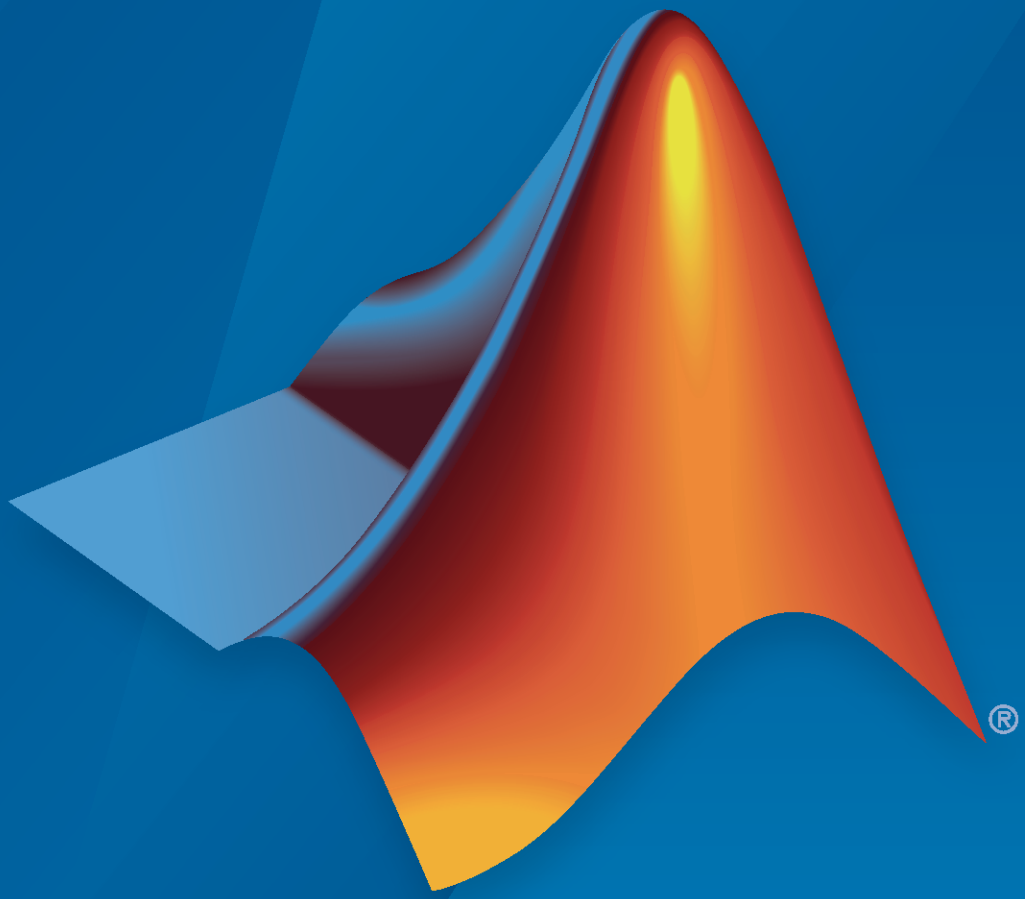
Simscape™ Multibody™

Reference



MATLAB® & SIMULINK®

R2023a

MathWorks®

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

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 *SimMechanics™ Reference*)
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)
September 2019	Online only	Revised for Version 7.0 (Release R2019b)
March 2020	Online only	Revised for Version 7.1 (Release R2020a)
September 2020	Online only	Revised for Version 7.2 (Release R2020b)
March 2021	Online only	Revised for Version 7.3 (Release R2021a)
September 2021	Online only	Revised for Version 7.4 (Release R2021b)
March 2022	Online only	Revised for Version 7.5 (Release R2022a)
September 2022	Online only	Revised for Version 7.6 (Release R2022b)
March 2023	Online only	Revised for Version 7.7 (Release R2023a)

Contents

Blocks

Angle Constraint

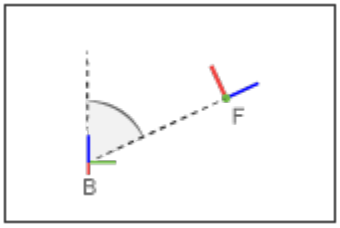
Fixed angle between two frame Z axes

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

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Distance Constraint | Point on Curve Constraint | Common Gear Constraint | Bevel Gear Constraint | Rack and Pinion Constraint | “Deep Learning Toolbox” | “Reinforcement Learning Toolbox” | “Global Optimization Toolbox” | `ga` (Global Optimization Toolbox) | `rlDDPGAgent` (Reinforcement Learning Toolbox)

Bearing Joint

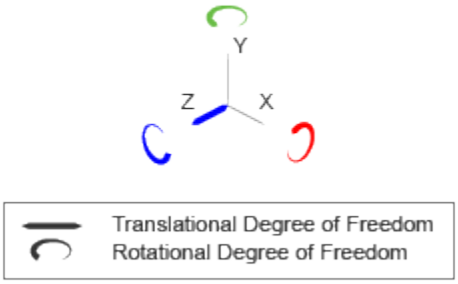
Joint with one prismatic and three revolute primitives

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.

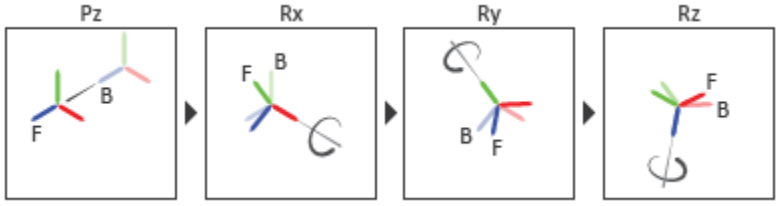
At each time step during the simulation, the joint block applies the sequence of time-varying frame transformations in this order:

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.

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.

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.

Mode Configuration

Specify the mode of the joint. The joint mode can be normal or disengaged throughout the simulation, or you can provide an input signal to change the mode during the simulation.

Mode

Select one of the following options to specify the mode of the joint. The default setting is `Normal`.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.
`Provided by Input`	This option exposes the **mode** port that you can connect to an input signal to change the joint mode during the simulation. The joint mode is normal when the input signal is `0` and disengaged when the input signal is `-1`. The joint mode can be changed many times during the simulation.

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 primitives provide the following actuation ports:

- tx, ty, tz — Actuation torques acting on the X, Y, and Z revolute joint primitives
- qx, qy, qz — Desired rotations of the X, Y, and Z revolute joint primitives

Sensing Ports

The prismatic 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
- fllz — Force due to contact with the lower limit of the Z prismatic joint primitive
- fulz — Force due to contact with the upper limit of the Z prismatic joint primitive

The revolute primitives provide the following sensing ports:

- qx, qy, qz — Angular positions of the X, Y, and Z revolute joint primitives
- wx, wy, wz — Angular velocities of the X, Y, and Z revolute joint primitives
- bx, by, bz — Angular accelerations of the X, Y, and Z revolute joint primitives
- tx, ty, tz — Actuation torques acting on the X, Y, and Z revolute joint primitives
- tllx, tlly, tllz — Torques due to contact with the lower limits of the X, Y, and Z revolute joint primitives
- tulx, tuly, tulz — Torques due to contact with the upper limits of the X, Y, and Z 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

Mode Port

Mode configuration provides the following port:

- mode — Value of the mode of the joint. If the input is equal to `0`, the joint behaves normally. If the input is equal to `-1`, the joint behaves as disengaged.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Prismatic Joint | Revolute Joint

Topics

“Actuating and Sensing with Physical Signals”
“Motion Sensing”
“Rotational Measurements”
“Translational Measurements”

Belt-Cable End

Tip of the cord of a pulley system

Libraries:
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.

Version History

Introduced in R2018a

Extended Capabilities

C/C++ Code Generation

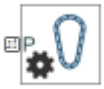
Generate C and C++ code using Simulink® Coder™.

See Also

Pulley | Belt-Cable Spool | Belt-Cable Properties

Belt-Cable Properties

General characteristics of the cord of a pulley system

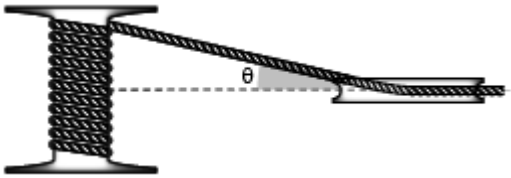


Libraries:
Simscape / Multibody / Belts and Cables

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.

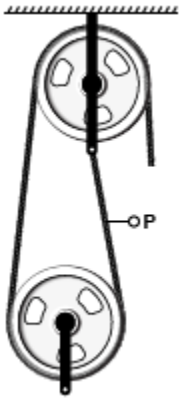
By default, the cord can enter and exit a drum—pulley or spool—at an angle to its center plane. This angle can vary during simulation—for example, due to translation of the drum on a prismatic joint. While the contact point is always in the center plane of the drum, the drum can move when mounted on 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`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered solid and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`

2 **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Version History

Introduced in R2018a

Extended Capabilities

C/C++ Code Generation

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

See Also

Belt-Cable End | Belt-Cable Spool | Pulley

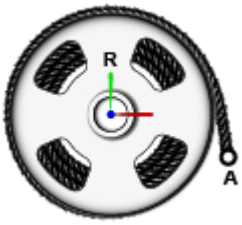
Belt-Cable Spool

Source and sink of cord in a pulley system

Libraries:
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.

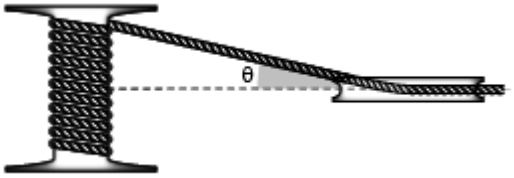
The degrees of freedom of the spool are a function of the joint by which the spool connects to other components. It is common for a revolute joint to provide those degrees of freedom; they reduce in this case to rotation about a single axis (that of the spool). Actuation inputs, specified directly through 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 Cylindrical Solid or Inertia block. Consider the Cylindrical 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 -π/2 to +π/2. This range is open. The measured value can vary between -π/2 and +π/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.

Version History

Introduced in R2018a

Extended Capabilities

C/C++ Code Generation

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

See Also

Pulley | Belt-Cable End | Belt-Cable Properties

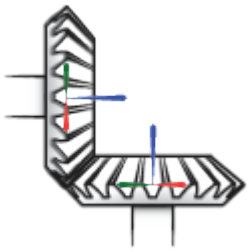
Bevel Gear Constraint

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

Libraries:
Simscape / Multibody / Gears and Couplings / Gears

Description

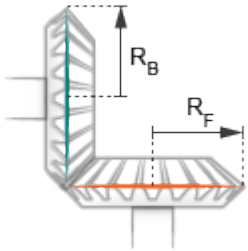
The Bevel Gear Constraint block represents a kinematic constraint between two gear bodies with intersecting rotation axes held at a specified angle. The base and follower frame ports identify the connection frames on the gear bodies. The gear rotation axes coincide with the connection frame *z*-axes. 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.

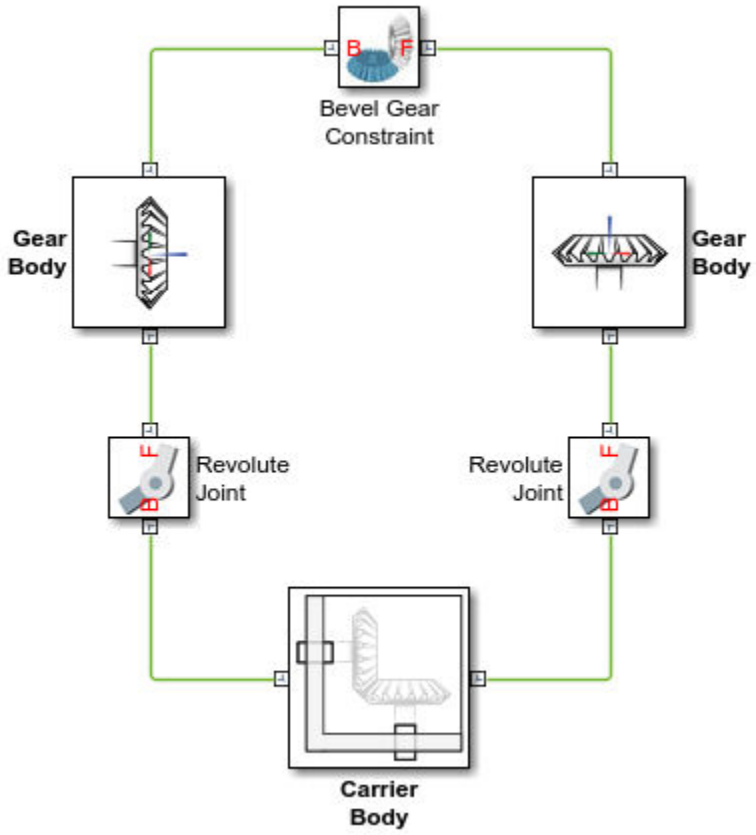
Gear Geometry

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

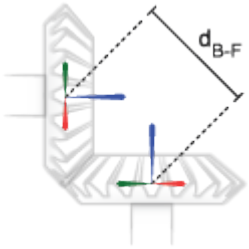
The block imposes special restrictions on the relative positions and orientations of the gear connection frames. The restrictions ensure that the gears assemble only at distances and angles suitable for meshing. The block enforces the restrictions during model assembly, when it first attempts to place 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 $d_{\text{B-F}}$, follows from the law of cosines:

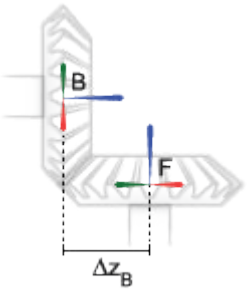
  $$d_{B-F} = \sqrt{R_B^2 + R_F^2 - 2R_B R_F \cos(\pi - \theta)},$$

  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.



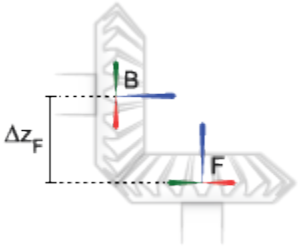
- The distance between the base and follower frame origins along the *z*-axis of the base frame, denoted Δz_B, must be equal to:

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



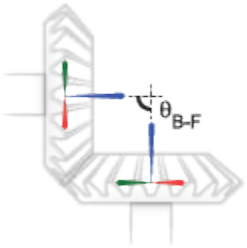
- The distance between the base and follower frame origins along the *z*-axis of the follower frame, denoted Δz_F, must be equal to:

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



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 $\theta_{B\text{-}F}$ 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 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`.

Version History

Introduced in R2013b

Extended Capabilities

C/C++ Code Generation

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

See Also

Rack and Pinion Constraint | Common Gear Constraint | Brick Solid | Cylindrical Solid | Ellipsoidal Solid | Extruded Solid | Revolved Solid | Spherical Solid

Topics

“Bevel Gear”

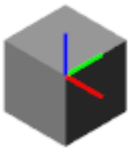
Brick Solid

Solid brick element with geometry, inertia, and color

Libraries:
Simscape / Multibody / Body Elements

Description

The Brick Solid block is a prismatic shape with geometry center coincident with the reference frame origin and prismatic surfaces normal to the reference frame x, y, and z axes.



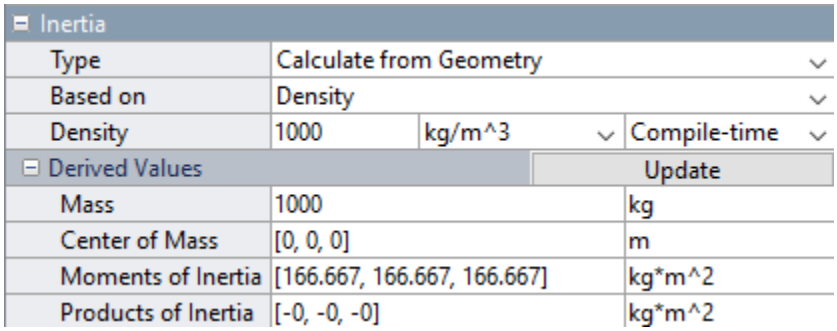
The Brick Solid block adds to the attached frame a solid element with geometry, inertia, and color. The brick solid element can be a simple rigid body or part of a compound rigid body—a group of rigidly connected solids, often separated in space through rigid transformations. Combine Brick Solid and other solid blocks with the Rigid Transform blocks to model a compound rigid body.

By default, this block automatically computes the mass properties of the solid. 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.

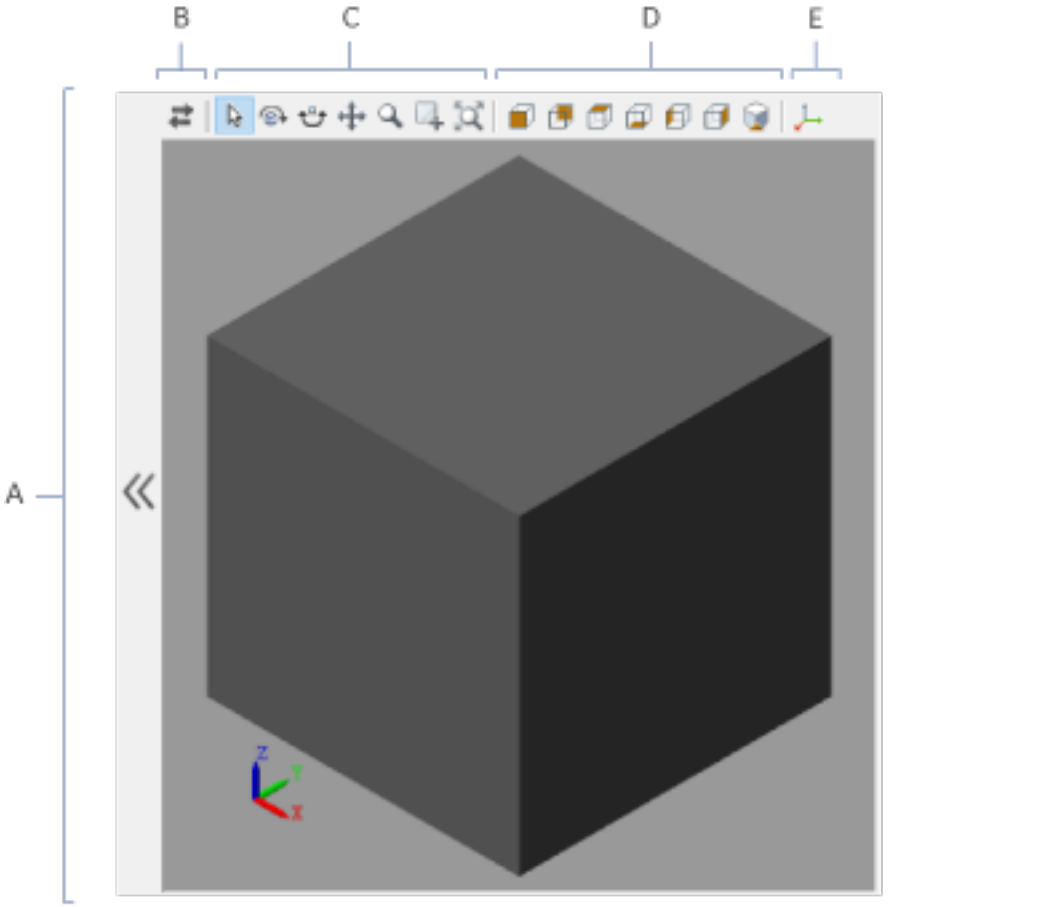


Derived Values Display

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.



A — Show/Hide Visualization Pane
B — Update Visualization
C — Select | Rotate | Roll | Pan | Zoom | Zoom to Region | Fit to View
D — Front | Back | Top | Bottom | Left | Right | Isometric Views
E — Show/Hide Frame

Brick 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 brick solid. This frame is fixed with respect to the solid geometry. 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”.

Geometry

G — Geometry
geometry

Geometry that represents the solid. Connect this port to a Spatial Contact Force block to model contacts on the solid.

Dependencies

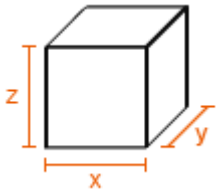
To enable this port, under **Geometry**, expand **Export** and select **Entire Geometry**.

Parameters

Geometry

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.



Entire Geometry — Export the true geometry of the block
`off` (default) | `on`

Select **Entire Geometry** to export the true geometry of the Brick Solid block which can be used for other blocks, such as the Spatial Contact Force block.

Dependencies

To enable this option, select **Entire Geometry** under the **Export**.

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/m^3` (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*m^2` (default) | three-element vector with units of mass*length^2

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:

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

Custom: Products of Inertia — Off-diagonal elements of inertia tensor
`[0 0 0] kg*m^2` (default) | three-element vector with units of mass*length^2

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:

- $I_{yz} = -\int_m yz\,dm$
- $I_{zx} = -\int_m zx\,dm$
- $I_{xy} = -\int_m xy\,dm$

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

Type of the visual representation of the solid, specified as `From Geometry`, `Marker`, or `None`. Set the parameter to `From Geometry` to show the visual representation of the solid. Set the parameter to `Marker` to represent the solid as a marker. Set the parameter to `None` to hide the solid in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Shape — Shape of marker to represent 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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
`10 pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered solid and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Frames

Show Port R — Show reference frame port for connection to other blocks
on (default) | off

Select to expose the **R** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button  to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the solid block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** of the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the solid.
  - **At Center of Mass**: Make the new frame origin coincident with the center of mass of the solid.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

- **Along Reference Frame Axis**: Selects an axis of the reference frame of the solid.
- **Along Principal Inertia Axis**: Selects an axis of the principal inertia axis of the solid.
- **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the solid. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button  to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button  to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2019b

Extended Capabilities

C/C++ Code Generation

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

See Also

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

Topics

“Creating Custom Solid Frames”
“Manipulate the Color of a Solid”
“Model a Simple Link”
“Model a Simple Pendulum”
“Modeling Bodies”
“Representing Solid Geometry”
“Specifying Custom Inertias”
“Working with Frames”

Bushing Joint

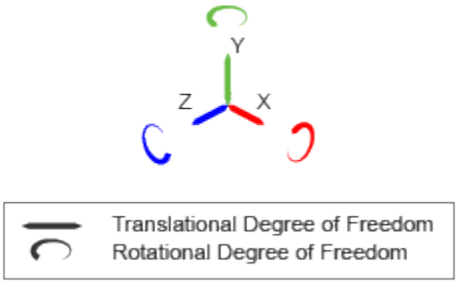
Joint with three prismatic and three revolute primitives

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.

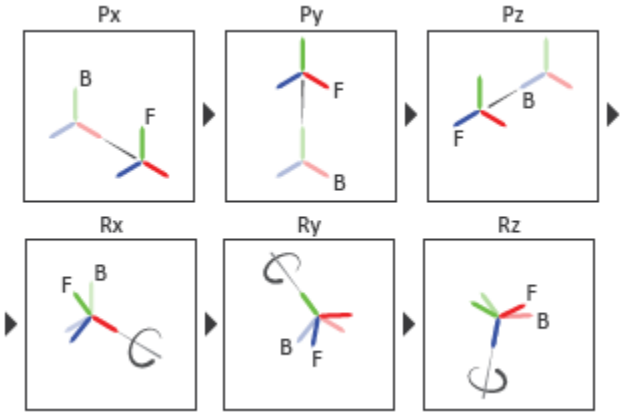
2 Rotation:

   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.
`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: 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.

Mode Configuration

Specify the mode of the joint. The joint mode can be normal or disengaged throughout the simulation, or you can provide an input signal to change the mode during the simulation.

Mode

Select one of the following options to specify the mode of the joint. The default setting is `Normal`.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.
`Provided by Input`	This option exposes the **mode** port that you can connect to an input signal to change the joint mode during the simulation. The joint mode is normal when the input signal is `0` and disengaged when the input signal is `-1`. The joint mode can be changed many times during the simulation.

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:

- fx, fy, fz — Actuation forces acting on the X, Y, and Z prismatic joint primitives
- px, py, pz — Desired trajectories of the X, Y, Z prismatic joint primitives

The revolute joint primitives provide the following actuation ports:

- tx, ty, tz — Actuation torques acting on the X, Y, and Z revolute joint primitives
- qx, qy, qz — Desired rotations of the X, Y, and Z revolute joint primitives

Sensing Ports

The prismatic joint primitives provide the following sensing ports:

- px, py, pz — Positions of the X, Y, and Z prismatic joint primitives
- vx, vy, vz — Velocities of the X, Y, and Z prismatic joint primitives
- ax, ay, az — Accelerations of the X, Y, and Z prismatic joint primitives
- fx, fy, fz — Actuation forces acting on the X, Y, and Z prismatic joint primitives
- fllx, flly, fllz — Forces due to contact with the lower limits of the X, Y, and Z prismatic joint primitives
- fulx, fuly, fulz — Forces due to contact with the upper limits of the X, Y, and Z prismatic joint primitives

The revolute joint primitives provide the following sensing ports:

- qx, qy, qz — Angular positions of the X, Y, and Z revolute joint primitives
- wx, wy, wz — Angular velocities of the X, Y, and Z revolute joint primitives
- bx, by, bz — Angular accelerations of the X, Y, and Z revolute joint primitives
- tx, ty, tz — Actuation torques acting on the X, Y, and Z revolute joint primitives
- tllx, tlly, tllz — Torques due to contact with the lower limits of the X, Y, and Z revolute joint primitives
- tulx, tuly, tulz — Torques due to contact with the upper limits of the X, Y, and Z 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

Mode Port

Mode configuration provides the following port:

- mode — Value of the mode of the joint. If the input is equal to `0`, the joint behaves normally. If the input is equal to `-1`, the joint behaves as disengaged.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

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

Topics

“Actuating and Sensing with Physical Signals”
“Cable Robot”
“Computing Actuator Torques Using Inverse Dynamics”
“Motion Sensing”
“Rotational Measurements”
“Translational Measurements”
“Using the Lead Screw Joint Block - Linear Actuator”

Cartesian Joint

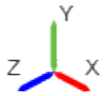
Joint with three prismatic primitives

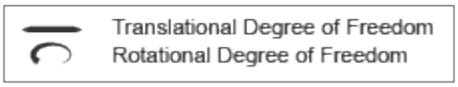
Library

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.





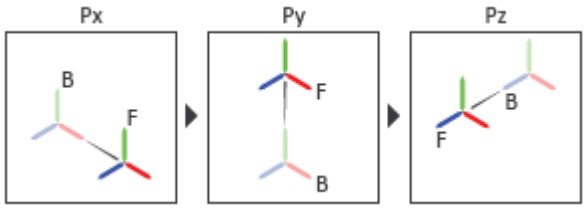
Joint Degrees of Freedom

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.

Mode Configuration

Specify the mode of the joint. The joint mode can be normal or disengaged throughout the simulation, or you can provide an input signal to change the mode during the simulation.

Mode

Select one of the following options to specify the mode of the joint. The default setting is `Normal`.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.

Method	Description
`Provided by Input`	This option exposes the **mode** port that you can connect to an input signal to change the joint mode during the simulation. The joint mode is normal when the input signal is `0` and disengaged when the input signal is `-1`. The joint mode can be changed many times during the simulation.

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:

- fx, fy, fz — Actuation forces acting on the X, Y, and Z prismatic joint primitives
- px, py, pz — Desired trajectories of the X, Y, Z prismatic joint primitives

Sensing Ports

The prismatic joint primitives provide the following sensing ports:

- px, py, pz — Positions of the X, Y, and Z prismatic joint primitives
- vx, vy, vz — Velocities of the X, Y, and Z prismatic joint primitives
- ax, ay, az — Accelerations of the X, Y, and Z prismatic joint primitives
- fx, fy, fz — Actuation forces acting on the X, Y, and Z prismatic joint primitives
- fllx, flly, fllz — Forces due to contact with the lower limits of the X, Y, and Z prismatic joint primitives
- fulx, fuly, fulz — Forces due to contact with the upper limits of the X, Y, and Z 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

Mode Port

Mode configuration provides the following port:

- mode — Value of the mode of the joint. If the input is equal to `0`, the joint behaves normally. If the input is equal to `-1`, the joint behaves as disengaged.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Prismatic Joint | Rectangular Joint

Topics

“Actuating and Sensing with Physical Signals”
“Motion Sensing”
“Specifying Variable Inertias”
“Translational Measurements”

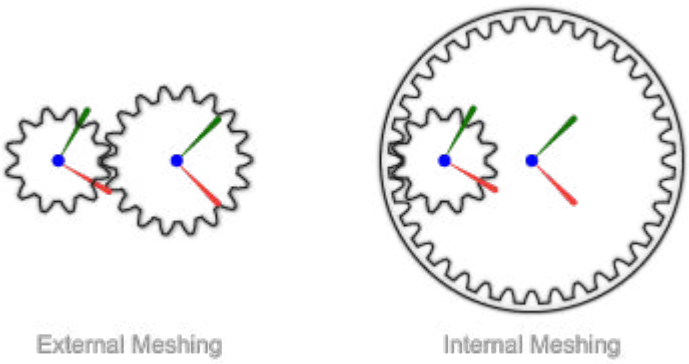
Common Gear Constraint

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

Libraries:
Simscape / Multibody / Gears and Couplings / Gears

Description

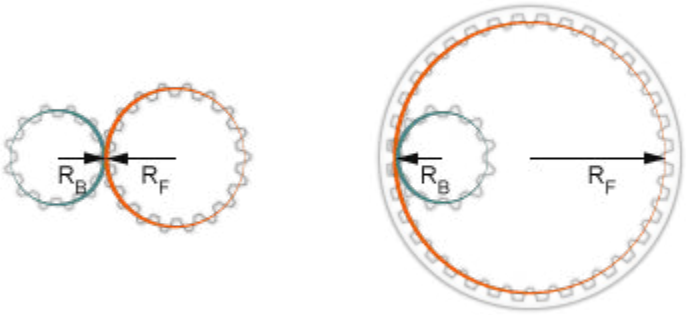
The Common Gear Constraint block represents a kinematic constraint between two coplanar spur gear bodies with parallel rotation axes. The gear meshing can be external to both gears or internal to one of the gears. The base and follower frame ports identify the connection frames on the spur gear bodies.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

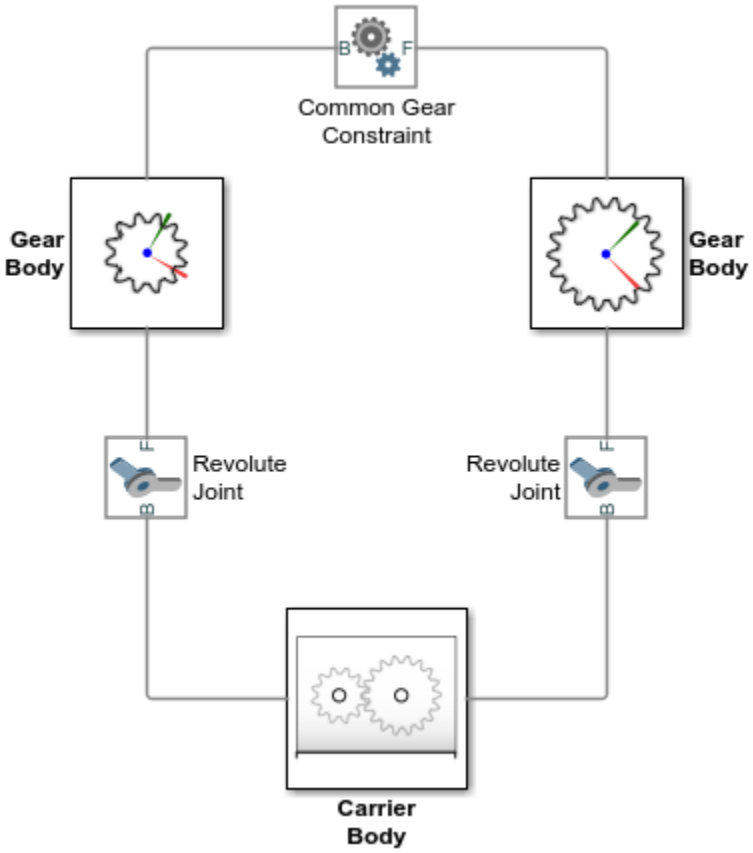
Gear Geometry

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.



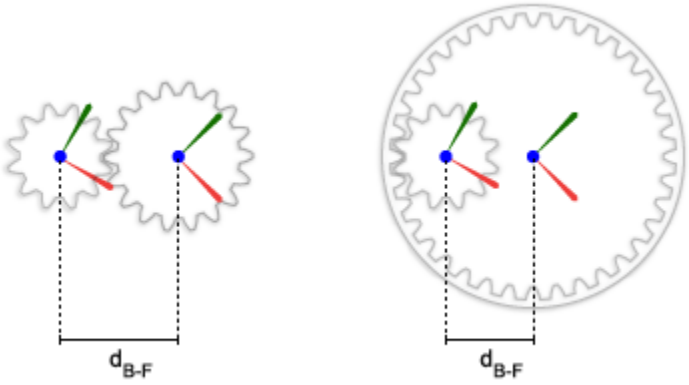
Assembly Requirements

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

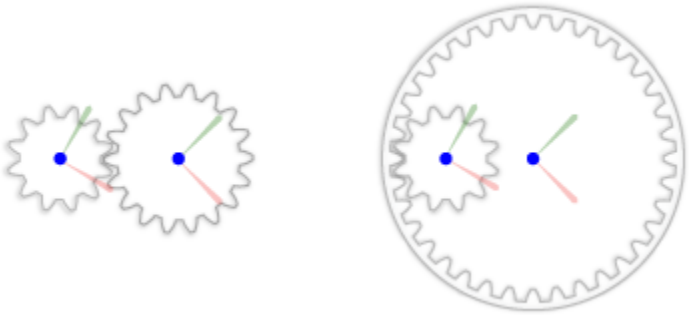
- The distance between the *z*-axes of the base and follower frame, denoted $d_{B\text{-}F}$ in the figure, must equal the distance between the gear centers. This constraint ensures that the rotation axes of the gears are at the proper distance for meshing.



- 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`.

Version History

Introduced in R2013a

Extended Capabilities

C/C++ Code Generation

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

See Also

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

Topics

“Assemble a Gear Model”
“External Spur Gear”
“Internal Spur Gear”
“Model a Compound Gear Train”
“Modeling Gear Constraints”
“Using the Common Gear Block - Cardan Gear Mechanism”
“Using the Common Gear Block”

Constant Velocity Joint

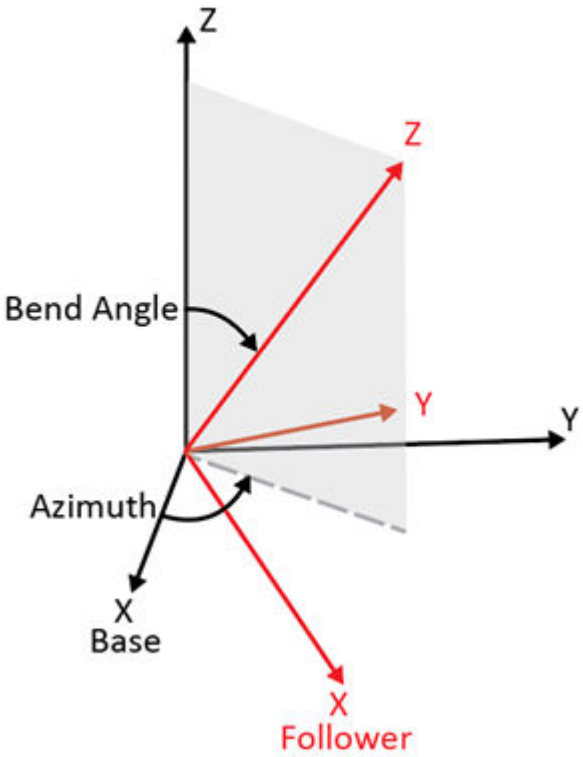
Joint that enforces a constant-velocity kinematic constraint between two shafts

Libraries:
Simscape / Multibody / Joints

Description

The Constant Velocity Joint block enforces a constant-velocity (CV) kinematic constraint between its base and follower frames, whose origins are coincident throughout the simulation. Specifically, if the *Z*-axes of the base and follower frames are both fixed with respect to a common reference frame, the *Z*-components of the two frames' angular velocities with respect to the common reference frame are equal.

The block has two degrees of freedom that allow the *Z*-axes of the base and follower frames to be arbitrarily oriented relative to each other. The figure shows an example.



The black and red frames indicate the base and follower frames of the block. The azimuth rotation about the *Z*-axis of the base frame locates the plane in which the bend angle occurs. The bend angle about the resulting *Y*-axis, the red *Y*-axis, specifies the orientation of the *Z*-axis of the follower frame with respect to the *Z*-axis of the base frame.

The block has two parameterizations, `Rotation Sequence (faster simulation)` or `Quaternion (allows zero bend angle)`, to specify the internal states of the joint. Use the rotation-sequence parameterization whenever possible because a simulation with this parameterization is generally faster than a simulation with the quaternion parameterization. See “Internal State” on page 1-0 for more information.

You can specify the desired initial states of the joint with the parameters under **State Targets**, such as the position and velocity of the azimuth and bend angle.

The block has a variety of sensing abilities. You can sense the azimuth, bend angle, and their time derivatives throughout the simulation. Furthermore, you can sense the forces and torques that act in the joint, such as constraint forces and total torque. See “Composite Force/Torque Sensing” on page 1-71 for more information.

Ports

Frame

B — Base frame
frame

Port associated with the base frame of the joint block. In typical applications where a CV joint couples two shafts, the *Z*-axis of base frame is aligned with the driving shaft.

F — Follower frame
frame

Port associated with the follower frame of the joint block. In typical applications where a CV joint couples two shafts, the *Z*-axis of follower frame is aligned with the driven shaft.

Input

mode — Input signal controlling joint mode
scalar

Input signal that controls the joint mode, specified as a unitless scalar. The joint mode is normal when the input signal is 0 and disengaged when the input signal is -1. The joint mode can be changed many times during a simulation.

Dependencies

To enable this port, under **Mode Configuration**, set **Mode** to `Provided by Input`.

Output

qb — Bend angle
scalar

Bend angle of the CV joint, returned as a scalar. This quantity is the angle between the *Z*-axes of the base and follower frames of the block.

Dependencies

To enable this port, under **Constant Velocity Primitive (CV) > Sensing > Bend Angle**, select **Position**.

wb — Velocity of bend angle
scalar

Velocity of the bend angle of the CV joint, returned as a scalar. This quantity equals the time derivative of the signal output from port **qb**.

Dependencies

To enable this port, under **Constant Velocity Primitive (CV) > Sensing > Bend Angle**, select **Velocity**.

bb — Acceleration of bend angle
scalar

Acceleration of the bend angle of the CV joint, returned as a scalar. This quantity equals the second time derivative of the signal output from port **qb**.

Dependencies

To enable this port, under **Constant Velocity Primitive (CV) > Sensing > Bend Angle**, select **Acceleration**.

qa — Azimuth
scalar

Azimuth of the CV joint, returned as a scalar. This quantity is the angle of the rotation about the *Z*-axis of the base frame. The rotation locates the plane in which the bend angle occurs.

Dependencies

To enable this port, under **Constant Velocity Primitive (CV) > Sensing > Azimuth**, select **Position**.

wa — Velocity of azimuth
scalar

Velocity of the azimuth of the CV joint, returned as a scalar. This quantity equals the time derivative of the signal output from port **qa**.

Dependencies

To enable this port, under **Constant Velocity Primitive (CV) > Sensing > Azimuth**, select **Velocity**.

ba — Acceleration of azimuth
scalar

Acceleration of the azimuth of the CV joint, returned as a scalar. This quantity equals the second time derivative of the signal output from port **qa**.

Dependencies

To enable this port, under **Constant Velocity Primitive (CV) > Sensing > Azimuth**, select **Acceleration**.

fc — Constraint force
scalar

Constraint force that acts in the joint, returned as a scalar. The force maintains the translational constraints of the joint. See “Measure Joint Constraint Forces” for more information.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Constraint Force**.

tc — Constraint torque
scalar

Constraint torque acting in the joint, returned as a scalar. The torque maintains the rotational constraints of the joint. See “Force and Torque Sensing” for more information.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Constraint Torque**.

ft — Total force
scalar

Total force acting in the joint, returned as a scalar. The total force is the sum of forces transmitted from one frame to the other through the joint. The force includes actuation, internal, and constraint forces. See “Force and Torque Sensing” for more information.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Total Force**.

tt — Total torque
scalar

Total torque acting in the joint, returned as a scalar. The total torque is the sum of torques transmitted from one frame to the other through the joint. The torque includes actuation, internal, and constraint torques. See “Force and Torque Sensing” for more information.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Total Torque**.

Parameters

Constant Velocity Primitive (CV)

Internal State — Internal state parameterization of joint
`Rotation Sequence (faster simulations)` (default) | `Quaternion (allows zero bend angle)`

Choose the parameterization to specify the internal states of the joint.

`Rotation Sequence (faster simulations)`

The Constant Velocity Joint block uses the *Z-Y-Z* rotation sequence parameterization. The angle of the second *Z*-axis rotation is the negative of the angle of the first *Z*-axis rotation.

Use this method whenever possible because a simulation with this parameterization is generally faster than a simulation with the quaternion parameterization. However, the *Z-Y-Z* rotation

sequence parameterization has a kinematic singularity when the bend angle is zero. If zero bend angle is required, set **Internal State** to `Quaternion (allows zero bend angle)`.

`Quaternion (allows zero bend angle)`

The Constant Velocity Joint block uses the quaternion parameterization. To enforce the CV kinematic constraint, the fourth component of the quaternion, the *Z*-component of the vector part, is always zero.

Use this method when zero bend angle is required. This method does not have a kinematic singularity at zero bend angle but has a kinematic singularity when the bend angle is 180 degrees. A simulation with this parameterization is generally slower than a simulation with the *Z-Y-Z* parameterization.

State Targets > Specify Position Target — Whether to specify position target
`off` (default) | `on`

Select this parameter to enable parameters for specifying the position target of the joint.

State Targets > Specify Position Target > Priority — Priority level of position target
`High (desired)` (default) | `Low (approximate)`

Set the priority level of the position target. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, under **Constant Velocity Primitive (CV) > State Targets**, select **Specify Position Target**.

State Targets > Specify Position Target > Value — Angles to specify position target
`Bend Angle Only` (default) | `Bend Angle and Azimuth`

Whether to specify the position target of the CV joint at the start of simulation using only the bend angle or the bend angle and the azimuth.

Dependencies

To enable this parameter, under **Constant Velocity Primitive (CV) > State Targets**, select **Specify Position Target**.

State Targets > Specify Position Target > Value > Bend Angle — Bend angle to specify
`45 deg` (default)

Specify the bend angle of the CV joint.

Dependencies

To enable this parameter, under **Constant Velocity Primitive (CV) > State Targets**, select **Specify Position Target**.

State Targets > Specify Position Target > Value > Azimuth — Azimuth to specify
`0 deg` (default)

Specify the azimuth of the CV joint.

Dependencies

To enable this parameter, under **Constant Velocity Primitive (CV)** > **State Targets** > **Specify Position Target** > **Value**, select **Bend Angle and Azimuth**.

State Targets > Specify Velocity Target — Whether to specify velocity target
`off` (default) | `on`

Select this parameter to enable parameters for specifying the velocity target of the joint.

State Targets > Specify Velocity Target > Priority — Priority level of velocity target
`High (desired)` (default) | `Low (approximate)`

Set the priority level of the velocity target. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, under **Constant Velocity Primitive (CV)** > **State Targets**, select **Specify Velocity Target**.

State Targets > Specify Velocity Target > Value — Parameters to specify velocity target
`Bend Angle Only` (default) | `Bend Angle and Azimuth`

Whether to specify the velocity target of the CV joint at the start of simulation using only the bend angle or the bend angle and the azimuth.

Dependencies

To enable this parameter, under **Constant Velocity Primitive (CV)** > **State Targets**, select **Specify Velocity Target**.

State Targets > Specify Velocity Target > Value > Bend Angle — Bend angle velocity to specify
0 `deg/s` (default)

Specify the bend angle velocity of the CV joint.

Dependencies

To enable this parameter, under **Constant Velocity Primitive (CV)** > **State Targets**, select **Specify Velocity Target**.

State Targets > Specify Velocity Target > Value > Azimuth — Azimuth velocity to specify
0 `deg/s` (default)

Specify the azimuth velocity of the CV joint.

Dependencies

To enable this parameter, under **Constant Velocity Primitive (CV)** > **State Targets** > **Specify Velocity Target** > **Value**, select **Bend Angle and Azimuth**.

Sensing > Bend Angle > Position — Whether to sense bend angle
off (default) | on

Select this parameter to enable the port **qb**.

Sensing > Bend Angle > Velocity — Whether to sense velocity of bend angle
off (default) | on

Select this parameter to enable the port **wb**.

Sensing > Bend Angle > Acceleration — Whether to sense acceleration of bend angle
off (default) | on

Select this parameter to enable the port **bb**.

Sensing > Azimuth > Position — Whether to sense azimuth
off (default) | on

Select this parameter to enable the port **qa**.

Sensing > Azimuth > Velocity — Whether to sense velocity of azimuth
off (default) | on

Select this parameter to enable the port **wa**.

Sensing > Azimuth > Acceleration — Whether to sense acceleration of azimuth
off (default) | on

Select this parameter to enable the port **ba**.

Mode Configuration

Mode — Joint mode
`Normal` (default) | `Disengaged` | `Provided by Input`

Specify the joint mode for the simulation.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.
`Provided by Input`	This option exposes the **mode** port that you can connect to an input signal to change the joint mode during the simulation. The joint mode is normal when the input signal is 0 and disengaged when the input signal is -1. The joint mode can be changed many times during the simulation.

Composite Force/Torque Sensing

Direction — Measurement direction
`Follower on Base` (default) | `Base on Follower`

Measurement direction, specified as `Follower on Base` or `Base on Follower`.

- `Follower on Base` — Sense the force and torque that the follower frame exerts on the base frame.
- `Base on Follower` — Sense the force and torque that the base frame exerts on the follower frame.

Reversing the direction changes the sign of the measurements. For more information see “Force and Torque Measurement Direction”.

Resolution Frame — Frame used to resolve measurements
`Base` (default) | `Follower`

Frame used to resolve the measurements, specified as `Base` or `Follower`.

- `Base` — The joint block resolves the measurements in the coordinates of the base frame.
- `Follower` — The joint block resolves the measurements in the coordinates of the follower frame.

Constraint Force — Whether to sense constraint force in joint
off (default) | on

Select this parameter to enable the port **fc**.

Constraint Torque — Whether to sense constraint torque in joint
off (default) | on

Select this parameter to enable the port **tc**.

Total Force — Whether to sense total force in joint
off (default) | on

Select this parameter to enable the port **ft**.

Total Torque — Whether to sense total torque in joint
off (default) | on

Select this parameter to enable the port **tt**.

Version History

Introduced in R2015a

Extended Capabilities

C/C++ Code Generation

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

See Also

Universal Joint

Topics

“Force and Torque Sensing”
“Guiding Assembly”
“Measure Joint Constraint Forces”

Cylindrical Joint

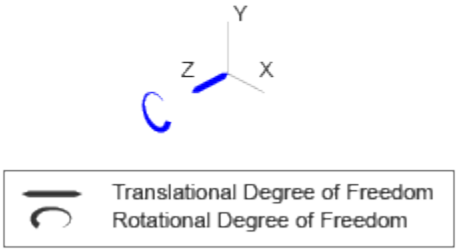
Joint with one prismatic and one revolute primitives possessing parallel motion axes

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.



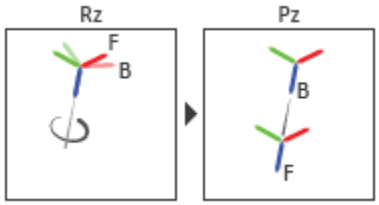
Joint Degrees of Freedom

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.

Mode Configuration

Specify the mode of the joint. The joint mode can be normal or disengaged throughout the simulation, or you can provide an input signal to change the mode during the simulation.

Mode

Select one of the following options to specify the mode of the joint. The default setting is `Normal`.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.

Method	Description
`Provided by Input`	This option exposes the **mode** port that you can connect to an input signal to change the joint mode during the simulation. The joint mode is normal when the input signal is `0` and disengaged when the input signal is `-1`. The joint mode can be changed many times during the simulation.

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
- fllz — Force due to contact with the lower limit of the Z prismatic joint primitive
- fulz — Force due to contact with the upper limit of 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
- tllz — Torque due to contact with the lower limit of the Z revolute joint primitive
- tulz — Torque due to contact with the upper limit of 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

Mode Port

Mode configuration provides the following port:

- mode — Value of the mode of the joint. If the input is equal to `0`, the joint behaves normally. If the input is equal to `-1`, the joint behaves as disengaged.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Prismatic Joint | Revolute Joint

Topics

“Actuating and Sensing with Physical Signals”
“Motion Sensing”
“Rotational Measurements”
“Translational Measurements”
“Using the Lead Screw Joint Block - Linear Actuator”

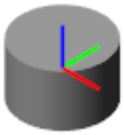
Cylindrical Solid

Solid cylindrical element with geometry, inertia, and color

Libraries:
Simscape / Multibody / Body Elements

Description

The Cylindrical Solid block is a cylindrical shape with geometry center coincident with the reference frame origin and symmetry axis coincident with the reference frame z axis.



The Cylindrical 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 Cylindrical Solid and other solid blocks with the 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.

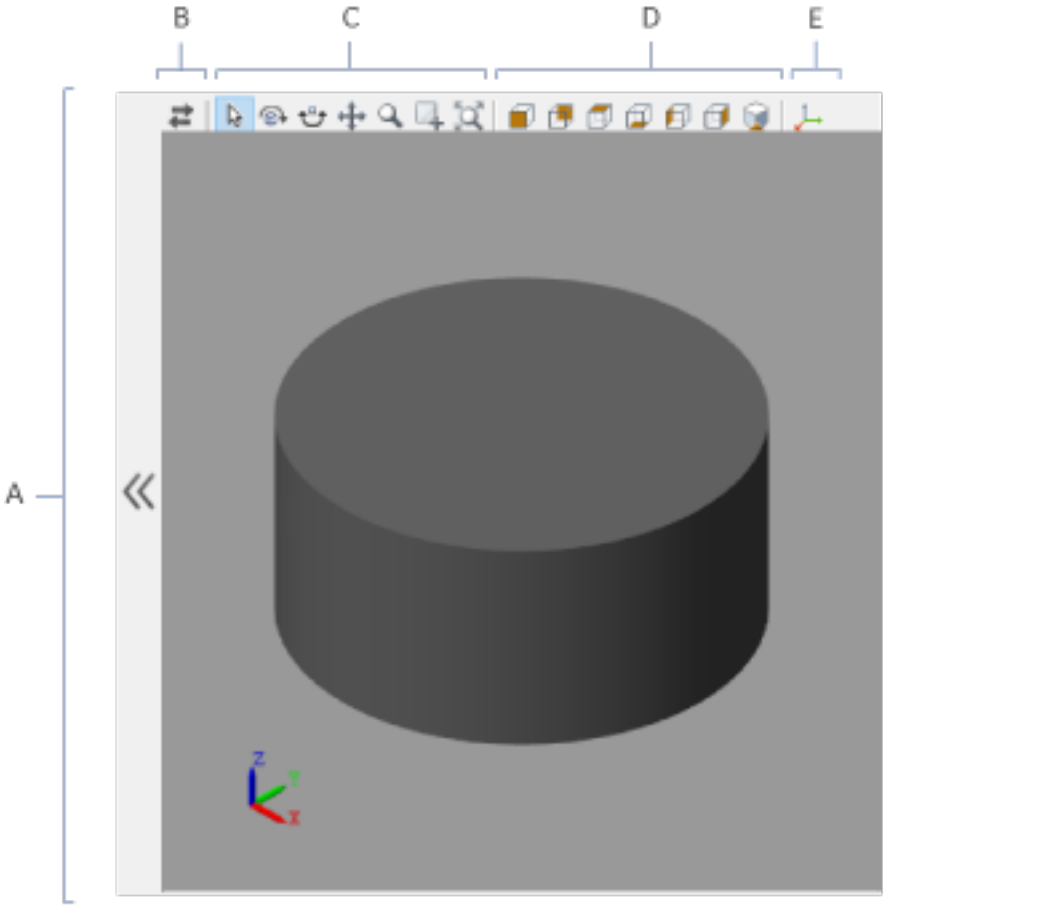
Inertia			
Type	Calculate from Geometry		
Based on	Density		
Density	1000	kg/m^3	Compile-time
Derived Values		Update	
Mass	1000		kg
Center of Mass	[0, 0, 0]		m
Moments of Inertia	[166.667, 166.667, 166.667]		kg*m^2
Products of Inertia	[-0, -0, -0]		kg*m^2

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.



Cylindrical 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. 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”.

Geometry

G — Geometry
geometry

Geometry that represents the solid. Connect this port to a Spatial Contact Force block to model contacts on the solid.

Dependencies

To enable this port, under **Geometry**, expand **Export** and select **Entire Geometry**.

Parameters

Geometry

Radius — Radius of the cylinder
`1 m` (default) | scalar with units of length

Distance between the axis of the cylinder and its surface.

Length — Length of the cylinder
`1 m` (default) | scalar with units of length

Distance between the opposing ends of the cylinder.

Entire Geometry — Export the true geometry of the block
`off` (default) | `on`

Select **Entire Geometry** to export the true geometry of the Cylindrical Solid block which can be used for other blocks, such as the Spatial Contact Force block.

Dependencies

To enable this option, select **Entire Geometry** under the **Export**.

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 if known.

