dynamic Cameras - Vehicle Slalom*.

The table provides a comparison of features relevant to the task of visualizing a multibody model. A green icon denotes a feature that is maximally supported. A yellow icon denotes a feature that is partially supported. A red icon denotes a feature that is not at all supported.

Feature	First Generation	Second Generation
Static visualization		
Dynamic visualization		
3-D model exploration		
Animation replay		
Visualization filtering		
Video Creation		
Dynamic Cameras		
"Go to Block" Navigation		

For more information, see: “Selective Model Visualization”

- “Selective Model Visualization”
- “Create a Model Animation Video”
- “Go to a Block from Mechanics Explorer”
- “Visualization Cameras”

Simulation and Analysis

Machine dimensionality and analysis mode are no longer required parameters in a model. In the first-generation environment, both were specified explicitly via the Machine Environment block. In the second-generation environment, dimensionality is determined automatically from the relative placement of joints and constraints. The types of analysis allowed during simulation are a direct consequence of the actuation inputs specified at, and the sensing outputs provided at, joints.

The table provides a comparison of features relevant to the task of analyzing model parameters and simulation data. A green icon denotes a feature that is maximally supported. A yellow icon denotes a feature that is partially supported. A red icon denotes a feature that is not at all supported.

Feature	First Generation	Second Generation
2-D and 3-D simulation	●	●
Forward and inverse dynamics	●	●
Trimming and linearization	●	●
Simscape data logging	●	●
Simscape variable viewer	●	●
Simscape statistics viewer	●	●

Reproducing the First-Generation Analysis Modes

The (first-generation) Machine Environment block provides a selection of four analysis modes—Forward dynamics, Inverse dynamics, Kinematics, Trimming. These terms are used in the sense outlined here:

- **Forward dynamics** — Compute the motions of bodies (their positions and velocities) given some force and torque inputs and some initial positions and velocities.
- **Inverse dynamics** — Compute the forces and torques acting on bodies arranged in an open-loop structure given some position and velocity inputs and some initial positions and velocities.
- **Kinematics** — Compute the forces and torques acting on bodies arranged in a closed-loop structure given some position and velocity inputs and some initial positions and velocities. This mode is the merely the closed-loop analogue of **Inverse dynamics**.
- **Trimming** — Configure a model for trimming via the Simulink `trim` function or the more powerful Control Design Toolbox `findop` function. Trimming is defined as the discovery of steady-state operating points. Models are often linearized about such points, for example, by means of the Simulink `linmod` function.

You can trim and linearize a second-generation model using the appropriate Simulink tools. No special Simscape Multibody setting is required to perform either task. The size of perturbations applied in linearization tasks is specified through the Mechanism Configuration block.

Third-Party Model Import

CAD import remains an important modeling workflow in the second-generation environment. The `smimport` function replaces the `mech_import` function as the means of importing a CAD assembly model. URDF model import is now also possible by means of the same function, as is Onshape model export, by means of the `smexportonshape` function.

The table provides a comparison of features relevant to the task of importing a third-party multibody model into the Simscape Multibody environment. A green icon denotes a feature that is maximally supported. A yellow icon denotes a feature that is partially supported. A red icon denotes a feature that is not at all supported.

Feature	First Generation	Second Generation
CAD assembly import	●	●
CAD assembly update	●	●
Imported-model simplification	●	●

Feature	First Generation	Second Generation
URDF assembly import	●	●
Onshape assembly export	●	●

For more information, see:

- “CAD Translation”
- “URDF Import”
- “Onshape Import”