Density — Mass per unit volume of material
`1000 kg/m^3` (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*m^2` (default) | three-element vector with units of mass*length^2

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:

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

Custom: Products of Inertia — Off-diagonal elements of inertia tensor
`[0 0 0] kg*m^2` (default) | three-element vector with units of mass*length^2

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:

- $I_{yz} = -\int_m yz\,dm$
- $I_{zx} = -\int_m zx\,dm$
- $I_{xy} = -\int_m xy\,dm$

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

Type of the visual representation of the solid, specified as `From Geometry`, `Marker`, or `None`. Set the parameter to `From Geometry` to show the visual representation of the solid. Set the parameter to `Marker` to represent the solid as a marker. Set the parameter to `None` to hide the solid in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Shape — Shape of marker to represent 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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
10 `pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered solid and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Frames

Show Port R — Show reference frame port for connection to other blocks
on (default) | off

Select to expose the **R** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button  to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the solid block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** of the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the solid.
  - **At Center of Mass**: Make the new frame origin coincident with the center of mass of the solid.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining

two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

- **Along Reference Frame Axis**: Selects an axis of the reference frame of the solid.
- **Along Principal Inertia Axis**: Selects an axis of the principal inertia axis of the solid.
- **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the solid. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button  to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button  to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2019b

Extended Capabilities

C/C++ Code Generation

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

See Also

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

Topics

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

Disk

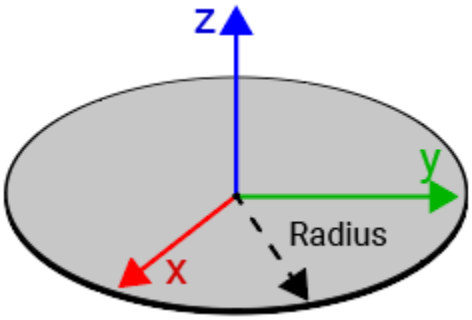
Disk for contact modeling

Libraries:
Simscape / Multibody / Curves and Surfaces

Description

The Disk block creates a 2-D circular disk that you can use to model contact problems. You can model the contact between disk and many other geometries, including an infinite plane, brick, cylinder, ellipsoid, sphere, or convex hull.

The Disk block has two ports. The **R** port corresponds to the reference frame that is coincident with the origin of the disk geometry, and the disk lies on the *xy*-plane of the reference frame. The **G** port is a geometry port that represents the geometry of the disk.



To specify the size of the disk, use the **Radius** parameter.

Ports

Frame

R — Reference frame
frame

Disk reference frame. Connect this frame to another block to specify the location and orientation of the disk.

Geometry

G — Geometry
geometry

Geometry that represents the disk. Connect this port to a Spatial Contact Force block to model contacts on the disk.

Parameters

Radius — Radius of disk
`1 m` (default) | positive integer

Radius of the disk, specified as a positive integer with a unit of length.

Graphic

Type — Visual representation of disk
`From Geometry` (default) | `None`

Visual representation of the disk. Set this parameter to `From Geometry` to show the visual representation of the disk. Set this parameter to `None` to hide the disk in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify diffuse color and opacity. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry`.

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered object and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set **Type** to `From Geometry`.

Opacity — Opacity of rendered object
1.0 (default) | scalar in the range of 0 to 1

Opacity of the rendered object, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The specular highlight is the bright spot on the rendered object due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

This property specifies a general level of illumination that does not come directly from a light source. Use this property to change the shadow color of the rendered object.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0 0 0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Shininess — Shininess of rendered object
75 (default) | scalar in the range of 0 to 128

Shininess of the rendered object, specified as a scalar in the range of 0 to 128. This property affects the sharpness of the specular reflections of the rendered object. An object with high shininess has a mirror-like appearance, and an object with low shininess has a more low-gloss or satin appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Version History

Introduced in R2022a

Extended Capabilities

C/C++ Code Generation

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

See Also

Spatial Contact Force

Distance Constraint

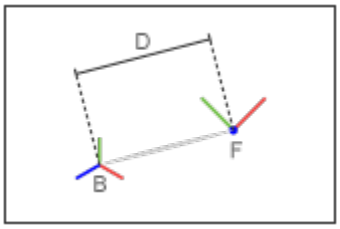
Fixed distance between two frame origins

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
- fm — Signed magnitude of the distance constraint force

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

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

Topics

“Model a Compound Gear Train”

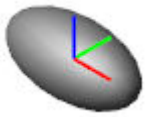
Ellipsoidal Solid

Solid ellipsoidal element with geometry, inertia, and color

Libraries:
Simscape / Multibody / Body Elements

Description

The Ellipsoidal Solid block is a 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.



The Ellipsoidal 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 Ellipsoidal Solid and other solid blocks with the 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.

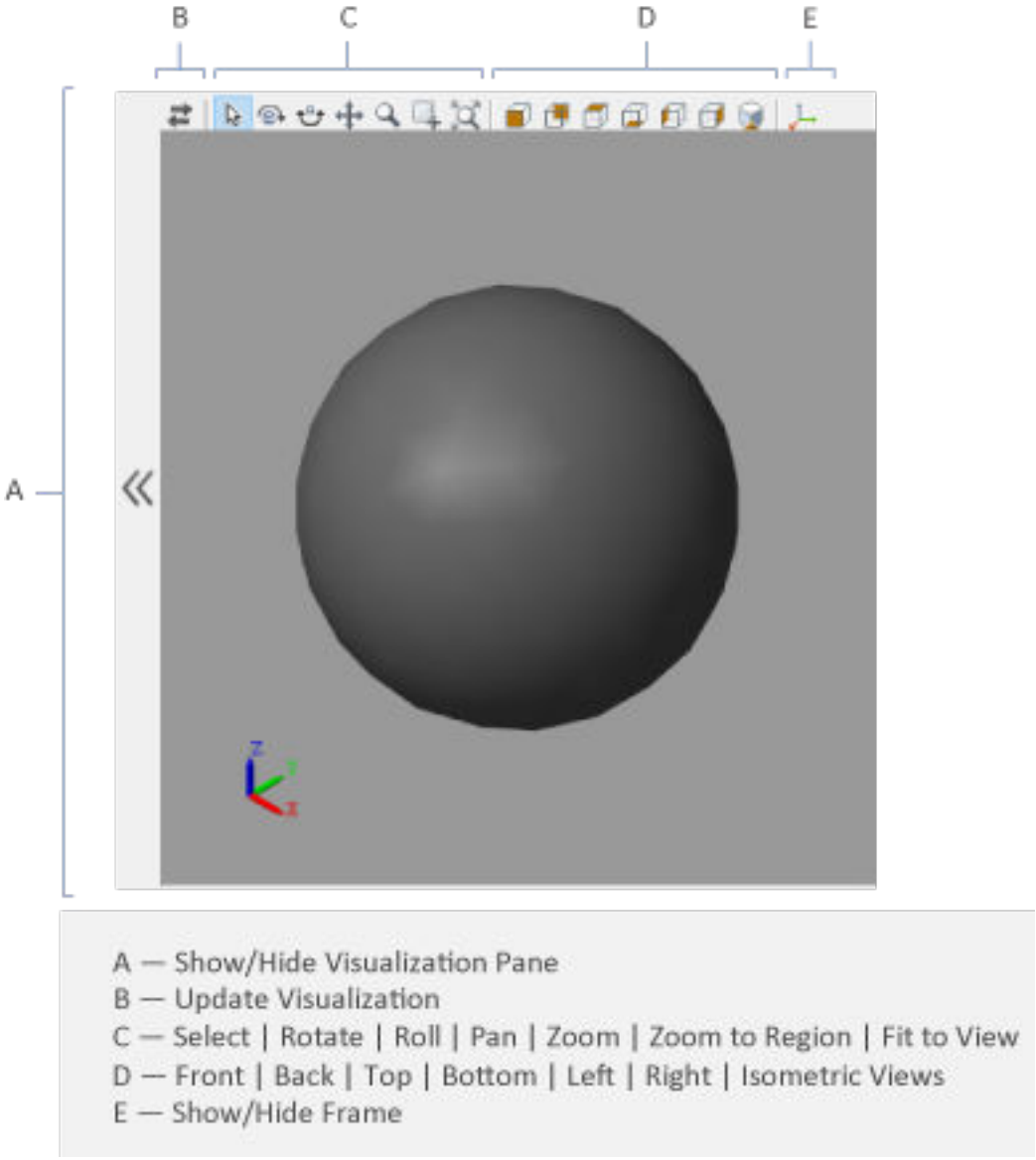
Inertia			
Type	Calculate from Geometry		
Based on	Density		
Density	1000	kg/m^3	Compile-time
Derived Values		Update	
Mass	1000		kg
Center of Mass	[0, 0, 0]		m
Moments of Inertia	[166.667, 166.667, 166.667]		kg*m^2
Products of Inertia	[-0, -0, -0]		kg*m^2

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.



Ellipsoidal 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. 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”.

Geometry

G — Geometry
geometry

Geometry that represents the solid. Connect this port to a Spatial Contact Force block to model contacts on the solid.

Dependencies

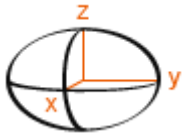
To enable this port, under **Geometry**, expand **Export** and select **Entire Geometry**.

Parameters

Geometry

Radii — Ellipsoid radii along the *x*, *y*, and *z* semiprincipal axes
`[1 1 2] 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.



Entire Geometry — Export the true geometry of the block
`off` (default) | `on`

Select **Entire Geometry** to export the true geometry of the Ellipsoidal Solid block which can be used for other blocks, such as the Spatial Contact Force block.

Dependencies

To enable this option, select **Entire Geometry** under the **Export**.

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 if known.

Density — Mass per unit volume of material
`1000 kg/m^3` (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*m^2` (default) | three-element vector with units of mass*length^2

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:

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

Custom: Products of Inertia — Off-diagonal elements of inertia tensor
`[0 0 0] kg*m^2` (default) | three-element vector with units of mass*length^2

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:

- $I_{yz} = -\int_m yz\,dm$
- $I_{zx} = -\int_m zx\,dm$
- $I_{xy} = -\int_m xy\,dm$

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

Type of the visual representation of the solid, specified as `From Geometry`, `Marker`, or `None`. Set the parameter to `From Geometry` to show the visual representation of the solid. Set the parameter to `Marker` to represent the solid as a marker. Set the parameter to `None` to hide the solid in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Shape — Shape of marker to represent 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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
`10 pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered solid and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`

2 **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Frames

Show Port R — Show reference frame port for connection to other blocks
on (default) | off

Select to expose the **R** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button  to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the solid block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** of the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the solid.
  - **At Center of Mass**: Make the new frame origin coincident with the center of mass of the solid.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

  - **Along Reference Frame Axis**: Selects an axis of the reference frame of the solid.
  - **Along Principal Inertia Axis**: Selects an axis of the principal inertia axis of the solid.
  - **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the solid. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the

visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button  to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button  to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2019b

Extended Capabilities

C/C++ Code Generation

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

See Also

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

Topics

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

External Force and Torque

Apply external force and/or torque to connected frame

Libraries:
Simscape / Multibody / Forces and Torques

Description

The External Force and Torque block applies the external force and/or torque to the frame that connects to the **F** port of the block. The forces act at the origin of the frame and the torques act about the corresponding axes of the frame. The block accepts physical signals to specify the forces and torques. The signal can be a constant or time-varying value throughout a simulation.

You can specify the resolution frame to resolve the force and torque inputs. A force input with a positive value acts along the positive direction of the associated frame axis. A torque input with a positive value acts about the associated frame axis according to the right-hand rule.

Ports

Frame

F — Follower frame
frame

Follower frame at which the external forces and torques apply.

Input

Force

fx — External force along *x*-axis
physical signal

Physical signal port that accepts the external force in the *x*-direction of the force resolution frame. The input signal is a scalar that can vary over time.

Dependencies

To enable this port, under **Force**, select **Force (X)**.

fy — External force along *y*-axis
physical signal

Physical signal port that accepts the external force in the *y*-direction of the force resolution frame. The input signal is a scalar that can vary over time.

Dependencies

To enable this port, under **Force**, select **Force (Y)**.

fz — External force along *z*-axis
physical signal

Physical signal port that accepts the external force in the *z*-direction of the force resolution frame. The input signal is a scalar that can vary over time.

Dependencies

To enable this port, under **Force**, select **Force (Z)**.

f — External force vector
physical signal

Physical signal port that accepts the external force. The input signal is a 1-by-3 or 3-by-1 vector, [F_x F_y F_z], that can vary over time.

Dependencies

To enable this port, under **Force**, select **Force**.

Torque

tx — External torque about *x*-axis
physical signal

Physical signal port that accepts the external torque about the *x*-axis of the torque resolution frame. The input signal is a scalar that can vary over time.

Dependencies

To enable this port, under **Torque**, select **Torque (X)**.

ty — External torque about *y*-axis
physical signal

Physical signal port that accepts the external torque about the *y*-axis of the torque resolution frame. The input signal is a scalar that can vary over time.

Dependencies

To enable this port, under **Torque**, select **Torque (Y)**.

tz — External torque about *z*-axis
physical signal

Physical signal port that accepts the external torque about the *z*-axis of the torque resolution frame. The input signal is a scalar that can vary over time.

Dependencies

To enable this port, under **Torque**, select **Torque (Z)**.

t — External torque vector
physical signal

Physical signal port that accepts the external torque. The input signal is a 1-by-3 or 3-by-1 vector, [T_x T_y T_z], that can vary over time.

Dependencies

To enable this port, under **Torque**, select **Torque**.

Parameters

Force

Force Resolution Frame — Frame to use to resolve each force input
`Attached Frame` (default) | `World`

Frame used to resolve the force inputs, specified as:

- `Attached Frame` — The block resolves the inputs in the frame that connects to the follower frame of the External Force and Torque block.
- `World` — The block resolves the inputs in the world frame of the model.

Torque

Torque Resolution Frame — Frame to use to resolve each torque input
`Attached Frame` (default) | `World`

Frame used to resolve the torque inputs, specified as:

- `Attached Frame` — The block resolves the inputs in the frame that connects to the follower frame of the External Force and Torque block.
- `World` — The block resolves the inputs in the world frame of the model.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Internal Force

Topics

“Force and Torque Sensing”

Extruded Solid

Solid extruded element with geometry, inertia, and color

Libraries:
Simscape / Multibody / Body Elements

Description

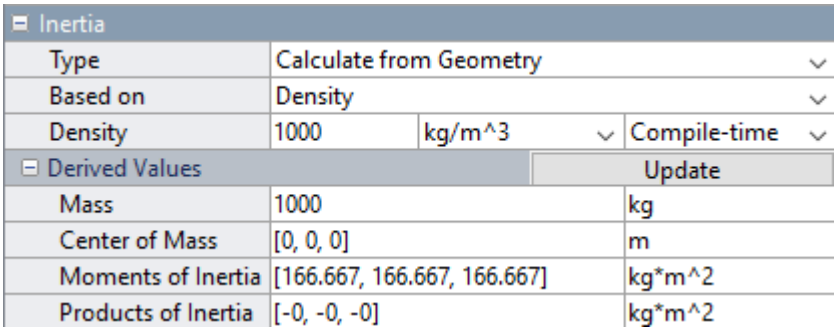
The Extruded Solid block adds to the attached frame a solid element with geometry, inertia, and color. The solid element can be a simple rigid body or part of a compound rigid body—a group of rigidly connected solids, often separated in space through rigid transformations. Combine Extruded Solid and other solid blocks with the 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.



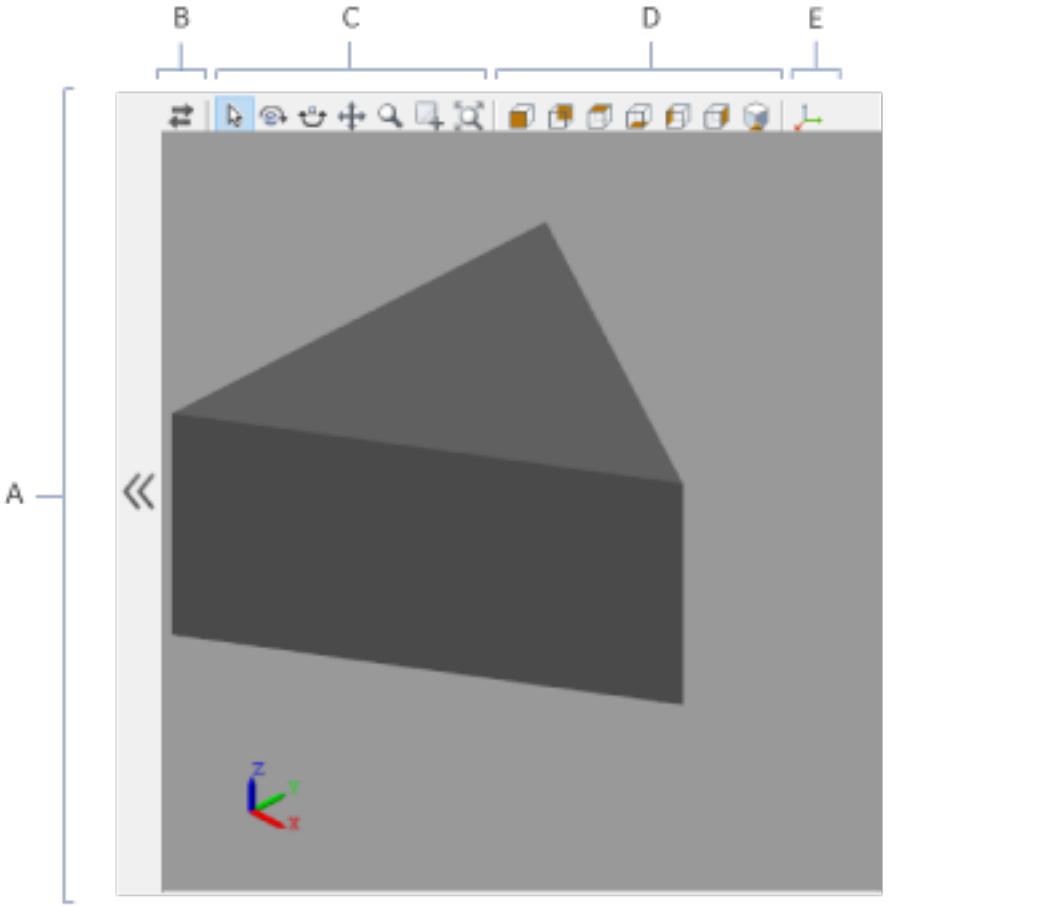
Derived Values Display

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.



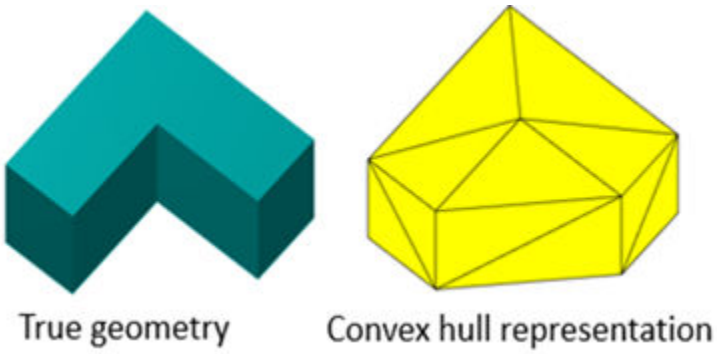
Extruded Solid Visualization Pane

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.

Exporting Geometry Properties

The Extruded Solid block can generate a convex hull geometry representation from an extruded solid. This geometric data can be used to model spatial contact forces.

As shown in the figure, the convex hull geometry is an approximation of the true geometry. Note that the block calculates the physical properties, such as mass and inertia, based on its true geometry.



L-shape sold

Ports

Frame

R — Reference frame
frame

Local reference frame of the solid. This frame is fixed with respect to the solid geometry. 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”.

Geometry

G — Geometry
geometry

Geometry that represents the solid. Connect this port to a Spatial Contact Force block to model contacts on the solid.

Dependencies

To enable this port, under **Geometry**, expand **Export** and select **Entire Geometry**.

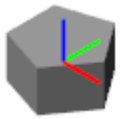
Parameters

Geometry

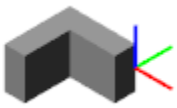
Extrusion Type — Shape parameterization to use
`Regular` (default) | `General`

Shape parameterization to use. Select `Regular` or `General`.

- `Regular` — 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` — 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.

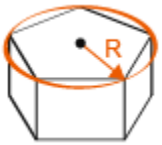


Regular Extrusion: Number of Sides — Number of sides of the extrusion cross-section
3 (default) | scalar with units of length

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.



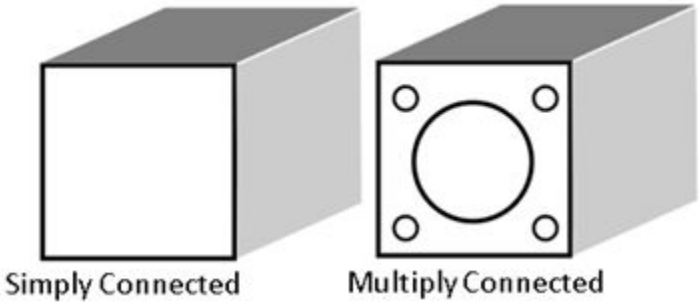
Regular Extrusion: Length — Sweep length of the extrusion
`1 m` (default) | scalar with units of length

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.

Entire Geometry — Export the true geometry of the block
`off` (default) | `on`

Select **Entire Geometry** to export the true geometry of the Extruded Solid block which can be used for other blocks, such as the Spatial Contact Force block.

Dependencies

To enable this option, set **Extrusion Type** to `Regular` and select **Entire Geometry** under the **Export**.

Convex Hull — Generate the convex hull representation of the true geometry
`off` (default) | `on`

Select **Convex Hull** to generate the convex hull representation of the true geometry. This convex hull can be used for contacts by connecting the Spatial Contact Force block.

Dependencies

To enable this option, set **Extrusion Type** to `General` and select **Convex Hull** under the **Export**.

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 if known.

Density — Mass per unit volume of material
`1000 kg/m^3` (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*m^2` (default) | three-element vector with units of mass*length^2

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:

- $I_{xx} = \int_m \left(y^2 + z^2\right) dm$
- $I_{yy} = \int_m \left(x^2 + z^2\right) dm$

- $I_{zz} = \int_m \left(x^2 + y^2\right) dm$

Custom: Products of Inertia — Off-diagonal elements of inertia tensor
`[0 0 0] kg*m^2` (default) | three-element vector with units of mass*length^2

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:

- $I_{yz} = -\int_m yz\, dm$
- $I_{zx} = -\int_m zx\, dm$
- $I_{xy} = -\int_m xy\, dm$

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

Type of the visual representation of the solid, specified as `From Geometry`, `Marker`, or `None`. Set the parameter to `From Geometry` to show the visual representation of the solid. Set the parameter to `Marker` to represent the solid as a marker. Set the parameter to `None` to hide the solid in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Shape — Shape of marker to represent 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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
10 `pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered solid and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Frames

Show Port R — Show reference frame port for connection to other blocks
on (default) | off

Select to expose the **R** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button  to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the solid block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** of the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the solid.
  - **At Center of Mass**: Make the new frame origin coincident with the center of mass of the solid.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

- **Along Reference Frame Axis**: Selects an axis of the reference frame of the solid.
- **Along Principal Inertia Axis**: Selects an axis of the principal inertia axis of the solid.
- **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the solid. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button  to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button  to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2019b

Extended Capabilities

C/C++ Code Generation

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

See Also

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

Topics

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

File Solid

Solid element with properties derived from external file

Libraries:
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.

The File Solid block can read geometries created by various CAD software, as shown in the table.

Supported Software and File Formats

CAD Software	Release Supported
ACIS	Up to 2019
Autodesk Inventor	Up to 2020
CATIA V4	Up to 4.2.5
CATIA V5	Up to V5-6 R2019(R29)
CATIA V6	Up to V5-6 R2019(R29)
Creo-Pro/E	Pro/Engineer 19.0 to Creo 6.0
IGES	5.1, 5.2, and 5.3
JT	Up to v10.6
NX	V11.0 to NX 12.0, and 1953
Parasolid	Up to v33
Revit	2015 to 2021
Rhino3D	4, 5, and 6
Solid Edge	V19 - 20, ST - ST10, 2021
SolidWorks	97 to 2021
STEP	AP 203 E1/E2, AP 214, and AP 242
STL	All Versions

Note 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”.

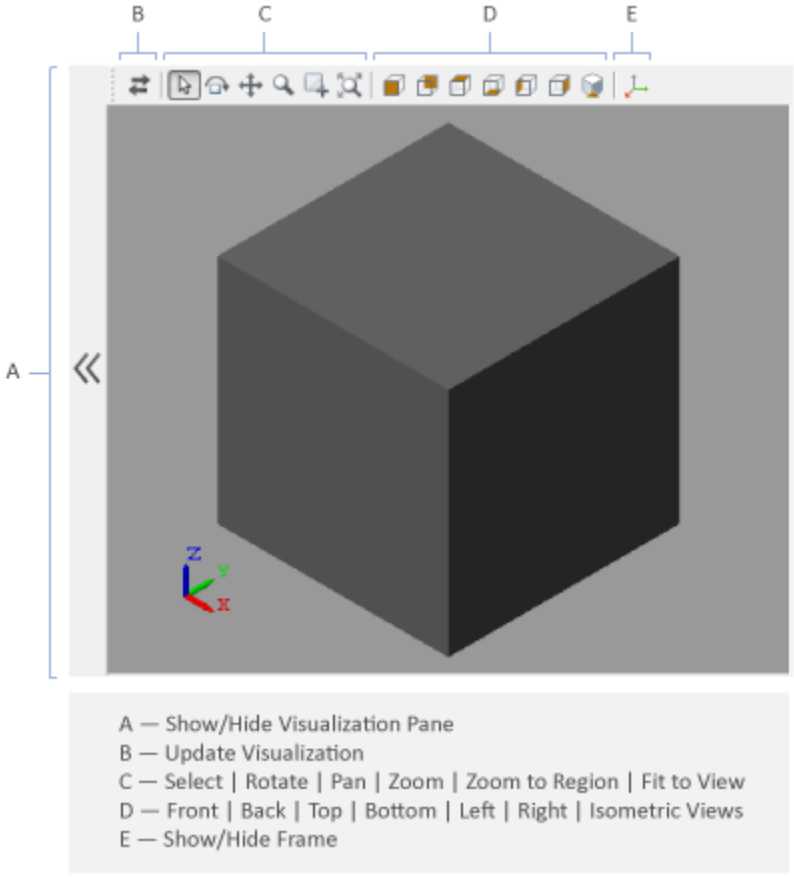
Derived Values

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.

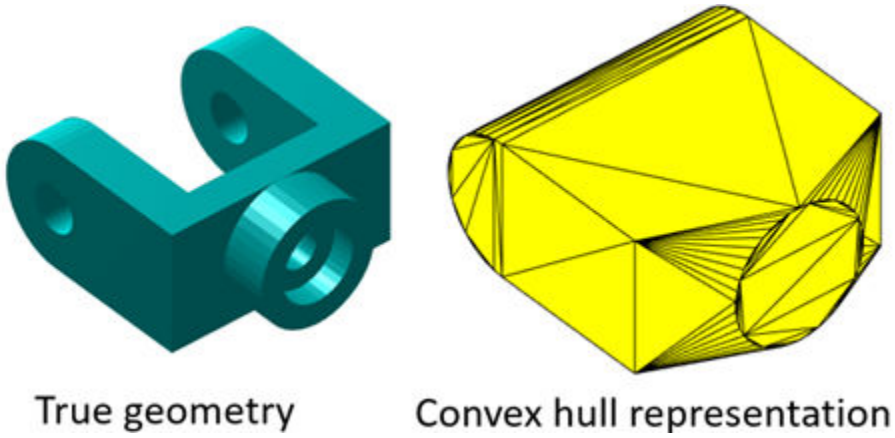
This approach suits complex models in which multiple blocks must share the same parameter value—a common density, say, or color, if defined as an RGB vector. When the MATLAB variable definition then changes, so do all block parameters that depend on it. Consider using MATLAB variables here if a parameter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

Exporting Geometry Properties

The File Solid block can generate a convex hull geometry representation of an imported CAD file in the Simscape Multibody environment. This geometric data can be used to model spatial contact forces.

As shown in the figure, the convex hull geometry is an approximation of the true geometry. Note that the block calculates the physical properties, such as mass and inertia, based on its true geometry.



Simple Part

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.

Geometry

CH — Convex hull representation
geometry

Convex hull that represents the geometry of the solid. Connect this port to a Spatial Contact Force block to model contacts on the convex hull.

Dependencies

To enable this port, under **Geometry**, expand **Export** and select **Convex Hull**.

Parameters

Geometry

File Name — Path of CAD geometry file
custom character vector

Path of the CAD file, specified as a custom character vector. The file location can be specified as an absolute path starting from the root directory of the file system or a relative path starting from a folder on the MATLAB® path.

See “Supported Software and File Formats” on page 1-124 for details of supported file formats.

Example: `'C:/Users/JDoe/Documents/myShape.STEP'` or `'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 geometry defined in a geometry file. Changing the units changes the scale of the imported geometry.

Convex Hull — Generate a convex hull representation of the true geometry
`off` (default) | `on`

Select **Convex Hull** to generate a convex hull representation of the true geometry. This convex hull can be used for contacts by connecting the Spatial Contact Force block.

Dependencies

To enable this option, select **Convex Hull** under the **Export**.

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/m^3` (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*m^2` (default) | three-element vector with units of mass*length^2

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:

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

Custom: Products of Inertia — Off-diagonal elements of inertia tensor
`[0 0 0] kg*m^2` (default) | three-element vector with units of mass*length^2

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:

- $I_{yz} = -\int_m yz\, dm$
- $I_{zx} = -\int_m zx\, dm$
- $I_{xy} = -\int_m xy\, dm$

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

Type of the visual representation of the solid, specified as `From Geometry`, `Marker`, or `None`. Set the parameter to `From Geometry` to show the visual representation of the solid. Set the parameter to `Marker` to represent the solid as a marker. Set the parameter to `None` to hide the solid in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Shape — Shape of marker to represent 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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
`10 pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered solid and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This

parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Frames

Show Port R — Show reference frame port for connection to other blocks
on (default) | off

Select to expose the **R** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button  to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the solid block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** of the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the solid.
  - **At Center of Mass**: Make the new frame origin coincident with the center of mass of the solid.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

  - **Along Reference Frame Axis**: Selects an axis of the reference frame of the solid.
  - **Along Principal Inertia Axis**: Selects an axis of the principal inertia axis of the solid.
  - **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the solid. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button  to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button  to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2018b

Extended Capabilities

C/C++ Code Generation

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

See Also

Brick Solid | Cylindrical Solid | Ellipsoidal Solid | Extruded Solid | Revolved Solid | Spherical Solid | Variable Brick Solid | Variable Cylindrical Solid | Variable Spherical Solid | Rigid Transform

Topics

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

Flexible Angle Beam

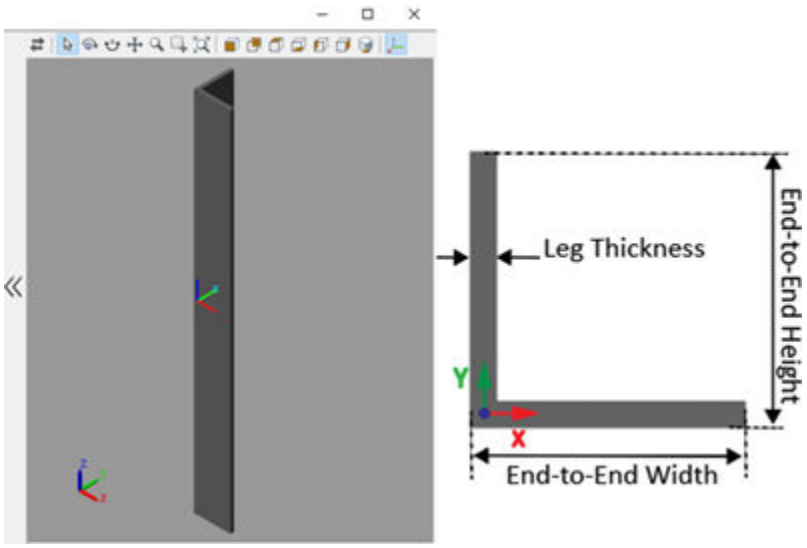
Angle beam with elastic properties for deformation

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

Description

The Flexible Angle Beam block models a slender beam with an L-shaped cross-section, also known as an L-beam. The L-beam consists of one horizontal component and one vertical component, which are called beam legs. The L-beam can have small and linear deformations. These deformations include extension, bending, and torsion. The block calculates the beam cross-sectional properties, such as the axial, flexural, and torsional rigidities, based on the geometry and material properties that you specify.

The geometry of the L-beam is an extrusion of its cross-section. The beam cross-section, defined in the *xy*-plane, is extruded along the *z*-axis. To define the cross-section, you can specify its dimensions in the **Geometry** section of the block dialog box. The figure shows an L-beam and its cross-section. The reference frame of the beam is located at the midpoint of the intersection line of the mid-planes of the two legs.



This block supports two damping methods and a discretization option to increase the accuracy of the modeling. For more information, see “Overview of Flexible Beams”.

Stiffness and Inertia Properties

The block provides two ways to specify the stiffness and inertia properties for a beam. To model a beam made of homogeneous, isotropic, and linearly elastic material, in the **Stiffness and Inertia** section, set the **Type** parameter to `Calculate from Geometry`. Then specify the density, Young’s

modulus, and Poisson’s ratio or shear modulus. See the **Derived Values** parameter for more information about the calculated stiffness and inertia properties.

Alternatively, you can manually specify the stiffness and inertia properties, such as flexural rigidity and mass moment of inertia density, by setting the **Type** parameter to `Custom`. Use this option to model a beam that is made of anisotropic materials. This option decouples the mechanical properties from the beam cross section so you can specify desired mechanical properties without capturing the details of the exact cross section, such as fillet, rounds, chamfers, and tapers.

The manually entered stiffness properties must be calculated with respect to the frame located at the bending centroid. The frame must be in the same orientation as the beam reference frame. The stiffness matrix is

$$\bar{S}^b = \begin{bmatrix} S^b & 0 & 0 & 0 \\ 0 & H_x^b & -H_{xy}^b & 0 \\ 0 & -H_{xy}^b & H_y^b & 0 \\ 0 & 0 & 0 & H_z^b \end{bmatrix}$$

where:

- S^b is the axial stiffness along the beam.
- H_x^b is the centroidal bending stiffness about the *x*-axis.
- H_y^b is the centroidal bending stiffness about the *y*-axis.
- H_z^b is the torsional stiffness.
- H_{xy}^b is the centroidal cross bending stiffness.

The manually entered inertia properties must be calculated with respect to a frame located at the center of mass. The frame must be in the same orientation as the beam reference frame. The mass matrix includes rotatory inertia

$$\bar{M}^c = \begin{bmatrix} m^c & 0 & 0 & 0 & 0 & 0 \\ 0 & m^c & 0 & 0 & 0 & 0 \\ 0 & 0 & m^c & 0 & 0 & 0 \\ 0 & 0 & 0 & i_x^c & -i_{xy}^c & 0 \\ 0 & 0 & 0 & -i_{xy}^c & i_y^c & 0 \\ 0 & 0 & 0 & 0 & 0 & i_z^c \end{bmatrix}$$

where:

- m^c is the mass per unit length.
- i_x^c is the mass moment of inertia density about the *x*-axis.
- i_y^c is the mass moment of inertia density about the *y*-axis.
- i_z^c is the polar mass moment of inertia density.

- i_{xy}^{c} is the mass product of inertia density.

The beam block uses the classical beam theory where the relation between sectional strains and stress resultants is

$$\begin{bmatrix} \bar{F}_z \\ \bar{M}_x \\ \bar{M}_y \\ \bar{M}_z \end{bmatrix} = \bar{S}^b \begin{bmatrix} \bar{\varepsilon}_z \\ \bar{\kappa}_x \\ \bar{\kappa}_y \\ \bar{\kappa}_z \end{bmatrix}$$

where:

- $\bar{F}_z$ is the axial force along the beam.
- $\bar{M}_x$ is the bending moment about the *x*-axis.
- $\bar{M}_y$ is the bending moment about the *y*-axis.
- $\bar{M}_z$ is the torsional moment about the *z*-axis.
- $\bar{\varepsilon}_z$ is the axial strain along the beam.
- $\bar{\kappa}_x$ is the bending curvature about the *x*-axis.
- $\bar{\kappa}_y$ is the bending curvature about the *y*-axis.
- $\bar{\kappa}_z$ is the torsional twist about the *z*-axis.

Ports

Frame

A — Connection frame
frame

Frame 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 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

End-to-End Width — Distance between ends of horizontal leg
`1 m` (default) | positive scalar

Distance between the ends of the horizontal leg.

Note The **End-to-End Width** must be larger than the **Vertical Leg Thickness**.

End-to-End Height — Distance between ends of vertical leg
`1 m` (default) | positive scalar

Distance between the ends of the vertical leg.

Note The **End-to-End Height** must be larger than the **Horizontal Leg Thickness**.

Horizontal Leg Thickness — Distance between faces of horizontal leg
`0.1 m` (default) | positive scalar

Distance between the two faces of the horizontal leg.

Vertical Leg Thickness — Distance between faces of vertical leg
`0.1 m` (default) | positive scalar

Distance between the two faces of the vertical leg.

Length — Extrusion length of beam
`10 m` (default) | positive scalar

Extrusion length of the beam. The beam is modeled 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

Type — Method to use to specify stiffness and inertia properties
`Calculate from Geometry` (default) | `Custom`

Method to use to specify the stiffness and inertia properties, specified as `Calculate from Geometry` or `Custom`.

When you set the parameter to `Calculate from Geometry`, the block calculates the stiffness and inertia properties based on the specified density, Young's modulus, and Poisson's ratio or shear modulus. Set the parameter to `Custom` to manually specify the stiffness and inertia properties.

Density — Mass per unit volume of material
`2700 kg/m^3` (default) | positive scalar

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

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

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.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Young's Modulus — Ratio of axial stress to axial strain
`70 GPa` (default) | positive scalar

Young's modulus of elasticity of the beam. The greater its value, the stronger the resistance to bending and axial deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Poisson's Ratio — Ratio of transverse to longitudinal strains
0.33 (default) | scalar in the range [0, 0.5)

Poisson's ratio of the beam. The value specified must be greater than or equal to `0` and smaller than `0.5`. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's Modulus and Poisson's Ratio**.

Shear Modulus — Ratio of shear stress to engineering shear strain
`26 GPa` (default) | positive scalar

Shear modulus (or modulus of rigidity) of the beam. The greater its value, the stronger the resistance to torsional deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's and Shear Modulus**.

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 calculate and display 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:

$$[I_x, I_y] = \left[\int_A (y - y_c)^2 dA, \int_A (x - x_c)^2 dA\right],$$

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

$$I_{xy} = \int_A (x - x_c)(y - y_c) dA,$$

- I_p — Centroidal polar moment of area:

$$I_P = I_x + I_y,$$

where x_c and y_c are the coordinates of the centroid.

Geometric Centroid — Centroid of beam cross section
`[0 0] m` (default) | 2-by-1 array

Centroid of the beam cross section, specified as a 2-by-1 array. The array specifies the x and y coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Bending Centroid — Intersection of neutral axis and cross section
`[0 0] m` (default) | 2-by-1 array

Intersection of the neutral axis and cross section, specified as a 2-by-1 array. The array specifies the x and y coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Shear Center — Beam shear center
[0 0] `m` (default) | 2-by-1 array

Beam shear center, specified as a 2-by-1 array. The array specifies the x and y coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Axial Rigidity — Resistance against axial deformation
1 `Pa*m^2` (default) | scalar

Resistance against the deformation along the beam longitudinal direction, specified as a scalar. This parameter specifies the S^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Flexural Rigidity — Resistance against bending deformations
[1 1] `Pa*m^4` (default) | 2-by-1 array

Resistance against bending deformations, specified as a 2-by-1 array. The array specifies the H_x^b and H_y^b elements of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Cross Flexural Rigidity — Resistance against bending deformation about x-axis in response to bending moment about the y-axis
0 `Pa*m^4` (default) | scalar

Resistance against the bending deformation about the x-axis in response to bending moment about the y-axis, specified as a scalar. This parameter specifies the H_{xy}^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Torsional Rigidity — Resistance against twisting
1 `Pa*m^4` (default) | scalar

Resistance against twisting about the longitudinal axis of the beam, specified as a scalar. This parameter specifies the H_z^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Center of Mass — Center of Mass
[0 0] m (default) | 2-by-1 array

Center of mass, specified as a 2-by-1 array. The array specifies the x and y coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Per Unit Length — Mass per unit length
1 kg/m (default) | scalar

Mass per unit length, specified as a scalar. This parameter specifies the m^c element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Moment of Inertia Density — Resistance against rotation about x and y axes
[1 1] kg*m (default) | 2-by-1 array

Resistance against rotation about the x and y axes, specified as a 2-by-1 array. The array specifies the i_x^c and i_y^c elements of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Product of Inertia Density — Resistance against rotation about x-axis due to moment about y-axis
0 kg*m (default) | scalar

Resistance against rotation about the x-axis due to the moment about the y-axis, specified as a scalar. This parameter specifies the i_{xy}^c element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Damping

Type — Type of damping method
`Proportional` (default) | `Uniform Modal` | `None`

Damping method to apply to the beam:

- Select `None` to model undamped beams.
- Select `Proportional` to apply the proportional (or Rayleigh) damping method. This method defines the damping matrix $[C]$ as a linear combination of the mass matrix $[M]$ and stiffness matrix $[K]$:

  $$[C] = \alpha[M] + \beta[K],$$

  where α and β are the scalar coefficients.

- Select `Uniform Modal` to apply the uniform modal damping method. This method applies a single damping ratio to all the vibration modes of the beam. The larger the value, the faster vibrations decay.

Mass Coefficient — Coefficient of mass matrix
`0 1/s` (default) | nonnegative scalar

Coefficient, α, of the mass matrix. This parameter defines damping proportional to the mass matrix $[M]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Stiffness Coefficient — Coefficient of stiffness matrix
`0.001 s` (default) | nonnegative scalar

Coefficient, β, of the stiffness matrix. This parameter defines damping proportional to the stiffness matrix $[K]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Damping Ratio — Damping ratio for uniform modal damping method
0.01 (default) | unitless nonnegative scalar

Damping ratio, ζ, applied to all beam vibration modes in the uniform modal damping model. The larger the value, the faster beam vibrations decay.

- Use $\zeta = 0$ to model undamped beams.
- Use $\zeta < 1$ to model underdamped beams.
- Use $\zeta = 1$ to model critically damped beams.
- Use $\zeta > 1$ to model overdamped beams.

Dependencies

To enable this parameter, set **Type** to `Uniform Modal`.

Data Types: `double`

Discretization

Number of Elements — Number of beam finite elements
1 (default) | positive integer

Number of finite elements in the beam model. Increasing the number of elements always improves accuracy of the simulation. But practically, at some point, the increase in accuracy is negligible when there are many elements. Additionally, a higher number of elements increases the computational cost and slows down the speed of the simulation.

Fidelity

Reduction — Method to model flexible bodies
`None` (default) | `Modally Reduced`

Method to use to model flexible bodies, specified as `None` or `Modally Reduced`. Set the parameter to `None` to use full nodal elastic coordinates generated by the finite-element method or set the parameter to `Modally Reduced` to use the modal transformation method to reduce the elastic coordinates of the body. For both settings, the block uses the floating frame of the reference formulation [1-2] to couple the body with its elastic deformation.

Number of Retained Modes — Retained modes
1 (default) | nonnegative integer

Retained modes, specified as an integer in range [0, *n*], where *n* is the number of elastic degrees of freedom of the body. If you set the number to 0 the flexible body is treated as a rigid body.

Dependencies

To enable this parameter, set **Reduction** to `Modally Reduced`.

Graphic

Type — Graphic to use in the visualization of beam
`From Geometry` (default) | `None`

Type of the visual representation of the beam, specified as `From Geometry` or `None`. Set the parameter to `From Geometry` to show the visual representation of the beam. Set the parameter to `None` to hide the beam in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered beam and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered beam due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered beam.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the beam itself. When a beam has a emissive color, the beam can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Frames

Show Port A — Show port A for connection to other blocks
on (default) | off

Select to expose the **A** port.

Show Port B — Show port B for connection to other blocks
on (default) | off

Select to expose the **B** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button  to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the beam block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** for the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the undeformed beam.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected undeformed geometry feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

  - **Along Reference Frame Axis**: Selects an axis of the reference frame of the undeformed beam.
  - **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the undeformed beam. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button  to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button  to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2020a

References

[1] Shabana, Ahmed A. *Dynamics of Multibody Systems*. Fourth edition. New York: Cambridge University Press, 2014.

[2] Agrawal, Om P., and Ahmed A. Shabana. “Dynamic Analysis of Multibody Systems Using Component Modes.” *Computers & Structures* 21, no. 6 (January 1985): 1303–12. https://doi.org/10.1016/0045-7949(85)90184-1.

Extended Capabilities

C/C++ Code Generation

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

See Also

Extruded Solid | Reduced Order Flexible Solid | Flexible Channel Beam | Flexible Cylindrical Beam | Flexible I Beam | Flexible Rectangular Beam | Flexible T Beam | General Flexible Beam | Rigid Transform

Topics

“Overview of Flexible Beams”
“Modeling Bodies”
“Manipulate the Color of a Solid”

Flexible Channel Beam

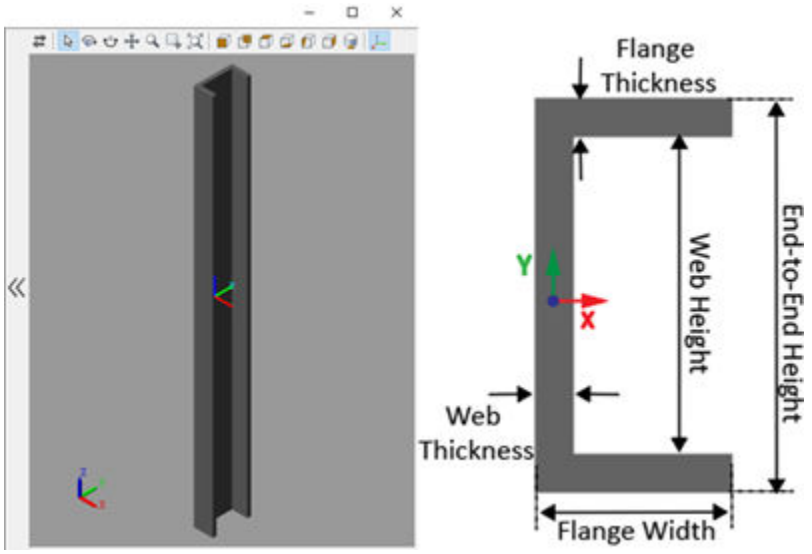
Channel beam with elastic properties for deformation

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

Description

The Flexible Channel Beam block models a slender beam with a C-shaped cross-section, also known as a C-beam. The C-beam consists of two horizontal components, known as flanges, that are connected by one vertical component, which is called a web. The C-beam can have small and linear deformations. These deformations include extension, bending, and torsion. The block calculates the beam cross-sectional properties, such as the axial, flexural, and torsional rigidities, based on the geometry and material properties that you specify.

The geometry of the C-beam is an extrusion of its cross-section. The beam cross-section, defined in the *xy*-plane, is extruded along the *z*-axis. To define the cross-section, you can specify its dimensions in the **Geometry** section of the block dialog box. The figure shows a C-beam and its cross-section. The reference frame of the beam is located at the centroid of the web.



This block supports two damping methods and a discretization option to increase the accuracy of the modeling. For more information, see “Overview of Flexible Beams”.

Stiffness and Inertia Properties

The block provides two ways to specify the stiffness and inertia properties for a beam. To model a beam made of homogeneous, isotropic, and linearly elastic material, in the **Stiffness and Inertia** section, set the **Type** parameter to `Calculate from Geometry`. Then specify the density, Young’s modulus, and Poisson’s ratio or shear modulus. See the **Derived Values** parameter for more information about the calculated stiffness and inertia properties.

Alternatively, you can manually specify the stiffness and inertia properties, such as flexural rigidity and mass moment of inertia density, by setting the **Type** parameter to `Custom`. Use this option to model a beam that is made of anisotropic materials. This option decouples the mechanical properties from the beam cross section so you can specify desired mechanical properties without capturing the details of the exact cross section, such as fillet, rounds, chamfers, and tapers.

The manually entered stiffness properties must be calculated with respect to the frame located at the bending centroid. The frame must be in the same orientation as the beam reference frame. The stiffness matrix is

$$\bar{S}^b = \begin{bmatrix} S^b & 0 & 0 & 0 \\ 0 & H_x^b & -H_{xy}^b & 0 \\ 0 & -H_{xy}^b & H_y^b & 0 \\ 0 & 0 & 0 & H_z^b \end{bmatrix}$$

where:

- S^b is the axial stiffness along the beam.
- H_x^b is the centroidal bending stiffness about the *x*-axis.
- H_y^b is the centroidal bending stiffness about the *y*-axis.
- H_z^b is the torsional stiffness.
- H_{xy}^b is the centroidal cross bending stiffness.

The manually entered inertia properties must be calculated with respect to a frame located at the center of mass. The frame must be in the same orientation as the beam reference frame. The mass matrix includes rotatory inertia

$$\bar{M}^c = \begin{bmatrix} m^c & 0 & 0 & 0 & 0 & 0 \\ 0 & m^c & 0 & 0 & 0 & 0 \\ 0 & 0 & m^c & 0 & 0 & 0 \\ 0 & 0 & 0 & i_x^c & -i_{xy}^c & 0 \\ 0 & 0 & 0 & -i_{xy}^c & i_y^c & 0 \\ 0 & 0 & 0 & 0 & 0 & i_z^c \end{bmatrix}$$

where:

- m^c is the mass per unit length.
- i_x^c is the mass moment of inertia density about the *x*-axis.
- i_y^c is the mass moment of inertia density about the *y*-axis.
- i_z^c is the polar mass moment of inertia density.
- i_{xy}^c is the mass product of inertia density.

The beam block uses the classical beam theory where the relation between sectional strains and stress resultants is

$$\begin{bmatrix} \bar{F}_z \\ \bar{M}_x \\ \bar{M}_y \\ \bar{M}_z \end{bmatrix} = \bar{S}^b \begin{bmatrix} \bar{\varepsilon}_z \\ \bar{\kappa}_x \\ \bar{\kappa}_y \\ \bar{\kappa}_z \end{bmatrix}$$

where:

- $\bar{F}_z$ is the axial force along the beam.
- $\bar{M}_x$ is the bending moment about the *x*-axis.
- $\bar{M}_y$ is the bending moment about the *y*-axis.
- $\bar{M}_z$ is the torsional moment about the *z*-axis.
- $\bar{\varepsilon}_z$ is the axial strain along the beam.
- $\bar{\kappa}_x$ is the bending curvature about the *x*-axis.
- $\bar{\kappa}_y$ is the bending curvature about the *y*-axis.
- $\bar{\kappa}_z$ is the torsional twist about the *z*-axis.

Ports

Frame

A — Connection frame
frame

Frame 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 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

End-to-End Height — Distance between outer faces of flanges
`1 m` (default) | positive scalar

Distance between the outer faces of the two flanges. The `End-to-End Height` is also known as the beam depth.

Note The `End-to-End Height` must be larger than the sum of the two flange thicknesses.

Web Thickness — Distance between faces of web
0.1 m (default) | positive scalar

Distance between the two faces of the web.

Note The Web Thickness must be smaller than the width of the flanges.

Top Flange Width — Distance between ends of top flange
0.5 m (default) | positive scalar

Distance between the two ends of the top flange.

Top Flange Thickness — Distance between faces of top flange
0.1 m (default) | positive scalar

Distance between the two faces of the top flange.

Bottom Flange Width — Distance between ends of bottom flange
0.5 m (default) | positive scalar

Distance between the two ends of the bottom flange.

Bottom Flange Thickness — Distance between faces of bottom flange
0.1 m (default) | positive scalar

Distance between the two faces of the bottom flange.

Length — Extrusion length of beam
10 m (default) | positive scalar

Extrusion length of the beam. The beam is modeled 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

Type — Method to use to specify stiffness and inertia properties
Calculate from Geometry (default) | Custom

Method to use to specify the stiffness and inertia properties, specified as Calculate from Geometry or Custom.

When you set the parameter to Calculate from Geometry, the block calculates the stiffness and inertia properties based on the specified density, Young's modulus, and Poisson's ratio or shear modulus. Set the parameter to Custom to manually specify the stiffness and inertia properties.

Density — Mass per unit volume of material
2700 kg/m^3 (default) | positive scalar

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

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

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.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Young's Modulus — Ratio of axial stress to axial strain
`70 GPa` (default) | positive scalar

Young's modulus of elasticity of the beam. The greater its value, the stronger the resistance to bending and axial deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Poisson's Ratio — Ratio of transverse to longitudinal strains
0.33 (default) | scalar in the range [0, 0.5)

Poisson's ratio of the beam. The value specified must be greater than or equal to `0` and smaller than `0.5`. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's Modulus and Poisson's Ratio**.

Shear Modulus — Ratio of shear stress to engineering shear strain
`26 GPa` (default) | positive scalar

Shear modulus (or modulus of rigidity) of the beam. The greater its value, the stronger the resistance to torsional deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's and Shear Modulus**.

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 calculate and display 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:

$$[I_x, I_y] = \left[\int_A (y - y_c)^2 dA, \int_A (x - x_c)^2 dA\right],$$

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

$$I_{xy} = \int_A (x - x_c)(y - y_c) dA,$$

- I_p — Centroidal polar moment of area:

$$I_P = I_x + I_y,$$

where x_c and y_c are the coordinates of the centroid.

Geometric Centroid — Centroid of beam cross section
[0 0] m (default) | 2-by-1 array

Centroid of the beam cross section, specified as a 2-by-1 array. The array specifies the x and y coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Bending Centroid — Intersection of neutral axis and cross section
`[0 0] m` (default) | 2-by-1 array

Intersection of the neutral axis and cross section, specified as a 2-by-1 array. The array specifies the *x* and *y* coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Shear Center — Beam shear center
`[0 0] m` (default) | 2-by-1 array

Beam shear center, specified as a 2-by-1 array. The array specifies the *x* and *y* coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Axial Rigidity — Resistance against axial deformation
`1 Pa*m^2` (default) | scalar

Resistance against the deformation along the beam longitudinal direction, specified as a scalar. This parameter specifies the S^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Flexural Rigidity — Resistance against bending deformations
`[1 1] Pa*m^4` (default) | 2-by-1 array

Resistance against bending deformations, specified as a 2-by-1 array. The array specifies the H_x^b and H_y^b elements of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Cross Flexural Rigidity — Resistance against bending deformation about x-axis in response to bending moment about the y-axis
`0 Pa*m^4` (default) | scalar

Resistance against the bending deformation about the *x*-axis in response to bending moment about the *y*-axis, specified as a scalar. This parameter specifies the H_{xy}^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Torsional Rigidity — Resistance against twisting
`1 Pa*m^4` (default) | scalar

Resistance against twisting about the longitudinal axis of the beam, specified as a scalar. This parameter specifies the H_z^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Center of Mass — Center of Mass
`[0 0] m` (default) | 2-by-1 array

Center of mass, specified as a 2-by-1 array. The array specifies the x and y coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Per Unit Length — Mass per unit length
`1 kg/m` (default) | scalar

Mass per unit length, specified as a scalar. This parameter specifies the m^c element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Moment of Inertia Density — Resistance against rotation about x and y axes
`[1 1] kg*m` (default) | 2-by-1 array

Resistance against rotation about the x and y axes, specified as a 2-by-1 array. The array specifies the i_x^c and i_y^c elements of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Product of Inertia Density — Resistance against rotation about x-axis due to moment about y-axis
`0 kg*m` (default) | scalar

Resistance against rotation about the x-axis due to the moment about the y-axis, specified as a scalar. This parameter specifies the i_{xy}^c element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Damping

Type — Type of damping method
`Proportional` (default) | `Uniform Modal` | `None`

Damping method to apply to the beam:

- Select `None` to model undamped beams.

- Select `Proportional` to apply the proportional (or Rayleigh) damping method. This method defines the damping matrix $[C]$ as a linear combination of the mass matrix $[M]$ and stiffness matrix $[K]$:

  $$[C] = \alpha[M] + \beta[K],$$

  where α and β are the scalar coefficients.
- Select `Uniform Modal` to apply the uniform modal damping method. This method applies a single damping ratio to all the vibration modes of the beam. The larger the value, the faster vibrations decay.

Mass Coefficient — Coefficient of mass matrix
0 `1/s` (default) | nonnegative scalar

Coefficient, α, of the mass matrix. This parameter defines damping proportional to the mass matrix $[M]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Stiffness Coefficient — Coefficient of stiffness matrix
0.001 `s` (default) | nonnegative scalar

Coefficient, β, of the stiffness matrix. This parameter defines damping proportional to the stiffness matrix $[K]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Damping Ratio — Damping ratio for uniform modal damping method
0.01 (default) | unitless nonnegative scalar

Damping ratio, ζ, applied to all beam vibration modes in the uniform modal damping model. The larger the value, the faster beam vibrations decay.

- Use $\zeta = 0$ to model undamped beams.
- Use $\zeta < 1$ to model underdamped beams.
- Use $\zeta = 1$ to model critically damped beams.
- Use $\zeta > 1$ to model overdamped beams.

Dependencies

To enable this parameter, set **Type** to `Uniform Modal`.

Data Types: `double`

Discretization

Number of Elements — Number of beam finite elements
1 (default) | positive integer

Number of finite elements in the beam model. Increasing the number of elements always improves accuracy of the simulation. But practically, at some point, the increase in accuracy is negligible when

there are many elements. Additionally, a higher number of elements increases the computational cost and slows down the speed of the simulation.

Fidelity

Reduction — Method to model flexible bodies
`None` (default) | `Modally Reduced`

Method to use to model flexible bodies, specified as `None` or `Modally Reduced`. Set the parameter to `None` to use full nodal elastic coordinates generated by the finite-element method or set the parameter to `Modally Reduced` to use the modal transformation method to reduce the elastic coordinates of the body. For both settings, the block uses the floating frame of the reference formulation [1-2] to couple the body with its elastic deformation.

Number of Retained Modes — Retained modes
1 (default) | nonnegative integer

Retained modes, specified as an integer in range [0, *n*], where *n* is the number of elastic degrees of freedom of the body. If you set the number to 0 the flexible body is treated as a rigid body.

Dependencies

To enable this parameter, set **Reduction** to `Modally Reduced`.

Graphic

Type — Graphic to use in the visualization of beam
`From Geometry` (default) | `None`

Type of the visual representation of the beam, specified as `From Geometry` or `None`. Set the parameter to `From Geometry` to show the visual representation of the beam. Set the parameter to `None` to hide the beam in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered beam and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered beam due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered beam.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the beam itself. When a beam has a emissive color, the beam can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Frames

Show Port A — Show port A for connection to other blocks
on (default) | off

Select to expose the **A** port.

Show Port B — Show port B for connection to other blocks
on (default) | off

Select to expose the **B** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the beam block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** for the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the undeformed beam.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected undeformed geometry feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

  - **Along Reference Frame Axis**: Selects an axis of the reference frame of the undeformed beam.
  - **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the undeformed beam. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2020a

References

[1] Shabana, Ahmed A. *Dynamics of Multibody Systems*. Fourth edition. New York: Cambridge University Press, 2014.

[2] Agrawal, Om P., and Ahmed A. Shabana. “Dynamic Analysis of Multibody Systems Using Component Modes.” *Computers & Structures* 21, no. 6 (January 1985): 1303–12. https://doi.org/10.1016/0045-7949(85)90184-1.

Extended Capabilities

C/C++ Code Generation

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

See Also

Flexible Angle Beam | Flexible Cylindrical Beam | Flexible I Beam | Flexible Rectangular Beam | Flexible T Beam | General Flexible Beam | Extruded Solid | Reduced Order Flexible Solid | Rigid Transform

Topics

“Overview of Flexible Beams”
“Modeling Bodies”
“Manipulate the Color of a Solid”

Flexible Cylindrical Beam

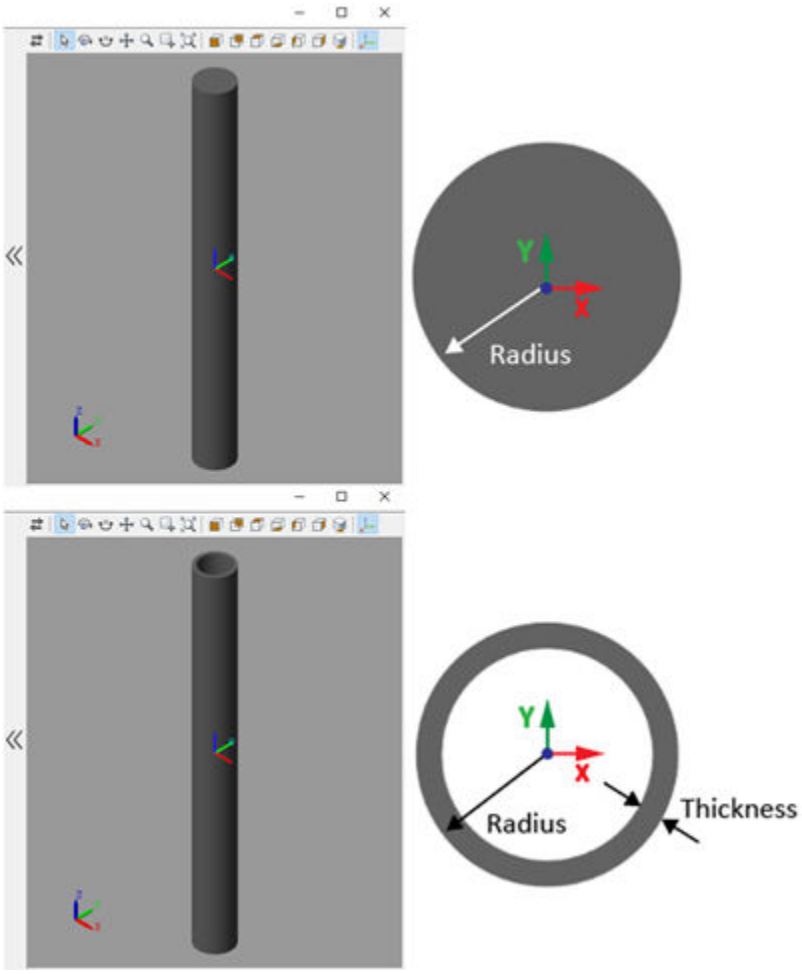
Cylindrical beam with elastic properties for deformation

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

Description

The Flexible Cylindrical Beam block models a slender beam with a circular cross-section that can be solid or hollow. The cylindrical beam can have small and linear deformations. These deformations include extension, bending, and torsion. The block calculates the beam cross-sectional properties, such as the axial, flexural, and torsional rigidities, based on the geometry and material properties that you specify.

The geometry of the cylindrical beam is an extrusion of its cross-section. The beam cross-section, defined in the *xy*-plane, is extruded along the *z*-axis. To define the cross-section, you can specify its dimensions in the **Geometry** section of the block dialog box. The figure shows a solid beam and a hollow beam. The reference frame is located at the centroid of the beam.



This block supports two damping methods and a discretization option to increase the accuracy of the modeling. For more information, see “Overview of Flexible Beams”.

Stiffness and Inertia Properties

The block provides two ways to specify the stiffness and inertia properties for a beam. To model a beam made of homogeneous, isotropic, and linearly elastic material, in the **Stiffness and Inertia** section, set the **Type** parameter to `Calculate from Geometry`. Then specify the density, Young’s modulus, and Poisson’s ratio or shear modulus. See the **Derived Values** parameter for more information about the calculated stiffness and inertia properties.

Alternatively, you can manually specify the stiffness and inertia properties, such as flexural rigidity and mass moment of inertia density, by setting the **Type** parameter to `Custom`. Use this option to model a beam that is made of anisotropic materials. This option decouples the mechanical properties from the beam cross section so you can specify desired mechanical properties without capturing the details of the exact cross section, such as fillet, rounds, chamfers, and tapers.

The manually entered stiffness properties must be calculated with respect to the frame located at the bending centroid. The frame must be in the same orientation as the beam reference frame. The stiffness matrix is

$$\bar{S}^b = \begin{bmatrix} S^b & 0 & 0 & 0 \\ 0 & H_x^b & -H_{xy}^b & 0 \\ 0 & -H_{xy}^b & H_y^b & 0 \\ 0 & 0 & 0 & H_z^b \end{bmatrix}$$

where:

- S^b is the axial stiffness along the beam.
- H_x^b is the centroidal bending stiffness about the *x*-axis.
- H_y^b is the centroidal bending stiffness about the *y*-axis.
- H_z^b is the torsional stiffness.
- H_{xy}^b is the centroidal cross bending stiffness.

The manually entered inertia properties must be calculated with respect to a frame located at the center of mass. The frame must be in the same orientation as the beam reference frame. The mass matrix includes rotatory inertia

$$\bar{M}^c = \begin{bmatrix} m^c & 0 & 0 & 0 & 0 & 0 \\ 0 & m^c & 0 & 0 & 0 & 0 \\ 0 & 0 & m^c & 0 & 0 & 0 \\ 0 & 0 & 0 & i_x^c & -i_{xy}^c & 0 \\ 0 & 0 & 0 & -i_{xy}^c & i_y^c & 0 \\ 0 & 0 & 0 & 0 & 0 & i_z^c \end{bmatrix}$$

where:

- m^c is the mass per unit length.
- i_x^c is the mass moment of inertia density about the *x*-axis.
- i_y^c is the mass moment of inertia density about the *y*-axis.
- i_z^c is the polar mass moment of inertia density.
- i_{xy}^c is the mass product of inertia density.

The beam block uses the classical beam theory where the relation between sectional strains and stress resultants is

$$\begin{bmatrix} \bar{F}_z \\ \bar{M}_x \\ \bar{M}_y \\ \bar{M}_z \end{bmatrix} = \bar{S}^b \begin{bmatrix} \bar{\varepsilon}_z \\ \bar{\kappa}_x \\ \bar{\kappa}_y \\ \bar{\kappa}_z \end{bmatrix}$$

where:

- $\bar{F}_z$ is the axial force along the beam.
- $\bar{M}_x$ is the bending moment about the x-axis.
- $\bar{M}_y$ is the bending moment about the y-axis.
- $\bar{M}_z$ is the torsional moment about the z-axis.
- $\bar{\varepsilon}_z$ is the axial strain along the beam.
- $\bar{\kappa}_x$ is the bending curvature about the x-axis.
- $\bar{\kappa}_y$ is the bending curvature about the y-axis.
- $\bar{\kappa}_z$ is the torsional twist about the z-axis.

Ports

Frame

A — Connection frame
frame

Frame 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 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

Type — Choice of solid or hollow cross-section
`Solid` (default) | `Hollow`

Choice of solid or hollow cross-section:

- Select `Solid` to model a beam with a solid cross-section.
- Select `Hollow` to model a beam with a hollow cross-section.

Radius — Distance between central axis and outer surface of beam
0.5 `m` (default) | positive scalar

Distance between the central axis and the outer surface of the beam.

Thickness — Wall thickness of hollow cross-section
0.1 `m` (default) | positive scalar

Wall thickness of the hollow cross-section. This parameter specifies the distance between the inner and outer surfaces of the beam.

Dependencies

To enable this parameter, set **Type** to `Hollow`.

Length — Extrusion length of beam
`10 m` (default) | positive scalar

Extrusion length of the beam. The beam is modeled 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

Type — Method to use to specify stiffness and inertia properties
`Calculate from Geometry` (default) | `Custom`

Method to use to specify the stiffness and inertia properties, specified as `Calculate from Geometry` or `Custom`.

When you set the parameter to `Calculate from Geometry`, the block calculates the stiffness and inertia properties based on the specified density, Young's modulus, and Poisson's ratio or shear modulus. Set the parameter to `Custom` to manually specify the stiffness and inertia properties.

Density — Mass per unit volume of material
`2700 kg/m^3` (default) | positive scalar

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

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

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.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Young's Modulus — Ratio of axial stress to axial strain
`70 GPa` (default) | positive scalar

Young's modulus of elasticity of the beam. The greater its value, the stronger the resistance to bending and axial deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Poisson's Ratio — Ratio of transverse to longitudinal strains
0.33 (default) | scalar in the range [0, 0.5)

Poisson's ratio of the beam. The value specified must be greater than or equal to `0` and smaller than `0.5`. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's Modulus and Poisson's Ratio**.

Shear Modulus — Ratio of shear stress to engineering shear strain
26 `GPa` (default) | positive scalar

Shear modulus (or modulus of rigidity) of the beam. The greater its value, the stronger the resistance to torsional deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's and Shear Modulus**.

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 calculate and display 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**: [ρI_x, ρI_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:

$$[I_x, I_y] = \left[\int_A (y - y_c)^2 dA, \int_A (x - x_c)^2 dA\right],$$

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

$$I_{xy} = \int_A (x - x_c)(y - y_c) dA,$$

- I_p — Centroidal polar moment of area:

$$I_P = I_x + I_y,$$

where x_c and y_c are the coordinates of the centroid.

Geometric Centroid — Centroid of beam cross section
`[0 0] m` (default) | 2-by-1 array

Centroid of the beam cross section, specified as a 2-by-1 array. The array specifies the x and y coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Bending Centroid — Intersection of neutral axis and cross section
`[0 0] m` (default) | 2-by-1 array

Intersection of the neutral axis and cross section, specified as a 2-by-1 array. The array specifies the x and y coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Shear Center — Beam shear center
`[0 0] m` (default) | 2-by-1 array

Beam shear center, specified as a 2-by-1 array. The array specifies the x and y coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Axial Rigidity — Resistance against axial deformation
`1 Pa*m^2` (default) | scalar

Resistance against the deformation along the beam longitudinal direction, specified as a scalar. This parameter specifies the S^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Flexural Rigidity — Resistance against bending deformations
`[1 1] Pa*m^4` (default) | 2-by-1 array

Resistance against bending deformations, specified as a 2-by-1 array. The array specifies the H_x^b and H_y^b elements of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Cross Flexural Rigidity — Resistance against bending deformation about x-axis in response to bending moment about the y-axis
`0 Pa*m^4` (default) | scalar

Resistance against the bending deformation about the *x*-axis in response to bending moment about the *y*-axis, specified as a scalar. This parameter specifies the H_{xy}^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Torsional Rigidity — Resistance against twisting
`1 Pa*m^4` (default) | scalar

Resistance against twisting about the longitudinal axis of the beam, specified as a scalar. This parameter specifies the H_z^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Center of Mass — Center of Mass
`[0 0] m` (default) | 2-by-1 array

Center of mass, specified as a 2-by-1 array. The array specifies the *x* and *y* coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Per Unit Length — Mass per unit length
`1 kg/m` (default) | scalar

Mass per unit length, specified as a scalar. This parameter specifies the m^c element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Moment of Inertia Density — Resistance against rotation about x and y axes
`[1 1] kg*m` (default) | 2-by-1 array

Resistance against rotation about the *x* and *y* axes, specified as a 2-by-1 array. The array specifies the i_x^C and i_y^C elements of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Product of Inertia Density — Resistance against rotation about x-axis due to moment about y-axis
`0 kg*m` (default) | scalar

Resistance against rotation about the *x*-axis due to the moment about the *y*-axis, specified as a scalar. This parameter specifies the i_{xy}^C element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Damping

Type — Type of damping method
`Proportional` (default) | `Uniform Modal` | `None`

Damping method to apply to the beam:

- Select `None` to model undamped beams.
- Select `Proportional` to apply the proportional (or Rayleigh) damping method. This method defines the damping matrix $[C]$ as a linear combination of the mass matrix $[M]$ and stiffness matrix $[K]$:

  $$[C] = \alpha[M] + \beta[K],$$

  where α and β are the scalar coefficients.
- Select `Uniform Modal` to apply the uniform modal damping method. This method applies a single damping ratio to all the vibration modes of the beam. The larger the value, the faster vibrations decay.

Mass Coefficient — Coefficient of mass matrix
`0 1/s` (default) | nonnegative scalar

Coefficient, α, of the mass matrix. This parameter defines damping proportional to the mass matrix $[M]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Stiffness Coefficient — Coefficient of stiffness matrix
`0.001 s` (default) | nonnegative scalar

Coefficient, β, of the stiffness matrix. This parameter defines damping proportional to the stiffness matrix [K].

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Damping Ratio — Damping ratio for uniform modal damping method
0.01 (default) | unitless nonnegative scalar

Damping ratio, ζ, applied to all beam vibration modes in the uniform modal damping model. The larger the value, the faster beam vibrations decay.

- Use $\zeta = 0$ to model undamped beams.
- Use $\zeta < 1$ to model underdamped beams.
- Use $\zeta = 1$ to model critically damped beams.
- Use $\zeta > 1$ to model overdamped beams.

Dependencies

To enable this parameter, set **Type** to `Uniform Modal`.

Data Types: `double`

Discretization

Number of Elements — Number of beam finite elements
1 (default) | positive integer

Number of finite elements in the beam model. Increasing the number of elements always improves accuracy of the simulation. But practically, at some point, the increase in accuracy is negligible when there are many elements. Additionally, a higher number of elements increases the computational cost and slows down the speed of the simulation.

Fidelity

Reduction — Method to model flexible bodies
`None` (default) | `Modally Reduced`

Method to use to model flexible bodies, specified as `None` or `Modally Reduced`. Set the parameter to `None` to use full nodal elastic coordinates generated by the finite-element method or set the parameter to `Modally Reduced` to use the modal transformation method to reduce the elastic coordinates of the body. For both settings, the block uses the floating frame of the reference formulation [1-2] to couple the body with its elastic deformation.

Number of Retained Modes — Retained modes
1 (default) | nonnegative integer

Retained modes, specified as an integer in range [0, n], where n is the number of elastic degrees of freedom of the body. If you set the number to 0 the flexible body is treated as a rigid body.

Dependencies

To enable this parameter, set **Reduction** to `Modally Reduced`.

Graphic

Type — Graphic to use in the visualization of beam
`From Geometry` (default) | `None`

Type of the visual representation of the beam, specified as `From Geometry` or `None`. Set the parameter to `From Geometry` to show the visual representation of the beam. Set the parameter to `None` to hide the beam in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered beam and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered beam due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered beam.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the beam itself. When a beam has a emissive color, the beam can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Frames

Show Port A — Show port A for connection to other blocks
on (default) | off

Select to expose the **A** port.

Show Port B — Show port B for connection to other blocks
on (default) | off

Select to expose the **B** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button  to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the beam block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** for the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the undeformed beam.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected undeformed geometry feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining

two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

- **Along Reference Frame Axis**: Selects an axis of the reference frame of the undeformed beam.
- **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the undeformed beam. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button  to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button  to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2020a

References

[1] Shabana, Ahmed A. *Dynamics of Multibody Systems*. Fourth edition. New York: Cambridge University Press, 2014.

[2] Agrawal, Om P., and Ahmed A. Shabana. “Dynamic Analysis of Multibody Systems Using Component Modes.” *Computers & Structures* 21, no. 6 (January 1985): 1303–12. https://doi.org/10.1016/0045-7949(85)90184-1.

Extended Capabilities

C/C++ Code Generation

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

See Also

Flexible Angle Beam | Flexible Channel Beam | Flexible I Beam | Flexible Rectangular Beam | Flexible T Beam | General Flexible Beam | Cylindrical Solid | Revolved Solid | Reduced Order Flexible Solid | Rigid Transform

Topics

“Overview of Flexible Beams”
“Modeling Bodies”

“Manipulate the Color of a Solid”

Flexible I Beam

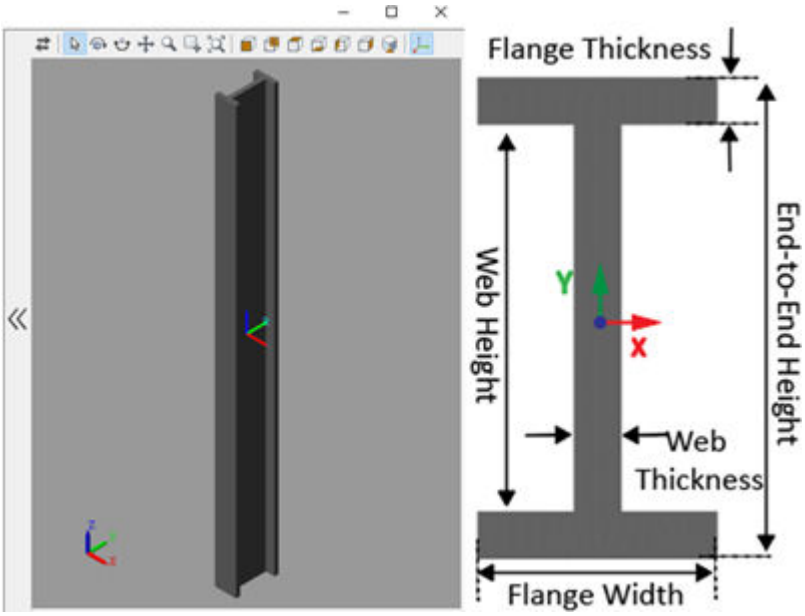
I-beam with elastic properties for deformation

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

Description

The Flexible I Beam block models a slender beam with an I-shaped cross-section, also known as an I-beam. The I-beam consists of two horizontal components, known as flanges, that are connected by one vertical component, which is called a web. The I-beam can have small and linear deformations. These deformations include extension, bending, and torsion. The block calculates the beam cross-sectional properties, such as the axial, flexural, and torsional rigidities, based on the geometry and material properties that you specify.

The geometry of the I-beam is an extrusion of its cross-section. The beam cross-section, defined in the *xy*-plane, is extruded along the *z*-axis. To define the cross-section, you can specify its dimensions in the **Geometry** section of the block dialog box. The figure shows an I-beam and its cross-section. The reference frame of the beam is located at the centroid of the web.



This block supports two damping methods and a discretization option to increase the accuracy of the modeling. For more information, see “Overview of Flexible Beams”.

Stiffness and Inertia Properties

The block provides two ways to specify the stiffness and inertia properties for a beam. To model a beam made of homogeneous, isotropic, and linearly elastic material, in the **Stiffness and Inertia**

section, set the **Type** parameter to `Calculate from Geometry`. Then specify the density, Young's modulus, and Poisson's ratio or shear modulus. See the **Derived Values** parameter for more information about the calculated stiffness and inertia properties.

Alternatively, you can manually specify the stiffness and inertia properties, such as flexural rigidity and mass moment of inertia density, by setting the **Type** parameter to `Custom`. Use this option to model a beam that is made of anisotropic materials. This option decouples the mechanical properties from the beam cross section so you can specify desired mechanical properties without capturing the details of the exact cross section, such as fillet, rounds, chamfers, and tapers.

The manually entered stiffness properties must be calculated with respect to the frame located at the bending centroid. The frame must be in the same orientation as the beam reference frame. The stiffness matrix is

$$\bar{S}^b = \begin{bmatrix} S^b & 0 & 0 & 0 \\ 0 & H_x^b & -H_{xy}^b & 0 \\ 0 & -H_{xy}^b & H_y^b & 0 \\ 0 & 0 & 0 & H_z^b \end{bmatrix}$$

where:

- S^b is the axial stiffness along the beam.
- H_x^b is the centroidal bending stiffness about the *x*-axis.
- H_y^b is the centroidal bending stiffness about the *y*-axis.
- H_z^b is the torsional stiffness.
- H_{xy}^b is the centroidal cross bending stiffness.

The manually entered inertia properties must be calculated with respect to a frame located at the center of mass. The frame must be in the same orientation as the beam reference frame. The mass matrix includes rotatory inertia

$$\overline{M}^c = \begin{bmatrix} m^c & 0 & 0 & 0 & 0 & 0 \\ 0 & m^c & 0 & 0 & 0 & 0 \\ 0 & 0 & m^c & 0 & 0 & 0 \\ 0 & 0 & 0 & i_x^c & -i_{xy}^c & 0 \\ 0 & 0 & 0 & -i_{xy}^c & i_y^c & 0 \\ 0 & 0 & 0 & 0 & 0 & i_z^c \end{bmatrix}$$

where:

- m^c is the mass per unit length.
- i_x^c is the mass moment of inertia density about the *x*-axis.
- i_y^c is the mass moment of inertia density about the *y*-axis.

- i_z^C is the polar mass moment of inertia density.
- i_{xy}^C is the mass product of inertia density.

The beam block uses the classical beam theory where the relation between sectional strains and stress resultants is

$$\begin{bmatrix} \bar{F}_z \\ \bar{M}_x \\ \bar{M}_y \\ \bar{M}_z \end{bmatrix} = \bar{S}^b \begin{bmatrix} \bar{\varepsilon}_z \\ \bar{\kappa}_x \\ \bar{\kappa}_y \\ \bar{\kappa}_z \end{bmatrix}$$

where:

- $\bar{F}_z$ is the axial force along the beam.
- $\bar{M}_x$ is the bending moment about the *x*-axis.
- $\bar{M}_y$ is the bending moment about the *y*-axis.
- $\bar{M}_z$ is the torsional moment about the *z*-axis.
- $\bar{\varepsilon}_z$ is the axial strain along the beam.
- $\bar{\kappa}_x$ is the bending curvature about the *x*-axis.
- $\bar{\kappa}_y$ is the bending curvature about the *y*-axis.
- $\bar{\kappa}_z$ is the torsional twist about the *z*-axis.

Ports

Frame

A — Connection frame
frame

Frame 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 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

End-to-End Height — Distance between outer faces of flanges
`1 m` (default) | positive scalar

Distance between the outer faces of the two flanges. The `End-to-End Height` is also known as the beam depth.

Note The `End-to-End Height` must be larger than the sum of the two flange thicknesses.

Web Thickness — Distance between faces of web
0.1 `m` (default) | positive scalar

Distance between the two faces of the web.

Note The `Web Thickness` must be smaller than the width of the flanges.

Top Flange Width — Distance between ends of top flange
0.5 `m` (default) | positive scalar

Distance between the two ends of the top flange.

Top Flange Thickness — Distance between faces of top flange
0.1 `m` (default) | positive scalar

Distance between the two faces of the top flange.

Bottom Flange Width — Distance between ends of bottom flange
0.5 `m` (default) | positive scalar

Distance between the two ends of the bottom flange.

Bottom Flange Thickness — Distance between faces of bottom flange
0.1 `m` (default) | positive scalar

Distance between the two faces of the bottom flange.

Length — Extrusion length of beam
10 `m` (default) | positive scalar

Extrusion length of the beam. The beam is modeled 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

Type — Method to use to specify stiffness and inertia properties
`Calculate from Geometry` (default) | `Custom`

Method to use to specify the stiffness and inertia properties, specified as `Calculate from Geometry` or `Custom`.

When you set the parameter to `Calculate from Geometry`, the block calculates the stiffness and inertia properties based on the specified density, Young's modulus, and Poisson's ratio or shear modulus. Set the parameter to `Custom` to manually specify the stiffness and inertia properties.

Density — Mass per unit volume of material
2700 kg/m^3 (default) | positive scalar

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

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

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.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Young's Modulus — Ratio of axial stress to axial strain
70 GPa (default) | positive scalar

Young's modulus of elasticity of the beam. The greater its value, the stronger the resistance to bending and axial deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Poisson's Ratio — Ratio of transverse to longitudinal strains
0.33 (default) | scalar in the range [0, 0.5)

Poisson's ratio of the beam. The value specified must be greater than or equal to `0` and smaller than `0.5`. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's Modulus and Poisson's Ratio**.

Shear Modulus — Ratio of shear stress to engineering shear strain
26 GPa (default) | positive scalar

Shear modulus (or modulus of rigidity) of the beam. The greater its value, the stronger the resistance to torsional deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's and Shear Modulus**.

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 calculate and display 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:

$$[I_x, I_y] = \left[\int_A (y - y_c)^2 dA, \int_A (x - x_c)^2 dA\right],$$

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

$$I_{xy} = \int_A (x - x_c)(y - y_c) dA,$$

- I_p — Centroidal polar moment of area:

$$I_P = I_x + I_y,$$

where x_c and y_c are the coordinates of the centroid.

Geometric Centroid — Centroid of beam cross section
[0 0] m (default) | 2-by-1 array

Centroid of the beam cross section, specified as a 2-by-1 array. The array specifies the *x* and *y* coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Bending Centroid — Intersection of neutral axis and cross section
`[0 0] m` (default) | 2-by-1 array

Intersection of the neutral axis and cross section, specified as a 2-by-1 array. The array specifies the *x* and *y* coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Shear Center — Beam shear center
`[0 0] m` (default) | 2-by-1 array

Beam shear center, specified as a 2-by-1 array. The array specifies the *x* and *y* coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Axial Rigidity — Resistance against axial deformation
`1 Pa*m^2` (default) | scalar

Resistance against the deformation along the beam longitudinal direction, specified as a scalar. This parameter specifies the S^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Flexural Rigidity — Resistance against bending deformations
`[1 1] Pa*m^4` (default) | 2-by-1 array

Resistance against bending deformations, specified as a 2-by-1 array. The array specifies the H_x^b and H_y^b elements of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Cross Flexural Rigidity — Resistance against bending deformation about x-axis in response to bending moment about the y-axis
`0 Pa*m^4` (default) | scalar

Resistance against the bending deformation about the *x*-axis in response to bending moment about the *y*-axis, specified as a scalar. This parameter specifies the H_{xy}^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Torsional Rigidity — Resistance against twisting
1 Pa*m^4 (default) | scalar

Resistance against twisting about the longitudinal axis of the beam, specified as a scalar. This parameter specifies the H_z^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Center of Mass — Center of Mass
[0 0] m (default) | 2-by-1 array

Center of mass, specified as a 2-by-1 array. The array specifies the x and y coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Per Unit Length — Mass per unit length
1 kg/m (default) | scalar

Mass per unit length, specified as a scalar. This parameter specifies the m^c element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Moment of Inertia Density — Resistance against rotation about x and y axes
[1 1] kg*m (default) | 2-by-1 array

Resistance against rotation about the x and y axes, specified as a 2-by-1 array. The array specifies the i_x^c and i_y^c elements of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Product of Inertia Density — Resistance against rotation about x-axis due to moment about y-axis
0 kg*m (default) | scalar

Resistance against rotation about the x-axis due to the moment about the y-axis, specified as a scalar. This parameter specifies the i_{xy}^c element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Damping

Type — Type of damping method
`Proportional` (default) | `Uniform Modal` | `None`

Damping method to apply to the beam:

- Select `None` to model undamped beams.
- Select `Proportional` to apply the proportional (or Rayleigh) damping method. This method defines the damping matrix $[C]$ as a linear combination of the mass matrix $[M]$ and stiffness matrix $[K]$:

  $$[C] = \alpha[M] + \beta[K],$$

  where α and β are the scalar coefficients.
- Select `Uniform Modal` to apply the uniform modal damping method. This method applies a single damping ratio to all the vibration modes of the beam. The larger the value, the faster vibrations decay.

Mass Coefficient — Coefficient of mass matrix
0 `1/s` (default) | nonnegative scalar

Coefficient, α, of the mass matrix. This parameter defines damping proportional to the mass matrix $[M]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Stiffness Coefficient — Coefficient of stiffness matrix
0.001 `s` (default) | nonnegative scalar

Coefficient, β, of the stiffness matrix. This parameter defines damping proportional to the stiffness matrix $[K]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Damping Ratio — Damping ratio for uniform modal damping method
0.01 (default) | unitless nonnegative scalar

Damping ratio, ζ, applied to all beam vibration modes in the uniform modal damping model. The larger the value, the faster beam vibrations decay.

- Use $\zeta = 0$ to model undamped beams.
- Use $\zeta < 1$ to model underdamped beams.
- Use $\zeta = 1$ to model critically damped beams.
- Use $\zeta > 1$ to model overdamped beams.

Dependencies

To enable this parameter, set **Type** to `Uniform Modal`.

Data Types: `double`

Discretization

Number of Elements — Number of beam finite elements
1 (default) | positive integer

Number of finite elements in the beam model. Increasing the number of elements always improves accuracy of the simulation. But practically, at some point, the increase in accuracy is negligible when there are many elements. Additionally, a higher number of elements increases the computational cost and slows down the speed of the simulation.

Fidelity

Reduction — Method to model flexible bodies
`None` (default) | `Modally Reduced`

Method to use to model flexible bodies, specified as `None` or `Modally Reduced`. Set the parameter to `None` to use full nodal elastic coordinates generated by the finite-element method or set the parameter to `Modally Reduced` to use the modal transformation method to reduce the elastic coordinates of the body. For both settings, the block uses the floating frame of the reference formulation [1-2] to couple the body with its elastic deformation.

Number of Retained Modes — Retained modes
1 (default) | nonnegative integer

Retained modes, specified as an integer in range [0, *n*], where *n* is the number of elastic degrees of freedom of the body. If you set the number to 0 the flexible body is treated as a rigid body.

Dependencies

To enable this parameter, set **Reduction** to `Modally Reduced`.

Graphic

Type — Graphic to use in the visualization of beam
`From Geometry` (default) | `None`

Type of the visual representation of the beam, specified as `From Geometry` or `None`. Set the parameter to `From Geometry` to show the visual representation of the beam. Set the parameter to `None` to hide the beam in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered beam and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered beam due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered beam.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the beam itself. When a beam has a emissive color, the beam can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Frames

Show Port A — Show port A for connection to other blocks
on (default) | off

Select to expose the **A** port.

Show Port B — Show port B for connection to other blocks
on (default) | off

Select to expose the **B** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the beam block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** for the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the undeformed beam.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected undeformed geometry feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

  - **Along Reference Frame Axis**: Selects an axis of the reference frame of the undeformed beam.
  - **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the undeformed beam. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2020a

References

[1] Shabana, Ahmed A. *Dynamics of Multibody Systems*. Fourth edition. New York: Cambridge University Press, 2014.

[2] Agrawal, Om P., and Ahmed A. Shabana. "Dynamic Analysis of Multibody Systems Using Component Modes." *Computers & Structures* 21, no. 6 (January 1985): 1303–12. https://doi.org/10.1016/0045-7949(85)90184-1.

Extended Capabilities

C/C++ Code Generation

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

See Also

Flexible Angle Beam | Flexible Channel Beam | Flexible Cylindrical Beam | Flexible Rectangular Beam | Flexible T Beam | General Flexible Beam | Extruded Solid | Reduced Order Flexible Solid | Rigid Transform

Topics

"Overview of Flexible Beams"
"Modeling Bodies"
"Manipulate the Color of a Solid"
"Custom Solid Frames"

Flexible Rectangular Beam

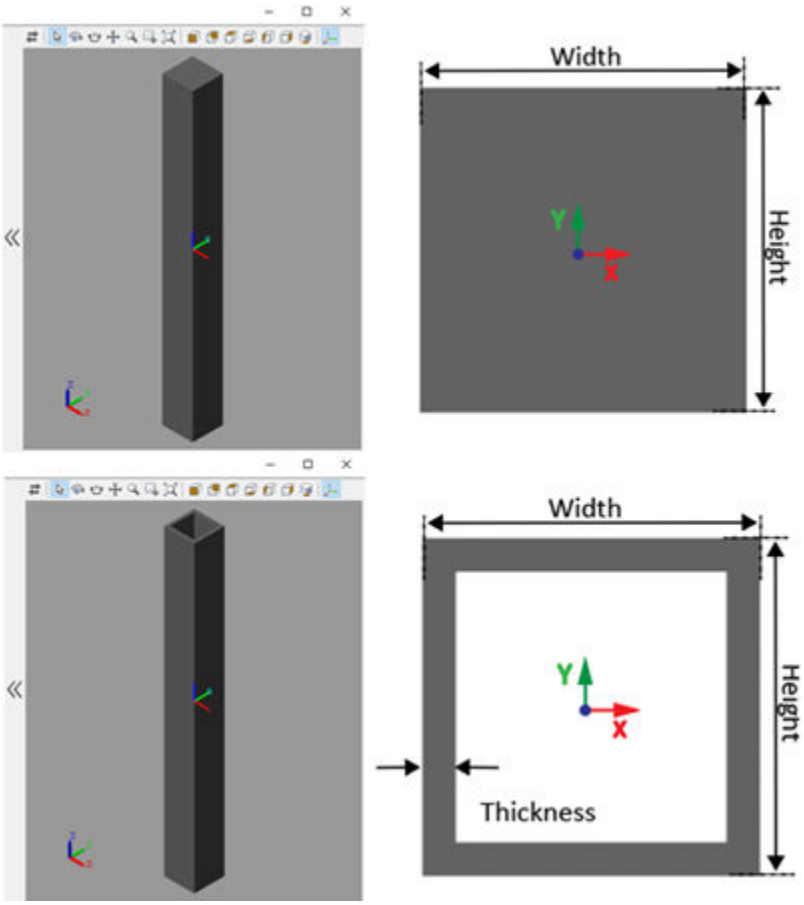
Rectangular beam with elastic properties for deformation

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

Description

The Flexible Rectangular Beam block models a slender beam with a rectangular cross-section that can be solid or hollow. The rectangular beam can have small and linear deformations. These deformations include extension, bending, and torsion. The block calculates the beam cross-sectional properties, such as the axial, flexural, and torsional rigidities, based on the geometry and material properties that you specify.

The geometry of the rectangular beam is an extrusion of its cross-section. The beam cross-section, defined in the *xy*-plane, is extruded along the *z*-axis. To define the cross-section, you can specify its dimensions in the **Geometry** section of the block dialog box. The figure shows a solid beam and a hollow beam. The reference frame is located at the centroid of the beam.



This block supports two damping methods and a discretization option to increase the accuracy of the modeling. For more information, see “Overview of Flexible Beams”.

Stiffness and Inertia Properties

The block provides two ways to specify the stiffness and inertia properties for a beam. To model a beam made of homogeneous, isotropic, and linearly elastic material, in the **Stiffness and Inertia** section, set the **Type** parameter to `Calculate from Geometry`. Then specify the density, Young’s modulus, and Poisson’s ratio or shear modulus. See the **Derived Values** parameter for more information about the calculated stiffness and inertia properties.

Alternatively, you can manually specify the stiffness and inertia properties, such as flexural rigidity and mass moment of inertia density, by setting the **Type** parameter to `Custom`. Use this option to model a beam that is made of anisotropic materials. This option decouples the mechanical properties from the beam cross section so you can specify desired mechanical properties without capturing the details of the exact cross section, such as fillet, rounds, chamfers, and tapers.

The manually entered stiffness properties must be calculated with respect to the frame located at the bending centroid. The frame must be in the same orientation as the beam reference frame. The stiffness matrix is

$$\bar{S}^b = \begin{bmatrix} S^b & 0 & 0 & 0 \\ 0 & H_x^b & -H_{xy}^b & 0 \\ 0 & -H_{xy}^b & H_y^b & 0 \\ 0 & 0 & 0 & H_z^b \end{bmatrix}$$

where:

- S^b is the axial stiffness along the beam.
- H_x^b is the centroidal bending stiffness about the *x*-axis.
- H_y^b is the centroidal bending stiffness about the *y*-axis.
- H_z^b is the torsional stiffness.
- H_{xy}^b is the centroidal cross bending stiffness.

The manually entered inertia properties must be calculated with respect to a frame located at the center of mass. The frame must be in the same orientation as the beam reference frame. The mass matrix includes rotatory inertia

$$\bar{M}^c = \begin{bmatrix} m^c & 0 & 0 & 0 & 0 & 0 \\ 0 & m^c & 0 & 0 & 0 & 0 \\ 0 & 0 & m^c & 0 & 0 & 0 \\ 0 & 0 & 0 & i_x^c & -i_{xy}^c & 0 \\ 0 & 0 & 0 & -i_{xy}^c & i_y^c & 0 \\ 0 & 0 & 0 & 0 & 0 & i_z^c \end{bmatrix}$$

where:

- m^c is the mass per unit length.
- i_x^c is the mass moment of inertia density about the *x*-axis.
- i_y^c is the mass moment of inertia density about the *y*-axis.
- i_z^c is the polar mass moment of inertia density.
- i_{xy}^c is the mass product of inertia density.

The beam block uses the classical beam theory where the relation between sectional strains and stress resultants is

$$\begin{bmatrix} \bar{F}_z \\ \bar{M}_x \\ \bar{M}_y \\ \bar{M}_z \end{bmatrix} = \bar{S}^b \begin{bmatrix} \bar{\varepsilon}_z \\ \bar{\kappa}_x \\ \bar{\kappa}_y \\ \bar{\kappa}_z \end{bmatrix}$$

where:

- $\bar{F}_z$ is the axial force along the beam.
- $\bar{M}_x$ is the bending moment about the *x*-axis.
- $\bar{M}_y$ is the bending moment about the *y*-axis.
- $\bar{M}_z$ is the torsional moment about the *z*-axis.
- $\bar{\varepsilon}_z$ is the axial strain along the beam.
- $\bar{\kappa}_x$ is the bending curvature about the *x*-axis.
- $\bar{\kappa}_y$ is the bending curvature about the *y*-axis.
- $\bar{\kappa}_z$ is the torsional twist about the *z*-axis.

Ports

Frame

A — Connection frame
frame

Frame 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 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

Type — Choice of solid or hollow cross-section
`Solid` (default) | `Hollow`

Choice of solid or hollow cross-section:

- Select `Solid` to model a beam with a solid cross-section.
- Select `Hollow` to model a beam with a hollow cross-section.

Width — Distance between outer vertical faces of beam
`1 m` (default) | positive scalar

Distance between the outer vertical faces of the beam.

Height — Distance between outer horizontal faces of beam
`1 m` (default) | positive scalar

Distance between the outer horizontal faces of the beam.

Thickness — Wall thickness of hollow cross-section
`0.1 m` (default) | positive scalar

Wall thickness of the hollow cross-section. This parameter specifies the distance between the inner and outer surfaces of the beam.

Dependencies

To enable this parameter, set **Type** to `Hollow`.

Length — Extrusion length of beam
`10 m` (default) | positive scalar

Extrusion length of the beam. The beam is modeled 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

Type — Method to use to specify stiffness and inertia properties
`Calculate from Geometry` (default) | `Custom`

Method to use to specify the stiffness and inertia properties, specified as `Calculate from Geometry` or `Custom`.

When you set the parameter to `Calculate from Geometry`, the block calculates the stiffness and inertia properties based on the specified density, Young's modulus, and Poisson's ratio or shear modulus. Set the parameter to `Custom` to manually specify the stiffness and inertia properties.

Density — Mass per unit volume of material
`2700 kg/m^3` (default) | positive scalar

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

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

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.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Young's Modulus — Ratio of axial stress to axial strain
`70 GPa` (default) | positive scalar

Young's modulus of elasticity of the beam. The greater its value, the stronger the resistance to bending and axial deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Poisson's Ratio — Ratio of transverse to longitudinal strains
0.33 (default) | scalar in the range [0, 0.5)

Poisson's ratio of the beam. The value specified must be greater than or equal to `0` and smaller than `0.5`. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's Modulus and Poisson's Ratio**.

Shear Modulus — Ratio of shear stress to engineering shear strain
`26 GPa` (default) | positive scalar

Shear modulus (or modulus of rigidity) of the beam. The greater its value, the stronger the resistance to torsional deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's and Shear Modulus**.

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 calculate and display 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:

$$[I_x, I_y] = \left[\int_A (y - y_c)^2 dA, \int_A (x - x_c)^2 dA\right],$$

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

$$I_{xy} = \int_A (x - x_c)(y - y_c) dA,$$

- I_p — Centroidal polar moment of area:

$$I_P = I_x + I_y,$$

where x_c and y_c are the coordinates of the centroid.

Geometric Centroid — Centroid of beam cross section
[0 0] m (default) | 2-by-1 array

Centroid of the beam cross section, specified as a 2-by-1 array. The array specifies the x and y coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Bending Centroid — Intersection of neutral axis and cross section
`[0 0] m` (default) | 2-by-1 array

Intersection of the neutral axis and cross section, specified as a 2-by-1 array. The array specifies the x and y coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Shear Center — Beam shear center
`[0 0] m` (default) | 2-by-1 array

Beam shear center, specified as a 2-by-1 array. The array specifies the x and y coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Axial Rigidity — Resistance against axial deformation
`1 Pa*m^2` (default) | scalar

Resistance against the deformation along the beam longitudinal direction, specified as a scalar. This parameter specifies the S^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Flexural Rigidity — Resistance against bending deformations
`[1 1] Pa*m^4` (default) | 2-by-1 array

Resistance against bending deformations, specified as a 2-by-1 array. The array specifies the H_x^b and H_y^b elements of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Cross Flexural Rigidity — Resistance against bending deformation about x-axis in response to bending moment about the y-axis
`0 Pa*m^4` (default) | scalar

Resistance against the bending deformation about the x-axis in response to bending moment about the y-axis, specified as a scalar. This parameter specifies the H_{xy}^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Torsional Rigidity — Resistance against twisting
`1 Pa*m^4` (default) | scalar

Resistance against twisting about the longitudinal axis of the beam, specified as a scalar. This parameter specifies the H_z^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Center of Mass — Center of Mass
`[0 0] m` (default) | 2-by-1 array

Center of mass, specified as a 2-by-1 array. The array specifies the *x* and *y* coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Per Unit Length — Mass per unit length
`1 kg/m` (default) | scalar

Mass per unit length, specified as a scalar. This parameter specifies the m^c element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Moment of Inertia Density — Resistance against rotation about x and y axes
`[1 1] kg*m` (default) | 2-by-1 array

Resistance against rotation about the *x* and *y* axes, specified as a 2-by-1 array. The array specifies the i_x^c and i_y^c elements of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Product of Inertia Density — Resistance against rotation about x-axis due to moment about y-axis
`0 kg*m` (default) | scalar

Resistance against rotation about the *x*-axis due to the moment about the *y*-axis, specified as a scalar. This parameter specifies the i_{xy}^c element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Damping

Type — Type of damping method
`Proportional` (default) | `Uniform Modal` | `None`

Damping method to apply to the beam:

- Select `None` to model undamped beams.

- Select `Proportional` to apply the proportional (or Rayleigh) damping method. This method defines the damping matrix $[C]$ as a linear combination of the mass matrix $[M]$ and stiffness matrix $[K]$:

  $$[C] = \alpha[M] + \beta[K],$$

  where α and β are the scalar coefficients.
- Select `Uniform Modal` to apply the uniform modal damping method. This method applies a single damping ratio to all the vibration modes of the beam. The larger the value, the faster vibrations decay.

Mass Coefficient — Coefficient of mass matrix
0 `1/s` (default) | nonnegative scalar

Coefficient, α, of the mass matrix. This parameter defines damping proportional to the mass matrix $[M]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Stiffness Coefficient — Coefficient of stiffness matrix
0.001 `s` (default) | nonnegative scalar

Coefficient, β, of the stiffness matrix. This parameter defines damping proportional to the stiffness matrix $[K]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Damping Ratio — Damping ratio for uniform modal damping method
0.01 (default) | unitless nonnegative scalar

Damping ratio, ζ, applied to all beam vibration modes in the uniform modal damping model. The larger the value, the faster beam vibrations decay.

- Use $\zeta = 0$ to model undamped beams.
- Use $\zeta < 1$ to model underdamped beams.
- Use $\zeta = 1$ to model critically damped beams.
- Use $\zeta > 1$ to model overdamped beams.

Dependencies

To enable this parameter, set **Type** to `Uniform Modal`.

Data Types: `double`

Discretization

Number of Elements — Number of beam finite elements
1 (default) | positive integer

Number of finite elements in the beam model. Increasing the number of elements always improves accuracy of the simulation. But practically, at some point, the increase in accuracy is negligible when

there are many elements. Additionally, a higher number of elements increases the computational cost and slows down the speed of the simulation.

Fidelity

Reduction — Method to model flexible bodies
`None` (default) | `Modally Reduced`

Method to use to model flexible bodies, specified as `None` or `Modally Reduced`. Set the parameter to `None` to use full nodal elastic coordinates generated by the finite-element method or set the parameter to `Modally Reduced` to use the modal transformation method to reduce the elastic coordinates of the body. For both settings, the block uses the floating frame of the reference formulation [1-2] to couple the body with its elastic deformation.

Number of Retained Modes — Retained modes
1 (default) | nonnegative integer

Retained modes, specified as an integer in range [0, *n*], where *n* is the number of elastic degrees of freedom of the body. If you set the number to 0 the flexible body is treated as a rigid body.

Dependencies

To enable this parameter, set **Reduction** to `Modally Reduced`.

Graphic

Type — Graphic to use in the visualization of beam
`From Geometry` (default) | `None`

Type of the visual representation of the beam, specified as `From Geometry` or `None`. Set the parameter to `From Geometry` to show the visual representation of the beam. Set the parameter to `None` to hide the beam in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry`
2 **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry`
2 **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered beam and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry`
2 **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered beam due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry`
2 **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered beam.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the beam itself. When a beam has a emissive color, the beam can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Frames

Show Port A — Show port A for connection to other blocks
on (default) | off

Select to expose the **A** port.

Show Port B — Show port B for connection to other blocks
on (default) | off

Select to expose the **B** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the beam block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** for the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the undeformed beam.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected undeformed geometry feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

  - **Along Reference Frame Axis**: Selects an axis of the reference frame of the undeformed beam.
  - **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the undeformed beam. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2020a

References

[1] Shabana, Ahmed A. *Dynamics of Multibody Systems*. Fourth edition. New York: Cambridge University Press, 2014.

[2] Agrawal, Om P., and Ahmed A. Shabana. "Dynamic Analysis of Multibody Systems Using Component Modes." *Computers & Structures* 21, no. 6 (January 1985): 1303–12. https://doi.org/10.1016/0045-7949(85)90184-1.

Extended Capabilities

C/C++ Code Generation

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

See Also

Flexible Angle Beam | Flexible Channel Beam | Flexible Cylindrical Beam | Flexible I Beam | Flexible T Beam | General Flexible Beam | Extruded Solid | Reduced Order Flexible Solid | Rigid Transform

Topics

"Overview of Flexible Beams"
"Modeling Bodies"
"Manipulate the Color of a Solid"

Flexible T Beam

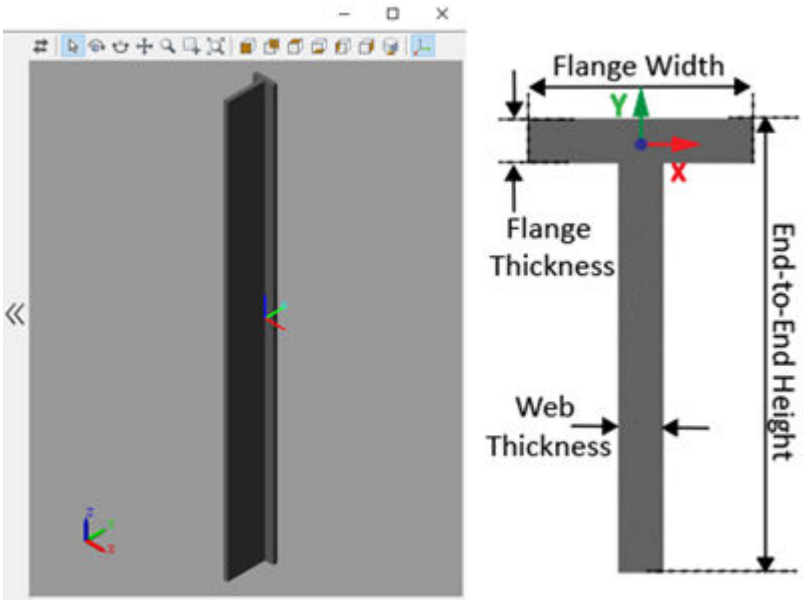
T-beam with elastic properties for deformation

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

Description

The Flexible T Beam block models a slender beam with a T-shaped cross-section, also known as a T-beam. The T-beam consists of one horizontal component, known as a flange, and one vertical component, which is called a web. The T-beam can have small and linear deformations. These deformations include extension, bending, and torsion. The block calculates the beam cross-sectional properties, such as the axial, flexural, and torsional rigidities, based on the geometry and material properties that you specify.

The geometry of the T-beam is an extrusion of its cross-section. The beam cross-section, defined in the *xy*-plane, is extruded along the *z*-axis. To define the cross-section, you can specify its dimensions in the **Geometry** section of the block dialog box. The figure shows a T-beam and its cross-section. The reference frame of the beam is located at the midpoint of the intersection line of the mid-planes of the web and flange.



This block supports two damping methods and a discretization option to increase the accuracy of the modeling. For more information, see “Overview of Flexible Beams”.

Stiffness and Inertia Properties

The block provides two ways to specify the stiffness and inertia properties for a beam. To model a beam made of homogeneous, isotropic, and linearly elastic material, in the **Stiffness and Inertia**

section, set the **Type** parameter to `Calculate from Geometry`. Then specify the density, Young's modulus, and Poisson's ratio or shear modulus. See the **Derived Values** parameter for more information about the calculated stiffness and inertia properties.

Alternatively, you can manually specify the stiffness and inertia properties, such as flexural rigidity and mass moment of inertia density, by setting the **Type** parameter to `Custom`. Use this option to model a beam that is made of anisotropic materials. This option decouples the mechanical properties from the beam cross section so you can specify desired mechanical properties without capturing the details of the exact cross section, such as fillet, rounds, chamfers, and tapers.

The manually entered stiffness properties must be calculated with respect to the frame located at the bending centroid. The frame must be in the same orientation as the beam reference frame. The stiffness matrix is

$$\bar{S}^b = \begin{bmatrix} S^b & 0 & 0 & 0 \\ 0 & H_x^b & -H_{xy}^b & 0 \\ 0 & -H_{xy}^b & H_y^b & 0 \\ 0 & 0 & 0 & H_z^b \end{bmatrix}$$

where:

- S^b is the axial stiffness along the beam.
- H_x^b is the centroidal bending stiffness about the *x*-axis.
- H_y^b is the centroidal bending stiffness about the *y*-axis.
- H_z^b is the torsional stiffness.
- H_{xy}^b is the centroidal cross bending stiffness.

The manually entered inertia properties must be calculated with respect to a frame located at the center of mass. The frame must be in the same orientation as the beam reference frame. The mass matrix includes rotatory inertia

$$\overline{M}^c = \begin{bmatrix} m^c & 0 & 0 & 0 & 0 & 0 \\ 0 & m^c & 0 & 0 & 0 & 0 \\ 0 & 0 & m^c & 0 & 0 & 0 \\ 0 & 0 & 0 & i_x^c & -i_{xy}^c & 0 \\ 0 & 0 & 0 & -i_{xy}^c & i_y^c & 0 \\ 0 & 0 & 0 & 0 & 0 & i_z^c \end{bmatrix}$$

where:

- m^c is the mass per unit length.
- i_x^c is the mass moment of inertia density about the *x*-axis.
- i_y^c is the mass moment of inertia density about the *y*-axis.

- i_z^c is the polar mass moment of inertia density.
- i_{xy}^c is the mass product of inertia density.

The beam block uses the classical beam theory where the relation between sectional strains and stress resultants is

$$\begin{bmatrix} \bar{F}_z \\ \bar{M}_x \\ \bar{M}_y \\ \bar{M}_z \end{bmatrix} = \bar{S}^b \begin{bmatrix} \bar{\varepsilon}_z \\ \bar{\kappa}_x \\ \bar{\kappa}_y \\ \bar{\kappa}_z \end{bmatrix}$$

where:

- $\bar{F}_z$ is the axial force along the beam.
- $\bar{M}_x$ is the bending moment about the *x*-axis.
- $\bar{M}_y$ is the bending moment about the *y*-axis.
- $\bar{M}_z$ is the torsional moment about the *z*-axis.
- $\bar{\varepsilon}_z$ is the axial strain along the beam.
- $\bar{\kappa}_x$ is the bending curvature about the *x*-axis.
- $\bar{\kappa}_y$ is the bending curvature about the *y*-axis.
- $\bar{\kappa}_z$ is the torsional twist about the *z*-axis.

Ports

Frame

A — Connection frame
frame

Frame 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 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

End-to-End Height — Distance from top face of flange to bottom end of web
`1 m` (default) | positive scalar

Distance from the top face of the flange to the bottom end of the web. The `End-to-End Height` is also known as the beam depth.

Note The **End-to-End Height** must be larger than the **Flange Thickness**.

Web Thickness — Distance between faces of web
`0.1 m` (default) | positive scalar

Distance between the two faces of the web.

Note The **Web Thickness** must be smaller than the **Flange Width**.

Flange Width — Distance between ends of flange
`0.5 m` (default) | positive scalar

Distance between the two ends of the flange.

Flange Thickness — Distance between faces of flange
`0.1 m` (default) | positive scalar

Distance between the two faces of the flange.

Length — Extrusion length of beam
`10 m` (default) | positive scalar

Extrusion length of the beam. The beam is modeled 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

Type — Method to use to specify stiffness and inertia properties
`Calculate from Geometry` (default) | `Custom`

Method to use to specify the stiffness and inertia properties, specified as `Calculate from Geometry` or `Custom`.

When you set the parameter to `Calculate from Geometry`, the block calculates the stiffness and inertia properties based on the specified density, Young's modulus, and Poisson's ratio or shear modulus. Set the parameter to `Custom` to manually specify the stiffness and inertia properties.

Density — Mass per unit volume of material
`2700 kg/m^3` (default) | positive scalar

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

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

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.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Young's Modulus — Ratio of axial stress to axial strain
`70 GPa` (default) | positive scalar

Young's modulus of elasticity of the beam. The greater its value, the stronger the resistance to bending and axial deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Poisson's Ratio — Ratio of transverse to longitudinal strains
0.33 (default) | scalar in the range [0, 0.5)

Poisson's ratio of the beam. The value specified must be greater than or equal to `0` and smaller than `0.5`. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's Modulus and Poisson's Ratio**.

Shear Modulus — Ratio of shear stress to engineering shear strain
`26 GPa` (default) | positive scalar

Shear modulus (or modulus of rigidity) of the beam. The greater its value, the stronger the resistance to torsional deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's and Shear Modulus**.

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 calculate and display 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:

$$[I_x, I_y] = \left[\int_A (y - y_c)^2 dA, \int_A (x - x_c)^2 dA\right],$$

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

$$I_{xy} = \int_A (x - x_c)(y - y_c) dA,$$

- I_p — Centroidal polar moment of area:

$$I_P = I_x + I_y,$$

where x_c and y_c are the coordinates of the centroid.

Geometric Centroid — Centroid of beam cross section
[0 0] m (default) | 2-by-1 array

Centroid of the beam cross section, specified as a 2-by-1 array. The array specifies the x and y coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Bending Centroid — Intersection of neutral axis and cross section
[0 0] m (default) | 2-by-1 array

Intersection of the neutral axis and cross section, specified as a 2-by-1 array. The array specifies the x and y coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Shear Center — Beam shear center
`[0 0] m` (default) | 2-by-1 array

Beam shear center, specified as a 2-by-1 array. The array specifies the x and y coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Axial Rigidity — Resistance against axial deformation
`1 Pa*m^2` (default) | scalar

Resistance against the deformation along the beam longitudinal direction, specified as a scalar. This parameter specifies the S^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Flexural Rigidity — Resistance against bending deformations
`[1 1] Pa*m^4` (default) | 2-by-1 array

Resistance against bending deformations, specified as a 2-by-1 array. The array specifies the H_x^b and H_y^b elements of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Cross Flexural Rigidity — Resistance against bending deformation about x-axis in response to bending moment about the y-axis
`0 Pa*m^4` (default) | scalar

Resistance against the bending deformation about the x-axis in response to bending moment about the y-axis, specified as a scalar. This parameter specifies the H_{xy}^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Torsional Rigidity — Resistance against twisting
`1 Pa*m^4` (default) | scalar

Resistance against twisting about the longitudinal axis of the beam, specified as a scalar. This parameter specifies the H_z^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Center of Mass — Center of Mass
`[0 0] m` (default) | 2-by-1 array

Center of mass, specified as a 2-by-1 array. The array specifies the x and y coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Per Unit Length — Mass per unit length
`1 kg/m` (default) | scalar

Mass per unit length, specified as a scalar. This parameter specifies the m^c element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Moment of Inertia Density — Resistance against rotation about x and y axes
`[1 1] kg*m` (default) | 2-by-1 array

Resistance against rotation about the x and y axes, specified as a 2-by-1 array. The array specifies the i_x^c and i_y^c elements of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Product of Inertia Density — Resistance against rotation about x-axis due to moment about y-axis
`0 kg*m` (default) | scalar

Resistance against rotation about the x-axis due to the moment about the y-axis, specified as a scalar. This parameter specifies the i_{xy}^c element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Damping

Type — Type of damping method
`Proportional` (default) | `Uniform Modal` | `None`

Damping method to apply to the beam:

- Select `None` to model undamped beams.
- Select `Proportional` to apply the proportional (or Rayleigh) damping method. This method defines the damping matrix $[C]$ as a linear combination of the mass matrix $[M]$ and stiffness matrix $[K]$:

$$[C] = \alpha[M] + \beta[K],$$

where α and β are the scalar coefficients.

- Select `Uniform Modal` to apply the uniform modal damping method. This method applies a single damping ratio to all the vibration modes of the beam. The larger the value, the faster vibrations decay.

Mass Coefficient — Coefficient of mass matrix
0 `1/s` (default) | nonnegative scalar

Coefficient, α, of the mass matrix. This parameter defines damping proportional to the mass matrix $[M]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Stiffness Coefficient — Coefficient of stiffness matrix
0.001 `s` (default) | nonnegative scalar

Coefficient, β, of the stiffness matrix. This parameter defines damping proportional to the stiffness matrix $[K]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Damping Ratio — Damping ratio for uniform modal damping method
0.01 (default) | unitless nonnegative scalar

Damping ratio, ζ, applied to all beam vibration modes in the uniform modal damping model. The larger the value, the faster beam vibrations decay.

- Use $\zeta = 0$ to model undamped beams.
- Use $\zeta < 1$ to model underdamped beams.
- Use $\zeta = 1$ to model critically damped beams.
- Use $\zeta > 1$ to model overdamped beams.

Dependencies

To enable this parameter, set **Type** to `Uniform Modal`.

Data Types: `double`

Discretization

Number of Elements — Number of beam finite elements
1 (default) | positive integer

Number of finite elements in the beam model. Increasing the number of elements always improves accuracy of the simulation. But practically, at some point, the increase in accuracy is negligible when there are many elements. Additionally, a higher number of elements increases the computational cost and slows down the speed of the simulation.

Fidelity

Reduction — Method to model flexible bodies
`None` (default) | `Modally Reduced`

Method to use to model flexible bodies, specified as `None` or `Modally Reduced`. Set the parameter to `None` to use full nodal elastic coordinates generated by the finite-element method or set the parameter to `Modally Reduced` to use the modal transformation method to reduce the elastic coordinates of the body. For both settings, the block uses the floating frame of the reference formulation [1-2] to couple the body with its elastic deformation.

Number of Retained Modes — Retained modes
1 (default) | nonnegative integer

Retained modes, specified as an integer in range [0, *n*], where *n* is the number of elastic degrees of freedom of the body. If you set the number to 0 the flexible body is treated as a rigid body.

Dependencies

To enable this parameter, set **Reduction** to `Modally Reduced`.

Graphic

Type — Graphic to use in the visualization of beam
`From Geometry` (default) | `None`

Type of the visual representation of the beam, specified as `From Geometry` or `None`. Set the parameter to `From Geometry` to show the visual representation of the beam. Set the parameter to `None` to hide the beam in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered beam and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered beam due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered beam.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the beam itself. When a beam has a emissive color, the beam can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Frames

Show Port A — Show port A for connection to other blocks
on (default) | off

Select to expose the **A** port.

Show Port B — Show port B for connection to other blocks
on (default) | off

Select to expose the **B** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button  to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the beam block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** for the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the undeformed beam.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected undeformed geometry feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

  - **Along Reference Frame Axis**: Selects an axis of the reference frame of the undeformed beam.
  - **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the undeformed beam. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button  to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button  to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2020a

References

[1] Shabana, Ahmed A. *Dynamics of Multibody Systems*. Fourth edition. New York: Cambridge University Press, 2014.

[2] Agrawal, Om P., and Ahmed A. Shabana. “Dynamic Analysis of Multibody Systems Using Component Modes.” *Computers & Structures* 21, no. 6 (January 1985): 1303–12. https://doi.org/10.1016/0045-7949(85)90184-1.

Extended Capabilities

C/C++ Code Generation

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

See Also

General Flexible Beam | Flexible Angle Beam | Flexible Channel Beam | Flexible Cylindrical Beam | Flexible I Beam | Flexible Rectangular Beam | Extruded Solid | Reduced Order Flexible Solid | Rigid Transform

Topics

“Overview of Flexible Beams”
“Modeling Bodies”
“Manipulate the Color of a Solid”

General Flexible Beam

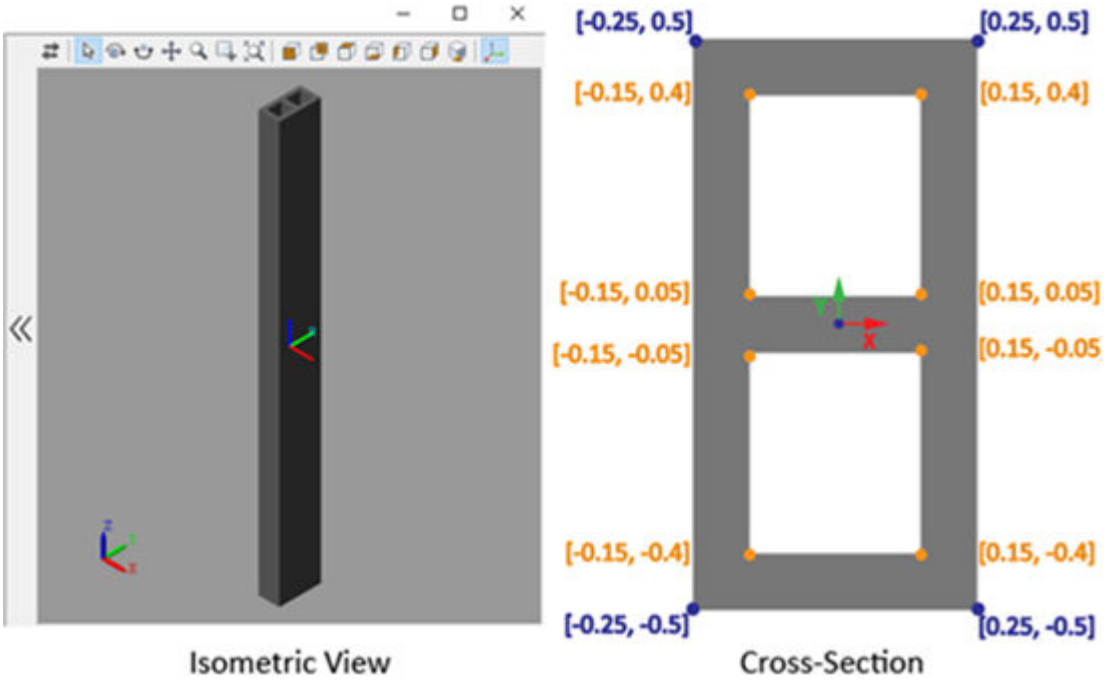
Slender extrusion with elastic properties for deformation

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

Description

The General Flexible Beam block models a slender beam of constant, general cross-section that can have small and linear deformations. These deformations include extension, bending, and torsion. The block calculates the beam cross-sectional properties, such as the axial, flexural, and torsional rigidities, based on the geometry and material properties that you specify.

The geometry of the flexible beam is an extrusion of its cross-section. The beam cross-section, defined in the *xy*-plane, is extruded along the *z*-axis. You can use this block to create flexible beams with simply or multiply connected cross-sections. For example, you can create the beam shown in the figure by entering these values for the **Cross-section** in the block's dialog box:
`{[-0.25,-0.50;0.25,-0.50;0.25,0.50;-0.25,0.50],`
`[-0.15,-0.40;0.15,-0.40;0.15,-0.05;-0.15,-0.05],`
`[-0.15,0.05;0.15,0.05;0.15,0.40;-0.15,0.40]}`.



This block supports two damping methods and a discretization option to increase the accuracy of the modeling. For more information, see “Overview of Flexible Beams”.

Stiffness and Inertia Properties

The block provides two ways to specify the stiffness and inertia properties for a beam. To model a beam made of homogeneous, isotropic, and linearly elastic material, in the **Stiffness and Inertia** section, set the **Type** parameter to `Calculate from Geometry`. Then specify the density, Young's modulus, and Poisson's ratio or shear modulus. See the **Derived Values** parameter for more information about the calculated stiffness and inertia properties.

Alternatively, you can manually specify the stiffness and inertia properties, such as flexural rigidity and mass moment of inertia density, by setting the **Type** parameter to `Custom`. Use this option to model a beam that is made of anisotropic materials. This option decouples the mechanical properties from the beam cross section so you can specify desired mechanical properties without capturing the details of the exact cross section, such as fillet, rounds, chamfers, and tapers.

The manually entered stiffness properties must be calculated with respect to the frame located at the bending centroid. The frame must be in the same orientation as the beam reference frame. The stiffness matrix is

$$\bar{S}^b = \begin{bmatrix} S^b & 0 & 0 & 0 \\ 0 & H_x^b & -H_{xy}^b & 0 \\ 0 & -H_{xy}^b & H_y^b & 0 \\ 0 & 0 & 0 & H_z^b \end{bmatrix}$$

where:

- S^b is the axial stiffness along the beam.
- H_x^b is the centroidal bending stiffness about the x-axis.
- H_y^b is the centroidal bending stiffness about the y-axis.
- H_z^b is the torsional stiffness.
- H_{xy}^b is the centroidal cross bending stiffness.

The manually entered inertia properties must be calculated with respect to a frame located at the center of mass. The frame must be in the same orientation as the beam reference frame. The mass matrix includes rotatory inertia

$$\overline{M}^c = \begin{bmatrix} m^c & 0 & 0 & 0 & 0 & 0 \\ 0 & m^c & 0 & 0 & 0 & 0 \\ 0 & 0 & m^c & 0 & 0 & 0 \\ 0 & 0 & 0 & i_x^c & -i_{xy}^c & 0 \\ 0 & 0 & 0 & -i_{xy}^c & i_y^c & 0 \\ 0 & 0 & 0 & 0 & 0 & i_z^c \end{bmatrix}$$

where:

- m^c is the mass per unit length.

- i_x^c is the mass moment of inertia density about the *x*-axis.
- i_y^c is the mass moment of inertia density about the *y*-axis.
- i_z^c is the polar mass moment of inertia density.
- i_{xy}^c is the mass product of inertia density.

The beam block uses the classical beam theory where the relation between sectional strains and stress resultants is

$$\begin{bmatrix} \bar{F}_z \\ \bar{M}_x \\ \bar{M}_y \\ \bar{M}_z \end{bmatrix} = \bar{S}^b \begin{bmatrix} \bar{\varepsilon}_z \\ \bar{\kappa}_x \\ \bar{\kappa}_y \\ \bar{\kappa}_z \end{bmatrix}$$

where:

- $\bar{F}_z$ is the axial force along the beam.
- $\bar{M}_x$ is the bending moment about the *x*-axis.
- $\bar{M}_y$ is the bending moment about the *y*-axis.
- $\bar{M}_z$ is the torsional moment about the *z*-axis.
- $\bar{\varepsilon}_z$ is the axial strain along the beam.
- $\bar{\kappa}_x$ is the bending curvature about the *x*-axis.
- $\bar{\kappa}_y$ is the bending curvature about the *y*-axis.
- $\bar{\kappa}_z$ is the torsional twist about the *z*-axis.

Ports

Frame

A — Connection frame
frame

Frame 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 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.

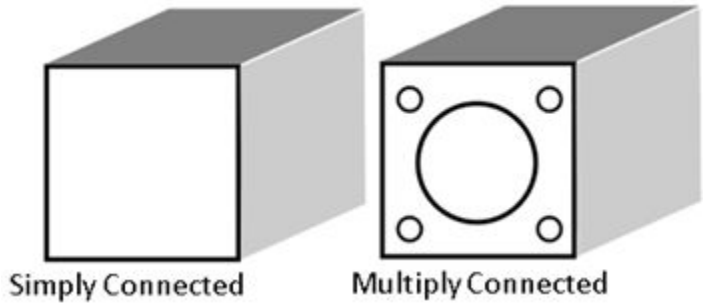
Parameters

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) | *N*-by-2 matrix | *M*-by-1 or 1-by-*M* cell array of *N*-by-2 matrices

Coordinates used to specify the boundaries of a beam cross-section. Specify the beam cross-section using one of the following methods:

- Use an *N*-by-2 matrix of *xy* coordinates to specify a simply connected section. Each row gives the [*x*,*y*] coordinates of a point on the cross-section boundary. 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.
- Use an *M*-by-1 or 1-by-*M* cell array of *N*-by-2 matrices of *xy* coordinates to specify a multiply connected section. The first entry in the cell represents the outer boundary and subsequent entries specify the hole boundaries.



Note To properly define the cross-section of beams, any two boundaries should not intersect, overlap, or touch.

Additionally, each individual boundary should have:

- No repeated vertices.
- No self-intersections.
- At least three non-collinear points.

Length — Extrusion length of beam
10 `m` (default) | positive scalar

The beam's length. The beam is modeled 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

Type — Method to use to specify stiffness and inertia properties
`Calculate from Geometry` (default) | `Custom`

Method to use to specify the stiffness and inertia properties, specified as `Calculate from Geometry` or `Custom`.

When you set the parameter to `Calculate from Geometry`, the block calculates the stiffness and inertia properties based on the specified density, Young's modulus, and Poisson's ratio or shear modulus. Set the parameter to `Custom` to manually specify the stiffness and inertia properties.

Density — Mass per unit volume of material
`2700 kg/m^3` (default) | positive scalar

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

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

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.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Young's Modulus — Ratio of axial stress to axial strain
`70 GPa` (default) | positive scalar

Young's modulus of elasticity of the beam. The greater its value, the stronger the resistance to bending and axial deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry`.

Poisson's Ratio — Ratio of transverse to longitudinal strains
0.33 (default) | scalar in the range [0, 0.5)

Poisson's ratio of the beam. The value specified must be greater than or equal to `0` and smaller than `0.5`. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's Modulus and Poisson's Ratio**.

Shear Modulus — Ratio of shear stress to engineering shear strain
`26 GPa` (default) | positive scalar

Shear modulus (or modulus of rigidity) of the beam. The greater its value, the stronger the resistance to torsional deformation. The default value corresponds to aluminum.

Dependencies

To enable this parameter, under the **Stiffness and Inertia** section, set **Type** parameter to `Calculate from Geometry` and set the **Specify** parameter to **Young's and Shear Modulus**.

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 calculate and display 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:

  $$[I_x, I_y] = \left[\int_A (y - y_c)^2 dA, \int_A (x - x_c)^2 dA\right],$$

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

  $$I_{xy} = \int_A (x - x_c)(y - y_c) dA,$$

- I_p — Centroidal polar moment of area:

  $$I_P = I_x + I_y,$$

where x_c and y_c are the coordinates of the centroid.

Geometric Centroid — Centroid of beam cross section
`[0 0] m` (default) | 2-by-1 array

Centroid of the beam cross section, specified as a 2-by-1 array. The array specifies the *x* and *y* coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Bending Centroid — Intersection of neutral axis and cross section
`[0 0] m` (default) | 2-by-1 array

Intersection of the neutral axis and cross section, specified as a 2-by-1 array. The array specifies the *x* and *y* coordinates of the centroid with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Shear Center — Beam shear center
`[0 0] m` (default) | 2-by-1 array

Beam shear center, specified as a 2-by-1 array. The array specifies the *x* and *y* coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Axial Rigidity — Resistance against axial deformation
`1 Pa*m^2` (default) | scalar

Resistance against the deformation along the beam longitudinal direction, specified as a scalar. This parameter specifies the S^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Flexural Rigidity — Resistance against bending deformations
[1 1] `Pa*m^4` (default) | 2-by-1 array

Resistance against bending deformations, specified as a 2-by-1 array. The array specifies the H_x^b and H_y^b elements of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Cross Flexural Rigidity — Resistance against bending deformation about x-axis in response to bending moment about the y-axis
0 `Pa*m^4` (default) | scalar

Resistance against the bending deformation about the *x*-axis in response to bending moment about the *y*-axis, specified as a scalar. This parameter specifies the H_{xy}^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Torsional Rigidity — Resistance against twisting
1 `Pa*m^4` (default) | scalar

Resistance against twisting about the longitudinal axis of the beam, specified as a scalar. This parameter specifies the H_z^b element of the stiffness matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Center of Mass — Center of Mass
[0 0] `m` (default) | 2-by-1 array

Center of mass, specified as a 2-by-1 array. The array specifies the *x* and *y* coordinates of the point with respect to the beam reference frame.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Per Unit Length — Mass per unit length
1 `kg/m` (default) | scalar

Mass per unit length, specified as a scalar. This parameter specifies the m^c element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Moment of Inertia Density — Resistance against rotation about x and y axes
`[1 1] kg*m` (default) | 2-by-1 array

Resistance against rotation about the *x* and *y* axes, specified as a 2-by-1 array. The array specifies the i_x^C and i_y^C elements of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Mass Product of Inertia Density — Resistance against rotation about x-axis due to moment about y-axis
`0 kg*m` (default) | scalar

Resistance against rotation about the *x*-axis due to the moment about the *y*-axis, specified as a scalar. This parameter specifies the i_{xy}^C element of the mass matrix.

Dependencies

To enable this parameter, in the **Stiffness and Inertia** section, set **Type** to `Custom`.

Damping

Type — Type of damping method
`Proportional` (default) | `Uniform Modal` | `None`

Damping method to apply to the beam:

- Select `None` to model undamped beams.
- Select `Proportional` to apply the proportional (or Rayleigh) damping method. This method defines the damping matrix $[C]$ as a linear combination of the mass matrix $[M]$ and stiffness matrix $[K]$:

  $$[C] = \alpha[M] + \beta[K],$$

  where α and β are the scalar coefficients.
- Select `Uniform Modal` to apply the uniform modal damping method. This method applies a single damping ratio to all the vibration modes of the beam. The larger the value, the faster vibrations decay.

Mass Coefficient — Coefficient of mass matrix
`0 1/s` (default) | nonnegative scalar

Coefficient, α, of the mass matrix. This parameter defines damping proportional to the mass matrix $[M]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Stiffness Coefficient — Coefficient of stiffness matrix
`0.001 s` (default) | nonnegative scalar

Coefficient, β, of the stiffness matrix. This parameter defines damping proportional to the stiffness matrix $[K]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Damping Ratio — Damping ratio for uniform modal damping method
0.01 (default) | unitless nonnegative scalar

Damping ratio, ζ, applied to all beam vibration modes in the uniform modal damping model. The larger the value, the faster beam vibrations decay.

- Use $\zeta = 0$ to model undamped beams.
- Use $\zeta < 1$ to model underdamped beams.
- Use $\zeta = 1$ to model critically damped beams.
- Use $\zeta > 1$ to model overdamped beams.

Dependencies

To enable this parameter, set **Type** to `Uniform Modal`.

Data Types: `double`

Discretization

Number of Elements — Number of beam finite elements
1 (default) | positive integer

Number of finite elements in the beam model. Increasing the number of elements always improves accuracy of the simulation. But practically, at some point, the increase in accuracy is negligible when there are many elements. Additionally, a higher number of elements increases the computational cost and slows down the speed of the simulation.

Fidelity

Reduction — Method to model flexible bodies
`None` (default) | `Modally Reduced`

Method to use to model flexible bodies, specified as `None` or `Modally Reduced`. Set the parameter to `None` to use full nodal elastic coordinates generated by the finite-element method or set the parameter to `Modally Reduced` to use the modal transformation method to reduce the elastic coordinates of the body. For both settings, the block uses the floating frame of the reference formulation [1-2] to couple the body with its elastic deformation.

Number of Retained Modes — Retained modes
1 (default) | nonnegative integer

Retained modes, specified as an integer in range [0, n], where n is the number of elastic degrees of freedom of the body. If you set the number to 0 the flexible body is treated as a rigid body.

Dependencies

To enable this parameter, set **Reduction** to `Modally Reduced`.

Graphic

Type — Graphic to use in the visualization of beam
`From Geometry` (default) | `None`

Type of the visual representation of the beam, specified as `From Geometry` or `None`. Set the parameter to `From Geometry` to show the visual representation of the beam. Set the parameter to `None` to hide the beam in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered beam and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered beam due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered beam.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the beam itself. When a beam has a emissive color, the beam can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`

2 **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry`
2 **Visual Properties** to `Advanced`

Frames

Show Port A — Show port A for connection to other blocks
on (default) | off

Select to expose the **A** port.

Show Port B — Show port B for connection to other blocks
on (default) | off

Select to expose the **B** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button  to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the beam block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** for the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the undeformed beam.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected undeformed geometry feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

  - **Along Reference Frame Axis**: Selects an axis of the reference frame of the undeformed beam.

- **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the undeformed beam. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button  to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button  to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2018b

References

[1] Shabana, Ahmed A. *Dynamics of Multibody Systems*. Fourth edition. New York: Cambridge University Press, 2014.

[2] Agrawal, Om P., and Ahmed A. Shabana. “Dynamic Analysis of Multibody Systems Using Component Modes.” *Computers & Structures* 21, no. 6 (January 1985): 1303–12. https://doi.org/10.1016/0045-7949(85)90184-1.

Extended Capabilities

C/C++ Code Generation

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

See Also

Flexible Angle Beam | Flexible Channel Beam | Flexible Cylindrical Beam | Flexible I Beam | Flexible Rectangular Beam | Flexible T Beam | Extruded Solid | Reduced Order Flexible Solid | Rigid Transform

Topics

“Overview of Flexible Beams”
“Modeling Bodies”
“Manipulate the Color of a Solid”

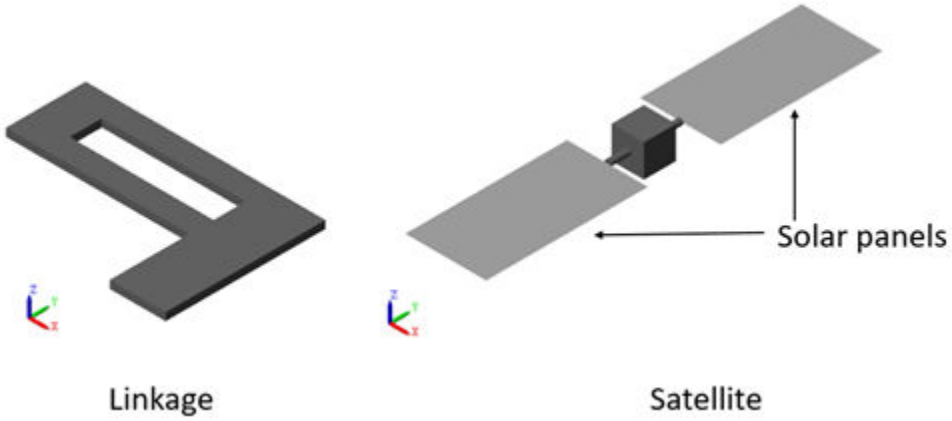
General Flexible Plate

Thin plate with elastic properties for deformation

Libraries:
Simscape / Multibody / Body Elements / Flexible Bodies / Plates and Shells

Description

The General Flexible Plate block models thin, flat structure capable of elastic deformations, including stretch, bending, and shear effects. The block applies the shear deformation Mindlin plate theory [1] [2][3] on page 1-242 and uses the finite element method [4] on page 1-242 for its solution. You can use this block to model thin, flat structures, such as linkages and satellite solar panels.



To specify the geometry of a plate, use the **Midsurface** and **Thickness** parameters. The midsurface of the plate is in the *xy* plane and the thickness is along the *z* axis. The thickness must be much smaller than the width and length of the plate, and the plate is symmetric about the midsurface. See the “Geometry” on page 1-236 section for more details.

The block models a flexible plate made of homogeneous, isotropic, and linearly elastic materials. You can specify the density, Young's modulus, and Poisson's ratio or shear modulus of the plate in the **Stiffness and Inertia** section. Additionally, the block supports two damping methods to control the performance of the modeling. To add custom frames to the plate, specify parameters in the **Frames** section.

Ports

Frame

FN — Custom frame
frame

Custom body-attached frames. Specified the name of the port using the “New Frame” on page 1-0 parameter. If you do not specify the name of the custom frame, the block names the frame **FN**, where N is an identifying number.

Dependencies

To enable a custom frame port, create a frame by clicking **New Frame**.

Parameters

Geometry

Midsurface — Coordinates of plate midsurface
[1 1; -1 1; -1 -1; 1 -1] `m` | *N*-by-2 matrix | *M*-by-1 or 1-by-*M* cell array of *N*-by-2 matrices

Coordinates used to specify the midsurface boundaries of the plate on the local *xy* plane. You can specify the midsurface by using:

- An *N*-by-2 matrix of *xy* coordinates to specify a midsurface. Each row gives the [*x*,*y*] coordinates of a point on the mid-surface boundaries. The points connect in the specified order to form a closed polyline. To ensure that the polyline is closed, the block inserts a line segment between the last and first points.
- An *M*-by-1 or 1-by-*M* cell array of *N*-by-2 matrices of *xy* coordinates to specify a midsurface with holes. The first element in the array represents the outer boundary and subsequent elements specify the hole boundaries.

Note Ensure that any two boundaries do not intersect, overlap, or touch.

Additionally, each individual boundary must have:

- No repeated vertices
- No self-intersections
- At least three non-collinear points

Thickness — Plate thickness
0.1 `m` (default) | positive scalar

Thickness of the plate. The block models the plate by extruding the specified midsurface along the local *z* axis of the plate. The extrusion is symmetric about the local *xy* plane of the pate. The thickness should be much smaller than the overall midsurface dimensions for plate theory to apply.

Stiffness and Inertia

Density — Mass per unit volume of material
2700 `kg/m^3` (default) | positive scalar

Mass per unit volume of material. The default value corresponds to aluminum.

Specify — Elastic properties used to parameterize plate
`Young's Modulus and Poisson's Ratio` (default) | `Young's and Shear Modulus`

Elastic properties used to parameterize the plate. You can specify either `Young's Modulus and Poisson's Ratio` or `Young's and Shear Modulus`. These properties are commonly available in materials databases.

Young's Modulus — Ratio of axial stress to axial strain
`70 GPa` (default) | positive scalar

Young's modulus of the elasticity of the plate. The greater the value of this parameter, the stronger the resistance to bending and in-plane normal deformation. The default value corresponds to aluminum.

Poisson's Ratio — Ratio of transverse to normal strains
0.33 (default) | scalar in the range [0, 0.5)

Poisson's ratio of the plate. The value specified must be greater than or equal to 0 and smaller than 0.5. The default value corresponds to aluminum.

Dependencies

To enable this parameter, set **Specify** to `Young's Modulus and Poisson's Ratio`.

Shear Modulus — Ratio of shear stress to engineering shear strain
`26 GPa` (default) | positive scalar

Shear modulus, also known as the modulus of rigidity, of the plate. The larger value correlates to a stronger resistance to shearing and twisting. The default value corresponds to aluminum.

Dependencies

To enable this parameter, set **Specify** to `Young's and Shear Modulus`.

Damping

Type — Type of damping method
`Proportional` (default) | `Uniform Modal` | `None`

Damping method for the plate:

- Select `None` to model undamped plates.
- Select `Proportional` to apply the proportional (or Rayleigh) damping method. This method defines the damping matrix $[C]$ as a linear combination of the mass matrix $[M]$ and stiffness matrix $[K]$:

  $$[C] = \alpha[M] + \beta[K],$$

  where α and β are the scalar coefficients.
- Select `Uniform Modal` to apply the uniform modal damping method. This method applies a single damping ratio to all the vibration modes of the plate. The larger the value, the faster vibrations decay.

Mass Coefficient — Coefficient of mass matrix
`0 1/s` (default) | nonnegative scalar

Coefficient α of the mass matrix. This parameter defines damping proportional to the mass matrix $[M]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Stiffness Coefficient — Coefficient of stiffness matrix
0.001 s (default) | nonnegative scalar

Coefficient β of the stiffness matrix. This parameter defines damping proportional to the stiffness matrix [K].

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Damping Ratio — Damping ratio for uniform modal damping method
0.01 (default) | unitless nonnegative scalar

Damping ratio ζ applied to all vibration modes of a plate. The larger the value, the faster vibrations decay. The vibration modes are underdamped if $\zeta < 1$ and overdamped if $\zeta > 1$.

Dependencies

To enable this parameter, set **Type** to `Uniform Modal`.

Data Types: `double`

Fidelity

Reduction — Method to model flexible bodies
`None` (default) | `Modally Reduced`

Method to use to model flexible bodies, specified as `None` or `Modally Reduced`. Set the parameter to `None` to use full nodal elastic coordinates generated by the finite-element method or set the parameter to `Modally Reduced` to use the modal transformation method to reduce the elastic coordinates of the body. For both settings, the block uses the floating frame of the reference formulation [5-6] to couple the body with its elastic deformation.

Number of Retained Modes — Retained modes
1 (default) | nonnegative integer

Retained modes, specified as an integer in range [0, n], where n is the number of elastic degrees of freedom of the body. If you set the number to 0 the flexible body is treated as a rigid body.

Dependencies

To enable this parameter, set **Reduction** to `Modally Reduced`.

Graphic

Type — Graphic to use in the visualization of plate
`From Geometry` (default) | `None`

Type of the visual representation of the plate, specified as `From Geometry` or `None`. Set the parameter to `From Geometry` to show the visual representation of the plate. Set the parameter to `None` to hide the plate in 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 specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Diffuse Color — Graphic Color
[0.5 0.5 0.5] (default) | three-element vector | three-element vector

Color of the graphic under direct white 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 on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This

parameter changes the color of the specular highlight, which is the bright spot on the rendered beam due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered beam.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the beam itself. When a beam has a emissive color, the beam can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry`
2 **Visual Properties** to `Advanced`

Frames

New Frame — Create custom frame for connection to other blocks
button

Click the Create button  to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, enter a name in the **Frame Name** parameter. The name identifies the port on the block and in the tree view pane of the Mechanics Explorer.
- Select a frame origin in the **Frame Origin** section:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the undeformed plate.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected undeformed geometry feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** button to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

  - **Along Reference Frame Axis**: Selects an axis of the reference frame of the undeformed plate.
  - **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the undeformed plate. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** button to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you created. Specified the name of the port using the “New Frame” on page 1-0 parameter. If you do not specify the name of the custom frame, the block names the frame **FN**, where `N` is an identifying number.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button  to edit other aspects of the custom frame, such as the origin and axes.
- Click the Delete button  to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2021b

References

[1] Bathe, Klaus-Jürgen. *Finite Element Procedures*. 2nd ed. Englewood Cliffs, N.J: Prentice-Hall, 2014.

[2] Cook, Robert Davis. *Concepts and Applications of Finite Element Analysis*. 4th ed. New York, NY: Wiley, 2001.

[3] Dvorkin, Eduardo N., and Klaus-Jürgen Bathe. "A Continuum Mechanics Based Four-node Shell Element for General Non-linear Analysis." *Engineering Computations* 1, no. 1 (January 1984): 77–88. https://doi.org/10.1108/eb023562.

[4] Bucalem, M. L., and K. J. Bathe. "Finite Element Analysis of Shell Structures." *Archives of Computational Methods in Engineering* 4, no. 1 (March 1997): 3–61. https://doi.org/10.1007/BF02818930.

[5] Shabana, Ahmed A. *Dynamics of Multibody Systems*. Fourth edition. New York: Cambridge University Press, 2014.

[6] Agrawal, Om P., and Ahmed A. Shabana. "Dynamic Analysis of Multibody Systems Using Component Modes." *Computers & Structures* 21, no. 6 (January 1985): 1303–12. https://doi.org/10.1016/0045-7949(85)90184-1.

Extended Capabilities

C/C++ Code Generation

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

See Also

General Flexible Beam

General Variable Mass

Mass element with variable inertial properties

Libraries:
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.

Limitations

- The General Variable Mass block does not conserve the angular momentum when simulated without an external moment. Instead of the angular momentum, the angular velocity remains unchanged.

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 × length^2

Input port for the time-varying inertia tensor. Specify the tensor elements in the order `[Ixx Ixy I xz; 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 * m^2` (default) | three-element vector with units of mass*length^2

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:

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

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 * m^2` (default) | 3-element array with units of mass * length^2

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:

- $I_{yz} = -\int_m yz\,dm$
- $I_{zx} = -\int_m zx\,dm$
- $I_{xy} = -\int_m xy\,dm$

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 variable mass, specified as `Equivalent Inertia Ellipsoid`, `Marker`, or `None`. Set the parameter to `Equivalent Inertia Ellipsoid` to represent the variable mass as an ellipsoid. Set the parameter to `Marker` to represent the variable mass as a marker. Set the parameter to `None` to hide the variable mass in the model visualization.

Shape — Shape of marker to represent to variable mass
`Sphere` (default) | `Cube` | `Frame`

Shape of the marker by means of which to visualize the variable mass, specified as `Sphere`, `Cube`, or `Frame`. The motion of the marker reflects the motion of the variable mass itself.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
`10 pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `Equivalent Inertia Ellipsoid` or `Marker`.

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the graphic and provides shading that gives the graphic a three-dimensional appearance.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `Equivalent Inertia Ellipsoid` or `Marker`
2. **Visual Properties** to `Simple`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the graphic due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `Equivalent Inertia Ellipsoid` or `Marker`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the graphic.

Dependencies

To enable this parameter, set:

1. **Type** to `Equivalent Inertia Ellipsoid` or `Marker`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the graphic itself. When a graphic has a emissive color, the graphic can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `Equivalent Inertia Ellipsoid` or `Marker`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `Equivalent Inertia Ellipsoid` or `Marker`
2. **Visual Properties** to `Advanced`

Version History

Introduced in R2016b

Extended Capabilities

C/C++ Code Generation

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

See Also

Brick Solid | Cylindrical Solid | Ellipsoidal Solid | Extruded Solid | Revolved Solid | Spherical Solid | Inertia | Spline

Topics

“Representing Solid Inertia”

“Manipulate the Color of a Solid”
“Specifying Variable Inertias”
“Specifying Custom Inertias”

Gimbal Joint

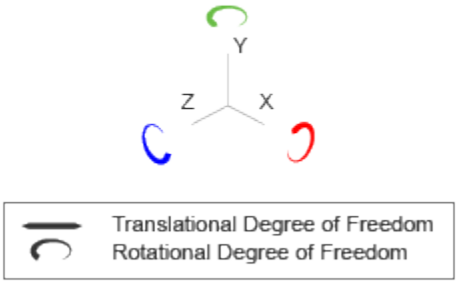
Joint with three revolute primitives

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.



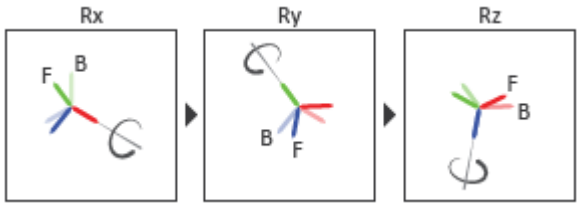
Joint Degrees of Freedom

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.

Mode Configuration

Specify the mode of the joint. The joint mode can be normal or disengaged throughout the simulation, or you can provide an input signal to change the mode during the simulation.

Mode

Select one of the following options to specify the mode of the joint. The default setting is `Normal`.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.

Method	Description
`Provided by Input`	This option exposes the **mode** port that you can connect to an input signal to change the joint mode during the simulation. The joint mode is normal when the input signal is `0` and disengaged when the input signal is `-1`. The joint mode can be changed many times during the simulation.

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:

- tx, ty, tz — Actuation torques acting on the X, Y, and Z revolute joint primitives
- qx, qy, qz — Desired rotations of the X, Y, and Z revolute joint primitives

Sensing Ports

The revolute joint primitives provide the following sensing ports:

- qx, qy, qz — Angular positions of the X, Y, and Z revolute joint primitives
- wx, wy, wz — Angular velocities of the X, Y, and Z revolute joint primitives
- bx, by, bz — Angular accelerations of the X, Y, and Z revolute joint primitives
- tx, ty, tz — Actuation torques acting on the X, Y, and Z revolute joint primitives
- tllx, tlly, tllz — Torques due to contact with the lower limits of the X, Y, and Z revolute joint primitives
- tulx, tuly, tulz — Torques due to contact with the upper limits of the X, Y, and Z 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

Mode Port

Mode configuration provides the following port:

- mode — Value of the mode of the joint. If the input is equal to `0`, the joint behaves normally. If the input is equal to `-1`, the joint behaves as disengaged.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Revolute Joint | Spherical Joint | Bushing Joint

Topics

“Actuating and Sensing with Physical Signals”
“Motion Sensing”
“Rotational Measurements”

Graphic

Marker with graphic properties

Libraries:
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, specified as `Sphere`, `Cube`, or `Frame`. 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 `pixels` (default) | scalar

Width of the marker in pixels, specified as a scalar. 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

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set **Visual Properties** to `Simple`.

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set **Visual Properties** to `Advanced`.

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction.

Dependencies

To enable this parameter, set **Visual Properties** to `Advanced`.

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the marker itself. When a marker has a emissive color, the marker can be seen even if there is no external light source.

Dependencies

To enable this parameter, set **Visual Properties** to `Advanced`.

Shininess — Shininess of marker
75 (default) | scalar in the range of 1 to 128

Shininess of the marker, specified as a scalar in the range of 0 to 128. This parameter affects the sharpness of the specular reflections of the marker. A marker with high shininess has a mirror-like appearance, and marker with low shininess has a more low-gloss or satin appearance.

Dependencies

To enable this parameter, set **Visual Properties** to `Advanced`.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

See Also

Inertia | Brick Solid | Cylindrical Solid | Ellipsoidal Solid | Extruded Solid | Revolved Solid | Spherical Solid | Spline

Topics
“Visualize Simscape Multibody Frames”
“Manipulate the Color of a Solid”

Gravitational Field

Field of force due to point mass

Library

Forces and Torques

Description

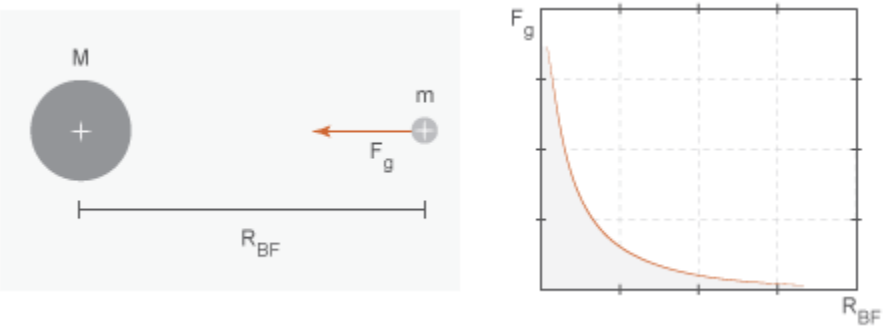
This block represents the gravitational field of a point mass. This field applies a gravitational force at the center of mass of each rigid body. The force magnitude decays with the square distance from the field origin, coincident with the base port frame origin. The force on a rigid body follows from 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, 6.67384×10^{-11} $m^3kg^{-1}s^{-2}$.
- 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.

Version History

Introduced in R2014a

Extended Capabilities

C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

See Also

Inverse Square Law Force | Mechanism Configuration

Topics
“Modeling Gravity”
“Model Gravity in a Planetary System”

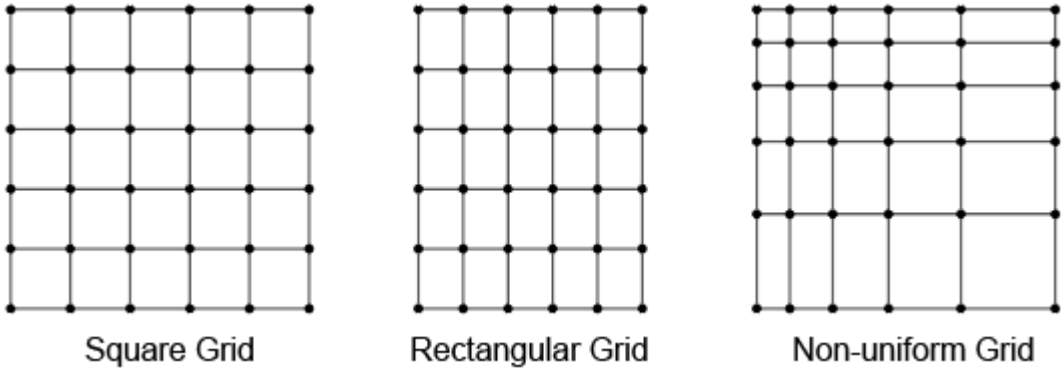
Grid Surface

Grid surface for contact modeling

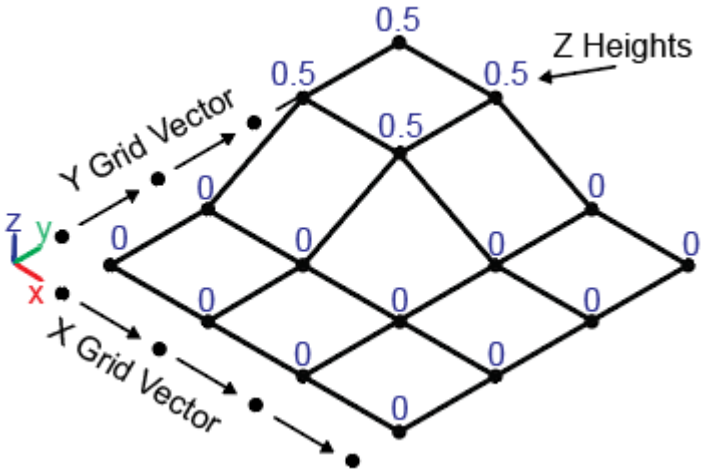
Libraries:
Simscape / Multibody / Curves and Surfaces

Description

The Grid Surface block creates a Cartesian grid surface that you can use to model contact problems. The grid surface has a similar format to the grid generated by the `ndgrid` function in MATLAB. You can use the block to create a surface that represents a square, rectangular, or non-uniform grid.



To model a grid surface, use the **X Grid Vector** and **Y Grid Vector** parameters to specify the coordinates in the *x* and *y* directions of the grid surface, then use the **Z Heights** parameter to specify the elevations of the grid points in the *z* direction. The image shows an illustration of a grid surface whose **Z Heights** parameter is specified as `[0 0 0.5 0.5; 0 0 0.5 0.5; 0 0 0 0; 0 0 0 0]`. The columns of the matrix are along the *x*-axis, and the rows of the matrix are along the *y*-axis.



To create a grid surface using LiDAR data, you must convert the scattered data to gridded data before using it in the Grid Surface block. To convert the data, you can use a `scatteredInterpolant` object to create an interpolant function by using the *x*, *y*, and *z* values of the LiDAR data, create *x* and *y* grid vectors that represent the domain of the grid, then use the grid vectors as the query points of the interpolant function to produce *z* values of the grid surface. The produced *z* values form a matrix in the `ndgrid` format that can be used to specify the **Z Heights** parameter. You can also use the `griddata` function to produce *z* values for a grid surface, but you need to transpose the output matrix to the `ndgrid` format.

Note

- Do not use the `ndgrid` function to create the *x* and *y* grid data for a grid surface. The `ndgrid` function replicates the input grid vectors into a matrix. Using grid vectors instead of full matrices can save memory.
- To have the best simulation performance, the spacing in the *x* and *y* grid vectors should be as large as possible but small enough to represent the surface with sufficient accuracy. To change the spacing of the grid vectors and compute the corresponding *z* values , you can use the `griddedInterpolant` object or `interp2` function.

Generally, you need to clean the LiDAR data before converting it to gridded data, such as removing outliers or *NaN* values. To clean the data, use the `isnan` and `isoutlier` functions.

Ports

Frame

R — Reference frame
frame

Grid surface reference frame. Connect this frame to another block to specify the location and orientation of the grid surface.

Geometry

G — Geometry
geometry

Geometry that represents the grid surface defined by this block. Connect this port to a Spatial Contact Force block to model contacts on the grid surface. The Grid Surface block supports only the contact with the Point or Point Cloud block.

Parameters

X Grid Vector — Coordinates in *x*-direction
[0 1 2 3] `m` (default) | *1-by-m* vector

Coordinates in the *x*-direction of the grid surface, specified as a real *1-by-m* vector. *m* must be greater than 2, and the vector elements must be strictly monotonic and increasing.

Y Grid Vector — Coordinates in *y*-direction
[0 1 2 3] `m` (default) | *1-by-n* vector

Coordinates in the *y*-direction of the grid surface, specified as a real *1-by-n* vector. *n* must be greater than 2, and the vector elements must be strictly monotonic and increasing.

Z Heights — Elevations of grid points
`[0 0 .5 .5; 0 0 .5 .5; 0 0 0 0; 0 0 0 0]` `m` (default) | *m-by-n* matrix

Elevations of the grid points in the grid surface, specified as a real *m-by-n* matrix. *m* and *n* are the lengths of the vectors for the **X Grid Vector** and **Y Grid Vector** parameters.

Graphic

Type — Visual representation of grid surface
`From Geometry` (default) | `None`

Visual representation of the grid surface. Set this parameter to `From Geometry` to show the visual representation of the grid surface. Set this parameter to `None` to hide the grid surface in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify diffuse color and opacity. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry`.

Diffuse Color — Graphic color
`[0.5 0.5 0.5]` (default) | three-element vector | four-element vector

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set **Type** to `From Geometry`.

Opacity — Graphic opacity
`1.0` (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Simple`

Specular Color — Highlight color
`[0.5 0.5 0.5 1.0]` (default) | three-element vector | four-element vector

Color of the 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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Ambient Color — Shadow color
[0.15 0.15 0.15 1.0] (default) | three-element vector | four-element vector

Color of the 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 (A) specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | three-element vector | four-element vector

Graphic 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 (A) specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
75 (default) | scalar in the range of 1 to 128

Sharpness of the 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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry`
2. **Visual Properties** to `Advanced`

Version History

Introduced in R2021b

Extended Capabilities

C/C++ Code Generation

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

See Also

Spatial Contact Force | Point Cloud | `griddata` | `isnan` | `isoutlier` | `scatteredInterpolant` | `griddata` | `griddedInterpolant` | `interp2`

Inertia

Mass element with fixed inertial properties

Libraries:
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*m^2` (default) | three-element vector with units of mass*length^2

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:

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

Custom: Products of Inertia — Off-diagonal elements of inertia tensor
`[0 0 0] kg*m^2` (default) | three-element vector with units of mass*length^2

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:

- $I_{yz} = -\int_m yz\, dm$
- $I_{zx} = -\int_m zx\, dm$
- $I_{xy} = -\int_m xy\, dm$

Graphic

Type — Graphic to use in the visualization of the inertia
`Marker` (default) | `None`

Type of graphic to use in the visualization of the inertia, specified as `Marker` or `None`. Set this parameter to `Marker` to represent the inertia as a marker, and set this parameter to `None` to hide the inertia from the model visualization.

Shape — Shape of marker to represent to the inertia
`Inertia Icon` (default) | `Sphere` | `Cube` | `Frame`

Shape of the marker, specified as `Inertia Icon`, `Sphere`, `Cube`, or `Frame`. The motion of the marker reflects the motion of the inertia itself.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
20 `pixels` (default) | scalar

Width of the marker in pixels, specified as a scalar. 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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `Marker`
2. **Visual Properties** to `Simple`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `Marker`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction.

Dependencies

To enable this parameter, set:

1. **Type** to `Marker`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the marker itself. When a marker has a emissive color, the marker can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `Marker`
2. **Visual Properties** to `Advanced`

Shininess — Shininess of marker
75 (default) | scalar in the range of 1 to 128

Shininess of the marker, specified as a scalar in the range of 0 to 128. This parameter affects the sharpness of the specular reflections of the marker. A marker with high shininess has a mirror-like appearance, and marker with low shininess has a more low-gloss or satin appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `Marker`
2. **Visual Properties** to `Advanced`

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Topics

“Representing Solid Inertia”
“Specifying Custom Inertias”

Inertia Sensor

Sensor to measure the inertial properties of body groups or mechanisms

Libraries:
Simscape / Multibody / Body Elements

Description

Use the Inertia Sensor block to measure the inertial properties for collections of body elements in your Simscape Multibody model. Parameters that the Inertia Sensor block can measure include:

- Mass
- Center of mass
- Inertia matrix
- Centered inertia matrix
- Principal inertia matrix
- Orientation of principal inertia frame

The Inertia Sensor block connects to a frame in your Simscape Multibody model and uses the frame to determine which body group or mechanism is measured.

A body group is the set of all body elements that are connected to each other directly or through rigid transforms. The body group may include solid blocks, inertia blocks, variable mass blocks, or flexible body blocks. By default, two body elements that are connected by a weld joint are not considered to be a part of the same body group, though you may choose to group these bodies by enabling the **Span Weld Joints** parameter.

A mechanism is the set of all connected body elements within the model. With mechanisms, you can choose to exclude grounded bodies by enabling the **Exclude Grounded Bodies** parameter. A grounded body is any rigid body that is rigidly connected to a World Frame block.

Ports

Input

S — Sensor port
frame

Sensor port to attach to a frame in the model. Use this port to determine which body group or mechanism is measured.

M — Custom measurement frame port
frame

When `Custom` is selected, a second port, M, appears on the Inertia Sensor block. The Inertia Sensor block measures inertial properties relative to the frame connected to the M port.

Dependencies

To enable this port, set the **Measurement Frame** parameter to `Custom`.

Output

m — Mass
scalar

Total mass for the collection of body elements.

Dependencies

To enable this port, select the **Mass** check box.

com — Center of Mass
vector

The center of mass for the collection of body elements.

Dependencies

To enable this port, select the **Center of Mass** check box.

I — Inertia Matrix
matrix

The inertia matrix for the collection of body elements.

Dependencies

To enable this port, select the **Inertia Matrix** check box.

Ic — Centered Inertia Matrix
matrix

The centered inertia matrix for the collection of body elements.

Dependencies

To enable this port, select the **Centered Inertia Matrix** check box.

Ip — Principal Inertia Matrix
matrix

The principal inertia matrix for the collection of body elements.

Dependencies

To enable this port, select the **Principal Inertia Matrix** check box.

R — Rotation Matrix
matrix

The rotation matrix for orientation of principal inertia frame relative to measurement frame.

Dependencies

To enable this port, select the **Rotation Matrix** check box.

Parameters

Sensor Extent

Sensor Extent — Determine the extent that the Inertia Sensor block measures
`Body Group` (default) | `Mechanism`

Extent that the Inertia Sensor block measures, defined as Body Group or Mechanism.

`Body Group`

When `Body Group` is selected, the Inertia Sensor block measures inertial properties for the collection of body elements that are connected to each other directly or through rigid transforms. This includes solid, inertia, variable mass, or flexible bodies.

`Mechanism`

When `Mechanism` is selected, the Inertia Sensor block measures inertial properties for all connected body elements within the model.

Sensor Extent: Span Weld Joints — Include bodies connected by weld joints in a body group
`cleared` (default) | checked

If selected, this option causes body groups to include body elements that are connected via a weld joint.

Dependencies

To enable this option, set **Sensor Extent** to `Body Group`.

Sensor Extent: Exclude Grounded Bodies — Do not measure inertial properties of grounded bodies
`cleared` (default) | checked

If selected, this option causes the Inertia Sensor block to measure inertial properties for the collection of body elements connected to the same mechanism as the Inertia Sensor block excluding any grounded bodies.

Dependencies

To enable this option, set **Sensor Extent** to `Mechanism`.

Measurement Frame — Determine the frame that the Inertia Sensor block uses for measurements
`Attached` (default) | `World` | `Custom`

`Attached`

When `Attached` is selected, the Inertia Sensor block measures inertial properties relative to the same frame that is connected to the S port.

`World`

When `World` is selected, the Inertia Sensor block measures inertial properties relative to the World Frame.

`Custom`

When `Custom` is selected, the M port is exposed. The Inertia Sensor block measures inertial properties relative to the frame connected to the M port.

Mass — Total mass of measured bodies
`cleared` (default) | checked

Select the **Mass** check box to measure the mass of the collection of body elements.

Center of Mass — Center of mass coordinates
`cleared` (default) | checked

Select the **Center of Mass** check box to measure the center of mass of the collection of body elements, output as a 3-by-1 vector relative to the measurement frame. If the mass of the measured body group or mechanism is zero, the center of mass is undefined and a runtime error occurs.

Inertia Matrix — Measurement of the inertia tensor
`cleared` (default) | checked

Select the **Inertia Matrix** check box to measure the inertia tensor of the collection of body elements, output as a 3-by-3 matrix relative to the measurement frame.

Centered Inertia Matrix — Measurement of the inertia tensor relative to the centered frame
`cleared` (default) | checked

Select the **Centered Inertia Matrix** check box to measure the inertia tensor of the collection of body elements, output as a 3-by-3 matrix relative to the centered frame. The centered frame is a frame whose origin coincides with the center of mass and whose axes are aligned with those of the measurement frame. If the mass of the measured body group or mechanism is zero, the centered inertia matrix is undefined and a runtime error occurs.

Principal Inertia Matrix — Measurement of the inertia tensor with respect to the principal inertia frame
`cleared` (default) | checked

Select the **Principal Inertia Matrix** check box to measure the inertia tensor of the collection of body elements, output as a 3-by-3 matrix with respect to the principal inertia frame.

The principal inertia frame is a frame with an origin that coincides with the center of mass and axes that are aligned with the principal axes of inertia. If the mass of the measured body group or mechanism is zero, the principal inertia matrix is undefined and a runtime error occurs.

Rotation Matrix — Orientation of the principal inertia frame
`cleared` (default) | checked

Select the **Rotation Matrix** check box to measure the orientation of the principal axes of inertia with respect to the measurement frame as a 3-by-3 matrix. If the mass of the measured body group or mechanism is zero, the rotation matrix is undefined and a runtime error occurs.

Graphic

Type — Sensor visualization setting
`Principal Inertia Frame` (default) | `Equivalent Inertia Ellipsoid` | `None`

Display of the inertial properties for the sensed collection of body elements, specified as `Principal Inertia Frame`, `Equivalent Inertia Ellipsoid`, or `None`.

Set the parameter to `Principal Inertia Frame` to represent the sensed inertial properties as the principal inertia frame. The principal inertia frame is a frame with an origin that coincides with the center of mass and axes that are aligned with the principal axes of inertia.

Set the parameter to `Equivalent Inertia Ellipsoid` to represent the inertia properties as an ellipsoid that has the same mass, moments, and products of inertia as the group of bodies being sensed.

Set the parameter to `None` to hide the visualization of the sensed inertia properties.

Size — Display size for the principal inertia frame
20 `pixels` (default) | positive scalar

Display size for the principal inertia frame, specified as a positive scalar.

Dependencies

To enable this parameter, set **Type** to `Principal Inertia Frame`.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `Principal Inertia Frame` or `Equivalent Inertia Ellipsoid`.

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the graphic and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set **Type** to `Principal Inertia Frame` or `Equivalent Inertia Ellipsoid`.

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `Principal Inertia Frame` or `Equivalent Inertia Ellipsoid`
2. **Visual Properties** to `Simple`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the graphic due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `Principal Inertia Frame` or `Equivalent Inertia Ellipsoid`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the graphic.

Dependencies

To enable this parameter, set:

1. **Type** to `Principal Inertia Frame` or `Equivalent Inertia Ellipsoid`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the graphic itself. When a graphic has a emissive color, the graphic can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `Principal Inertia Frame` or `Equivalent Inertia Ellipsoid`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `Principal Inertia Frame` or `Equivalent Inertia Ellipsoid`
2. **Visual Properties** to `Advanced`

Version History

Introduced in R2019b

Extended Capabilities

C/C++ Code Generation

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

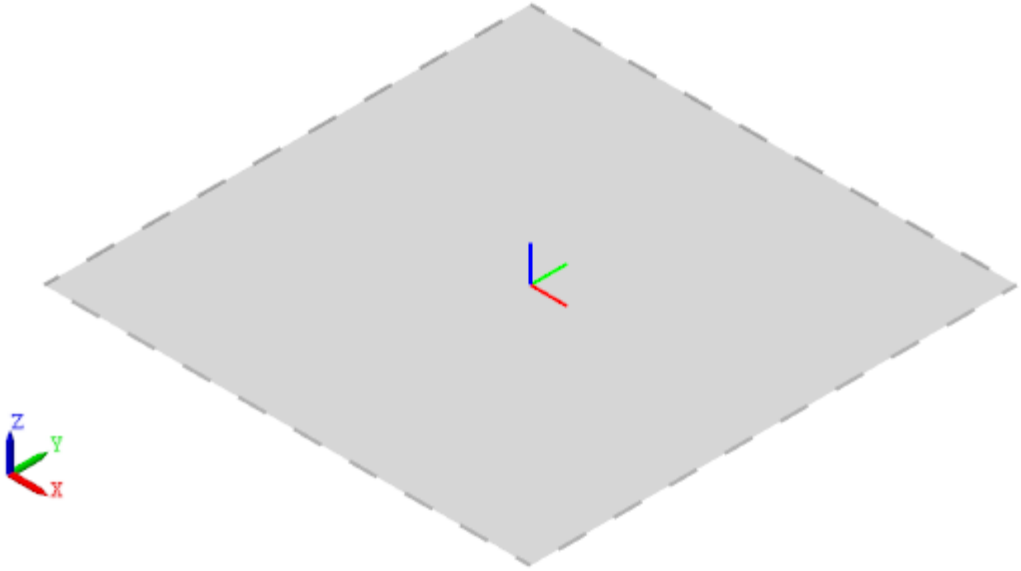
Infinite Plane

Infinite plane for contact modeling

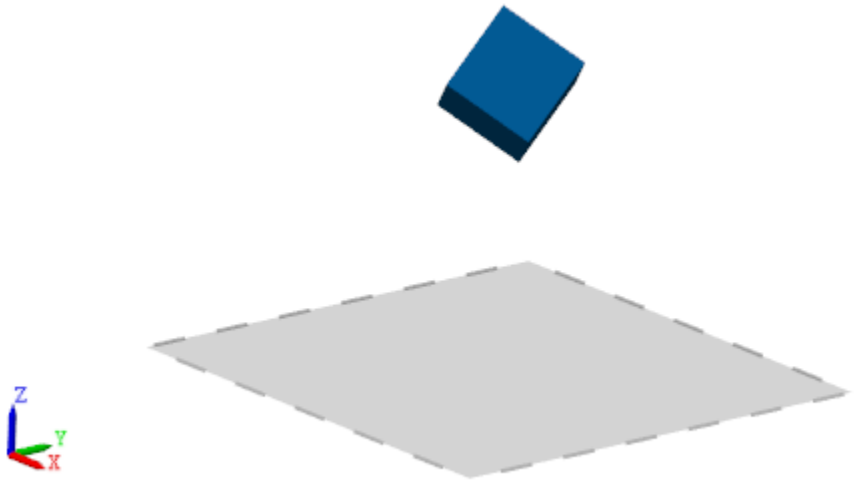
Libraries:
Simscape / Multibody / Curves and Surfaces

Description

The Infinite Plane block exports an infinite plane for modeling contact problems. You can model contact between the plane and many types of geometries, such as all types of solids and convex hulls in the Body Element library and the point geometry in the Curves and Surfaces library. Note that for contact, you must use the positive normal side of the plane for contact, which is in the direction indicated by the blue axis of the frame. Otherwise, the contacting part will be ejected to the normal side through the plane.



The Infinite Plane block is useful in planar contact modeling. For example, the Infinite Plane block is much more effective at modeling a ground plane than the Brick Solid block.



Ports

Frame

R — Reference frame
frame

Infinite plane reference frame. Connect this frame to another block to specify the location and orientation of the infinite plane.

Geometry

G — Geometry
frame

Geometry that represents the plane defined by this block. Connect this port to a Spatial Contact Force block to model contacts on the plane.

Parameters

Graphic

Type — Visual representation of plane
`Rectangle` (default) | `None`

Visual representation of the plane. Set this parameter to `Rectangle` to show the visual representation of the infinite plane as a rectangle with dash-line boundaries. Set this parameter to `None` to eliminate the plane from the model visualization.

Width (X) — Width of plane visual representation
`1 m` (default) | positive integer

Width of the plane's visual representation. Note that the actual plane has infinite size.

Dependencies

To enable this parameter, set **Type** to `Rectangle`.

Height (Y) — Height of plane visual representation
`1 m` (default) | positive integer

Height of the plane's visual representation. Note that the actual plane has infinite size.

Dependencies

To enable this parameter, set **Type** to `Rectangle`.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify diffuse color and opacity. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `Rectangle`.

Diffuse Color — Graphic color
[0.5 0.5 0.5] (default) | three-element vector | four-element vector

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `Rectangle`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
0.25 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `Rectangle`
2. **Visual Properties** to `Simple`

Specular Color — Highlight color
[0.5 0.5 0.5 1.0] (default) | three-element vector | four-element vector

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.

Dependencies

To enable this parameter, set:

1. **Type** to `Rectangle`
2. **Visual Properties** to `Advanced`

Ambient Color — Shadow color
[0.15 0.15 0.15 1.0] (default) | three-element vector | four-element vector

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 (A) specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `Rectangle`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | three-element vector | four-element vector

Graphic 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 (A) specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `Rectangle`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
75 (default) | scalar in the range of 1 to 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.

Dependencies

To enable this parameter, set:

1. **Type** to `Rectangle`
2. **Visual Properties** to `Advanced`

Version History

Introduced in R2020b

Extended Capabilities

C/C++ Code Generation

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

See Also

Spatial Contact Force

Topics

“Modeling Contact Force Between Two Solids”

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.

Version History

Introduced in R2013a

Extended Capabilities

C/C++ Code Generation

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

See Also

External Force and Torque | Spring and Damper Force | Inverse Square Law Force

Topics

“Actuating and Sensing with Physical Signals”

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**.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

External Force and Torque | Internal Force | Spring and Damper Force

Topics

“Actuating and Sensing with Physical Signals”
“Modeling Gravity”

Lead Screw Joint

Joint with coupled rotational and translational degrees of freedom

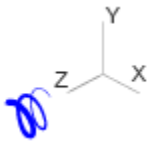
Library

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.
`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.

Mode Configuration

Specify the mode of the joint. The joint mode can be normal or disengaged throughout the simulation, or you can provide an input signal to change the mode during the simulation.

Mode

Select one of the following options to specify the mode of the joint. The default setting is `Normal`.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.

Method	Description
`Provided by Input`	This option exposes the **mode** port that you can connect to an input signal to change the joint mode during the simulation. The joint mode is normal when the input signal is `0` and disengaged when the input signal is `-1`. The joint mode can be changed many times during the simulation.

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:

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

Mode Port

Mode configuration provides the following port:

- mode — Value of the mode of the joint. If the input is equal to `0`, the joint behaves normally. If the input is equal to `-1`, the joint behaves as disengaged.

Version History

Introduced in R2015a

Extended Capabilities

C/C++ Code Generation

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

See Also

Prismatic Joint | Revolute Joint

Topics

“Cartesian 3D Printer”

“Using the Lead Screw Joint Block - Linear Actuator”
“Lead Screw with Friction”

Mechanism Configuration

Mechanism-wide gravity and simulation parameters

Libraries:
Simscape / Multibody / Utilities

Description

The Mechanism Configuration block specifies the gravity and simulation parameters of the mechanism to which the block connects. The simulation parameters include the perturbation value for computing numerical partial derivatives during linearization and the iteration number for the joint mode transition.

A Simscape Multibody network can have at most one instance of the Mechanism Configuration block. If a model does not have the Mechanism Configuration block, the uniform gravity applied to the mechanism is zero.

If a Simscape Multibody network contains a Gravitational Field block, set the **Uniform Gravity** parameter of the Mechanism Configuration block to `None`.

Ports

Frame

C — Port to connect mechanism
frame

Port to use to connect the mechanism. The settings of the Mechanism Configuration block apply only to the mechanism to which the block connects.

Input

g — Uniform gravity
physical signal

Uniform gravity of the mechanism to which the Mechanism Configuration block connects.

Dependencies

To enable this port, set **Uniform Gravity** to `Time-Varying`.

Parameters

Uniform Gravity

Uniform Gravity — Method to specify gravitational acceleration vector
`Constant` (default) | `Time-Varying` | `None`

Method used to calculate the uniform gravity for the mechanism to which the Mechanism Configuration block connects, specified as one of these options:

Method	Description
`Constant`	Use the **Gravity** parameter to specify the gravitational acceleration vector that remains constant in space and in time.
`Time-Varying`	Use the physical port **g** to specify the gravitational acceleration vector that remains constant in space but varies in time.
`None`	Specify zero uniform gravity for the mechanism. This option must be used if the Simscape Multibody network contains a Gravitational Field block.

Gravity — Constant gravitational acceleration vector
`[0 0 –9.80665] m/s^2` (default) | 1-by-3 vector

Constant gravitational acceleration vector, specified as a 1-by-3 vector, $[g_x\ g_y\ g_z]$. The block resolves the vector in the world frame of the mechanical system.

Dependencies

To enable this parameter, specify **Uniform Gravity** to `Constant`.

Linearization Delta — Perturbation value for linearization
0.001 (default) | positive scalar

Perturbation value for computing numerical partial derivatives during linearization, specified as a positive scalar. For more information about linearization, see “Linearize Nonlinear Models” (Simulink Control Design) and `linmod`.

Joint Mode Transition

Nonlinear Iterations — Number of Newton iterations for joints transitioning from disengaged to normal mode
2 (default) | positive integer

Number of Newton iterations to use for solving the positions of all the joints in the mechanism when the mechanism has one or more joints that transition from disengaged to normal mode, specified as a positive integer.

When a joint transitions from disengaged to normal mode, if that joint has large discrepancies in its location and orientation, the Simscape Multibody network may need a larger number of iterations to compute the required positions of all joints in the mechanism that satisfy the kinematic constraints of that transitioning joint.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Gravitational Field | `linmod`

Topics

“Modeling Gravity”
“Linearize Nonlinear Models” (Simulink Control Design)
“Linearize Simulink Model Using Model Linearizer” (Simulink Control Design)

Magic Formula Tire Force and Torque

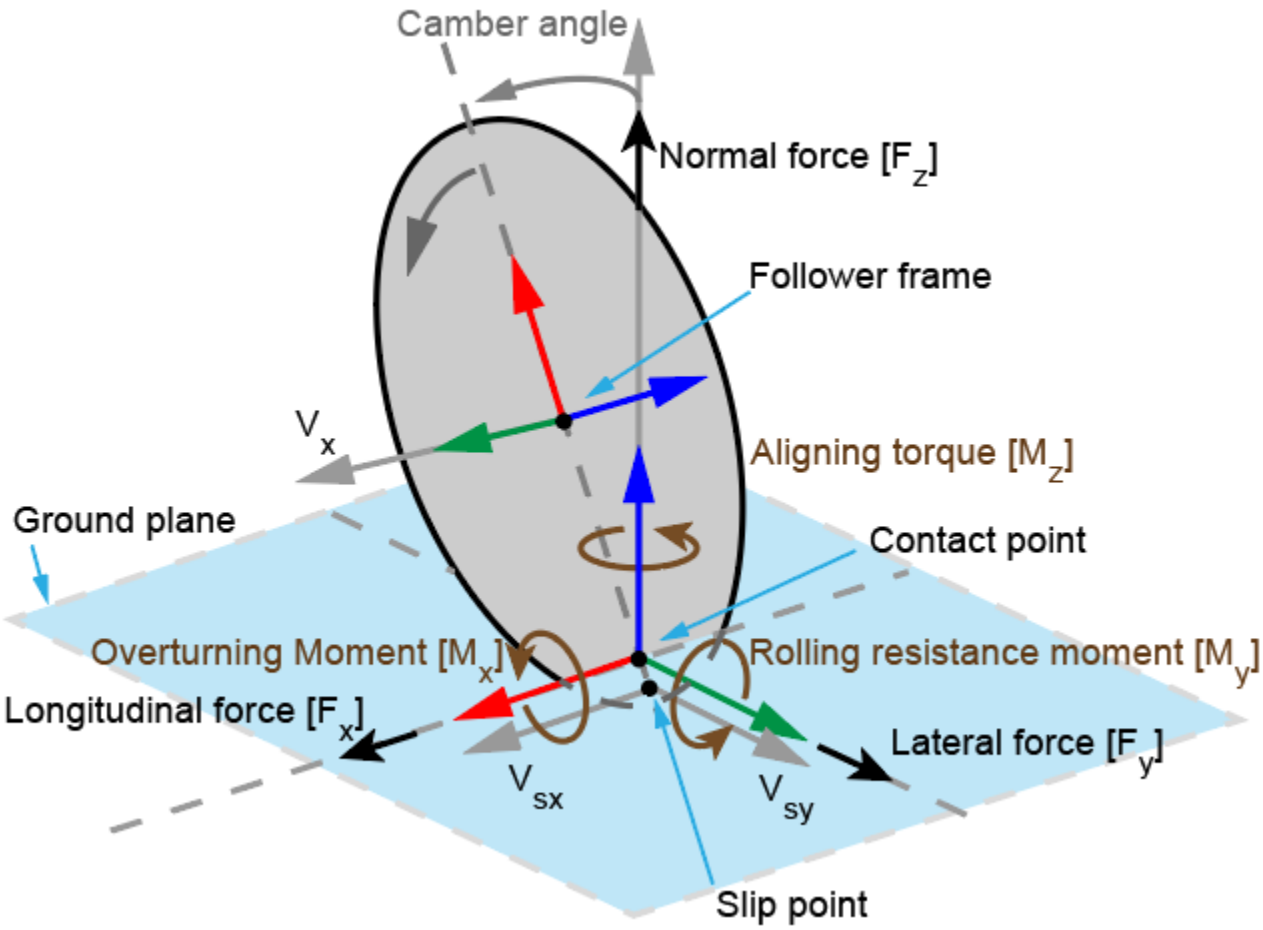
Apply steady-state tire force and torque by using Magic Formula tire equations

Libraries:
Simscape / Multibody / Forces and Torques

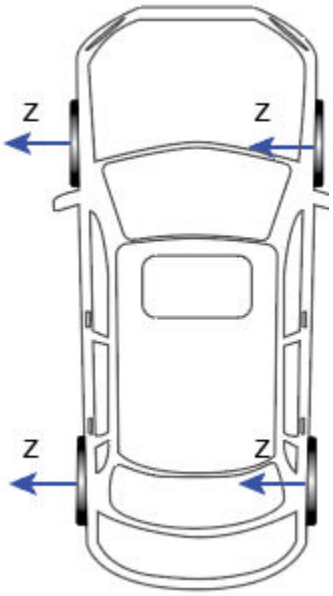
Description

The Magic Formula Tire Force and Torque block implements the combined slip steady-state magic formula model and can optionally include turn slip effects [1]. The block applies the force and torque to the follower frame.

The block calculates only the tire force and torque. To model the geometry and inertia properties of the tire, you must use a solid block, such as the Cylindrical Solid block. The block supports only passenger car tires. The magic formula tire model assumes that passenger car tires are disks, as shown in the diagram.



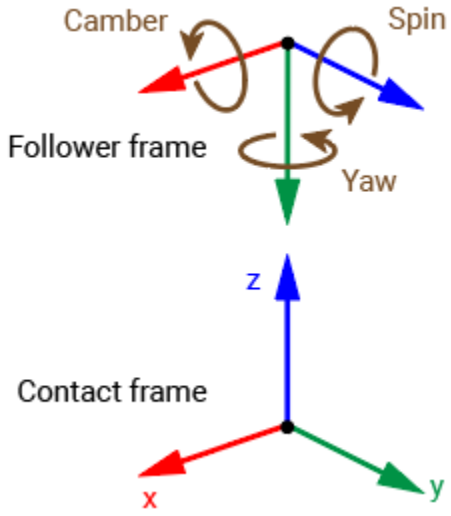
The contact frame is located at the contact point between the tire and ground plane. The follower frame is located at the center of the tire and rotates with the tire. To correctly orient the tires on a vehicle, you must align the *z*-axes of the follower frames with the blue arrows, as shown in the diagram.



To specify the properties of a tire model, generate a scalar structure array by using the `simscape.multibody.tirread` function and enter the array in the **Tire Parameters** parameter. Note that the block observes the ISO sign convention. To indicate which side of the vehicle the tire is mounted to, use the **Tire Side** parameter.

The **B** port, which represents the surface that the tire contacts, must be connected to an Infinite Plane or Grid Surface block. The geometry of the contact surface can move or be fixed relative to the world frame. The **F** port represents the follower frame. Note that if the follower frame penetrates the ground connected to the **B** port, the magic formula tire force and torque equations become degenerate and the block does not apply force and torque. This case happens when the follower frame of the block is below the geometry connected to the **B** port.

The yaw, camber, and spin angles correspond to a *y*-*x*-*z* sequence rotation about the follower frame of a tire. The image shows the contact and follower frames of the tire at zero configuration.



Ports

Geometry

B — Base geometry
geometry

Base geometry that represents the ground pane that the tire contacts. You must connect this port to the Infinite Plane or Grid Surface block.

Frame

F — Follower frame
frame

Follower frame that represents the tire. The frame origin is located at the center of the tire.

Input

lmux — Scaling factor of longitudinal friction coefficient
physical signal

Physical signal port that accepts the scaling factor of the longitudinal friction coefficient of the tire.

Dependencies

To enable this port, under **Scaling Coefficients**, set **LMUX** to `Provided by Input`.

lmuy — Scaling factor of lateral friction coefficient
physical signal

Physical signal port that accepts the scaling factor of the lateral friction coefficient of the tire.

Dependencies

To enable this port, under **Scaling Coefficients**, set **LMUY** to `Provided by Input`.

Output

Force/Torque

ft — Tire force
physical signal

Physical signal output port that provides the magic formula tire force that is applied to the follower frame of the block. The output contains three parts:

- F_x is the longitudinal force tangential to the ground surface at the contact point.
- F_y is the lateral force which is orthogonal to the plane defined by F_x and F_z.
- F_z is the normal force that is normal to the ground surface at the contact point.

Dependencies

To enable this port, under **Sensing > Force/Torque**, select **Tire Force**.

tt — Tire torque
physical signal

Physical signal output port that provides the magic formula tire torque that is applied to the follower frame of the block.

Dependencies

To enable this port, under **Sensing > Force/Torque**, select **Tire Torque**.

t — Pneumatic trail
physical signal

Physical signal output port that provides the distance from the contact point to the point of the resultant lateral force. The value has a unit of length.

You can use the pneumatic trail to compute the aligning torque, M_z.

Dependencies

To enable this port, under **Sensing > Force/Torque**, select **Pneumatic Trail**.

Slip

kappa — Longitudinal slip
physical signal

Physical signal output port that provides the ratio of the longitudinal slip velocity to the longitudinal speed of the tire. The value is dimensionless.

Dependencies

To enable this port, under **Sensing > Slip**, select **Longitudinal Slip**.

kappas — Saturated longitudinal slip
physical signal

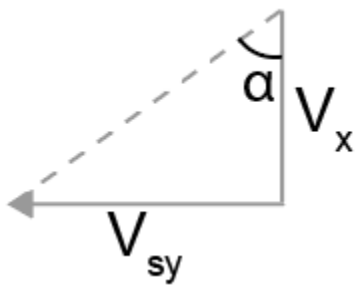
Physical signal output port that provides the longitudinal slip saturated to always be within the limits defined by the KPUMIN and KPUMAX parameters.

Dependencies

To enable this port, under **Sensing > Slip**, select **Saturated Longitudinal Slip**.

alpha — Slip angle
physical signal

Physical signal output port that provides the angle of the right triangle made by the lateral slip velocity, V_{sy} and the longitudinal speed, V_x. The value has a unit of angle.



Dependencies

To enable this port, under **Sensing > Slip**, select **Slip Angle**.

alphas — Saturated slip angle
physical signal

Physical signal output port that provides the slip angle saturated to always be within the limits defined by the ALPMIN and ALPMAX parameters. The value has a unit of angle.

Dependencies

To enable this port, under **Sensing > Slip**, select **Saturated Slip Angle**.

phit — Turn slip
physical signal

Physical signal output port that provides the ratio of the tire yaw velocity to the magnitude of the tire velocity in the *xy*-plane of the contact frame. The value has a unit of angle/length.

Turn slip is useful when modeling low speed cornering, such as parking maneuvers.

Dependencies

To enable this port, under **Sensing > Slip**, select **Turn Slip**.

Linear Velocity

vx — Relative longitudinal velocity
physical signal

Physical signal output port that provides the component of the relative velocity between the tire frame and the contact point on the geometry along the *x*-direction of the contact frame. The value has a unit of length/time.

Dependencies

To enable this port, under **Sensing > Linear Velocity**, select **Relative Longitudinal Velocity**.

vy — Relative lateral velocity
physical signal

Physical signal output port that provides the component of the relative velocity between the tire frame and the contact point on the geometry along the *y*-direction of the contact frame. The unit of the meters/sec.

Dependencies

To enable this port, under **Sensing > Linear Velocity**, select **Relative Lateral Velocity**.

vsx — Longitudinal slip velocity
physical signal

Physical signal output port that provides the component of the relative velocity between the slip point on the tire and the coincident point on the geometry along the *x*-direction of the contact frame. The value has a unit of length/time.

Dependencies

To enable this port, under **Sensing > Linear Velocity**, select **Longitudinal Slip Velocity**.

vsy — Lateral slip velocity
physical signal

Physical signal output port that provides the component of the relative velocity between the slip point on the tire and the coincident point on the geometry along the *y*-direction of the contact frame. The value has a unit of length/time.

Dependencies

To enable this port, under **Sensing > Linear Velocity**, select **Lateral Slip Velocity**.

Yaw

psid — Yaw velocity
physical signal

Physical signal output port that provides the first derivative of the yaw angle. The value has a unit of angle/time.

Dependencies

To enable this port, under **Sensing > Yaw**, select **Velocity**.

Camber

gamma — Camber angle
physical signal

Physical signal output port that provides the camber angle of the tire. The value has a unit of angle.

Dependencies

To enable this port, under **Sensing > Camber**, select **Angle**.

gammas — Saturated camber angle
physical signal

Physical signal output port that provides the camber angle of the tire saturated to always be within the limits defined by the CAMMIN and CAMMAX parameters. The value has a unit of angle.

Dependencies

To enable this port, under **Sensing > Camber**, select **Angle**.

gammad — Camber velocity
physical signal

Physical signal output port that provides the first derivative of the camber angle. The value has a unit of angle/time.

Dependencies

To enable this port, under **Sensing > Camber**, select **Velocity**.

Spin

omega — Spin velocity
physical signal

Physical signal output port that provides the first derivative of the spin angle. The value has a unit of angle/time.

Dependencies

To enable this port, under **Sensing > Spin**, select **Velocity**.

Tire Radius

romega — Free radius of tire
physical signal

Physical signal output port that provides the free radius of the tire. The radius increases as the tire rotates faster. The value has a unit of meter.

Dependencies

To enable this port, under **Sensing > Tire Radius**, select **Free Radius**.

rl — Loaded radius of tire
physical signal

Physical signal output port that provides the distance from the tire frame to the contact point. The value has a unit of meter.

Dependencies

To enable this port, under **Sensing > Tire Radius**, select **Loaded Radius**.

re — Effective rolling radius
physical signal

Physical signal output port that provides the distance from the tire frame to the slip point. The value has a unit of meter.

Dependencies

To enable this port, under **Sensing > Tire Radius**, select **Effective Rolling Radius**.

Friction

mux — Longitudinal friction coefficient
physical signal

Physical signal output port that provides the longitudinal friction coefficient of the tire computed by the magic formula equations.

Dependencies

To enable this port, under **Sensing > Friction**, select **Longitudinal Friction Coefficient**.

muy — Lateral friction coefficient
physical signal

Physical signal output port that provides the lateral friction coefficient of the tire computed by the magic formula equations.

Dependencies

To enable this port, under **Sensing > Friction**, select **Lateral Friction Coefficient**.

Contact Frame

Rb — Base rotation
physical signal

Physical signal port that outputs a 3-by-3 rotation matrix that maps the vectors in the contact frame to vectors in the reference frame of the base geometry. The output signal is resolved in the reference frame associated with the base geometry.

Dependencies

To enable this port, in the **Sensing > Contact Frame** section, select **Base Rotation**.

pb — Base translation
physical signal

Physical signal port that outputs at 3-by-1 vector that contains the coordinates of the origin of the contact frame resolved in the reference frame of the base geometry.

Dependencies

To enable this port, in the **Sensing > Contact Frame** section, select **Base Translation**.

Rf — Follower rotation
physical signal

Physical signal port that outputs a 3-by-3 rotation matrix that maps vectors in the contact frame to vectors in the reference frame of the follower geometry. The output signal is resolved in the reference frame associated with the follower geometry.

Dependencies

To enable this port, in the **Sensing > Contact Frame** section, select **Follower Rotation**.

pf — Follower translation
physical signal

Physical signal port that outputs at 3-by-1 vector that contains the coordinates of the origin of the contact frame resolved in the reference frame of the follower geometry.

Dependencies

To enable this port, in the **Sensing > Contact Frame** section, select **Follower Translation**.

Parameters

Tire Side — Tire position during modeling
`Left` (default) | `Right`

Tire position during modeling, specified as either `Left` or `Right`. Set the parameter to the side of the vehicle to which the tire is mounted.

Tire Parameters — Parameters of tire
[] (default) | scalar structure array

Tire parameters, specified as a scalar structure array. Use the `simscape.multibody.tirread` function to generate the structure array from a TIR file.

Slip Mode — Slip mode
`Combined` (default) | `Combined + Turn`

Slip mode, specified as either `Combined` or `Combined + Turn`.

To model combined slip, select `Combined`. To model combined slip with turn slip effects, select `Combined + Turn`.

Scaling Coefficients

LMUX — Scaling factor of longitudinal friction coefficient
`From Tire Parameters` (default) | `Provided by Input`

Scaling factor for the longitudinal friction coefficient. Select `From Tire Parameters` to use the constant scaling factor provided by the TIR file, or select `Provided by Input` to use input signals as scaling factors.

LMUY — Scaling factor of lateral friction coefficient
`From Tire Parameters` (default) | `Provided by Input`

Scaling factor for the lateral friction coefficient. Select `From Tire Parameters` to use the constant scaling factor provided by the TIR file, or select `Provided by Input` to use input signals as scaling factors.

Sensing

Force/Torque

Resolution Frame — Frame to resolve measurements
`Contact` (default) | `Follower`

Frame used to resolve the calculated tire force and torque, specified as either `Contact` or `Follower`.

Version History

Introduced in R2021b

R2023a: Base geometry supports the Grid Surface block

Use the Grid Surface block to represent the ground surface.

References

[1] Pacejka, Hans B., and Igo Besselink. *Tire and Vehicle Dynamics*. 3rd. Engineering Automotive Engineering. Amsterdam: Elsevier/Butterworth-Heinemann, 2012.

[2] Besselink, I. J.M., A. J.C. Schmeitz, and H. B. Pacejka. "An Improved Magic Formula/Swift Tyre Model That Can Handle Inflation Pressure Changes." *Vehicle System Dynamics* 48, no. sup1 (December 2010): 337–52. https://doi.org/10.1080/00423111003748088.

[3] van der Hofstad, R. H. M. T. "Study on improving the MF-Swift tyre model." (2010).

Extended Capabilities

C/C++ Code Generation

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

See Also

`simscape.multibody.tirread`

Topics

"Vehicle Dynamics - Car with Heave and Roll"

Pin Slot Joint

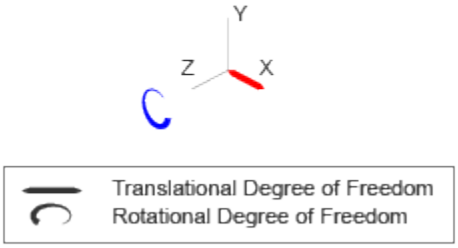
Joint with one prismatic and one revolute primitives possessing mutually orthogonal motion axes

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.



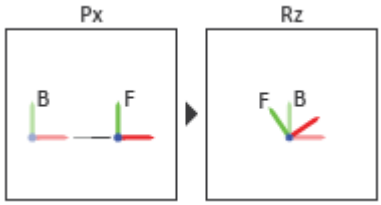
Joint Degrees of Freedom

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 `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.

Mode Configuration

Specify the mode of the joint. The joint mode can be normal or disengaged throughout the simulation, or you can provide an input signal to change the mode during the simulation.

Mode

Select one of the following options to specify the mode of the joint. The default setting is `Normal`.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.

Method	Description
`Provided by Input`	This option exposes the **mode** port that you can connect to an input signal to change the joint mode during the simulation. The joint mode is normal when the input signal is `0` and disengaged when the input signal is `-1`. The joint mode can be changed many times during the simulation.

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:

- fx — Actuation force acting on the X prismatic joint primitive
- px — Desired trajectory of the X 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:

- px — Position of the X prismatic joint primitive
- vx — Velocity of the X prismatic joint primitive
- ax — Acceleration of the X prismatic joint primitive
- fx — Actuation force acting on the X prismatic joint primitive
- fllx — Force due to contact with the lower limit of the X prismatic joint primitive
- fulx — Force due to contact with the upper limit of the X 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
- tllz — Torque due to contact with the lower limit of the Z revolute joint primitive
- tulz — Torque due to contact with the upper limit of 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

Mode Port

Mode configuration provides the following port:

- mode — Value of the mode of the joint. If the input is equal to `0`, the joint behaves normally. If the input is equal to `-1`, the joint behaves as disengaged.

Version History

Introduced in R2013a

Extended Capabilities

C/C++ Code Generation

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

See Also

Cylindrical Joint | Revolute Joint | Prismatic Joint

Topics

“Actuating and Sensing with Physical Signals”
“Motion Sensing”
“Ratchet Lifter”
“Rotational Measurements”
“Translational Measurements”
“Using the Rack-Pinion Block - Windshield Wiper Mechanism”
“Using the Point-On-Curve Block : Flapping Wing Mechanism”

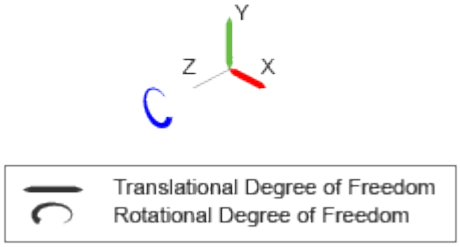
Planar Joint

Joint with one rotational and two translational degrees of freedom

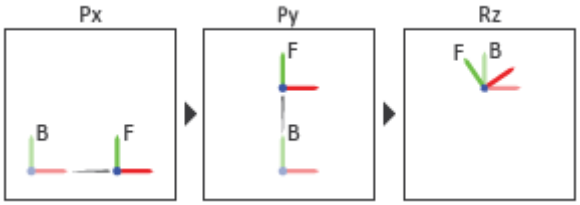
Libraries:
Simscape / Multibody / Joints

Description

The Planar Joint block provides one rotational and two translational degrees of freedom between two frames.



The block constrains the origin of the follower frame to the *x*-*y* plane of the base frame and the relative transformation of the follower frame follows this order:



First, the follower frame moves along the *x* and *y* axes of the base frame, respectively, and then rotates about the *z*-axis of the follower frame generated after the translations.

For information about how to specify joint blocks, see “Modeling Joint Connections”.

Ports

Frame

B — Base frame
frame

Base frame of the joint block.

F — Follower frame
frame

Follower frame of the joint block.

Input

X Prismatic Primitive (Px)

fx — Actuation force
physical signal

Physical signal input port that accepts the actuation force for the joint primitive. The block applies the force equally and oppositely to the base and follower frames of the joint along the *x*-axis of the base frame.

Dependencies

To enable this port, under **X Prismatic Primitive (Px)** > **Actuation**, set **Force** to `Provided by Input`.

px — Motion profile
physical signal

Physical signal input port that accepts the motion profile for the joint primitive. The signal provides the displacement of the follower frame with respect to the base frame along the *x*-axis of the base frame. The signal must also contain the first and second derivatives of the displacement.

Dependencies

To enable this port, under **X Prismatic Primitive (Px)** > **Actuation**, set **Motion** to `Provided by Input`.

Y Prismatic Primitive (Py)

fy — Actuation force
physical signal

Physical signal input port that accepts the actuation force for the joint primitive. The block applies this force equally and oppositely to the base and follower frames of the joint along the *y*-axis of the base frame.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py)** > **Actuation**, set **Force** to `Provided by Input`.

py — Motion profile
physical signal

Physical signal input port that accepts the motion profile for the joint primitive. The signal provides the displacement of the follower frame with respect to the base frame along the *y*-axis of the base frame. The signal must also contain the first and second derivatives of the displacement.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py)** > **Actuation**, set **Motion** to `Provided by Input`.

Z Revolute Primitive (Rz)

tz — Actuation torque
physical signal

Physical signal input port that accepts the actuation torque for the joint primitive. The block applies this torque equally and oppositely to both the base and follower frames of the joint primitive. The torque is about the *z*-axis of the base frame. The *z*-axes of the follower and base frames are aligned with each other during simulation.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz)** > **Actuation**, set **Torque** to `Provided by Input`.

qz — Motion profile
physical signal

Physical signal input port that accepts the motion profile for the joint primitive. The signal provides the rotation of the follower frame with respect to the base frame about the *z*-axis of the follower frame. The signal must also contain the first and second derivatives of the rotation.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz)** > **Actuation**, set **Motion** to `Provided by Input`.

Mode Configuration

mode — Joint mode control
scalar

Input port that controls the mode of the joint. The signal is a unitless scalar. The joint operates in normal mode when the input signal is `0` and operates in disengaged mode when the input signal is `-1`. You can change between the two modes at any time during simulation.

Dependencies

To enable this port, under **Mode Configuration**, set **Mode** to `Provided by Input`.

Output

X Prismatic Primitive (Px)

px — Position of primitive
physical signal

Physical signal port that outputs the position of the joint primitive. The value is the displacement of the follower frame with respect to the base frame in the *x*-direction of the base frame.

Dependencies

To enable this port, under **X Prismatic Primitive (Px)** > **Sensing**, select **Position**.

vx — Velocity of primitive
physical signal

Physical signal port that outputs the velocity of the joint primitive. The value is the first derivative of the signal from the port **px**.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Sensing**, select **Velocity**.

ax — Acceleration of primitive
physical signal

Physical signal port that outputs the acceleration of the joint primitive. The value is the second derivative of the signal from the port **px**.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Sensing**, select **Acceleration**.

fx — Actuator force acting on joint primitive
physical signal

Physical signal port that outputs the actuator force acting on the joint primitive.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Sensing**, select **Actuator Force**.

fllx — Lower-limit force
physical signal

Physical signal port that outputs the lower-limit force. The block applies this force when the joint primitive position is less than the lower bound of the free region. The block applies this force to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Sensing**, select **Lower-Limit Force**.

fulx — Upper-limit force
physical signal

Physical signal port that outputs the upper-limit force. The block applies this force when the joint primitive position exceeds the upper bound of the free region. The block applies this force to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Sensing**, select **Upper-Limit Force**.

Y Prismatic Primitive (Py)

py — Position of primitive
physical signal

Physical signal port that outputs the position of the joint primitive. The value is the displacement of the follower frame with respect to the base frame in the *y*-direction of the base frame.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Sensing**, select **Position**.

vy — Velocity of primitive
physical signal

Physical signal port that outputs the velocity of the joint primitive. The value is the first derivative of the signal from the port **py**.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Sensing**, select **Velocity**.

ay — Acceleration of primitive
physical signal

Physical signal port that outputs the acceleration of the joint primitive. The value is the second derivative of the signal from the port **py**.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Sensing**, select **Acceleration**.

fy — Actuator force acting on joint primitive
physical signal

Physical signal port that outputs the actuator force acting on the joint primitive.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Sensing**, select **Actuator Force**.

flly — Lower-limit force
physical signal

Physical signal port that outputs the lower-limit force. The block applies this force when the joint primitive position is less than the lower bound of the free region. The block applies this force to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Sensing**, select **Lower-Limit Force**.

fuly — Upper-limit force
physical signal

Physical signal port that outputs the upper-limit force. The block applies this force when the joint primitive position exceeds the upper bound of the free region. The block applies this force to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Sensing**, select **Upper-Limit Force**.

Z Revolute Primitive (Rz)

qz — Position of primitive
physical signal

Physical signal port that outputs the position of the joint primitive. The value is the rotation angle of the follower frame with respect to the base frame about the *z*-axis of the follower frame.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz) > Sensing**, select **Position**.

wz — Angular velocity of primitive
physical signal

Physical signal port that outputs the angular velocity of the joint primitive. The value is the first derivative of the signal from the port **qz**.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz) > Sensing**, select **Velocity**.

bz — Angular acceleration of primitive
physical signal

Physical signal port that outputs the angular acceleration of the joint primitive. The value is the second derivative of the signal from the port **qz**.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz) > Sensing**, select **Acceleration**.

tz — Actuator torque acting on joint primitive
physical signal

Physical signal port that outputs the actuator torque acting on the joint primitive.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz) > Sensing**, select **Actuator Torque**.

tllz — Lower-limit torque
physical signal

Physical signal port that outputs the lower-limit torque. The block applies this torque when the joint primitive position is less than the lower bound of the free region. The block applies this torque to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz) > Sensing**, select **Lower-Limit Torque**.

tulz — Upper-limit torque
physical signal

Physical signal port that outputs the upper-limit torque. The block applies this torque when the joint primitive position exceeds the upper bound of the free region. The block applies this torque to both

the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz) > Sensing**, select **Upper-Limit Torque**.

Composite Force/Torque Sensing

fc — Constraint force
physical signal

Physical signal port that outputs the constraint force that acts across the joint. The force maintains the translational constraints of the joint. For more information, see “Measure Joint Constraint Forces”.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Constraint Force**.

tc — Constraint torque
physical signal

Physical signal port that outputs the constraint torque that acts across the joint. The torque maintains the rotational constraints of the joint. For more information, see “Force and Torque Sensing”.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Constraint Torque**.

ft — Total force
physical signal

Physical signal port that outputs the total force that acts across the joint. The total force is the sum of the forces transmitted from one frame to the other through the joint. The force includes the actuation, internal, limit, and constraint forces. See “Force and Torque Sensing” for more information.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Total Force**.

tt — Total torque
physical signal

Physical signal port that outputs the total torque that acts across the joint. The total torque is the sum of the torques transmitted from one frame to the other through the joint. The torque includes the actuation, internal, limit, and constraint torques. For more information, see “Force and Torque Sensing”.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Total Torque**.

Parameters

X Prismatic Primitive (Px)

State Targets

Specify Position Target — Whether to specify position target
`off` (default) | `on`

Select this parameter to specify the position target for the *x* prismatic primitive.

Priority — Priority level of position target
`High (desired)` (default) | `Low (approximate)`

Priority level of the position target, specified as `High (desired)` or `Low (approximate)`. See "Guiding Assembly" for more information.

Dependencies

To enable this parameter, select **Specify Position Target**.

Value — Position target
`0 m` (default) | scalar

Position target of the *x* prismatic primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Position Target**.

Specify Velocity Target — Whether to specify linear velocity target
`off` (default) | `on`

Select this parameter to specify the linear velocity target for the *x* prismatic primitive.

Priority — Priority level of linear velocity target
`High (desired)` (default) | `Low (approximate)`

Priority level of the linear velocity target, specified as `High (desired)` or `Low (approximate)`. See "Guiding Assembly" for more information.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Value — Velocity target of
`0 m/s` (default) | scalar

Linear velocity target for the *x* prismatic primitive, specified as a scalar.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Internal Mechanics

Equilibrium Position — Position where internal force is zero
`0 m` (default) | scalar

Position where the spring force is zero, specified as a scalar in units of length.

Spring Stiffness — Stiffness of force law
0 `N/m` (default) | scalar

Stiffness of the internal spring-damper force law for the joint primitive, specified as a scalar in units of linear stiffness.

Damping Coefficient — Damping coefficient of force law
0 `N(m/s)` (default) | scalar

Damping coefficient of the internal spring-damper force law for the joint primitive, specified as a scalar in units of linear damping coefficient.

Limits

Specify Lower Limit — Whether to specify lower position limit
`off` (default) | `on`

Select this parameter to specify the lower limit of the *x* prismatic primitive.

Bound — Lower bound of free region
-1 `m` (default) | scalar

Lower bound of the free region of the *x* prismatic primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Spring Stiffness — Stiffness of spring at lower bound
1e6 `N/m` (default) | scalar

Stiffness of the spring at the lower bound, specified as a scalar in units of linear stiffness.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Damping Coefficient — Damping coefficient at lower bound
1e3 `N/(m/s)` (default) | scalar

Damping coefficient at the lower bound, specified as a scalar in units of linear damping coefficient.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Transition Region Width — Region to smooth spring and damper forces
1e-4 `m` (default) | scalar

Region to smooth the spring and damper forces, specified as a scalar in units of length.

The block applies the full value of the lower-limit force when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Specify Upper Limit — Whether to specify upper position limit
`off` (default) | `on`

Select this parameter to specify the upper limit of the *x* prismatic primitive..

Bound — Upper bound of free region
`1 m` (default) | scalar

Upper bound for the free region of the joint primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Spring Stiffness — Stiffness of spring at upper bound
`1e6 N/m` (default) | scalar

Stiffness of the spring at the upper bound, specified as a scalar in units of linear stiffness.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Damping Coefficient — Damping coefficient at upper bound
`1e3 N/(m/s)` (default) | scalar

Damping coefficient at the upper bound, specified as a scalar in units of linear damping coefficient.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Transition Region Width — Region to smooth spring and damper forces
`1e-4 m` (default) | scalar

Region to smooth the spring and damper forces, specified as a scalar in units of length.

The block applies the full value of the upper-limit force when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Actuation

Force — Option to provide actuator force
`None` (default) | `Provided by Input` | `Automatically Computed`

Option to provide the actuator force for the joint primitive, specified as one of these values:

Actuation Force Setting	Description
`None`	No actuator force.
`Provided by Input`	Input port **fx** specifies the actuator force for the *x* prismatic primitive.
`Automatically Computed`	The block automatically calculates the amount of force required to satisfy the motion inputs to the mechanism. If you set this parameter to `Automatically Computed`, you do not need to set **Motion** to `Provided by Input` for the same joint primitive. The automatically computed force may to satisfy a motion input elsewhere in the mechanism.

Motion — Option to provide motion
`Automatically Computed` (default) | `Provided by Input`

Option to provide the motion for the joint primitive, specified as one of these values:

Actuation Motion Setting	Description
`Automatically Computed`	The block computes and applies the joint primitive motion based on model dynamics.
`Provided by Input`	Input port **px** specifies the motion for the joint primitive.

Y Prismatic Primitive (Py)

State Targets

Specify Position Target — Whether to specify position target
`off` (default) | `on`

Select this parameter to specify the position target for the *y* prismatic primitive.

Priority — Priority level of position target
`High (desired)` (default) | `Low (approximate)`

Priority level of the position target, specified as `High (desired)` or `Low (approximate)`. See "Guiding Assembly" for more information.

Dependencies

To enable this parameter, select **Specify Position Target**.

Value — Position target
`0 m` (default) | scalar

Position target of the *y* prismatic primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Position Target**.

Specify Velocity Target — Whether to specify linear velocity target
`off` (default) | `on`

Select this parameter to specify the linear velocity target for the *y* prismatic primitive.

Priority — Priority level of linear velocity target
`High (desired)` (default) | `Low (approximate)`

Priority level of the linear velocity target, specified as `High (desired)` or `Low (approximate)`. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Value — Velocity target of
0 `m/s` (default) | scalar

Linear velocity target for the *y* prismatic primitive, specified as a scalar.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Internal Mechanics

Equilibrium Position — Position where internal force is zero
0 `m` (default) | scalar

Position where the spring force is zero, specified as a scalar in units of length.

Spring Stiffness — Stiffness of force law
0 `N/m` (default) | scalar

Stiffness of the internal spring-damper force law for the joint primitive, specified as a scalar in units of linear stiffness.

Damping Coefficient — Damping coefficient of force law
0 `N(m/s)` (default) | scalar

Damping coefficient of the internal spring-damper force law for the joint primitive, specified as a scalar in units of linear damping coefficient.

Limits

Specify Lower Limit — Whether to specify lower position limit
`off` (default) | `on`

Select this parameter to specify the lower limit of the *y* prismatic primitive.

Bound — Lower bound of free region
-1 `m` (default) | scalar

Lower bound of the free region of the *y* prismatic primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Spring Stiffness — Stiffness of spring at lower bound
1e6 `N/m` (default) | scalar

Stiffness of the spring at the lower bound, specified as a scalar in units of linear stiffness.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Damping Coefficient — Damping coefficient at lower bound
1e3 `N/(m/s)` (default) | scalar

Damping coefficient at the lower bound, specified as a scalar in units of linear damping coefficient.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Transition Region Width — Region to smooth spring and damper forces
1e-4 `m` (default) | scalar

Region to smooth the spring and damper forces, specified as a scalar in units of length.

The block applies the full value of the lower-limit force when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Specify Upper Limit — Whether to specify upper position limit
`off` (default) | `on`

Select this parameter to specify the upper limit of the *y* prismatic primitive.

Bound — Upper bound of free region
1 `m` (default) | scalar

Upper bound for the free region of the joint primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Spring Stiffness — Stiffness of spring at upper bound
1e6 `N/m` (default) | scalar

Stiffness of the spring at the upper bound, specified as a scalar in units of linear stiffness.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Damping Coefficient — Damping coefficient at upper bound
1e3 `N/(m/s)` (default) | scalar

Damping coefficient at the upper bound, specified as a scalar in units of linear damping coefficient.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Transition Region Width — Region to smooth spring and damper forces
`1e-4 m` (default) | scalar

Region to smooth the spring and damper forces, specified as a scalar in units of length.

The block applies the full value of the upper-limit force when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Actuation

Force — Option to provide actuator force
`None` (default) | `Provided by Input` | `Automatically Computed`

Option to provide the actuator force for the joint primitive, specified as one of these values:

Actuation Force Setting	Description
`None`	No actuator force.
`Provided by Input`	Input port **fy** specifies the actuator force for the *y* prismatic primitive.
`Automatically Computed`	The block automatically calculates the amount of force required to satisfy the motion inputs to the mechanism. If you set this parameter to `Automatically Computed`, you do not need to set **Motion** to `Provided by Input` for the same joint primitive. The automatically computed force may to satisfy a motion input elsewhere in the mechanism.

Motion — Option to provide motion
`Automatically Computed` (default) | `Provided by Input`

Option to provide the motion for the joint primitive, specified as one of these values:

Actuation Motion Setting	Description
`Automatically Computed`	The block computes and applies the joint primitive motion based on model dynamics.
`Provided by Input`	Input port **py** specifies the motion for the joint primitive.

Z Revolute Primitive (Rz)

State Targets

Specify Position Target — Whether to specify position target
`off` (default) | `on`

Select this parameter to specify the position target of the *z* revolute primitive.

Priority — Priority level of position target
`High (desired)` (default) | `Low (approximate)`

Priority level of the position target, specified as `High (desired)` or `Low (approximate)`. See "Guiding Assembly" for more information.

Dependencies

To enable this parameter, select **Specify Position Target**.

Value — Angle of position target
`0 deg` (default) | scalar

Angle at which to specify the position target, specified as a scalar with a unit of angle.

Dependencies

To enable this parameter, select **Specify Position Target**.

Specify Velocity Target — Whether to specify angular velocity target
`off` (default) | `on`

Select this parameter to specify the angular velocity target for the *z* revolute primitive.

Priority — Priority level of velocity target
`High (desired)` (default) | `Low (approximate)`

Priority level of the angular velocity target, specified as `High (desired)` or `Low (approximate)`. See "Guiding Assembly" for more information.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Value — Velocity target of *z* revolute primitive
`0 deg/s` (default) | scalar

Angular velocity target of the *z* revolute primitive, specified as a scalar with a unit of angular velocity.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Internal Mechanics

Equilibrium Position — Position where internal torque is zero
`0 deg` (default) | scalar

Position where the spring torque is zero, specified as a scalar with a unit of angle.

Spring Stiffness — Stiffness of force law
0 `N*m/deg` (default) | scalar

Stiffness of the internal spring-damper force law for the *z* revolute primitive, specified as a scalar with a unit of torsional stiffness.

Damping Coefficient — Damping coefficient of force law
0 `N*m/(deg/s)` (default) | scalar

Damping coefficient of the internal spring-damper force law for the *z* revolute primitive, specified as a scalar with a unit of damping coefficient.

Limits

Specify Lower Limit — Whether to specify lower position limit
`off` (default) | `on`

Select this parameter to specify the lower limit of the *z* revolute primitive.

Bound — Lower bound of free region
-90 `deg` (default) | scalar

Lower bound for the free region of the *z* revolute primitive, specified as a scalar with a unit of angle.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Spring Stiffness — Stiffness of spring at lower bound
1e4 `N*m/deg` (default) | scalar

Stiffness of the spring at the lower bound, specified as a scalar with a unit of torsional stiffness.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Damping Coefficient — Damping coefficient at lower bound
10 `N*m/(deg/s)` (default) | scalar

Damping coefficient at the lower bound, specified as a scalar with a unit of damping coefficient.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Transition Region Width — Region to smooth spring and damper torques
0.1 `deg` (default) | scalar

Region to smooth the spring and damper torques, specified as a scalar with a unit of angle.

The block applies the full value of the lower-limit torque when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Specify Upper Limit — Whether to specify upper position limit
`off` (default) | `on`

Select this parameter to specify the upper limit of the *z* revolute primitive.

Bound — Upper bound of free region
`90 deg` (default) | scalar

Upper bound for the free region of the *z* revolute primitive, specified as a scalar with a unit of angle.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Spring Stiffness — Stiffness of spring at upper bound
`1e4 N*m/deg` (default) | scalar

Stiffness of the spring at the upper bound, specified as a scalar with a unit of torsional stiffness.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Damping Coefficient — Damping coefficient at upper bound
`10 N*m/(deg/s)` (default) | scalar

Damping coefficient at the upper bound, specified as a scalar with a unit of damping coefficient.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Transition Region Width — Region to smooth spring and damper torques
`0.1 deg` (default) | scalar

Region to smooth the spring and damper torques, specified as a scalar with a unit of angle.

The block applies the full value of the upper-limit torque when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Actuation

Torque — Option to provide actuator torque
`None` (default) | `Provided by Input` | `Automatically Computed`

Option to provide the actuator torque for the joint primitive, specified as one of these values:

Actuation Torque Setting	Description
`None`	No actuator torque.
`Provided by Input`	Input port **tz** specifies the actuator torque for the *z* revolute primitive.
`Automatically Computed`	The block computes the torque automatically. If you set this parameter to `Automatically Computed`, you do not need to set **Motion** to `Provided by Input` for the for the same joint primitive. The automatically computed torque may satisfy a motion input somewhere else in the mechanism.

Motion — Option to provide motion
`Automatically Computed` (default) | `Provided by Input`

Option to provide the motion for the joint primitive, specified as one of these values:

Actuation Torque Setting	Description
`Automatically computed`	The block computes and applies the joint primitive motion based on model dynamics.
`Provided by Input`	Input port **qz** specifies the motion for the *z* revolute primitive.

Mode Configuration

Mode — Joint mode
`Normal` (default) | `Disengaged` | `Provided by Input`

Joint mode for the simulation, specified as one of these values:

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.
`Provided by Input`	The port **mode** specifies whether the joint behaves normally or is disengaged.

Composite Force/Torque Sensing

Direction — Measurement direction
`Follower on Base` (default) | `Base on Follower`

Measurement direction, specified as one of these values:

- `Follower on Base` — Sense the force and torque that the follower frame exerts on the base frame.
- `Base on Follower` — Sense the force and torque that the base frame exerts on the follower frame.

Note that this parameter only affects the output signals under the **Composite Force/Torque Sensing** section. Reversing the direction changes the sign of the measurements. For more information see “Force and Torque Measurement Direction”.

Resolution Frame — Frame used to resolve measurements
`Base` (default) | `Follower`

Frame used to resolve the measurements, specified as one of these values:

- `Base` — The joint block resolves the measurements in the coordinates of the base frame.
- `Follower` — The joint block resolves the measurements in the coordinates of the follower frame.

Note that this parameter only affects the output signals under the **Composite Force/Torque Sensing** section.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Rectangular Joint | `simscape.multibody.PlanarJoint`

Topics

“Assemble Bodies Using Joints and Constraints”
“Modeling Joint Connections”
“Motion Sensing”
“Rotational Measurements”
“Translational Measurements”

Point on Curve Constraint

Kinematic constraint between frame origin and curved path

Libraries:
Simscape / Multibody / Constraints

Description

The Point on Curve Constraint block confines a point to a curve. The frame attached to the point is free to rotate depending on other constraints in the model. Use the Point on Curve Constraint block to model point-on-curve constraints, such as that between a roller coaster and a track or a cam follower and a cam.

The Point on Curve Constraint block has two ports: **F** and **B**. Port **F** is a frame port whose origin corresponds to the point. Port **B** is a geometry port that you connect to a curve. Avoid curves with sharp changes in slope, as these can cause simulation issues.

You can use this block to sense the constraint forces and torques between the point and curve. Based on the setting of the **Resolution Frame** parameter, the block resolves the forces and torques in either the base or follower frame.

Ports

Geometry

B — Base frame
frame

Geometry port associated with the curve defined by the Spline block.

Frame

F — Follower frame
frame

Follower frame whose origin is confined to the curve connected to port **B**.

Output

f — Constraint force
vector

Constraint force that acts between the point and curve, returned as a vector. See “Measure Joint Constraint Forces” for more information. If you set the **Resolution Frame** parameter to `Base`, the sensed force is with respect to the reference frame of the curve connected to port **B**.

Dependencies

To enable this port, under **Constraint Force/Torque Sensing**, select **Force Vector**.

t — Constraint torque
vector

Constraint torque that acts between the point and curve, returned as a vector. See “Force and Torque Sensing” for more information. If you set the **Resolution Frame** parameter to `Base`, the sensed torque is with respect to the reference frame of the curve connected to the port **B**.

Dependencies

To enable this port, under **Constraint Force/Torque Sensing**, select **Torque Vector**.

Parameters

Direction — Measurement direction
`Follower on Base` (default) | `Base on Follower`

Direction in which to measure the constraint force and torque. Reversing this direction changes the sign of the measurements. See “Force and Torque Measurement Direction” for more information.

Resolution Frame — Frame to resolve measurements
`Base` (default) | `Follower`

Frame used to resolve the measured vector quantities. See “Selecting a Measurement Frame” for more information.

Force Vector — Whether to sense constraint force
`off` (default) | `on`

Select this parameter to expose port **f**, which outputs the constraint force vector between the **B** and **F** frames.

Torque Vector — Whether to sense constraint torque
`off` (default) | `on`

Select this parameter to expose port **t**, which outputs the constraint torque vector between the **B** and **F** frames.

Version History

Introduced in R2015b

Extended Capabilities

C/C++ Code Generation

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

See Also

Spline | Point on Surface Constraint

Topics

“Constrain a Point to a Curve”

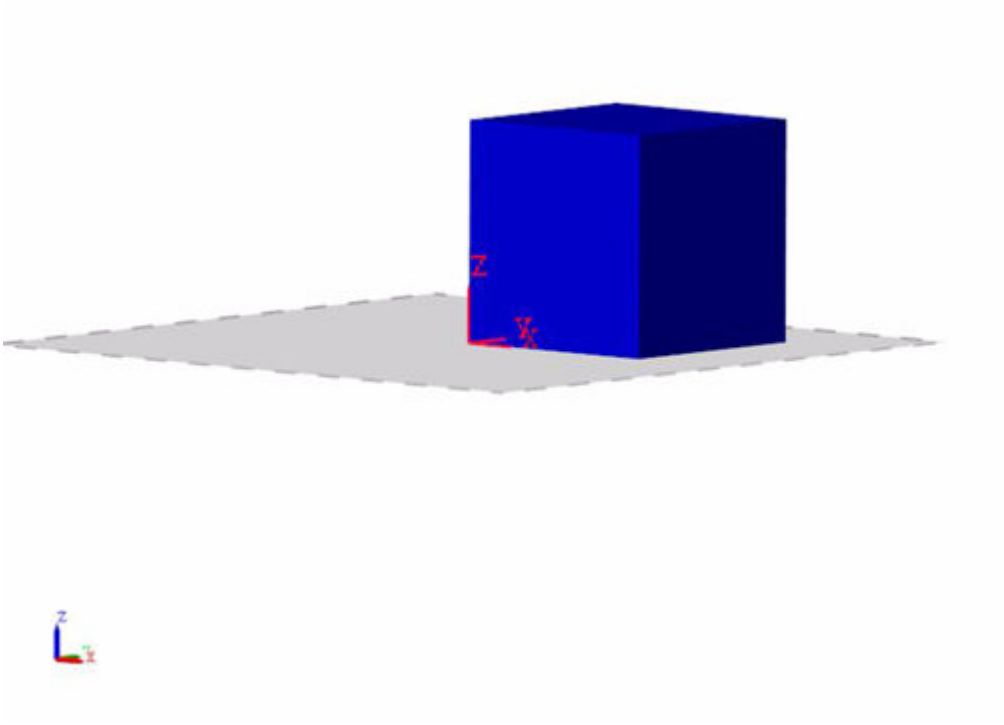
Point on Surface Constraint

Kinematic constraint between frame origin and 2-D surface

Libraries:
Simscape / Multibody / Constraints

Description

The Point on Surface Constraint block confines a point to a 2-D surface. The frame attached to the point has five degrees of freedom (three rotational and two translational). This animation shows an example of using the Point on Surface Constraint block to constrain one vertex of a brick to a plane. The vertex translates and rotates on the plane. Note that the gravity is in the -*Z* direction, and there is no contact between the brick and plane.



The Point on Surface Constraint block has two ports: **F** and **B**. Port **F** is a frame port whose origin corresponds to the point. Port **B** is a geometry port that you connect to the geometry of a 2-D surface. Currently, the Point on Surface Constraint block supports only the Infinite Plane block.

You can use this block to sense the constraint forces and torques between the point and 2-D surface. The forces and torques are resolved in either the base or follower frame, based on the setting of the **Resolution Frame** parameter.

Ports

Geometry

B — Base frame
frame

Geometry port associated with the constraint surface specified by blocks like the Infinite Plane block.

Frame

F — Follower frame
frame

Follower frame whose origin is confined to the 2-D surface connected to the port **B** of the Point on Surface Constraint block. The origin of the follower frame moves on the 2-D surface and has five degrees of freedom (three rotational and two translational).

Output

f — Constraint force
vector

Constraint force that acts between the point and surface, returned as a vector. See “Measure Joint Constraint Forces” for more information. If you set the **Resolution Frame** parameter to `Base`, the sensed force is with respect to the reference frame of the surface that connects to port **B**.

Dependencies

To enable this port, under **Constraint Force/Torque Sensing**, select **Force Vector**.

t — Constraint torque
vector

Constraint torque that acts between the point and surface, returned as a vector. See “Force and Torque Sensing” for more information. If you set the **Resolution Frame** parameter to `Base`, the sensed torque is with respect to the reference frame of the surface that connects to the port **B**.

Dependencies

To enable this port, under **Constraint Force/Torque Sensing**, select **Torque Vector**.

Parameters

Direction — Measurement direction
`Follower on Base` (default) | `Base on Follower`

Direction in which to measure the constraint force and torque. Reversing this direction changes the sign of the measurements. See “Force and Torque Measurement Direction” for more information.

Resolution Frame — Frame to resolve measurements
`Base` (default) | `Follower`

Frame used to resolve the measured vector quantities. See “Selecting a Measurement Frame” for more information.

Force Vector — Whether to sense constraint force
`off` (default) | `on`

Select this parameter to expose the port **f**, which outputs the constraint force vector between **B** and **F** frames.

Torque Vector — Whether to sense constraint torque
`off` (default) | `on`

Select this parameter to expose the port **t**, which outputs the constraint torque vector between **B** and **F** frames.

Version History

Introduced in R2021a

Extended Capabilities

C/C++ Code Generation

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

See Also

Point on Curve Constraint

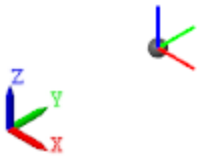
Point

Point geometry for contact modeling

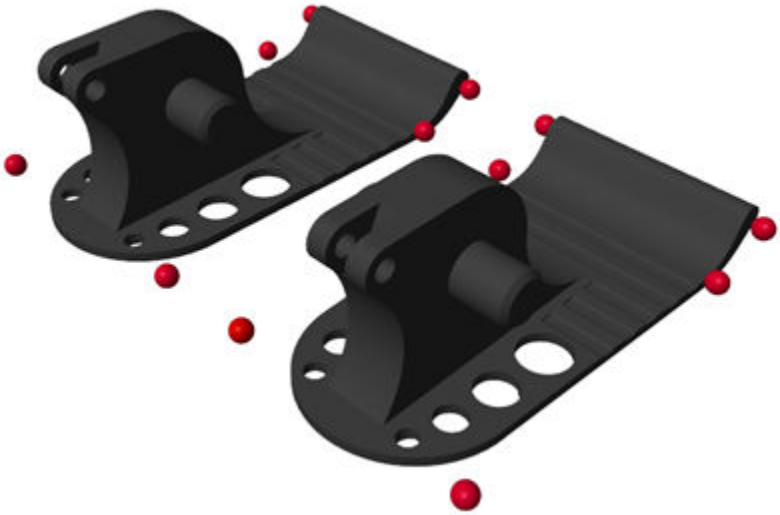
Libraries:
Simscape / Multibody / Curves and Surfaces

Description

The Point block exports a 0-D point geometry for modeling contact problems. You can model contacts between the point and many types of geometries, such as all the solids and convex hulls in the Body Element library and the infinite plane in the Curves and Surfaces library. However, you cannot model a contact between a pair of points because there is no penetration, which is necessary for the penalty method, between two points.



Using the point geometry can simplify and speed up contact modeling that involves complex geometries by eliminating unnecessary geometric details. For example, in the “Train Humanoid Walker” example, you can simplify the feet-ground contact by using several points to represent the robot feet.



Ports

Frame

R — Local reference frame
frame

Local reference frame to use to define the location and orientation of the point.

Geometry

G — Geometry frame
frame

Geometry frame that represents the point defined by this block. Connect this port to the Spatial Contact Force block to model contacts on the point.

Parameters

Graphic

Type — Graphic to use in visualization of point
`Marker` (default) | `None`

Graphic used to visualize the point. Specify this parameter to `None` to eliminate the point from the model visualization.

Shape — Shape of point marker
`Sphere` (default) | `Cube` | `Frame`

Shape of the marker used to visualize the point. The marker shape has no effect on the contact results of the point.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
10 `pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify diffuse color and opacity. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Diffuse Color — Graphic color
[0.5 0.5 0.5] (default) | three-element vector | four-element vector

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Version History

Introduced in R2020b

Extended Capabilities

C/C++ Code Generation

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

See Also

Spatial Contact Force

Topics

“Modeling Contact Force Between Two Solids”

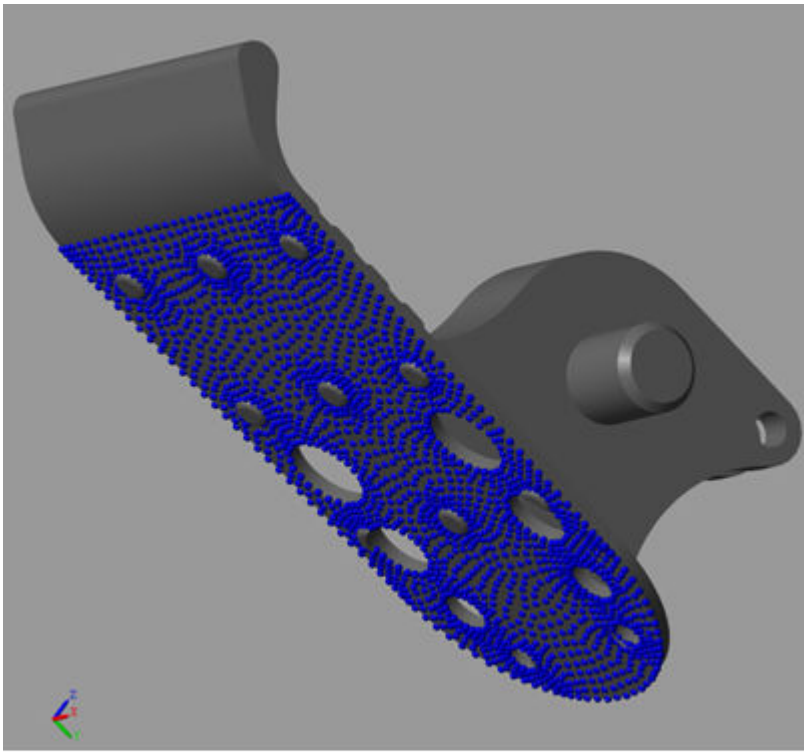
Point Cloud

Point cloud geometry for contact modeling

Libraries:
Simscape / Multibody / Curves and Surfaces

Description

The Point Cloud block exports a set of points in space for modeling contact problems. Each point has a rigid offset with respect to the reference frame of the Point Cloud block. You can use a Point Cloud block to approximate a geometry, such as a solid or convex hull. When modeling certain sustained and distributed contact problems, the Spatial Contact Force block might perform better with a Point Cloud block than with a geometry block, such as Brick Solid or File Solid. This image demonstrates how to use a point cloud to approximate the bottom of a robot foot.



The Point Cloud block has one reference frame port (**R**) and one geometry port (**G**). To use a Point Cloud block to model contact problems, connect port **G** to a Spatial Contact Force block. The Spatial Contact Force block treats a point cloud as an aggregation of *N*-point geometries and applies contact forces to each point independently. Each contact force is based on the penetration and velocity of the individual point of the cloud. Note that the Spatial Contact Force block does not support sensing when connected to a Point Cloud block.

To specify the locations of points, you can enter an *N*-by-3 matrix for the **Coordinates Matrix** parameter of the Point Cloud block. Each row of the matrix specifies the Cartesian coordinates of a point with respect to the reference frame of the Point Cloud block. An error occurs if the matrix has any repeated rows.

Tip You can use the `unique` function to remove repeated rows from an input matrix.

Ports

Frame

R — Reference frame
frame

Point cloud reference frame. To specify the location and orientation of the point cloud, connect this frame to another block, such as Brick Solid or File Solid.

Geometry

G — Geometry frame
frame

Geometry frame that represents the points defined by this block. To model contacts on the points, connect this port to a Spatial Contact Force block.

Parameters

Coordinates Matrix — Coordinates of points
`[0 0 0 ] m` (default) | *N*-by-3 matrix

Coordinates of the points, specified as a *N*-by-3 matrix. Each row of the matrix specifies the Cartesian coordinates of a point with respect to the reference frame of the Point Cloud block. An error occurs if the matrix has any repeated rows. You can use the `unique` function to remove repeated rows from an input matrix.

Data Types: `double`

Graphic

Type — Graphic to use in visualization of point cloud
`Marker Cloud` (default) | `None`

Graphic used to visualize the point cloud. To eliminate the point cloud from the model visualization, set this parameter to `None`.

Sphere Radius — Marker radius
`1e-2 m` (default) | positive scalar

Radius of the marker for each point of the point cloud.

Dependencies

To enable this parameter, set **Type** to `Marker Cloud`.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify diffuse color and opacity. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `Marker Cloud`.

Diffuse Color — Graphic color
[0.5 0.5 0.5] (default) | three-element vector | four-element vector

Color of the graphic under direct white light, specified as an [R G B] or [R G B A] vector on a 0–1 scale. The optional fourth element (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set **Type** to `Marker Cloud`.

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `Marker Cloud`
2. **Visual Properties** to `Simple`

Specular Color — Highlight color
[0.5 0.5 0.5 1.0] (default) | three-element vector | four-element vector

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.

Dependencies

To enable this parameter, set:

1. **Type** to `Marker Cloud`
2. **Visual Properties** to `Advanced`

Ambient Color — Shadow color
[0.15 0.15 0.15 1.0] (default) | three-element vector | four-element vector

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 (A) specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `Marker Cloud`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | three-element vector | four-element vector

Graphic 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 (A) specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `Marker Cloud`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
75 (default) | scalar in the range of 1 to 128

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

Dependencies

To enable this parameter, set:

1. **Type** to `Marker Cloud`
2. **Visual Properties** to `Advanced`

Version History

Introduced in R2021a

Extended Capabilities

C/C++ Code Generation

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

See Also

Spatial Contact Force | `unique`

Topics

“Modeling Contact Force Between Two Solids”

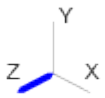
Prismatic Joint

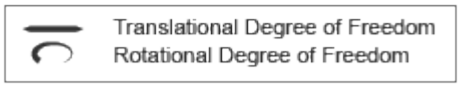
Joint that allows relative motion along single axis

Libraries:
Simscape / Multibody / Joints

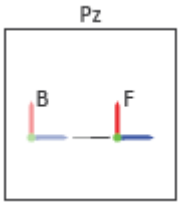
Description

The Prismatic Joint block models a connection that limits the relative motion between two frames to one translational degree of freedom. This type of motion is common in hydraulic or pneumatic cylinders.





The Prismatic Joint block has only one prismatic primitive, which is along the *z*-axis of the base frame. During simulation, the *z*-axes of the base and follower frames align, and the follower frame moves with respect to the base frame along the *z*-axis, as shown in the image.



Meanwhile, the *x* and *y* axes of the follower frame remain parallel to the *x* and *y* axes of the base frame, respectively.

To target the initial state of a joint primitive, use the parameters under **State Targets**. The position and velocity targets are resolved in the base frame. You can also set the priority levels for the targets. If not all the state targets can be satisfied, the priority level determines which targets to satisfy first and how closely to satisfy them. See “Guiding Assembly” for more information.

To model damping and spring behavior in a joint primitive, use the parameters under **Internal Mechanics**. Specify the **Damping Coefficient** parameter to model energy dissipation and the **Spring Stiffness** parameter to model energy storage. Springs resist attempts to displace the joint

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

To specify the limits of a joint primitive, use the parameters under **Limits**. The lower and upper bounds define the width of the free region of a joint primitive. The block applies a force to accelerate the joint position back to the free region when the position exceeds the bounds. The block uses a smoothed spring-damper method to compute the force. See the “Description” on page 1-415 section of the Spatial Contact Force block for more information about the smoothed spring-damper method.

The **Force** and **Motion** parameters in the **Actuation** section govern the motion of a primitive during a simulation. Additionally, the block has ports that output sensing data, such as position, velocity, acceleration, force, and torque, that enable you to perform analytical tasks on a model. See “Sensing” and “Force and Torque Sensing” for more information.

Ports

Frame

B — Base frame
frame

Base frame of the joint block.

F — Follower frame
frame

Follower frame of the joint block.

Input

f — Actuation force
physical signal

Physical signal input port that accepts the actuation force for the joint primitive. The signal provides the value of the force that applies on both the base and follower frames of the joint primitive.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz)** > **Actuation**, set **Force** to `Provided by Input`.

p — Motion profile
physical signal

Physical signal input port that accepts the motion profile for the joint primitive. The signal provides the displacement of the follower frame with respect to the base frame along the joint primitive axis. Note that the signal must also contain the first and second derivatives of the displacement.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz)** > **Actuation**, set **Motion** to `Provided by Input`.

mode — Joint mode control
scalar

Input port that controls the joint mode. The signal is a unitless scalar. The joint mode is normal when the input signal is 0 and disengaged when the input signal is -1. The joint mode can be changed many times during simulation.

Dependencies

To enable this port, under **Mode Configuration**, set **Mode** to `Provided by Input`.

Output

p — Position of primitive
physical signal

Physical signal port that outputs the position of the primitive. The value is the displacement of the follower frame with respect to the base frame.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Position**.

v — Velocity of primitive
physical signal

Physical signal port that outputs the velocity of the primitive.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Velocity**.

a — Acceleration of primitive
physical signal

Physical signal port that outputs the acceleration of the primitive.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Acceleration**.

f — Actuator force acting on joint primitive
physical signal

Physical signal port that outputs the actuator force acting on the joint primitive.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Actuator Force**.

fll — Lower-limit force
physical signal

Physical signal port that outputs the lower-limit force. The block applies the force when the joint primitive position is less than the lower bound of the free region. The force applies to both the base and follower frames of the joint primitive to accelerate the position back to the free region.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Lower-Limit Force**.

ful — Upper-limit force
physical signal

Physical signal port that outputs the upper-limit force. The block applies the force when the joint primitive position exceeds the upper bound of the free region. The force applies to both the base and follower frames of the joint primitive to accelerate the position back to the free region.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Upper-Limit Force**.

fc — Constraint force
physical signal

Physical signal port that outputs the constraint force that acts in the joint. The force maintains the translational constraints of the joint. For more information, see “Measure Joint Constraint Forces”.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Constraint Force**.

tc — Constraint torque
physical signal

Physical signal port that outputs the constraint torque that acts in the joint. The torque maintains the rotational constraints of the joint. For more information, see “Force and Torque Sensing”.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Constraint Torque**.

ft — Total force
physical signal

Physical signal port that outputs the total force that acts in the joint. The total force is the sum of the forces transmitted from one frame to the other through the joint. The force includes actuation, internal, limit, and constraint forces. See “Force and Torque Sensing” for more information.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Total Force**.

tt — Total torque
physical signal

Physical signal port that outputs the total torque that acts in the joint. The total torque is the sum of the torques transmitted from one frame to the other through the joint. The torque includes actuation, internal, limit, and constraint torques. For more information, see “Force and Torque Sensing”.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Total Torque**.

Parameters

Z Prismatic Primitive (Pz)

State Targets > Specify Position Target — Whether to specify position target
`off` (default) | `on`

Select this parameter to enable parameters that specify the position target of the joint primitive.

State Targets > Specify Position Target > Priority — Priority level of position target
`High (desired)` (default) | `Low (approximate)`

Priority level of the position target, specified as `High (desired)` or `Low (approximate)`.

Dependencies

To enable this parameter, under **Z Prismatic Primitive (Pz) > State Targets**, select **Specify Position Target**.

State Targets > Specify Position Target > Value — Position target
`0 m` (default) | scalar in units of length

Position target of the joint primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, under **Z Prismatic Primitive (Pz) > State Targets**, select **Specify Position Target**.

State Targets > Specify Velocity Target — Whether to specify velocity target
`off` (default) | `on`

Select this parameter to enable parameters that specify the velocity target of the joint primitive.

State Targets > Specify Velocity Target > Priority — Priority level of velocity target
`High (desired)` (default) | `Low (approximate)`

Priority level of the velocity target, specified as `High (desired)` or `Low (approximate)`.

Dependencies

To enable this parameter, under **Z Prismatic Primitive (Pz) > State Targets**, select **Specify Velocity Target**.

State Targets > Specify Velocity Target > Value — Velocity target of joint primitive
`0 m/s` (default) | scalar in units of linear velocity

Velocity target of the joint primitive, specified as a scalar in a unit of linear velocity.

Dependencies

To enable this parameter, under **Z Prismatic Primitive (Pz) > State Targets**, select **Specify Velocity Target**.

Internal Mechanics > Equilibrium Position — Position where internal force is zero
`0 m` (default) | scalar in units of length

Position where the spring force is zero, specified as a scalar in units of length. The value specifies the position of the primitive.

Internal Mechanics > Spring Stiffness — Stiffness of force law
`0 N/m` (default) | scalar in units of linear stiffness

Stiffness of the internal spring-damper force law for the joint primitive, specified as a scalar in units of linear stiffness.

Internal Mechanics > Damping Coefficient — Damping coefficient of force law
`0 N(m/s)` (default) | scalar in units of linear damping coefficient

Damping coefficient of the internal spring-damper force law for the joint primitive, specified as a scalar in units of linear damping coefficient.

Limits > Specify Lower Limit — Whether to specify lower position limit
`off` (default) | `on`

Select this parameter to enable parameters that specify the lower limit of the joint primitive.

Limits > Specify Lower Limit > Bound — Lower bound of free region
`-1 m` (default) | scalar in units of length

Lower bound of the free region of the joint primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, under **Z Prismatic Primitive (Pz) > Limits**, select **Specify Lower Limit**.

Limits > Specify Lower Limit > Spring Stiffness — Stiffness of spring at lower bound
`1e6 N/m` (default) | scalar in units of linear stiffness

Stiffness of the spring at the lower bound, specified as a scalar in units of linear stiffness.

Dependencies

To enable this parameter, under **Z Prismatic Primitive (Pz) > Limits**, select **Specify Lower Limit**.

Limits > Specify Lower Limit > Damping Coefficient — Damping coefficient at lower bound
`1e3 N/(m/s)` (default) | scalar in units of linear damping coefficient

Damping coefficient at the lower bound, specified as a scalar in units of linear damping coefficient.

Dependencies

To enable this parameter, under **Z Prismatic Primitive (Pz) > Limits**, select **Specify Lower Limit**.

Limits > Specify Lower Limit > Transition Region Width — Region to smooth spring and damper forces
`1e-4 m` (default) | scalar in units of length

Region to smooth the spring and damper forces, specified as a scalar in units of length.

The block applies the full value of this force when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, under **Z Prismatic Primitive (Pz) > Limits**, select **Specify Lower Limit**.

Limits > Specify Upper Limit — Whether to specify upper position limit
`off` (default) | `on`

Select this parameter to enable parameters for specifying the upper limit of the joint primitive.

Limits > Specify Upper Limit > Bound — Upper bound of free region
`1 m` (default) | scalar in units of length

Upper bound for the free region of the joint primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, under **Z Prismatic Primitive (Pz) > Limits**, select **Specify Upper Limit**.

Limits > Specify Upper Limit > Spring Stiffness — Stiffness of spring at upper bound
`1e6 N/m` (default) | scalar in units of linear stiffness

Stiffness of the spring at the upper bound, specified as a scalar in units of stiffness.

Dependencies

To enable this parameter, under **Z Prismatic Primitive (Pz) > Limits**, select **Specify Upper Limit**.

Limits > Specify Upper Limit > Damping Coefficient — Damping coefficient at upper bound
`1e3 N/(m/s)` (default) | scalar in units of linear damping coefficient

Damping coefficient at the upper bound, specified as a scalar in units of linear damping coefficient.

Dependencies

To enable this parameter, under **Z Prismatic Primitive (Pz) > Limits**, select **Specify Upper Limit**.

Limits > Specify Upper Limit > Transition Region Width — Region to smooth spring and damper forces
`1e-4 m` (default) | scalar in units of length

Region to smooth the spring and damper forces, specified as a scalar in units of length.

The block applies the full value of this force when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, under **Z Prismatic Primitive (Pz) > Limits**, select **Specify Upper Limit**.

Actuation > Force — Option to provide actuation force
`None` (default) | `Provided by Input` | `Automatically Computed`

Option to provide the actuation force for the joint primitive, specified as the values in the table.

Actuation Force Setting	Description
`None`	No actuation force.
`Provided by Input`	Input port **f** specifies the actuation force for the joint primitive..
`Automatically computed`	The block automatically calculates the amount of force required to satisfy the motion inputs to the mechanism. Note that if you set this parameter to `Automatically computed`, it doesn't mean that you must use `Provided by Input` for the **Motion** parameter for the same joint primitive. The automatically computed force could be to satisfy a motion input somewhere else in the mechanism.

Actuation > Motion — Option to provide actuation motion
`Automatically Computed` (default) | `Provided by Input`

Option to provide the actuation motion for the joint primitive, specified as the values in the table.

Actuation Motion Setting	Description
`Automatically computed`	The block computes and applies the joint primitive motion based on model dynamics.
`Provided by Input`	Input port **p** specifies the actuation motion for the joint primitive.

Sensing > Position — Whether to sense joint primitive position
off (default) | on

Select this parameter to enable the output port **p**.

Sensing > Velocity — Whether to sense joint primitive velocity
off (default) | on

Select this parameter to enable port **v**.

Sensing > Acceleration — Whether to sense joint primitive acceleration
off (default) | on

Select this parameter to enable port **a**.

Sensing > Actuator Force — Whether to sense joint actuator force
off (default) | on

Select this parameter to enable port **f**.

Sensing > Lower-Limit force — Whether sense to lower-limit force
off (default) | on

Select this parameter to enable port **fll**.

Sensing > Upper-Limit force — Whether sense to upper-limit force
off (default) | on

Select this parameter to enable port **ful**.

Mode Configuration

Mode — Joint mode
`Normal` (default) | `Disengaged` | `Provided by Input`

Joint mode for the simulation, specified as the values in the tables.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.
`Provided by Input`	Port **mode** specifies whether the joint behaves normally or is disengaged.

Composite Force/Torque Sensing

Direction — Measurement direction
`Follower on Base` (default) | `Base on Follower`

Measurement direction, specified as `Follower on Base` or `Base on Follower`:

- `Follower on Base` — Sense the force and torque that the follower frame exerts on the base frame.
- `Base on Follower` — Sense the force and torque that the base frame exerts on the follower frame.

Reversing the direction changes the sign of the measurements. For more information see “Force and Torque Measurement Direction”.

Resolution Frame — Frame used to resolve measurements
`Base` (default) | `Follower`

Frame used to resolve the measurements, specified as:

- `Base` — The joint block resolves the measurements in the coordinates of the base frame.
- `Follower` — The joint block resolves the measurements in the coordinates of the follower frame.

Constraint Force — Whether to sense constraint force in joint
off (default) | on

Select this parameter to enable the port **fc**.

Constraint Torque — Whether to sense constraint torque in joint
off (default) | on

Select this parameter to enable the port **tc**.

Total Force — Whether to sense total force in joint
off (default) | on

Select this parameter to enable the port **ft**.

Total Torque — Whether to sense total torque in joint
off (default) | on

Select this parameter to enable the port **tt**.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Topics

“Force and Torque Sensing”
“Guiding Assembly”
“Measure Joint Constraint Forces”

Pulley

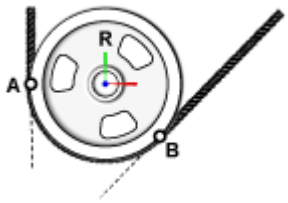
Wheel wrapped in a cord for the transmission of torque and motion

Libraries:
Simscape / Multibody / Belts and Cables

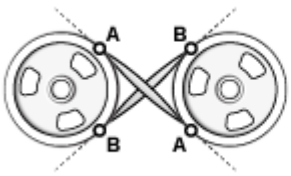
Description

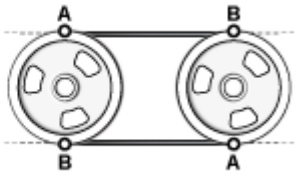
The Pulley block represents a grooved or toothed wheel wrapped in a cord, an arrangement used frequently in the transmission of torque and motion in part for the mechanical advantage that it can provide. The pulley (or sprocket if toothed) is *ideal*: massless and frictionless, with zero slip permitted 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.

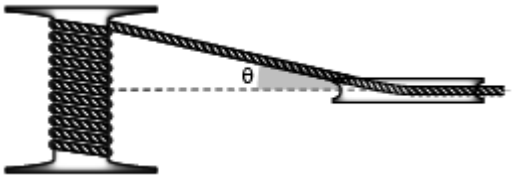




The degrees of freedom of the pulley depend entirely on the joints and constraints (if any) to which it connects. Attaching a pulley to a case by means of a revolute joint imparts to the pulley one rotational

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.

Initial Wrap Angle — Minimum wrap angle of a pulley at the start of simulation
`0.0 deg` (default) | positive scalar in degrees

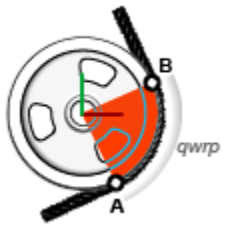
Lower Bound — Specify the lower bound on the initial wrap angle of the pulley. The wrap angle at the beginning of simulation is greater than or equal to this specified value.

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.

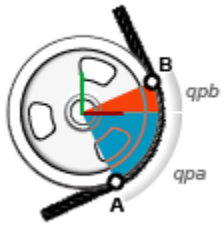


- **Pulley Angle A** — Angle, measured in the *xy* plane of the reference frame, from the local *x*-axis to the line between the frame origin and point of contact **A**.

  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 -π/2 to +π/2. This range is open. The measured value can vary between -π/2 and +π/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 -π/2 to +π/2. This range is open. The measured value can vary between -π/2 and +π/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.

Version History

Introduced in R2018a

Extended Capabilities

C/C++ Code Generation

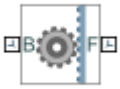
Generate C and C++ code using Simulink® Coder™.

See Also

Belt-Cable End | Belt-Cable Spool | Belt-Cable Properties

Rack and Pinion Constraint

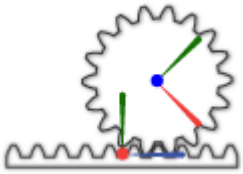
Kinematic constraint between a translating rack body and a rotating pinion body



Libraries:
Simscape / Multibody / Gears and Couplings / Gears

Description

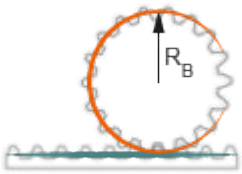
The Rack and Pinion Constraint block represents a kinematic constraint between a translating rack body and a rotating pinion body. The base frame port identifies the connection frame on the pinion body and the follower frame port identifies the connection frame on the rack body. The pinion rotation axis 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

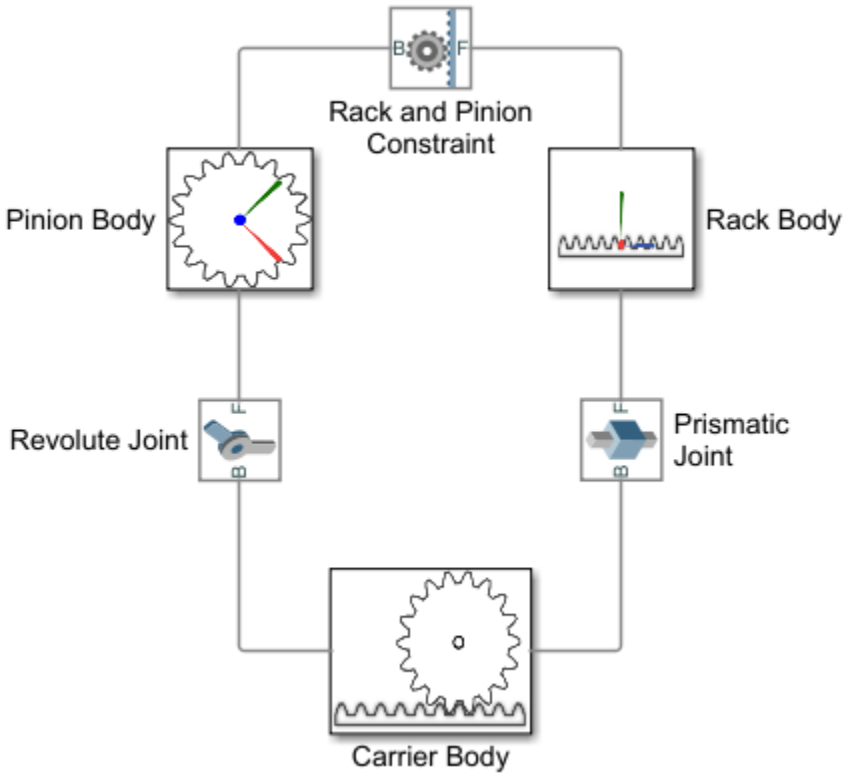
The rack-and-pinion constraint is parameterized in terms of the dimensions of the pinion pitch circle. The pitch circle is an imaginary circle concentric with the pinion body and tangent to the tooth contact point. The pitch radius, labeled R_B in the figure, is the radius that the pinion would have 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.



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.

Version History

Introduced in R2013a

Extended Capabilities

C/C++ Code Generation

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

See Also

Worm and Gear Constraint | Bevel Gear Constraint | Common Gear Constraint

Topics

“Rack and Pinion”
“Measure Forces and Torques Acting at Joints”

Rectangular Joint

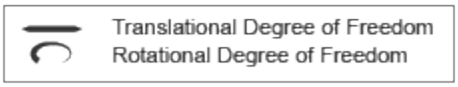
Joint with two prismatic primitives

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.



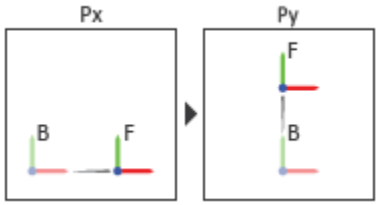
Joint Degrees of Freedom

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.

Mode Configuration

Specify the mode of the joint. The joint mode can be normal or disengaged throughout the simulation, or you can provide an input signal to change the mode during the simulation.

Mode

Select one of the following options to specify the mode of the joint. The default setting is `Normal`.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.

Method	Description
`Provided by Input`	This option exposes the **mode** port that you can connect to an input signal to change the joint mode during the simulation. The joint mode is normal when the input signal is `0` and disengaged when the input signal is `-1`. The joint mode can be changed many times during the simulation.

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:

- fx, fy — Actuation forces acting on the X and Y prismatic joint primitives
- px, py — 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
- fllx, flly — Forces due to contact with the lower limits of the X and Y prismatic joint primitives
- fulx, fuly — Forces due to contact with the upper limits of 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

Mode Port

Mode configuration provides the following port:

- mode — Value of the mode of the joint. If the input is equal to `0`, the joint behaves normally. If the input is equal to `-1`, the joint behaves as disengaged.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Prismatic Joint | Planar Joint

Topics

“Actuating and Sensing with Physical Signals”
“Motion Sensing”
“Translational Measurements”

Reduced Order Flexible Solid

Create flexible body using reduced-order model data

Libraries:
Simscape / Multibody / Body Elements / Flexible Bodies

Description

The Reduced Order Flexible Solid block creates a flexible body by using reduced-order model (ROM) data.

A reduced-order model is a computationally efficient model that characterizes the properties of a flexible body that can have linear elastic deformations. The ROM data must include:

- Coordinates and unit quaternions that specify the positions and orientations of all interface frames relative to a common reference frame on page 1-378. See “Interface Frames” on page 1-381.
- A symmetric stiffness matrix that describes the elastic properties of the flexible body. See “Stiffness Matrix” on page 1-0 .
- A symmetric mass matrix that describes the inertial properties of the flexible body. See “Mass Matrix” on page 1-0 .

To generate ROM data, you can use the **Flexible Body Model Builder** app or finite-element analysis tools, such as the Partial Differential Equation Toolbox™. By using the toolbox, you can start with the CAD geometry of the body, generate a finite-element mesh, apply the Craig-Bampton method, and generate the corresponding ROM data. For more information, see “Model an Excavator Dipper Arm as a Flexible Body”.

Common Reference Frame

The data provided to the Reduced Order Flexible Solid block must refer to a consistent common reference frame. This reference frame defines the *x*, *y*, and *z* directions that specify the relative position of all points in the body. The frame also defines the directions of the elastic degrees of freedom associated with each interface frame.

Requirements for the ROM Data

The ROM data for the Reduced Order Flexible Solid block must satisfy these requirements:

- The ROM data must have at least one boundary node. Each boundary node determines the location of an interface frame.
- Each boundary node must contribute six degrees of freedom to the reduced-order model. The degrees of freedom for node i must be retained in the order
  $U_i = [T_{xi}, T_{yi}, T_{zi}, R_{xi}, R_{yi}, R_{zi}]$,
  where:
  - T_{xi}, T_{yi}, and T_{zi} are translational degrees of freedom along the *x*, *y*, and *z* directions of the common reference frame.

- R_{xi}, R_{yi}, and R_{zi} are rotational degrees of freedom about the x, y, and z axes of the common reference frame.

The reduced-order model can also include additional degrees of freedom, D_1, D_2, ⋯, D_m, that correspond to retained normal vibration modes. For a reduced-order model that has n boundary nodes and m modal degrees of freedom, the degrees of the freedom of the model is:

$$U_{reduced} = [U_1, U_2, \cdots, U_n, D_1, D_2, \cdots, D_m].$$

The number of the degrees of freedom $N_{reduced}$ equals $6n + m$, which includes six rigid-body degrees of freedom. The number of elastic degrees of freedom equals $N_{reduced}$ - 6. The number of the degrees of freedom determines the size of the stiffness and mass matrices. For the matrices, the order of the rows and columns must correspond to the order of the $U_{reduced}$.

Reduce the Degrees of Freedom for a Flexible Body

You can use the block to further reduce the degrees of freedom for a flexible body. The fewer the degrees of freedom for flexible bodies, the faster the simulation.

For example, a ROM data set generated by the Craig-Bampton method with two boundary nodes and 10 fixed-interface normal modes has a total of 22 degrees of freedom. If you use the ROM data, the block generates an internal representation of the flexible body that has six rigid-body and 16 elastic degrees of freedom. To further reduce the number of the elastic degrees of freedom, you can set **Reduction** to `Modally Reduced`. If you specify **Number of Retained Modes** to 8, the flexible body retains the eight lowest-frequency modal degrees of freedom in addition to the six rigid-body degrees of freedom.

Damping

To specify the damping characteristics of the flexible bodies, the block has three damping methods: proportional damping, uniform modal damping, and damping matrix methods. For more informations, see “Damping” on page 1-381.

Simulation Performance

Flexible bodies can increase the numerical stiffness of a multibody model. To avoid simulation issues, use a stiff solver such as `ode15s` or `ode23t`.

Damping can significantly influence simulation performance. For example, when modeling a body with little or no damping, undesirable high-frequency modes in the response can slow the simulation. In that case, adding a small amount of damping can improve the speed of the simulation without significantly affecting the accuracy of the model.

Ports

Frame

F1, F2, ..., F*n* — Frames
frame

Frames that connect the flexible body to the model. Each interface frame corresponds to a boundary node in the reduced-order model. The frames allow you to connect the flexible body to other Simscape Multibody elements, such as joints, constraints, forces, and sensors. It is not required that all frame ports be connected.

Parameters

Unit System

Unit System — Standard or custom units for length, mass, and time
`SI (m, kg, s)` (default) | `CGS (cm, g, s)` | `English (ft, slug, s)` | `Custom`

System of units in which to express length, mass, time, and other derived units of measure used in all block parameters. Angle measure is always in radians.

Unit System	Base Units			Selected Derived Units		
	Length	Mass	Time	Force	Stress and Pressure	Density
`SI`	m	kg	s	N = kg·m/s^2	Pa = kg/($m·s^2$)	kg/m^3
`CGS`	cm	g	s	dyn = g·cm/s^2	Ba = g/($cm·s^2$)	g/cm^3
`English`	ft	slug	s	lbf = slug·ft/s^2	slug/($ft·s^2$)	$slug/ft^3$

To specify units of length, mass, and time individually, select `Custom`. For example, suppose that you specify a custom unit system with `mm` for length, `t` for mass, and `s` for time. The derived units include N = t·mm/s^2 for force, MPa = t/(mm·s^2) for stress and pressure, and t/mm^3 for density.

Note The specified units must be consistent with the imported data. For example, you must set the **Unit System** parameter to `SI (m, kg, s)` when using the ROM data generated by the **Flexible Body Model Builder** app because the generated ROM data use SI units.

Length Unit — Unit of measure for length
`m` (default) | `cm` | `mm` | `km` | `in` | `ft` | `yd` | `mi`

Unit of measure in which to express length.

Dependencies

To enable this parameter, set **Unit System** to `Custom`.

Mass Unit — Unit of measure for mass
`kg` (default) | `g` | `t` | `lbm` | `slug`

Unit of measure in which to express mass.

Dependencies

To enable this parameter, set **Unit System** to `Custom`.

Time Unit — Unit of measure for time
`s` (default) | `ms` | `min` | `hr`

Unit of measure in which to express time.

Dependencies

To enable this parameter, set **Unit System** to `Custom`.

Interface Frames

Number of Frames — Number of frame ports
`1` (default) | positive integer

Number of interface frame ports, specified as a positive integer. This number must match the number of boundary nodes defined in the ROM data. The parameter does not support variables or expressions.

Origins — Cartesian coordinates of interface frame origins
`[0 0 0]` (default) | *n*-by-*3* matrix

Cartesian coordinates of all interface frame origins, specified as a *n*-by-*3* matrix. Each row of the matrix represents one frame origin. All coordinates must be relative to the common reference frame. Each boundary node in the reduced-order model must have a corresponding interface frame on the block.

Orientations — Orientations of interface frames
`[1 0 0 0]` (default) | *n*-by-*4* matrix of unit quaternions

Orientations of interface frames, specified as an *n*-by-*4* matrix of unit quaternions, where *n* is the number of interface frames.

Reduced Order Matrices

Stiffness Matrix — Stiffness matrix from reduced-order model
`[]` (default) | *r*-by-*r* matrix

Stiffness matrix from the ROM data, specified as a *r*-by-*r* matrix. The stiffness matrix is a symmetric matrix that describes the elastic properties of the flexible body. For a ROM data set with *n* boundary nodes and *m* dynamic deformation modes, the stiffness matrix has $r = 6n + m$ rows and columns.

Mass Matrix — Mass matrix from reduced-order model
`[]` (default) | *r*-by-*r* matrix

Mass matrix from the ROM data, specified as a *r*-by-*r* matrix. The mass matrix is a symmetric matrix that describes the inertial properties of the flexible body. For a ROM data set with *n* boundary nodes and *m* dynamic deformation modes, the mass matrix has $r = 6n + m$ rows and columns.

Damping

Type — Type of damping method
`Proportional` (default) | `Uniform Modal` | `Damping Matrix` | `None`

Damping method to apply to the flexible body:

- Select `Proportional` to apply the proportional (or Rayleigh) damping method. This method defines the damping matrix $[C]$ as a linear combination of the mass matrix $[M]$ and stiffness matrix $[K]$:

  $$[C] = \alpha[M] + \beta[K],$$

where α and β are scalar coefficients.

- Select `Uniform Modal` to apply the uniform modal damping method. This method applies a single damping ratio to all the vibration modes of the solid. The larger the value, the faster vibrations decay.
- Select `Damping Matrix` to use a reduced-order damping matrix that you computed with the stiffness and mass matrices. For example, you can use this option to specify a modal damping model for the flexible body.
- Select `None` to model undamped body.

Mass Coefficient — Coefficient of mass matrix
`0 1/s` (default) | nonnegative scalar

Coefficient, α, of the mass matrix. This parameter defines damping proportional to the mass matrix $[M]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Stiffness Coefficient — Coefficient of stiffness matrix
`0.001 s` (default) | nonnegative scalar

Coefficient, β, of the stiffness matrix. This parameter defines damping proportional to the stiffness matrix $[K]$.

Dependencies

To enable this parameter, set **Type** to `Proportional`.

Damping Ratio — Damping ratio for uniform modal damping method
0.01 (default) | unitless nonnegative scalar

Damping ratio, ζ, applied to all vibration modes of the flexible body. The larger the value, the faster the vibrations decay.

- Use $\zeta = 0$ to model undamped solids.
- Use $\zeta < 1$ to model underdamped solids.
- Use $\zeta = 1$ to model critically damped solids.
- Use $\zeta > 1$ to model overdamped solids.

Dependencies

To enable this parameter, set **Type** to `Uniform Modal`.

Data Types: `double`

Damping Matrix — Reduced-order damping matrix
`[ ]` (default) | *r*-by-*r* matrix

Reduced-order damping matrix, specified as a *r*-by-*r* matrix. The damping matrix is a symmetric matrix that describes the damping properties of the flexible body. In a flexible body with n boundary nodes and m dynamic deformation modes, the damping matrix has $r = 6n + m$ rows and columns.

Dependencies

To enable this parameter, set **Type** to `Damping Matrix`.

Fidelity

Type — Method to model flexible bodies
`None` (default) | `Modally Reduced`

Method to use to model flexible bodies, specified as `None` or `Modally Reduced`.

- If you select `None`, the block models a flexible body with all the degrees of freedom.
- If you select `Modally Reduced`, the block performs an eigenvalue decomposition to the mass and stiffness matrices of the flexible body. You can then use the **Number of Retained Modes** parameter to specify the desired normal modes of the flexible body for the simulation.

Number of Retained Modes — Retained normal modes
1 (default) | nonnegative integer

Retained normal modes, specified as an integer in range [0, *n*], where *n* equals the number of elastic degrees of freedom of the flexible body.

The block uses this parameter to specify how many low-frequency modes to retain for the simulation. If you set this parameter to `0`, the flexible body is treated as a rigid body. The larger the number, the slower the simulation.

Dependencies

To enable this parameter, set **Type** to `Modally Reduced`.

Graphic

Type — Visual representation of flexible body
`None` (default) | `Partitioned Geometry` | `Deformed Geometry`

Visual representation of the flexible body, specified as `None`, `Partitioned Geometry`, or `Deformed Geometry`.

- To hide the body, use `None`.
- To show the deformed body by using an approximate representation that consists of several rigid pieces, use `Partitioned Geometry`. In the visualization, the block divides the body into several pieces according to the interface frames and rigidly attaches each piece to its corresponding interface frame.
- To show the reconstructed deformed shape of the body, use `Deformed Geometry`. The block uses the provided triangulation of the body and the recovery data to reconstruct the elastic deformations of the vertices on the triangulation.

File Name — Path of CAD geometry file
custom character vector

Path of the CAD file that defines the undeformed solid geometry, specified as a custom character vector. The file location can be specified as an absolute path starting from the root directory of the file system or a relative path starting from a folder on the MATLAB path.

The CAD file must define the geometry of the flexible body by using the same reference frame as the reduced-order model. See “Supported Software and File Formats” on page 1-124 for details of supported file formats.

Example: `'C:/Users/JDoe/Documents/myShape.STEP'` or `'Documents/myShape.STEP'`

Dependencies

To enable this parameter, set **Type** to `Partitioned Geometry`.

Unit Type — Source of length unit for geometry
`From File` (default) | `From Unit System`

Source of the solid geometry unit, specified as `From File` or `From Unit System`. Select `From File` to use the units specified in the imported file. Select `From Unit System` to use the units specified by the **Unit System** parameter.

Dependencies

To enable this parameter, set **Type** to `Partitioned Geometry`.

Vertex Coordinates — Coordinates of vertices on triangulation representation
`[ ]` (default) | n_v-by-*3* matrix

Coordinates of the vertices on the triangulation representation for the solid, specified as a n_v-by-*3* matrix. n_v is the number of vertices, and each row of the matrix contains the *x*, *y*, and *z* coordinates for a vertex. The coordinates refer to the common reference frame.

Dependencies

To enable this parameter, set **Type** to `Deformed Geometry`.

Facet Vertex Indices — Indices of vertices that define facets of triangulation representation
`[ ]` (default) | n_f-by-*3* matrix

Indices of vertices that define the facets of the triangulation representation for the solid, specified as a n_f-by-*3* matrix. n_f is the number of facets, and each row of the matrix contains the indices of three vertices that define their corresponding facet. The index of a vertex is the row number of the vertex in the matrix for the **Vertex Coordinates** parameter.

Dependencies

To enable this parameter, set **Type** to `Deformed Geometry`.

Reconstruction Matrix — Recovery data to visualize deformed body
`[ ]` (default) | $3n_v$-by-*r* matrix

Recovery data to visualize the deformed body, specified as a $3n_v$-by-*r* matrix. n_v is the number of vertices and *r* is the number of columns of the mass or stiffness matrix in the ROM data.

The Reduced Order Flexible Solid block uses the reconstruction matrix to calculate the displacements of the vertices on the triangulation representation to display the deformed shape of the body. Each set of three rows of the matrix corresponds to the *x*, *y*, and *z* displacements of a vertex.

Dependencies

To enable this parameter, set **Type** to `Deformed Geometry`.

Visual Properties — Parameterizations for color and opacity
`From File` (default) | `Advanced` | `Simple`

Parameterization for specifying visual properties.

- Select `From File` to use color data from the imported CAD geometry file. Note that not all file formats allow color data. In formats that allow color data, that data is often optional. If your file does not specify color, the solid is rendered in gray.
- Select `Simple` to specify **Color** and **Opacity**.
- Select `Advanced` to specify more visual properties, such as **Diffuse Color**, **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `Partitioned Geometry`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `Partitioned Geometry`.
2. **Visual Properties** to `Simple`.

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `Partitioned Geometry`.
2. **Visual Properties** to `Simple`.

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered solid and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `Partitioned Geometry`.
2. **Visual Properties** to `Advanced`.

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `Partitioned Geometry`.
2. **Visual Properties** to `Advanced`.

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1. **Type** to `Partitioned Geometry`.
2. **Visual Properties** to `Advanced`.

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `Partitioned Geometry`.
2. **Visual Properties** to `Advanced`.

Shininess — Shininess of rendered solid
75 (default) | scalar in the range of 1 to 128

Shininess of the rendered solid, specified as a scalar in the range of 0 to 128. This parameter affects the sharpness of the specular reflections of the rendered solid. A solid with high shininess has a mirror-like appearance, and a solid with low shininess has a more low-gloss or satin appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `Partitioned Geometry`.
2. **Visual Properties** to `Advanced`.

Version History

Introduced in R2019b

References

[1] Shabana, Ahmed A. *Dynamics of Multibody Systems*. Fourth edition. New York: Cambridge University Press, 2014.

[2] Agrawal, Om P., and Ahmed A. Shabana. “Dynamic Analysis of Multibody Systems Using Component Modes.” *Computers & Structures* 21, no. 6 (January 1985): 1303–12. https://doi.org/10.1016/0045-7949(85)90184-1.

Extended Capabilities

C/C++ Code Generation

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

See Also

General Flexible Beam | Rigid Transform

Topics

“Triangulation Representations”

Reference Frame

Non-inertial reference frame

Libraries:
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.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

World Frame | Rigid Transform

Topics

“Creating Connection Frames”
“Visualize Simscape Multibody Frames”
“Working with Frames”

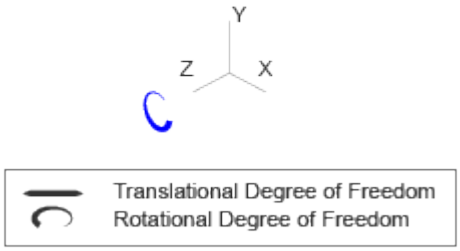
Revolute Joint

Joint with one revolute primitive

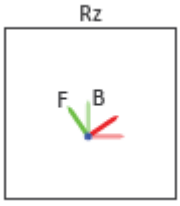
Libraries:
Simscape / Multibody / Joints

Description

The Revolute Joint block models a joint that has one rotational degree of freedom.



The joint constrains the motion of two arbitrary frames that connect to the base and follower frames of the joint to pure rotation about a common axis. The axis of rotation is aligned with the *z*-axis of the joint base frame. The base and follower frames have a common origin and *z*-axis, and the follower frame rotates about the *z*-axis, as shown in the image.



To target the initial state of the a joint primitive, use the parameters under **State Targets**. The position and velocity targets are resolved in the base frame. You can also set the priority levels for the targets. If the block cannot simultaneously satisfy the state targets, the priority level determines which targets to satisfy first and how closely to satisfy them. See “Guiding Assembly” for more information.

To model damping and spring behavior in a joint primitive, use the parameters under **Internal Mechanics**. Specify joint damping coefficients to model energy dissipation and joint spring stiffness to model energy storage. Springs resist attempts to displace the joint primitive from its equilibrium position. Joint dampers act as energy dissipation elements. Springs and dampers are strictly linear.

To specify the limits of a joint primitive, use the parameters under **Limits**. The lower and upper bounds define the width of the free region of a joint primitive. The block applies a force or torque to

accelerate the joint position back to the free region when the position exceeds the bounds. The block uses a smoothed spring-damper method to compute the force or torque. See “Description” on page 1-415 section of the Spatial Contact Force block for more information about the smoothed spring-damper method.

A revolute primitive provides two actuation parameters, **Torque** and **Motion**, that govern the motion of the primitive during a simulation. See “Specifying Joint Actuation Inputs” for more information. Additionally, the block has ports that output sensing data, such as position, velocity, acceleration, forces, and torques, that enable you to perform analytical tasks on a model. See “Sensing” and “Force and Torque Sensing” for more information.

Ports

Frame

B — Base frame
frame

Base frame of the joint block.

F — Follower frame
frame

Follower frame of the joint block.

Input

t — Actuation torque
physical signal

Physical signal input port that accepts the actuation torque for the joint primitive. The signal provides the value of the torque that applies on both the base and follower frames of the joint primitive.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz)** > **Actuation**, set **Torque** to `Provided by Input`.

q — Motion profile
physical signal

Physical signal input port that accepts the motion profile for the joint primitive. The signal provides the rotation of the follower frame with respect to the base frame about the joint primitive axis. Note that the signal must also contain the first and second derivatives of the rotation.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz)** > **Actuation**, set **Motion** to `Provided by Input`.

mode — Joint mode control
scalar

Input port that controls the joint mode. The signal is a unitless scalar. The joint mode is normal when the input signal is 0 and disengaged when the input signal is -1. The joint mode can be changed many times during a simulation.

Dependencies

To enable this port, under **Mode Configuration**, set **Mode** to `Provided by Input`.

Output

q — Position of joint primitive
physical signal

Physical signal port that outputs the position of the joint primitive. The value is the rotation angle of the follower frame with respect to the base frame.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz) > Sensing**, select **Position**.

w — First derivative of position of joint primitive
physical signal

Physical signal port that outputs the first derivative of position of the joint primitive.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz) > Sensing**, select **Velocity**.

b — Second derivative of position of joint primitive
physical signal

Physical signal port that outputs the second derivative of position of the joint primitive.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz) > Sensing**, select **Acceleration**.

t — Actuator torque acting on joint primitive
physical signal

Physical signal port that outputs the actuator torque acting on the joint primitive.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz) > Sensing**, select **Actuator Torque**.

tll — Lower-limit torque
physical signal

Physical signal port that outputs the lower-limit torque. The block applies the torque when the joint primitive position exceeds the lower bound of the free region. The torque applies to both the base and follower frames of the joint primitive to accelerate the position back to the free region.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz) > Sensing**, select **Lower-Limit Torque**.

tul — Upper-limit torque
physical signal

Physical signal port that outputs the upper-limit torque. The block applies the torque when the joint primitive position exceeds the upper bound of the free region. The torque applies to both the base and follower frames of the joint primitive to accelerate the position back to the free region.

Dependencies

To enable this port, under **Z Revolute Primitive (Rz) > Sensing**, select **Upper-Limit Torque**.

fc — Constraint force
physical signal

Physical signal port that outputs constraint force that acts in the joint. The force maintains the translational constraints of the joint. See “Measure Joint Constraint Forces” for more information.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Constraint Force**.

tc — Constraint torque
physical signal

Physical signal port that outputs constraint torque that acts in the joint. The torque maintains the rotational constraints of the joint. See “Force and Torque Sensing” for more information.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Constraint Torque**.

ft — Total force
physical signal

Physical signal port that outputs the total force that acts in the joint. The total force is the sum of the forces transmitted from one frame to the other through the joint. The force includes actuation, internal, limit and constraint forces. See “Force and Torque Sensing” for more information.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Total Force**.

tt — Total torque
physical signal

Physical signal port that outputs the total torque that acts in the joint. The total torque is the sum of the torques transmitted from one frame to the other through the joint. The torque includes actuation, internal, limit, and constraint torques. See “Force and Torque Sensing” for more information.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Total Torque**.

Parameters

Z Revolute Primitive (Rz)

State Targets > Specify Position Target — Whether to specify position target
`off` (default) | `on`

Select this parameter to enable parameters that specify the position target of the joint primitive.

State Targets > Specify Position Target > Priority — Priority level of position target
`High (desired)` (default) | `Low (approximate)`

Priority level of the position target, specified as `High (desired)` or `Low (approximate)`.

Dependencies

To enable this parameter, under **Z Revolute Primitive (Rz) > State Targets**, select **Specify Position Target**.

State Targets > Specify Position Target > Value — Angle of position target
0 `deg` (default) | scalar with a unit of angle

Angle to specify the position target, specified as a scalar with a unit of angle.

Dependencies

To enable this parameter, under **Z Revolute Primitive (Rz) > State Targets**, select **Specify Position Target**.

State Targets > Specify Velocity Target — Whether to specify velocity target
`off` (default) | `on`

Select this parameter to enable parameters for that specify the velocity target of the joint primitive.

State Targets > Specify Velocity Target > Priority — Priority level of velocity target
`High (desired)` (default) | `Low (approximate)`

Priority level of the velocity target, specified as `High (desired)` or `Low (approximate)`.

Dependencies

To enable this parameter, under **Z Revolute Primitive (Rz) > State Targets**, select **Specify Velocity Target**.

State Targets > Specify Velocity Target > Value — Velocity target of joint primitive
0 `deg/s` (default) | scalar with unit of angular velocity

Velocity target of the joint primitive, specified as a scalar with a unit of angular velocity.

Dependencies

To enable this parameter, under **Z Revolute Primitive (Rz) > State Targets**, select **Specify Velocity Target**.

Internal Mechanics > Equilibrium Position — Position where internal torque is zero
0 `deg` (default) | scalar with unit of angle

Position where the spring torque is zero, specified as a scalar with a unit of angle. The value specifies the rotation angle of the follower frame with respect to the base frame.

Internal Mechanics > Spring Stiffness — Stiffness of force law
0 `N*m/deg` (default) | scalar with unit of stiffness

Stiffness of the internal spring-damper force law for the joint primitive, specified as a scalar with a unit of stiffness.

Internal Mechanics > Damping Coefficient — Damping coefficient of force law
0 `N*m/(deg/s)` (default) | scalar with unit of damping coefficient

Damping coefficient of the internal spring-damper force law for the joint primitive, specified as a scalar with a unit of damping coefficient.

Limits > Specify Lower Limit — Whether to specify lower position limit
`off` (default) | `on`

Select this parameter to enable parameters that specify the lower limit of the joint primitive.

Limits > Specify Lower Limit > Bound — Lower bound of free region
-90 `deg` (default) | scalar with unit of angle

Lower bound for the free region of the joint primitive, specified as a scalar with a unit of angle.

Dependencies

To enable this parameter, under **Z Revolute Primitive (Rz) > Limits**, select **Specify Lower Limit**.

Limits > Specify Lower Limit > Spring Stiffness — Stiffness of spring at lower bound
1e4 `N*m/deg` (default) | scalar with unit of stiffness

Stiffness of the spring at lower bound, specified as a scalar with a unit of stiffness.

Dependencies

To enable this parameter, under **Z Revolute Primitive (Rz) > Limits**, select **Specify Lower Limit**.

Limits > Specify Lower Limit > Damping Coefficient — Damping coefficient at lower bound
10 `N*m/(deg/s)` (default) | scalar with unit of damping coefficient

Damping coefficient at lower bound, specified as a scalar with a unit of damping coefficient.

Dependencies

To enable this parameter, under **Z Revolute Primitive (Rz) > Limits**, select **Specify Lower Limit**.

Limits > Specify Lower Limit > Transition Region Width — Region to smooth spring and damper forces
0.1 `deg` (default) | scalar with unit of angle

Region to smooth the spring and damper forces, specified as a scalar with a unit of angle.

The forces get full value once the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, under **Z Revolute Primitive (Rz) > Limits**, select **Specify Lower Limit**.

Limits > Specify Upper Limit — Whether to specify upper position limit
`off` (default) | `on`

Select this parameter to enable parameters for specifying the upper limit of the joint primitive.

Limits > Specify Upper Limit > Bound — Upper bound of free region
90 deg (default) | scalar with unit of angle

Upper bound for the free region of the joint primitive, specified as a scalar with a unit of angle.

Dependencies

To enable this parameter, under **Z Revolute Primitive (Rz) > Limits**, select **Specify Upper Limit**.

Limits > Specify Upper Limit > Spring Stiffness — Stiffness of spring at upper bound
1e4 N*m/deg (default) | scalar with unit of stiffness

Stiffness of the spring at upper bound, specified as a scalar with a unit of stiffness.

Dependencies

To enable this parameter, under **Z Revolute Primitive (Rz) > Limits**, select **Specify Upper Limit**.

Limits > Specify Upper Limit > Damping Coefficient — Damping coefficient at upper bound
10 N*m/(deg/s) (default) | scalar with unit of damping coefficient

Damping coefficient at upper bound, specified as a scalar with a unit of damping coefficient.

Dependencies

To enable this parameter, under **Z Revolute Primitive (Rz) > Limits**, select **Specify Upper Limit**.

Limits > Specify Upper Limit > Transition Region Width — Region to smooth spring and damper forces
0.1 deg (default) | scalar with unit of angle

Region to smooth the spring and damper forces, specified as a scalar with a unit of angle.

The forces get full value once the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, under **Z Revolute Primitive (Rz) > Limits**, select **Specify Upper Limit**.

Actuation > Torque — Option to provide actuation torque
None (default) | Provided by Input | Automatically Computed

Option to provide the actuation torque for the joint primitive, specified as None, Provided by Input, or Automatically Computed.

Actuation Torque Setting	Description
None	No actuation torque.
Provided by Input	Expose the input port **t**.

Actuation Torque Setting	Description
`Automatically computed`	The torque required to satisfy the motion inputs to the mechanism. The block computes the torque automatically. Note that if you set this parameter to `Automatically computed`, it doesn't mean that you must use `Provided by Input` for the **Motion** parameter for the same joint primitive. The automatically computed torque could be to satisfy a motion input somewhere else in the mechanism.

Actuation > Motion — Option to provide actuation motion
`Automatically Computed` (default) | `Provided by Input`

Option to provide the actuation motion for the joint primitive, specified as `Automatically Computed` or `Provided by Input`.

Actuation Torque Setting	Description
`Automatically computed`	The block computes and applies the joint primitive motion based on model dynamics.
`Provided by Input`	Expose the input port **q**.

Sensing > Position — Whether to sense joint primitive position
off (default) | on

Select this parameter to enable the output port **q**.

Sensing > Velocity — Whether to sense joint primitive velocity
off (default) | on

Select this parameter to enable the port **w**.

Sensing > Acceleration — Whether to sense joint primitive acceleration
off (default) | on

Select this parameter to enable the port **b**.

Sensing > Lower-Limit Torque — Whether sense to lower-limit torque
off (default) | on

Select this parameter to enable the port **tll**.

Sensing > Upper-Limit Torque — Whether sense to upper-limit torque
off (default) | on

Select this parameter to enable the port **tul**.

Mode Configuration

Mode — Joint mode
`Normal` (default) | `Disengaged` | `Provided by Input`

Joint mode for the simulation, specified as `Normal`, `Disengaged`, or `Provided by Input`.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.
`Provided by Input`	Expose the port **mode**.

Composite Force/Torque Sensing

Direction — Measurement direction
`Follower on Base` (default) | `Base on Follower`

Measurement direction, specified as `Follower on Base` or `Base on Follower`.

- `Follower on Base` — Sense the force and torque that the follower frame exerts on the base frame.
- `Base on Follower` — Sense the force and torque that the base frame exerts on the follower frame.

Reversing the direction changes the sign of the measurements. For more information see “Force and Torque Measurement Direction”.

Resolution Frame — Frame used to resolve measurements
`Base` (default) | `Follower`

Frame used to resolve the measurements, specified as `Base` or `Follower`.

- `Base` — The joint block resolves the measurements in the coordinates of the base frame.
- `Follower` — The joint block resolves the measurements in the coordinates of the follower frame.

Constraint Force — Whether to sense constraint force in joint
off (default) | on

Select this parameter to enable the port **fc**.

Constraint Torque — Whether to sense constraint torque in joint
off (default) | on

Select this parameter to enable the port **tc**.

Total Force — Whether to sense total force in joint
off (default) | on

Select this parameter to enable the port **ft**.

Total Torque — Whether to sense total torque in joint
off (default) | on

Select this parameter to enable the port **tt**.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Topics

“Force and Torque Sensing”
“Guiding Assembly”
“Measure Joint Constraint Forces”

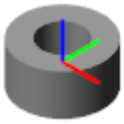
Revolved Solid

Solid revolved element with geometry, inertia, and color

Libraries:
Simscape / Multibody / Body Elements

Description

The Revolved Solid block is a 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.



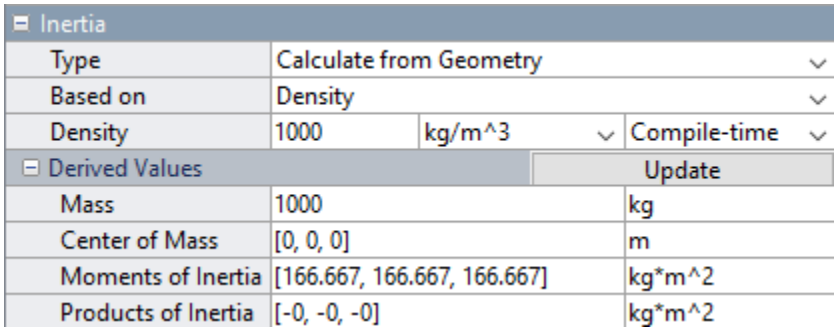
The Revolved Solid block adds to the attached frame a solid element with geometry, inertia, and color. The solid element can be a simple rigid body or part of a compound rigid body—a group of rigidly connected solids, often separated in space through rigid transformations. Combine Revolved Solid and other solid blocks with the 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.

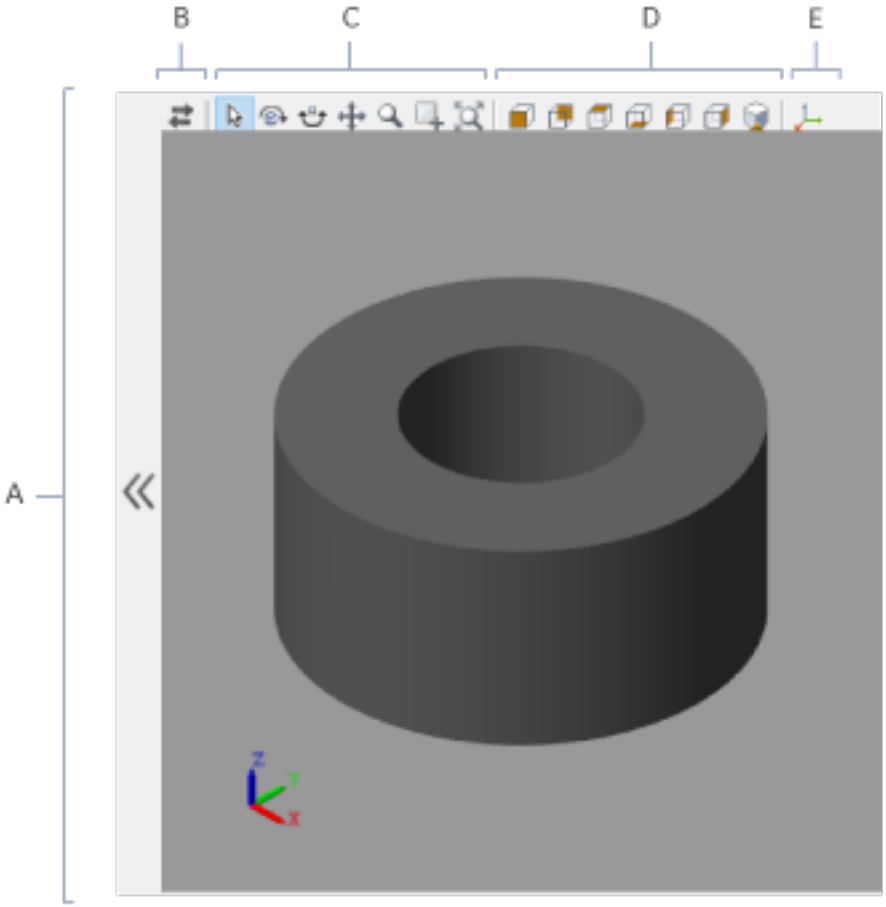


Derived Values Display

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.



A — Show/Hide Visualization Pane
B — Update Visualization
C — Select | Rotate | Roll | Pan | Zoom | Zoom to Region | Fit to View
D — Front | Back | Top | Bottom | Left | Right | Isometric Views
E — Show/Hide Frame

Revolved 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. 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”.

Geometry

CH — Convex hull representation
geometry

Convex hull that represents the geometry of the solid. Connect this port to a Spatial Contact Force block to model contacts on the convex hull.

Dependencies

To enable this port, under **Geometry**, expand **Export** and select **Convex Hull**.

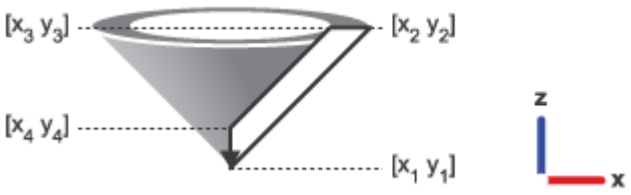
Parameters

Geometry

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

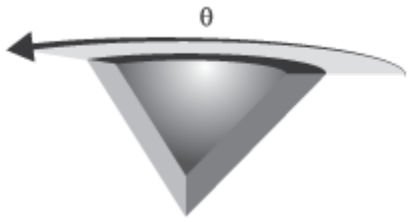
Cross-sectional shape specified as an [x,z] coordinate matrix, with each row corresponding to a point on the cross-sectional profile. The coordinates specified must define a closed loop with no self-intersecting segments.

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.

Convex Hull — Generate a convex hull representation of the true geometry
`off` (default) | `on`

Select **Convex Hull** to generate a convex hull representation of the true geometry. This convex hull can be used for contacts by connecting the Spatial Contact Force block.

Dependencies

To enable this option, select **Convex Hull** under the **Export**.

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 if known.

Density — Mass per unit volume of material
`1000 kg/m^3` (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*m^2` (default) | three-element vector with units of mass*length^2

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:

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

Custom: Products of Inertia — Off-diagonal elements of inertia tensor
`[0 0 0] kg*m^2` (default) | three-element vector with units of mass*length^2

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:

- $I_{yz} = -\int_m yz\, dm$
- $I_{zx} = -\int_m zx\, dm$
- $I_{xy} = -\int_m xy\, dm$

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

Type of the visual representation of the solid, specified as `From Geometry`, `Marker`, or `None`. Set the parameter to `From Geometry` to show the visual representation of the solid. Set the parameter to `Marker` to represent the solid as a marker. Set the parameter to `None` to hide the solid in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Shape — Shape of marker to represent 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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
`10 pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered solid and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Frames

Show Port R — Show reference frame port for connection to other blocks
on (default) | off

Select to expose the **R** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the solid block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** of the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the solid.
  - **At Center of Mass**: Make the new frame origin coincident with the center of mass of the solid.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

  - **Along Reference Frame Axis**: Selects an axis of the reference frame of the solid.
  - **Along Principal Inertia Axis**: Selects an axis of the principal inertia axis of the solid.
  - **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the solid. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2019b

Extended Capabilities

C/C++ Code Generation

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

See Also

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

Topics

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

Rigid Transform

Fixed spatial relationship between frames

Libraries:
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 to specify rotation transform
None (default) | `Aligned Axes` | `Standard Axis` | `Arbitrary Axis` | `Rotation Sequence` | `Rotation Matrix` | `Quaternion`

Method to use to specify the rotation transform between the base and follower frames. Set the parameter to None to constrain the base and follower frames to the same orientation.

Pair 1: Follower — Follower frame axis used to align with specified base frame axis
+X (default) | -X | +Y | -Y | +Z | -Z

Follower frame axis used to align with the base frame axis set by the **Pair 1: Base** parameter, specified as an orthogonal axis of the follower frame. The follower frame rotates with respect to the base frame to enable the alignment between the selected axes of the base and follower frames.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Aligned Axis`.

Pair 1: Base — Base frame axis
+Y (default) | +X | -X | -Y | +Z | -Z

Base frame axis to align with the follower frame specified by the **Pair 1: Follower**, specified as an orthogonal axis of the base frame.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Aligned Axis`.

Pair 2: Follower — Follower frame axis used to align with specified base frame axis
+Y | +X | -X | -Y | +Z | -Z

Base frame axis to align with the follower frame specified by the **Pair 2: Follower**, specified as an orthogonal axis of the follower frame. The follower frame rotates with respect to the base frame to enable the alignment between the selected axes of the base and follower frames.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Aligned Axis`.

Pair2: Base — Base frame axis
+Z (default) | +X | -X | +Y | -Y | -Z

Base frame axis used to let the follower frame axis set in the **Pair2: Follower** parameter to align with, specified as an orthogonal axis of the base frame. The axis choices for **Pair 2** depend on the **Pair 1** axis selections.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Aligned Axis`.

Axis — Standard axis of relative rotation
+Z (default) | +X | -X | +Y | -Y | -Z

Axis of the relative rotation, specified as an orthogonal axis of the base frame.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Standard Axis`.

Angle — Angle of relative rotation
0.0 deg (default) | scalar

Angle of the relative rotation, specified as a scalar. The angle indicates the rotation of the follower frame with respect to the base frame about the specified axis.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Standard Axis`.

Axis — Axis of relative rotation
[0 0 1] (default) | 3-by1 vector

Axis of the relative rotation, specified as a 3-by-1 unit vector. The vector is dimensionless and indicates the rotational axis resolved in the base frame.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Arbitrary Axis`.

Angle — Angle of relative rotation
0.0 `deg` (default) | scalar

Angle of the relative rotation, specified as a scalar. The angle indicates the rotation of the follower frame with respect to the base frame about the axis specified by the **Axis** parameter.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Arbitrary Axis`.

Rotation About — Frame whose axes to rotate follower frame about
`Follower Axes` (default) | `Base Axes`

Frame whose axes to rotate the follower frame about, specified as `Follower Axes` or `Base Axes`. If you set the parameter to `Follower Axes`, the follower frame rotates about its own axes, and the follower frame changes the orientation with each successive rotation. If you set the parameter to `Base Axes`, the follower frame rotates about the fixed base frame axes. See “Rotational Measurements” for more information.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Rotation Sequence`.

Sequence — Sequence of rotation axis
`X-Y-X` (default) | `X-Y-Z` | `X-Z-X` | `X-Z-Y` | `Y-X-Y` | `Y-X-Z` | `Y-Z-X` | `Y-Z-Y` | `Z-X-Y` | `Z-X-Z` | `Z-Y-X` | `Z-Y-Z`

Sequence of the rotation axis for three successive elementary rotations. See “Rotation Sequence Measurements” for more information.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Rotation Sequence`.

Angles — Angles for elementary rotations
`[0 0 0] deg` (default) | 3-by-1 vector

Angles for elementary rotations, specified as a 3-by-1 vector. See “Rotation Sequence Measurements” for more information.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Rotation Sequence`.

Matrix — Rotation matrix
`[1 0 0; 0 1 0; 0 0 1]` (default) | 3-by-3 matrix

Relative rotation, specified as a 3-by-3 matrix. The matrix must be orthogonal and have determinant 1. See “Rotational Measurements” for more information.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Rotation Matrix`.

Quaternion — Relative rotation using quaternion
[1 0 0 0] (default) | unit quaternion vector

Relative rotation, specified as a vector in quaternion parameterization. See “Rotational Measurements” for more information about the quaternion.

Dependencies

To enable this parameter, under **Rotation**, set **Method** parameter to `Quaternion`.

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
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.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

World Frame | Variable Brick Solid | Reference Frame | Transform Sensor

Topics

“Creating Connection Frames”
“Model a Simple Link”
“Working with Frames”
“Visualize Simscape Multibody Frames”

Spatial Contact Force

Model contact between two geometries

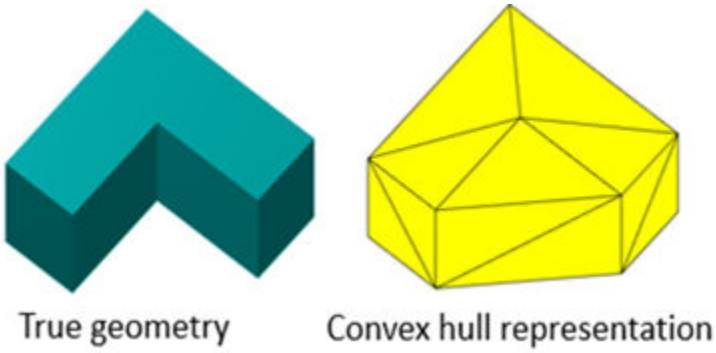
Libraries:
Simscape / Multibody / Forces and Torques

Description

The Spatial Contact Force block models the contact between geometries associated with a pair of solids. You can use the built-in penalty method or custom normal and friction force laws to model a contact.

Supported Geometries

The Spatial Contact Force block can model contacts between a variety of geometry pairs. The geometries can be sourced from the solid blocks in the Body Elements sublibrary or from the point and surface blocks in the Curves and Surfaces sublibrary. The geometry exported from a solid block is the convex hull of the solid. For a nonconvex body that can be modeled by using a File Solid, Extruded Solid, or Revolved Solid block, the block exports the convex hull for the Spatial Contact Force block to compute contact forces. When computing inertial properties, the solid blocks use the true geometry. The figure shows an example of a nonconvex geometry and its convex hull representation.



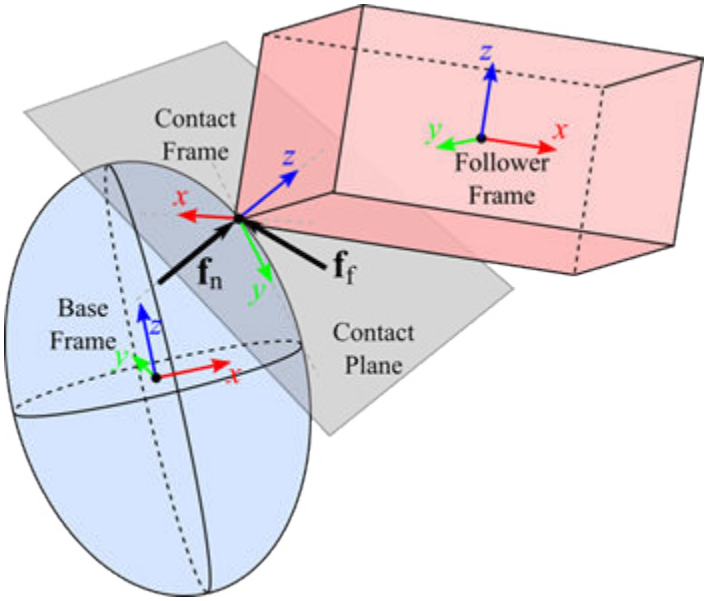
The Spatial Contact Force block does not model contacts between certain geometry pairs. For the full set of supported pairs, see the table.

	Convex hull of solid	Disk	Grid surface	Infinite plane	Point	Point cloud
Convex hull of solid	Yes	Yes	No	Yes	Yes	Yes
Disk	Yes	No	No	Yes	No	No
Grid surface	No	No	No	No	Yes	Yes

Infinite plane	Yes	Yes	No	No	Yes	Yes
Point	Yes	No	Yes	Yes	No	No
Point cloud	Yes	No	Yes	Yes	No	No

Contact Forces

The image shows how the Spatial Contact Force block models a spatial contact problem. In this case, the contact is between a blue base geometry and a red follower geometry.



During the contact, each geometry has a contact frame. The two contact frames are always coincident and located at the contact point. The z-direction of the contact frames is an outward normal vector for the base geometry, but inward normal vector for the follower geometry. During continuous contact, the contact frames move around the geometry as the contact point moves.

The block applies contact forces to the geometries at the origin of the contact frames in accordance with Newton's Third Law:

1. The normal force, f_n, which is aligned with the z-axis of the contact frame. This force pushes the geometries apart in order to reduce penetration.
2. The frictional force, f_f, which lies in the contact plane. This force opposes the relative tangential velocities between the geometries.

To specify a normal contact force, in the **Normal Force** section, set the **Method** parameter to `Smooth Spring-Damper` or `Provided by Input`. If you select `Smooth Spring-Damper`, the normal force is:

$$f_n = s(d, w) \cdot (k \cdot d + b \cdot d'),$$

where:

- f_n is the normal force applied in equal-and-opposite fashion to each contacting geometry.

- d is the penetration depth between two contacting geometries.
- w is the transition region width specified in the block.
- d' is the first time derivative of the penetration depth.
- k is the normal-force stiffness specified in the block.
- b is the normal-force damping specified in the block.
- $s(d, w)$ is the smoothing function.

The force law is smoothed near the onset of penetration. When $d < w$, the smoothing function increases continuously and monotonically over the interval $[0, w]$. The function is 0 when $d = 0$, the function is 1 when $d = w$, and the function has zero derivative with respect to d at the endpoints of the interval.

To better detect contacts when the value of the **Transition Region Width** parameter is small, the Spatial Contact Force block supports optional zero-crossing detection. The zero-crossing events only occur when the separation distance changes from positive or zero to negative and vice versa.

Note The zero-crossing detection of the Spatial Contact Force block is different than the zero-crossing detection of other Simulink® blocks, such as From File and Integrator, because the force equation of the Spatial Contact Force is continuous. For more information about zero-crossing detection in Simulink blocks, see “Zero-Crossing Detection”.

The Spatial Contact Force block clips the computed force to be always nonnegative. If the force law gives a negative force, the block applies zero force instead. This happens briefly as the geometries are separating and penetration is about to end. At that point, d is approaching zero and d' is negative. This modification ensures that the contact normal force is always repulsive and never attractive.

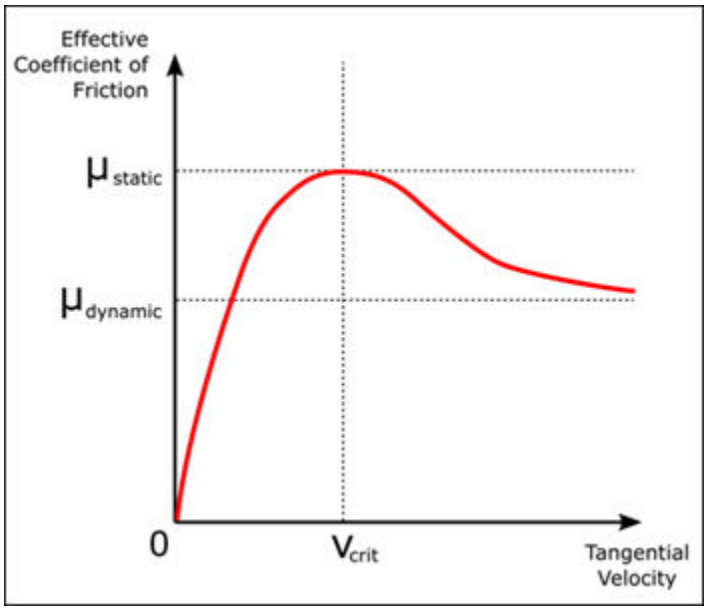
To specify a frictional force, in the **Frictional Force** section, set the **Method** parameter to `Smooth Stick-Slip`, `Provided by Input`, or `None`. If you select `Smooth Stick-Slip`, the frictional force is always directly opposed to the direction of the relative velocity at the contact point and is related to the normal force through a coefficient of friction that varies depending on the magnitude of the relative velocity:

$$|f_f| = \mu \cdot |f_n|,$$

where:

- f_f is the frictional force.
- f_n is the normal force.
- μ is the effective coefficient of friction.

The effective coefficient of friction is a function of the values of the **Coefficient of Static Friction**, **Coefficient of Dynamic Friction**, and **Critical Velocity** parameters, and the magnitude of the relative tangential velocity. At high relative velocities, the value of the effective coefficient of friction is close to that of the coefficient of dynamic friction. At the critical velocity, the effective coefficient of friction achieves a maximum value that is equal to the coefficient of static friction. The graph shows the basic relationship in the typical case where $\mu_{static} > \mu_{dynamic}$. In this case, the model is able to approximate stiction with a higher effective coefficient of friction near small tangential velocities.



Ports

Geometry

B — Base geometry
geometry

Geometry port associated with the base geometry.

F — Follower geometry
geometry

Geometry port associated with the follower geometry.

Input

fn — Normal contact force magnitude
physical signal

Physical signal input port that accepts the normal contact force magnitude between two geometries. The input signal is a scalar that specifies the normal contact force. The block clips negative values to zero.

Dependencies

To enable this port, in the **Normal Force** section, set **Method** to `Provided by Input`.

ffr — Friction force
physical signal

Physical signal input port that accepts the frictional force between two geometries. The input signal is a 2-by-1 vector that specifies the *x* and *y* components of the applied frictional force resolved in the contact frame.

Dependencies

To enable this port, in the **Frictional Force** section, set **Method** to `Provided by Input`.

Output

con — Contact signal
physical signal

Physical signal output port that provides the contact status of the base and follower geometries. The geometries are in contact if the value of the signal is 1 or separated if the value of the signal is 0.

Dependencies

To enable this port, in the **Sensing** section, select **Contact Signal**.

sep — Separation distance
physical signal

Physical signal output port that provides the separation distance between two geometries.

If the geometries are not penetrating each other, the signal has a nonnegative value that equals the minimum distance between the two geometries. If the geometries are penetrating, the signal has a negative value that equals the penetration depth.

Dependencies

To enable this port, in the **Sensing** section, select **Separation Distance**.

fn — Normal contact force magnitude
physical signal

Physical signal output port that provides the magnitude of the normal contact force between two geometries.

Dependencies

To enable this port, in the **Sensing** section, select **Normal Force Magnitude**.

ffrm — Frictional force magnitude
physical signal

Physical signal output port that provides the magnitude of the frictional contact force between the two geometries.

Dependencies

To enable this port, in the **Sensing** section, select **Frictional Force Magnitude**.

vn — Relative normal velocity
physical signal

Physical signal output port that provides the *z*-component of the relative velocity between the contact points of the base and follower geometries. The output value is resolved in the contact frame.

Dependencies

To enable this port, in the **Sensing** section, select **Relative Normal Velocity**.

vt — Relative tangential velocity
physical signal

Physical signal output port that provides the x and y components of the relative velocity between the contact points of the base and follower geometries. The output value is a 2-by-1 vector resolved in the contact frame.

Dependencies

To enable this port, in the **Sensing** section, select **Relative Tangential Velocity**.

Contact Frame

Rb — Base rotation
physical signal

Physical signal port that outputs a 3-by-3 rotation matrix that maps the vectors in the contact frame to vectors in the reference frame of the base geometry. The output signal is resolved in the reference frame associated with the base geometry.

Dependencies

To enable this port, in the **Sensing > Contact Frame** section, select **Base Rotation**.

pb — Base translation
physical signal

Physical signal port that outputs at 3-by-1 vector that contains the coordinates of the origin of the contact frame resolved in the reference frame of the base geometry.

Dependencies

To enable this port, in the **Sensing > Contact Frame** section, select **Base Translation**.

Rf — Follower rotation
physical signal

Physical signal port that outputs a 3-by-3 rotation matrix that maps vectors in the contact frame to vectors in the reference frame of the follower geometry. The output signal is resolved in the reference frame associated with the follower geometry.

Dependencies

To enable this port, in the **Sensing > Contact Frame** section, select **Follower Rotation**.

pf — Follower translation
physical signal

Physical signal port that outputs at 3-by-1 vector that contains the coordinates of the origin of the contact frame resolved in the reference frame of the follower geometry.

Dependencies

To enable this port, in the **Sensing > Contact Frame** section, select **Follower Translation**.

Parameters

Normal Force

Method — Methods to specify normal contact force
`Smooth Spring-Damper` (default) | `Provided by Input`

Methods to specify the normal contact force, specified as either `Smooth Spring-Damper` or `Provided by Input`.

Select `Smooth Spring-Damper` to use the modified spring-damper method to model the normal contact force, or select `Provided by Input` to input a custom force as the normal contact force.

Stiffness — Resistance of contact spring to geometric penetration
1e6 `N/m` (default) | scalar

Resistance of the contact spring to geometric penetration, specified as a scalar. The spring stiffness is a constant during the contact. The larger the value of the spring stiffness, the harder the contact between the geometries.

Dependencies

To enable this parameter, in the **Normal Force** section, set **Method** to `Smooth Spring-Damper`.

Damping — Resistance of contact damper to motion while geometries are penetrating
1e3 `N/(m/s)` (default) | scalar

Resistance of the contact damper to motion while the geometries are penetrating, specified as a scalar. The damping coefficient is a constant value that represents the lost energy from colliding geometries. The larger the value of the damping coefficient, the more energy is lost when geometries collide and the faster the contact vibrations are dampened. Use a value of zero to model perfectly elastic collisions, which conserve energy.

Dependencies

To enable this parameter, in the **Normal Force** section, set **Method** to `Smooth Spring-Damper`.

Transition Region Width — Region over which spring-damper force raises to its full value
1e-4 `m` (default) | scalar

Region over which the spring-damper force raises to its full value, specified as a scalar. The smaller the region, the sharper the onset of contact and the smaller the time step required for the solver. Reducing the transition region improves model accuracy while expanding the transition region improves simulation speed.

Dependencies

To enable this parameter, in the **Normal Force** section, set **Method** to `Smooth Spring-Damper`.

Frictional Force

Method — Methods to specify frictional force
`Smooth Stick-Slip` (default) | `None` | `Provided by Input`

Methods to specify the frictional force, specified as `Smooth Stick-Slip`, `None`, or `Provided by Input`.

Select `None` to omit friction, select `Smooth Stick-Slip` to use the modified stick-slip method to compute the frictional force, or use `Provided by Input` to input a custom frictional force.

Coefficient of Static Friction — Ratio of magnitude of frictional force to the magnitude of normal force when tangential velocity is close to zero
0.5 (default) | nonnegative scalar

Ratio of the magnitude of the frictional force to the magnitude of the normal force when the tangential velocity is close to zero, specified as a positive scalar.

This value is determined by the material properties of the contacting geometries. The value of this parameter is often less than one, although values greater than one are possible for high-friction materials. In most cases, this value should be higher than the coefficient of dynamic friction.

Dependencies

To enable this parameter, in the **Frictional Force** section, set **Method** to `Smooth Stick-Slip`.

Coefficient of Dynamic Friction — Ratio of magnitude of frictional force to magnitude of normal force when tangential velocity is large
0.3 (default) | nonnegative scalar

Ratio of the magnitude of the frictional force to the magnitude of the normal force when the tangential velocity is large, specified as a nonnegative scalar.

This value is determined by the material properties of the contacting geometries. The value of this parameter is often less than one, although values greater than one are possible for high-friction materials. In most cases, this value should be less than the coefficient of static friction.

Dependencies

To enable this parameter, in the **Frictional Force** section, set **Method** to `Smooth Stick-Slip`.

Critical Velocity — Velocity that determines blending between static and dynamic coefficients of friction
1e-3 `m/s` (default) | scalar

Velocity that determines the blending between the static and dynamic coefficients of friction, specified as a scalar.

When the critical velocity is equal to the magnitude of the tangential velocity, the effective coefficient of friction is equal to the value of the **Coefficient of Static Friction** parameter. As the magnitude of the tangential velocity increases beyond the specified critical velocity, the effective coefficient of friction asymptotically approaches the value of the **Coefficient of Dynamic Friction** parameter.

Dependencies

To enable this parameter, in the **Frictional Force** section, set **Method** to `Smooth Stick-Slip`.

Zero-Crossings

Detect Contact Start and End — Detect start and end of contacts as zero-crossing events
off (default) | on

Select to detect the start and end of each contact as zero-crossing events. The zero-crossing events occur when the separation distance changes from positive or zero to negative and vice versa.

Version History

Introduced in R2019b

Extended Capabilities

C/C++ Code Generation

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

See Also

Brick Solid | Cylindrical Solid | Extruded Solid | Spherical Solid | File Solid

Topics

“Modeling Contact Force Between Two Solids”
“Model Wheel Contact in a Car”
“Use Contact Proxies to Simulate Contact”
“Zero-Crossing Detection”

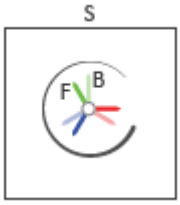
Spherical Joint

Joint allows 3-D rotations

Libraries:
Simscape / Multibody / Joints

Description

The Spherical Joint block provides three degrees of freedom between two frames. The follower frame can have an arbitrary 3-D rotation with respect to the base frame. During a simulation, the origins of the base and follower frames remain coincident.



Unlike a gimbal joint, the Spherical Joint does not have kinematic singularity because the rotation is encoded as a quaternion.

For information about how to specify joint blocks, see “Modeling Joint Connections”.

Ports

Frame

B — Base frame
frame

Base frame of the joint block.

F — Follower frame
frame

Follower frame of the joint block.

Input

tx — Actuation torque about *x*-axis of resolution frame
physical signal

Physical signal input port that accepts the actuation torque for the spherical primitive. The signal has a scalar format that represents the torque about the *x*-axis of the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Actuation**, set **Torque** to `Provided by Input` and select **Torque (X)**.

ty — Actuation torque about *y*-axis of resolution frame
physical signal

Physical signal input port that accepts the actuation torque for the spherical primitive. The signal has a scalar format that represents the torque about the *y*-axis of the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Actuation**, set **Torque** to `Provided by Input` and select **Torque (Y)**..

tz — Actuation torque about *z*-axis of resolution frame
physical signal

Physical signal input port that accepts the actuation torque for the spherical primitive. The signal has a scalar format that represents the torque about the *z*-axis of the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Actuation**, set **Torque** to `Provided by Input` and select **Torque (Z)**..

t — Actuation torque vector
physical signal

Physical signal input port that accepts the actuation torque for the spherical primitive. The vector represents the actuation torque to apply between the base and follower frame as expressed in resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Actuation**, set **Motion** to `Provided by Input` and select **Torque (XYZ)**..

Mode Configuration

mode — Joint mode control
scalar

Input port that controls the mode of the joint. The signal is a unitless scalar. The joint operates in normal mode when the input signal is `0` and operates in disengaged mode when the input signal is `-1`. You can change between the two modes at any time during simulation.

Dependencies

To enable this port, under **Mode Configuration**, set **Mode** to `Provided by Input`.

Output

Sensing

Q — Relative orientation in quaternion parameterization
physical signal

Orientation of the follower frame with respect to the base frame, returned as a unit quaternion. See “Quaternion Measurements” for more information.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Sensing**, select **Position**.

wx — *X*-coordinate of relative angular velocity
physical signal

X-coordinate of the relative angular velocity, returned as a scalar. The value is resolved in the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Sensing**, select **Velocity (X)**.

wy — *Y*-coordinate of relative angular velocity
physical signal

Y-coordinate of the relative angular velocity, returned as a scalar. The value is resolved in the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Sensing**, select **Velocity (Y)**.

wz — *Z*-coordinate of relative angular velocity
physical signal

Z-coordinate of the relative angular velocity, returned as a scalar. The value is resolved in the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Sensing**, select **Velocity (Z)**.

w — Angular velocity vector
physical signal

Relative angular velocity, returned as a 3-D vector resolved in the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Sensing**, select **Velocity**.

bx — *X*-coordinate of relative angular acceleration
physical signal

X-coordinate of the relative angular acceleration, returned as a scalar. This quantity equals the time derivative of the signal exported from the port **wx**.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Sensing**, select **Acceleration (X)**.

by — *Y*-coordinate of relative angular acceleration
physical signal

Y-coordinate of the relative angular acceleration, returned as a scalar. This quantity equals the time derivative of the signal exported from the port **wy**.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Acceleration (Y)**.

bz — *Z*-coordinate of relative angular acceleration
physical signal

Z-coordinate of the relative angular acceleration, returned as a scalar. This quantity equals the time derivative of the signal exported from the port **wz**.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Acceleration (Z)**.

b — Angular acceleration vector
physical signal

Relative angular acceleration, returned as a 3-D vector resolved in the resolution frame. This quantity equals the time derivative of the signal exported from the port **w**.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Acceleration**.

tll — Lower-limit torque magnitude
physical signal

Physical signal port that outputs the magnitude of the lower-limit torque. The block applies the lower-limit torque when the angle between the z-axes of the two frames is less than the bound of the lower limit. The torque applies to both the base and follower frames of the spherical primitive to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Signed Lower-Limit Torque Magnitude**.

tul — Upper-limit torque magnitude
physical signal

Physical signal port that outputs the magnitude of the upper-limit torque. The block applies the upper-limit torque when the angle between the z-axes of the two frames exceeds the upper bound. The torque applies to both the base and follower frames of the spherical primitive to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Upper-Limit Torque**.

Composite Force/Torque Sensing

fc — Constraint force
physical signal

Physical signal port that outputs the constraint force that acts across the joint. The force maintains the translational constraints of the joint. For more information, see “Measure Joint Constraint Forces”.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Constraint Force**.

tc — Constraint torque
physical signal

Physical signal port that outputs the constraint torque that acts across the joint. The torque maintains the rotational constraints of the joint. For more information, see “Force and Torque Sensing”.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Constraint Torque**.

ft — Total force
physical signal

Physical signal port that outputs the total force that acts across the joint. The total force is the sum of the forces transmitted from one frame to the other through the joint. The force includes the actuation, internal, limit, and constraint forces. See “Force and Torque Sensing” for more information.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Total Force**.

tt — Total torque
physical signal

Physical signal port that outputs the total torque that acts across the joint. The total torque is the sum of the torques transmitted from one frame to the other through the joint. The torque includes the actuation, internal, limit, and constraint torques. For more information, see “Force and Torque Sensing”.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Total Torque**.

Parameters

Spherical Primitive (S)

State Targets

Specify Position Target — Whether to specify relative orientation target
`off` (default) | `on`

Select this parameter to specify the target of the relative orientation between the base and follower frames.

Priority — Priority level of relative orientation target
`High (desired)` (default) | `Low (approximate)`

Priority level of the relative orientation target, specified as `High (desired)` or `Low (approximate)`. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, select **Specify Position Target**.

Value > Method — Method to specify relative orientation target
`None` (default) | `Aligned Axes` | `Standard Axis` | `Arbitrary Axis` | `Rotation Sequence` | `Rotation Matrix` | `Quaternion`

Method to use to specify the relative orientation target between the base and follower frames. When specifying the parameter to `None`, the follower and base frames have the same orientation at the beginning of the simulation.

Dependencies

To enable this parameter, select **Specify Position Target**.

Pair 1: Follower — Follower frame axis used to align with specified base frame axis
+X (default) | -X | +Y | -Y | +Z | -Z

Follower frame axis used to align with the base frame axis set by the **Pair 1: Base** parameter, specified as an orthogonal axis of the follower frame. The follower frame rotates with respect to the base frame to enable the alignment between the selected axes of the base and follower frames.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Aligned Axis`.

Pair 1: Base — Base frame axis
+Y (default) | +X | -X | -Y | +Z | -Z

Base frame axis to align with the follower frame specified by the **Pair 1: Follower**, specified as an orthogonal axis of the base frame.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Aligned Axis`.

Pair 2: Follower — Follower frame axis used to align with specified base frame axis
+Y | +X | -X | -Y | +Z | -Z

Base frame axis to align with the follower frame specified by the **Pair 2: Follower**, specified as an orthogonal axis of the follower frame. The follower frame rotates with respect to the base frame to enable the alignment between the selected axes of the base and follower frames.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Aligned Axis`.

Pair2: Base — Base frame axis
+Z (default) | +X | -X | +Y | -Y | -Z

Base frame axis used to let the follower frame axis set in the **Pair2: Follower** parameter to align with, specified as an orthogonal axis of the base frame. The axis choices for **Pair 2** depend on the **Pair 1** axis selections.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Aligned Axis`.

Axis — Standard axis of relative rotation
+Z (default) | +X | -X | +Y | -Y | -Z

Axis of the relative rotation, specified as an orthogonal axis of the base frame.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Standard Axis`.

Angle — Angle of relative rotation
0.0 deg (default) | scalar

Angle of the relative rotation, specified as a scalar. The angle indicates the rotation of the follower frame with respect to the base frame about the specified axis.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Standard Axis`.

Axis — Axis of relative rotation
[0 0 1] (default) | 3-by1 vector

Axis of the relative rotation, specified as a 3-by-1 unit vector. The vector is dimensionless and indicates the rotational axis resolved in the base frame.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Arbitrary Axis`.

Angle — Angle of relative rotation
0.0 deg (default) | scalar

Angle of the relative rotation, specified as a scalar. The angle indicates the rotation of the follower frame with respect to the base frame about the axis specified by the **Axis** parameter.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Arbitrary Axis`.

Rotation About — Frame whose axes to rotate follower frame about
`Follower Axes` (default) | `Base Axes`

Frame whose axes to rotate the follower frame about, specified as `Follower Axes` or `Base Axes`. If you set the parameter to `Follower Axes`, the follower frame rotates about its own axes, and the

follower frame changes the orientation with each successive rotation. If you set the parameter to `Base Axes`, the follower frame rotates about the fixed base frame axes. See “Rotational Measurements” for more information.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Rotation Sequence`.

Sequence — Sequence of rotation axis
`X-Y-X` (default) | `X-Y-Z` | `X-Z-X` | `X-Z-Y` | `Y-X-Y` | `Y-X-Z` | `Y-Z-X` | `Y-Z-Y` | `Z-X-Y` | `Z-X-Z` | `Z-Y-X` | `Z-Y-Z`

Sequence of the rotation axis for three successive elementary rotations. See “Rotation Sequence Measurements” for more information.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Rotation Sequence`.

Angles — Angles for rotation sequence parameterization
`[0 0 0] deg` (default) | 1-by-3 vector

Angles for the rotation sequence parameterization, specified as a 1-by-3 vector. See “Rotation Sequence Measurements” for more information.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Rotation Sequence`.

Rotation Matrix — Relative rotation using rotation matrix
`[1 0 0; 0 1 0; 0 0 1]` (default) | 3-by-3 matrix

Relative rotation, specified as a 3-by-3 matrix that maps vectors from the follower frame to the base frame. The matrix must be orthogonal and have determinant 1. See “Rotational Measurements” for more information.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Rotation Matrix`.

Quaternion — Relative rotation using quaternion
`[1 0 0 0]` (default) | unit quaternion vector

Relative rotation, specified as a unit quaternion vector. See “Rotational Measurements” for more information about the quaternion.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Quaternion`.

Specify Velocity Target — Whether to specify angular velocity target
`off` (default) | `on`

Select this parameter to specify the angular velocity target for the spherical primitive.

Priority — Priority level of angular velocity target
`High (desired)` (default) | `Low (approximate)`

Priority level of the angular velocity target, specified as `High (desired)` or `Low (approximate)`. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Value — Angular velocity target of spherical primitive
`[0 0 0] deg/s` (default) | 1-by-3 vector

Angular velocity target for the spherical primitive, specified as a 1-by-3 vector resoved in resolution frame.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Resolution Frame — Frame used to resolve specified angular velocity target
`Base` (default) | `Follower`

Frame used to resolve the specified angular velocity target, specified as one of these:

- `Base` — The joint block resolves the angular velocity target in the coordinates of the base frame.
- `Follower` — The joint block resolves the angular velocity target in the coordinates of the follower frame.

Internal Mechanics

Equilibrium Position > Method — Method to specify equilibrium frame
`None` (default) | `Aligned Axes` | `Standard Axis` | `Arbitrary Axis` | `Rotation Sequence` | `Rotation Matrix` | `Quaternion`

Method to use to specify the equilibrium frame with respect to the base frame. The equilibrium frame is fixed during the simulation. If the *z*-axes of the follower and equilibrium frames are aligned, the spring torque of the spherical primitive is zero.

Set the **Equilibrium Position > Method** parameter to `None` to let the equilibrium and base frames be coincident.

Pair 1: Follower — Equilibrium frame axis used to align with specified base frame axis
`+X` (default) | `-X` | `+Y` | `-Y` | `+Z` | `-Z`

Equilibrium frame axis used to align with the base frame axis set by the **Pair 1: Base** parameter, specified as an orthogonal axis of the equilibrium frame. The equilibrium frame rotates with respect to the base frame to enable the alignment between the selected axes of the base and equilibrium frames.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Aligned Axis`.

Pair 1: Base — Base frame axis
+Y (default) | +X | -X | -Y | +Z | -Z

Base frame axis to align with the equilibrium frame specified by the **Pair 1: Follower**, specified as an orthogonal axis of the base frame.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Aligned Axis`.

Pair 2: Follower — Equilibrium frame axis used to align with specified base frame axis
+Y | +X | -X | -Y | +Z | -Z

Base frame axis to align with the equilibrium frame specified by the **Pair 2: Follower**, specified as an orthogonal axis of the equilibrium frame. The equilibrium frame rotates with respect to the base frame to enable the alignment between the selected axes of the base and equilibrium frames.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Aligned Axis`.

Pair2: Base — Base frame axis
+Z (default) | +X | -X | +Y | -Y | -Z

Base frame axis used to let the equilibrium frame axis set in the **Pair2: Follower** parameter to align with, specified as an orthogonal axis of the base frame. The axis choices for **Pair 2** depend on the **Pair 1** axis selections.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Aligned Axis`.

Axis — Standard axis of relative rotation
+Z (default) | +X | -X | +Y | -Y | -Z

Axis of the relative rotation, specified as an orthogonal axis of the base frame.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Standard Axis`.

Angle — Angle of relative rotation
0.0 `deg` (default) | scalar

Angle of the relative rotation, specified as a scalar. The angle indicates the rotation of the equilibrium frame with respect to the base frame about the specified axis.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Standard Axis`.

Axis — Axis of relative rotation
[0 0 1] (default) | 3-by1 vector

Axis of the relative rotation, specified as a 3-by-1 unit vector. The vector is dimensionless and indicates the rotational axis resolved in the base frame.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Arbitrary Axis`.

Angle — Angle of relative rotation
`0.0 deg` (default) | scalar

Angle of the relative rotation, specified as a scalar. The angle indicates the rotation of the equilibrium frame with respect to the base frame about the axis specified by the **Axis** parameter.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Arbitrary Axis`.

Rotation About — Frame whose axes to rotate equilibrium frame about
`Follower Axes` (default) | `Base Axes`

Frame whose axes to rotate the equilibrium frame about, specified as `Follower Axes` or `Base Axes`. If you set the parameter to `Follower Axes`, the equilibrium frame rotates about its own axes, and the equilibrium frame changes the orientation with each successive rotation. If you set the parameter to `Base Axes`, the equilibrium frame rotates about the fixed base frame axes. See "Rotational Measurements" for more information.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Rotation Sequence`.

Sequence — Sequence of rotation axis
`X-Y-X` (default) | `X-Y-Z` | `X-Z-X` | `X-Z-Y` | `Y-X-Y` | `Y-X-Z` | `Y-Z-X` | `Y-Z-Y` | `Z-X-Y` | `Z-X-Z` | `Z-Y-X` | `Z-Y-Z`

Sequence of the rotation axis for three successive elementary rotations. See "Rotation Sequence Measurements" for more information.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Rotation Sequence`.

Angles — Angles for elementary rotations
`[0 0 0] deg` (default) | 1-by-3 vector

Angles for elementary rotations, specified as a 1-by-3 vector. See "Rotation Sequence Measurements" for more information.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Rotation Sequence`.

Matrix — Relative rotation using rotation matrix
`[1 0 0; 0 1 0; 0 0 1]` (default) | 3-by-3 matrix

Relative rotation, specified as a 3-by-3 matrix. The matrix must be orthogonal and have determinant 1. See "Rotational Measurements" for more information.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Rotation Matrix`.

Quaternion — Relative rotation using quaternion
`[1 0 0 0]` (default) | unit quaternion vector

Relative rotation, specified as a unit quaternion vector. See “Rotational Measurements” for more information about the quaternion.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Quaternion`.

Spring Stiffness — Stiffness of force law
`0 N*m/deg` (default) | scalar

Stiffness of the internal spring-damper force law for the spherical primitive, specified as a scalar with a unit of rotational stiffness.

The spring attempts to pull the follower frame so that the follower frame is aligned with the specified equilibrium fame.

Damping Coefficient — Damping coefficient of force law
`0 N*m/(deg/s)` (default) | scalar

Damping coefficient of the internal spring-damper force law for the spherical primitive, specified as a scalar with a unit of rotational damping coefficient.

Limits

Specify Lower Limit — Whether to specify lower position limit
`off` (default) | `on`

Select this parameter to specify the lower limit of the spherical primitive. Joint limits use spring-dampers to resist travel past the bounds of the range.

Bound — Lower bound of free region
`15 deg` (default) | scalar

Lower bound for the free region of the spherical primitive, specified as a scalar with a unit of angle.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Spring Stiffness — Stiffness of spring at lower bound
`1e4 N*m/deg` (default) | scalar

Stiffness of the spring at lower bound, specified as a scalar with a unit of rotational stiffness.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Damping Coefficient — Damping coefficient at lower bound
`10 N*m/(deg/s)` (default) | scalar

Damping coefficient at lower bound, specified as a scalar with a unit of rotational damping coefficient.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Transition Region Width — Region to smooth spring and damper torque
`0.1 deg` (default) | scalar

Region to smooth the spring and damper torques, specified as a scalar with a unit of angle.

The block applies the full value of the lower-limit torque when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of torques 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.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Specify Upper Limit — Whether to specify upper position limit
`off` (default) | `on`

Select this parameter to specify the upper limit of the spherical primitive. Joint limits use spring-dampers to resist travel past the bounds of the range.

Bound — Upper bound of free region
`45 deg` (default) | scalar

Upper bound for the free region of the spherical primitive, specified as a scalar with a unit of angle.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Spring Stiffness — Stiffness of spring at upper bound
`1e4 N*m/deg` (default) | scalar

Stiffness of the spring at upper bound, specified as a scalar with a unit of stiffness.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Damping Coefficient — Damping coefficient at upper bound
`10 N*m/(deg/s)` (default) | scalar

Damping coefficient at upper bound, specified as a scalar with a unit of damping coefficient.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Transition Region Width — Region to smooth spring and damper torques
`0.1 deg` (default) | scalar

Region to smooth the spring and damper torques, specified as a scalar with a unit of angle.

The block applies the full value of the upper-limit torque when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of torques 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.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Actuation

Torque — Option to provide actuation torque
`None` (default) | `Provided by Input`

Option to provide the actuation torque for the spherical primitive, specified as one of these values.

Actuation Torque Setting	Description
`None`	Apply no actuation torque.
`Provided by Input`	Apply actuation torques based on physical signals. The signal specifies the torque acting on the follower frame with respect to the base frame. The signal provides the value of the torque applied equally and oppositely to the base and follower frames. Selecting this option exposes additional parameters that you can use to enable input ports See the **Input** section for details.

Resolution Frame — Frame used to resolve actuation torque
`Base` (default) | `Follower`

Frame used to resolve the input actuation torques, specified as `Base` or `Follower`.

Sensing

Resolution Frame — Frame used to resolve sensing outputs
`Base` (default) | `Follower`

Frame used to resolve the sensing output signals, specified as `Base` or `Follower`. For more information about the output signals, see the **Output** section.

Mode Configuration

Mode — Joint mode
`Normal` (default) | `Disengaged` | `Provided by Input`

Joint mode for the simulation, specified as one of these values:

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.

Method	Description
`Provided by Input`	The port **mode** specifies whether the joint behaves normally or is disengaged.

Composite Force/Torque Sensing

Direction — Measurement direction
`Follower on Base` (default) | `Base on Follower`

Measurement direction, specified as one of these values:

- `Follower on Base` — Sense the force and torque that the follower frame exerts on the base frame.
- `Base on Follower` — Sense the force and torque that the base frame exerts on the follower frame.

Note that this parameter only affects the output signals under the **Composite Force/Torque Sensing** section. Reversing the direction changes the sign of the measurements. For more information see “Force and Torque Measurement Direction”.

Resolution Frame — Frame used to resolve measurements
`Base` (default) | `Follower`

Frame used to resolve the measurements, specified as one of these values:

- `Base` — The joint block resolves the measurements in the coordinates of the base frame.
- `Follower` — The joint block resolves the measurements in the coordinates of the follower frame.

Note that this parameter only affects the output signals under the **Composite Force/Torque Sensing** section.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation
Generate C and C++ code using Simulink® Coder™.

See Also

`simscape.multibody.SphericalJoint` | `simscape.multibody.SphericalPrimitive` | `simscape.multibody.SphericalSpringDamper`

Topics
“Assemble Bodies Using Joints and Constraints”
“Modeling Joint Connections”
“Motion Sensing”
“Rotational Measurements”
“Translational Measurements”

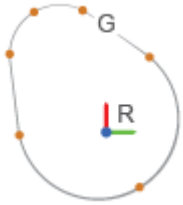
Spline

Cubic interpolating plane curve or space curve

Libraries:
Simscape / Multibody / Curves and Surfaces

Description

This Spline block creates a continuous spline curve based on cubic interpolation between the specified points. The curve can be two dimensional, such as a planar cam profile, or three dimensional, such as a roller coaster track. The dimension of the spline depends on the dimension of the coordinate matrix. An *N*-by-2 matrix specifies a 2-D curve in the *xy* plane. An *N*-by-3 matrix specifies a 3-D curve. All the coordinates are resolved in the local reference frame of the block. Moreover, according to the specified end conditions, the curve can be either open or closed.



Ports

Frame

R — Reference frame
frame

Spline curve reference frame. Connect this frame port to another block to specify the location and orientation of the spline curve in a model.

Geometry

G — Spline curve representation
geometry

Geometry data associated with the representation of a 2-D or 3-D continuous spline curve. It provides the spline curve specification to other blocks to which it connects, such as the Point on Curve Constraint block.

Parameters

Interpolation Points — Coordinates of interpolation points for specifying spline curve
[6 -2 9; -6 5 -4; -6 -7 9; 6 0 -4; -3 6 9; -3 -8 -4] m (default) | *N*-by-2 matrix | *N*-by-3 matrix

Matrix that includes the coordinates of the interpolation points for defining the spline curve. Use an *N*-by-2 matrix to specify a 2-D spline and an *N*-by-2 matrix to specify a 3-D spline. Each row of the matrix specifies the Cartesian coordinates of an interpolation point with respect to the reference frame of the Spline block. An error occurs if the matrix has any repeated rows.

Tips

You can use the `unique` function to remove repeated rows from an input matrix.

End Conditions — Treatment of the curve endpoints
`Periodic (Closed)` (default) | `Natural (Open)`

End conditions of the spline curve. The `Periodic (Closed)` end conditions correspond to a closed curve. For this condition, the block joins the first and last data points with a continuous curve. The `Natural (Open)` end condition corresponds to an open curve.

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 endpoints.

Graphic

Type — Spline visualization setting
`From Geometry` (default) | `Marker` | `None`

Visualization setting for this spline. Use the default setting, `From Geometry`, to show the spline. Select `Marker` to show a graphic marker such as a sphere or frame. Select `None` to disable visualization for this spline.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Diffuse Color — True color as [R,G,B,A]
[0.0 0.0 0.0] (default) | three-element vector | four-element vector

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 on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Shape — Shape of graphic marker
`Sphere` (default) | `Cube` | `Frame`

Geometrical shape of the graphic marker. The Mechanics Explorer shows the marker using the selected shape.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
10 `pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Version History

Introduced in R2015b

Extended Capabilities

C/C++ Code Generation

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

See Also

Point on Curve Constraint

Topics
“Constrain a Point to a Curve”
“Measure Joint Constraint Forces”
“Study the Joints and Constraints to Model”
“Assemble Bodies Using Joints and Constraints”

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**.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

External Force and Torque | Internal Force | Inverse Square Law Force

Topics

“Actuating and Sensing with Physical Signals”
“Specifying Variable Inertias”

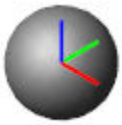
Spherical Solid

Solid spherical element with geometry, inertia, and color

Libraries:
Simscape / Multibody / Body Elements

Description

The Spherical Solid block is a spherical shape with geometry center coincident with the reference frame origin.



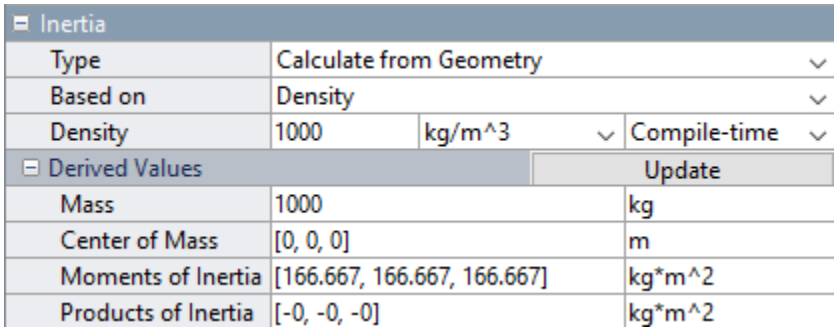
The Spherical Solid block adds to the attached frame a solid element with geometry, inertia, and color. The solid element can be a simple rigid body or part of a compound rigid body—a group of rigidly connected solids, often separated in space through rigid transformations. Combine Spherical Solid and other solid blocks with the 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.

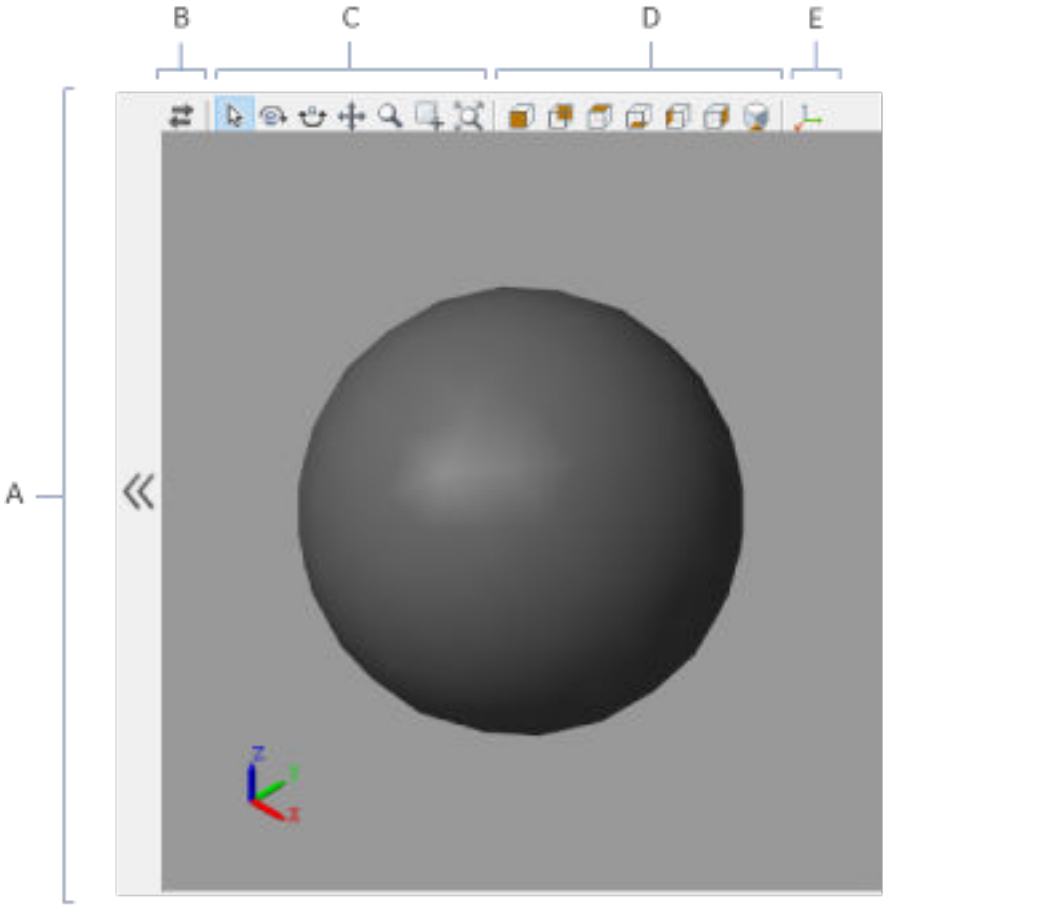


Derived Values Display

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. 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”.

Geometry

G — Geometry
geometry

Geometry that represents the solid. Connect this port to a Spatial Contact Force block to model contacts on the solid.

Dependencies

To enable this port, under **Geometry**, expand **Export** and select **Entire Geometry**.

Parameters

Geometry

Radius — Radius of the sphere
`1 m` (default) | scalar with units of length

Distance between the center of the sphere and its surface.

Entire Geometry — Export the true geometry of the block
`off` (default) | `on`

Select **Entire Geometry** to export the true geometry of the Spherical Solid block which can be used for other blocks, such as the Spatial Contact Force block.

Dependencies

To enable this option, select **Entire Geometry** under the **Export**.

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 if known.

Density — Mass per unit volume of material
`1000 kg/m^3` (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*m^2` (default) | three-element vector with units of mass*length^2

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:

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

Custom: Products of Inertia — Off-diagonal elements of inertia tensor
`[0 0 0] kg*m^2` (default) | three-element vector with units of mass*length^2

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:

- $I_{yz} = -\int_m yz\,dm$
- $I_{zx} = -\int_m zx\,dm$
- $I_{xy} = -\int_m xy\,dm$

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

Type of the visual representation of the solid, specified as `From Geometry`, `Marker`, or `None`. Set the parameter to `From Geometry` to show the visual representation of the solid. Set the parameter to `Marker` to represent the solid as a marker. Set the parameter to `None` to hide the solid in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Shape — Shape of marker to represent 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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
`10 pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the graphic 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 (A) specifies the color opacity on a scale of 0–1. Omitting the opacity element is equivalent to specifying a value of 1.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered solid and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`

2 **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Frames

Show Port R — Show reference frame port for connection to other blocks
on (default) | off

Select to expose the **R** port.

New Frame — Create custom frame for connection to other blocks
button

Click the Create button  to open a pane for creating a new body-attached frame. In this pane, you can specify the name, origin, and orientation for the frame.

- To name the custom frame, click the text field of the **Frame Name** parameter. The name identifies the corresponding port on the solid block and in the tree view pane of the Mechanics Explorer.
- To select the **Frame Origin** of the custom frame, use one of the following methods:
  - **At Reference Frame Origin**: Make the new frame origin coincident with the origin of the reference frame of the solid.
  - **At Center of Mass**: Make the new frame origin coincident with the center of mass of the solid.
  - **Based on Geometric Feature**: Make the new frame origin coincident with the center of the selected feature. Valid features include surfaces, lines, and points. Select a feature from the visualization pane, then click **Use Selected Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.
- To define the orientation of the custom frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the custom frame and then specify their directions.

  Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

  - **Along Reference Frame Axis**: Selects an axis of the reference frame of the solid.
  - **Along Principal Inertia Axis**: Selects an axis of the principal inertia axis of the solid.
  - **Based on Geometric Feature**: Selects the vector associated with the chosen geometry feature of the solid. Valid features include surfaces and lines. The corresponding vector is indicated by a white arrow in the visualization pane. You can select a feature from the

visualization pane and then click **Use Selected Feature** to confirm the selection. The name of the selected feature appears in the field below this option.

FrameN — Edit or delete existing custom frame
frame name

Frames that you have created. `N` is a unique identifying number for each custom frame.

- Click the text field to edit the name of an existing custom frame.
- Click the Edit button  to edit other aspects of the custom frame, such as origin and axes.
- Click the Delete button  to delete the custom frame.

Dependencies

To enable this parameter, create a frame by clicking **New Frame**.

Version History

Introduced in R2019b

Extended Capabilities

C/C++ Code Generation

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

See Also

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

Topics

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

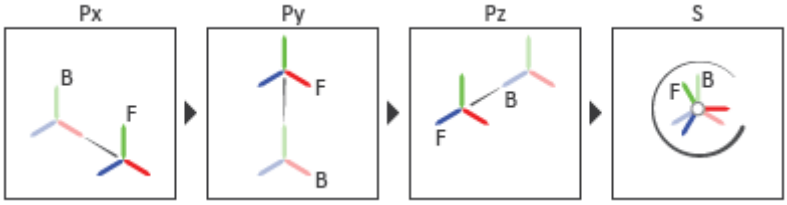
6-DOF Joint

Joint with six degrees of freedom and no kinematic singularity

Libraries:
Simscape / Multibody / Joints

Description

The 6-DOF Joint block provides three translational and three rotational degrees of freedom. The follower frame can have a 3-D transformation with respect to the base frame. The transformation contains three sequential translations and a 3-D rotation encoded as a quaternion. Quaternions do not have a kinematic singularity.



The translations are along the *x*, *y*, and *z* axes of the base frame, respectively. Before the rotation, the axes of the follower are parallel to the corresponding axes of the base frame. The 3-D rotation is with respect to the follower frame formed after the translations.

Ports

Frame

B — Base frame
frame

Base frame of the joint block.

F — Follower frame
frame

Follower frame of the joint block.

Input

X Prismatic Primitive (Px)

fx — Actuation force
physical signal

Physical signal input port that accepts the actuation force for the joint primitive. The block applies the force equally and oppositely to the base and follower frames of the joint along the *x*-axis of the base frame.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Actuation**, set **Force** to `Provided by Input`.

px — Motion profile
physical signal

Physical signal input port that accepts the motion profile for the joint primitive. The signal provides the displacement of the follower frame with respect to the base frame along the *x*-axis of the base frame. The signal must also contain the first and second derivatives of the displacement.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Actuation**, set **Motion** to `Provided by Input`.

Y Prismatic Primitive (Py)

fy — Actuation force
physical signal

Physical signal input port that accepts the actuation force for the joint primitive. The block applies this force equally and oppositely to the base and follower frames of the joint along the *y*-axis of the base frame.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Actuation**, set **Force** to `Provided by Input`.

py — Motion profile
physical signal

Physical signal input port that accepts the motion profile for the joint primitive. The signal provides the displacement of the follower frame with respect to the base frame along the *y*-axis of the base frame. The signal must also contain the first and second derivatives of the displacement.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Actuation**, set **Motion** to `Provided by Input`.

Z Prismatic Primitive (Pz)

pz — Position of primitive
physical signal

Physical signal port that outputs the position of the joint primitive. The value is the displacement of the follower frame with respect to the base frame in the *z*-direction of the base frame.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Position**.

vz — Velocity of primitive
physical signal

Physical signal port that outputs the velocity of the joint primitive. The value is the first derivative of the signal from the port **pz**.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Velocity**.

az — Acceleration of primitive
physical signal

Physical signal port that outputs the acceleration of the joint primitive. The value is the second derivative of the signal from the port **pz**.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Acceleration**.

fz — Actuator force acting on joint primitive
physical signal

Physical signal port that outputs the actuator force acting on the joint primitive.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Actuator Force**.

fllz — Lower-limit force
physical signal

Physical signal port that outputs the lower-limit force. The block applies this force when the joint primitive position is less than the lower bound of the free region. The block applies this force to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Lower-Limit Force**.

fulz — Upper-limit force
physical signal

Physical signal port that outputs the upper-limit force. The block applies this force when the joint primitive position exceeds the upper bound of the free region. The block applies this force to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Upper-Limit Force**.

Spherical Primitive (S)

tx — Actuation torque about *x*-axis of resolution frame
physical signal

Physical signal input port that accepts the actuation torque for the spherical primitive. The signal has a scalar format that represents the torque about the *x*-axis of the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Actuation**, set **Torque** to `Provided by Input` and select **Torque (X)**.

ty — Actuation torque about *y*-axis of resolution frame
physical signal

Physical signal input port that accepts the actuation torque for the spherical primitive. The signal has a scalar format that represents the torque about the *y*-axis of the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Actuation**, set **Torque** to `Provided by Input` and select **Torque (Y)**..

tz — Actuation torque about *z*-axis of resolution frame
physical signal

Physical signal input port that accepts the actuation torque for the spherical primitive. The signal has a scalar format that represents the torque about the *z*-axis of the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Actuation**, set **Torque** to `Provided by Input` and select **Torque (Z)**..

t — Actuation torque vector
physical signal

Physical signal input port that accepts the actuation torque for the spherical primitive. The signal has a vector format that represents the torque about an arbitrary axis [*x y z*] specified in the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S)** > **Actuation**, set **Motion** to `Provided by Input` and select **Torque (XYZ)**..

Mode Configuration

mode — Joint mode control
scalar

Input port that controls the mode of the joint. The signal is a unitless scalar. The joint operates in normal mode when the input signal is `0` and operates in disengaged mode when the input signal is `-1`. You can change between the two modes at any time during simulation.

Dependencies

To enable this port, under **Mode Configuration**, set **Mode** to `Provided by Input`.

Output

X Prismatic Primitive (Px)

px — Position of primitive
physical signal

Physical signal port that outputs the position of the joint primitive. The value is the displacement of the follower frame with respect to the base frame in the *x*-direction of the base frame.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Sensing**, select **Position**.

vx — Velocity of primitive
physical signal

Physical signal port that outputs the velocity of the joint primitive. The value is the first derivative of the signal from the port **px**.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Sensing**, select **Velocity**.

ax — Acceleration of primitive
physical signal

Physical signal port that outputs the acceleration of the joint primitive. The value is the second derivative of the signal from the port **px**.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Sensing**, select **Acceleration**.

fx — Actuator force acting on joint primitive
physical signal

Physical signal port that outputs the actuator force acting on the joint primitive.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Sensing**, select **Actuator Force**.

fllx — Lower-limit force
physical signal

Physical signal port that outputs the lower-limit force. The block applies this force when the joint primitive position is less than the lower bound of the free region. The block applies this force to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Sensing**, select **Lower-Limit Force**.

fulx — Upper-limit force
physical signal

Physical signal port that outputs the upper-limit force. The block applies this force when the joint primitive position exceeds the upper bound of the free region. The block applies this force to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **X Prismatic Primitive (Px) > Sensing**, select **Upper-Limit Force**.

Y Prismatic Primitive (Py)

py — Position of primitive
physical signal

Physical signal port that outputs the position of the joint primitive. The value is the displacement of the follower frame with respect to the base frame in the *y*-direction of the base frame.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Sensing**, select **Position**.

vy — Velocity of primitive
physical signal

Physical signal port that outputs the velocity of the joint primitive. The value is the first derivative of the signal from the port **py**.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Sensing**, select **Velocity**.

ay — Acceleration of primitive
physical signal

Physical signal port that outputs the acceleration of the joint primitive. The value is the second derivative of the signal from the port **py**.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Sensing**, select **Acceleration**.

fy — Actuator force acting on joint primitive
physical signal

Physical signal port that outputs the actuator force acting on the joint primitive.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Sensing**, select **Actuator Force**.

fly — Lower-limit force
physical signal

Physical signal port that outputs the lower-limit force. The block applies this force when the joint primitive position is less than the lower bound of the free region. The block applies this force to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Sensing**, select **Lower-Limit Force**.

fuly — Upper-limit force
physical signal

Physical signal port that outputs the upper-limit force. The block applies this force when the joint primitive position exceeds the upper bound of the free region. The block applies this force to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **Y Prismatic Primitive (Py) > Sensing**, select **Upper-Limit Force**.

Z Prismatic Primitive (Pz)

pz — Position of primitive
physical signal

Physical signal port that outputs the position of the joint primitive. The value is the displacement of the follower frame with respect to the base frame in the *z*-direction of the base frame.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Position**.

vz — Velocity of primitive
physical signal

Physical signal port that outputs the velocity of the joint primitive. The value is the first derivative of the signal from the port **pz**.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Velocity**.

az — Acceleration of primitive
physical signal

Physical signal port that outputs the acceleration of the joint primitive. The value is the second derivative of the signal from the port **pz**.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Acceleration**.

fz — Actuator force acting on joint primitive
physical signal

Physical signal port that outputs the actuator force acting on the joint primitive.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Actuator Force**.

fllz — Lower-limit force
physical signal

Physical signal port that outputs the lower-limit force. The block applies this force when the joint primitive position is less than the lower bound of the free region. The block applies this force to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Lower-Limit Force**.

fulz — Upper-limit force
physical signal

Physical signal port that outputs the upper-limit force. The block applies this force when the joint primitive position exceeds the upper bound of the free region. The block applies this force to both the base and follower frames of the joint primitive in order to accelerate the relative position back to the free region.

Dependencies

To enable this port, under **Z Prismatic Primitive (Pz) > Sensing**, select **Upper-Limit Force**.

Spherical Primitive (S)

Q — Relative orientation in quaternion parameterization
physical signal

Orientation of the follower frame with respect to the base frame, returned as a vector in quaternion parameterization in the resolution frame. See “Quaternion Measurements” for more information.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Position**.

wx — *X*-coordinate of relative angular velocity
physical signal

X-coordinate of the relative angular velocity, returned as a scalar. The value is resolved in the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Velocity (X)**.

wy — *Y*-coordinate of relative angular velocity
physical signal

Y-coordinate of the relative angular velocity, returned as a scalar. The value is resolved in the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Velocity (Y)**.

wz — *Z*-coordinate of relative angular velocity
physical signal

Z-coordinate of the relative angular velocity, returned as a scalar. The value is resolved in the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Velocity (Z)**.

w — Angular velocity vector
physical signal

Relative angular velocity, returned as a 3-D vector resolved in the resolution frame.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Velocity**.

bx — *X*-coordinate of relative angular acceleration
physical signal

X-coordinate of the relative angular acceleration, returned as a scalar. This quantity equals the time derivative of the signal exported from the port **wx**.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Acceleration (X)**.

by — *Y*-coordinate of relative angular acceleration
physical signal

Y-coordinate of the relative angular acceleration, returned as a scalar. This quantity equals the time derivative of the signal exported from the port **wy**.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Acceleration (Y)**.

bz — *Z*-coordinate of relative angular acceleration
physical signal

Z-coordinate of the relative angular acceleration, returned as a scalar. This quantity equals the time derivative of the signal exported from the port **wz**.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Acceleration (Z)**.

b — Angular acceleration vector
physical signal

Relative angular acceleration, returned as a 3-D vector resolved in the resolution frame. This quantity equals the time derivative of the signal exported from the port **w**

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Acceleration**.

tll — Lower-limit torque magnitude
physical signal

Physical signal port that outputs the lower-limit torque. The block applies the torque when the joint primitive position is less than the lower bound of the free region. The torque applies to both the base and follower frames of the joint primitive to accelerate the position back to the free region.

Dependencies

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Signed Lower-Limit Torque Magnitude**.

tul — Upper-limit torque magnitude
physical signal

Physical signal port that outputs the upper-limit torque. The block applies the torque when the joint primitive position exceeds the upper bound of the free region. The torque applies to both the base and follower frames of the joint primitive to accelerate the position back to the free region.

Dependencies

To enable this port, under

To enable this port, under **Spherical Primitive (S) > Sensing**, select **Upper-Limit Torque**.

Composite Force/Torque Sensing

fc — Constraint force
physical signal

Physical signal port that outputs the constraint force that acts across the joint. The force maintains the translational constraints of the joint. For more information, see “Measure Joint Constraint Forces”.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Constraint Force**.

tc — Constraint torque
physical signal

Physical signal port that outputs the constraint torque that acts across the joint. The torque maintains the rotational constraints of the joint. For more information, see “Force and Torque Sensing”.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Constraint Torque**.

ft — Total force
physical signal

Physical signal port that outputs the total force that acts across the joint. The total force is the sum of the forces transmitted from one frame to the other through the joint. The force includes the actuation, internal, limit, and constraint forces. See “Force and Torque Sensing” for more information.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Total Force**.

tt — Total torque
physical signal

Physical signal port that outputs the total torque that acts across the joint. The total torque is the sum of the torques transmitted from one frame to the other through the joint. The torque includes the actuation, internal, limit, and constraint torques. For more information, see “Force and Torque Sensing”.

Dependencies

To enable this port, under **Composite Force/Torque Sensing**, select **Total Torque**.

Parameters

X Prismatic Primitive (Px)

State Targets

Specify Position Target — Whether to specify position target
`off` (default) | `on`

Select this parameter to specify the position target for the *x* prismatic primitive.

Priority — Priority level of position target
`High (desired)` (default) | `Low (approximate)`

Priority level of the position target, specified as `High (desired)` or `Low (approximate)`. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, select **Specify Position Target**.

Value — Position target
`0 m` (default) | scalar

Position target of the *x* prismatic primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Position Target**.

Specify Velocity Target — Whether to specify linear velocity target
`off` (default) | `on`

Select this parameter to specify the linear velocity target for the *x* prismatic primitive.

Priority — Priority level of linear velocity target
`High (desired)` (default) | `Low (approximate)`

Priority level of the linear velocity target, specified as `High (desired)` or `Low (approximate)`. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Value — Velocity target of
`0 m/s` (default) | scalar

Linear velocity target for the *x* prismatic primitive, specified as a scalar.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Internal Mechanics

Equilibrium Position — Position where internal force is zero
`0 m` (default) | scalar

Position where the spring force is zero, specified as a scalar in units of length.

Spring Stiffness — Stiffness of force law
`0 N/m` (default) | scalar

Stiffness of the internal spring-damper force law for the joint primitive, specified as a scalar in units of linear stiffness.

Damping Coefficient — Damping coefficient of force law
`0 N(m/s)` (default) | scalar

Damping coefficient of the internal spring-damper force law for the joint primitive, specified as a scalar in units of linear damping coefficient.

Limits

Specify Lower Limit — Whether to specify lower position limit
`off` (default) | `on`

Select this parameter to specify the lower limit of the *x* prismatic primitive.

Bound — Lower bound of free region
`-1 m` (default) | scalar

Lower bound of the free region of the *x* prismatic primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Spring Stiffness — Stiffness of spring at lower bound
`1e6 N/m` (default) | scalar

Stiffness of the spring at the lower bound, specified as a scalar in units of linear stiffness.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Damping Coefficient — Damping coefficient at lower bound
`1e3 N/(m/s)` (default) | scalar

Damping coefficient at the lower bound, specified as a scalar in units of linear damping coefficient.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Transition Region Width — Region to smooth spring and damper forces
1e-4 `m` (default) | scalar

Region to smooth the spring and damper forces, specified as a scalar in units of length.

The block applies the full value of the lower-limit force when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Specify Upper Limit — Whether to specify upper position limit
`off` (default) | `on`

Select this parameter to specify the upper limit of the *x* prismatic primitive..

Bound — Upper bound of free region
1 `m` (default) | scalar

Upper bound for the free region of the joint primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Spring Stiffness — Stiffness of spring at upper bound
1e6 `N/m` (default) | scalar

Stiffness of the spring at the upper bound, specified as a scalar in units of linear stiffness.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Damping Coefficient — Damping coefficient at upper bound
1e3 `N/(m/s)` (default) | scalar

Damping coefficient at the upper bound, specified as a scalar in units of linear damping coefficient.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Transition Region Width — Region to smooth spring and damper forces
1e-4 `m` (default) | scalar

Region to smooth the spring and damper forces, specified as a scalar in units of length.

The block applies the full value of the upper-limit force when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Actuation

Force — Option to provide actuator force
`None` (default) | `Provided by Input` | `Automatically Computed`

Option to provide the actuator force for the joint primitive, specified as one of these values:

Actuation Force Setting	Description
`None`	No actuator force.
`Provided by Input`	Input port **fx** specifies the actuator force for the *x* prismatic primitive.
`Automatically Computed`	The block automatically calculates the amount of force required to satisfy the motion inputs to the mechanism. If you set this parameter to `Automatically Computed`, you do not need to set **Motion** to `Provided by Input` for the same joint primitive. The automatically computed force may to satisfy a motion input elsewhere in the mechanism.

Motion — Option to provide motion
`Automatically Computed` (default) | `Provided by Input`

Option to provide the motion for the joint primitive, specified as one of these values:

Actuation Motion Setting	Description
`Automatically Computed`	The block computes and applies the joint primitive motion based on model dynamics.
`Provided by Input`	Input port **px** specifies the motion for the joint primitive.

Y Prismatic Primitive (Py)

State Targets

Specify Position Target — Whether to specify position target
`off` (default) | `on`

Select this parameter to specify the position target for the *y* prismatic primitive.

Priority — Priority level of position target
`High (desired)` (default) | `Low (approximate)`

Priority level of the position target, specified as `High (desired)` or `Low (approximate)`. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, select **Specify Position Target**.

Value — Position target
`0 m` (default) | scalar

Position target of the *y* prismatic primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Position Target**.

Specify Velocity Target — Whether to specify linear velocity target
`off` (default) | `on`

Select this parameter to specify the linear velocity target for the *y* prismatic primitive.

Priority — Priority level of linear velocity target
`High (desired)` (default) | `Low (approximate)`

Priority level of the linear velocity target, specified as `High (desired)` or `Low (approximate)`. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Value — Velocity target of
`0 m/s` (default) | scalar

Linear velocity target for the *y* prismatic primitive, specified as a scalar.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Internal Mechanics

Equilibrium Position — Position where internal force is zero
`0 m` (default) | scalar

Position where the spring force is zero, specified as a scalar in units of length.

Spring Stiffness — Stiffness of force law
`0 N/m` (default) | scalar

Stiffness of the internal spring-damper force law for the joint primitive, specified as a scalar in units of linear stiffness.

Damping Coefficient — Damping coefficient of force law
`0 N(m/s)` (default) | scalar

Damping coefficient of the internal spring-damper force law for the joint primitive, specified as a scalar in units of linear damping coefficient.

Limits

Specify Lower Limit — Whether to specify lower position limit
`off` (default) | `on`

Select this parameter to specify the lower limit of the *y* prismatic primitive.

Bound — Lower bound of free region
`-1 m` (default) | scalar

Lower bound of the free region of the *y* prismatic primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Spring Stiffness — Stiffness of spring at lower bound
`1e6 N/m` (default) | scalar

Stiffness of the spring at the lower bound, specified as a scalar in units of linear stiffness.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Damping Coefficient — Damping coefficient at lower bound
`1e3 N/(m/s)` (default) | scalar

Damping coefficient at the lower bound, specified as a scalar in units of linear damping coefficient.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Transition Region Width — Region to smooth spring and damper forces
`1e-4 m` (default) | scalar

Region to smooth the spring and damper forces, specified as a scalar in units of length.

The block applies the full value of the lower-limit force when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Specify Upper Limit — Whether to specify upper position limit
`off` (default) | `on`

Select this parameter to specify the upper limit of the *y* prismatic primitive.

Bound — Upper bound of free region
`1 m` (default) | scalar

Upper bound for the free region of the joint primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Spring Stiffness — Stiffness of spring at upper bound
1e6 `N/m` (default) | scalar

Stiffness of the spring at the upper bound, specified as a scalar in units of linear stiffness.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Damping Coefficient — Damping coefficient at upper bound
1e3 `N/(m/s)` (default) | scalar

Damping coefficient at the upper bound, specified as a scalar in units of linear damping coefficient.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Transition Region Width — Region to smooth spring and damper forces
1e-4 `m` (default) | scalar

Region to smooth the spring and damper forces, specified as a scalar in units of length.

The block applies the full value of the upper-limit force when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Actuation

Force — Option to provide actuator force
`None` (default) | `Provided by Input` | `Automatically Computed`

Option to provide the actuator force for the joint primitive, specified as one of these values:

Actuation Force Setting	Description
`None`	No actuator force.
`Provided by Input`	Input port **fy** specifies the actuator force for the *y* prismatic primitive.
`Automatically Computed`	The block automatically calculates the amount of force required to satisfy the motion inputs to the mechanism. If you set this parameter to `Automatically Computed`, you do not need to set **Motion** to `Provided by Input` for the same joint primitive. The automatically computed force may to satisfy a motion input elsewhere in the mechanism.

Motion — Option to provide motion
`Automatically Computed` (default) | `Provided by Input`

Option to provide the motion for the joint primitive, specified as one of these values:

Actuation Motion Setting	Description
`Automatically Computed`	The block computes and applies the joint primitive motion based on model dynamics.
`Provided by Input`	Input port **py** specifies the motion for the joint primitive.

Z Prismatic Primitive (Pz)

State Targets

Specify Position Target — Whether to specify position target
`off` (default) | `on`

Select this parameter to specify the position target for the *z* prismatic primitive.

Priority — Priority level of position target
`High (desired)` (default) | `Low (approximate)`

Priority level of the position target, specified as `High (desired)` or `Low (approximate)`. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, select **Specify Position Target**.

Value — Position target
`0 m` (default) | scalar

Position target of the *z* prismatic primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Position Target**.

Specify Velocity Target — Whether to specify linear velocity target
`off` (default) | `on`

Select this parameter to specify the linear velocity target for the *z* prismatic primitive.

Priority — Priority level of linear velocity target
`High (desired)` (default) | `Low (approximate)`

Priority level of the linear velocity target, specified as `High (desired)` or `Low (approximate)`. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Value — Velocity target of
`0 m/s` (default) | scalar

Linear velocity target for the *z* prismatic primitive, specified as a scalar.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Internal Mechanics

Equilibrium Position — Position where internal force is zero
`0 m` (default) | scalar

Position where the spring force is zero, specified as a scalar in units of length.

Spring Stiffness — Stiffness of force law
`0 N/m` (default) | scalar

Stiffness of the internal spring-damper force law for the joint primitive, specified as a scalar in units of linear stiffness.

Damping Coefficient — Damping coefficient of force law
`0 N(m/s)` (default) | scalar

Damping coefficient of the internal spring-damper force law for the joint primitive, specified as a scalar in units of linear damping coefficient.

Limits

Specify Lower Limit — Whether to specify lower position limit
`off` (default) | `on`

Select this parameter to specify the lower limit of the *z* prismatic primitive.

Bound — Lower bound of free region
`-1 m` (default) | scalar

Lower bound of the free region of the *z* prismatic primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Spring Stiffness — Stiffness of spring at lower bound
`1e6 N/m` (default) | scalar

Stiffness of the spring at the lower bound, specified as a scalar in units of linear stiffness.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Damping Coefficient — Damping coefficient at lower bound
`1e3 N/(m/s)` (default) | scalar

Damping coefficient at the lower bound, specified as a scalar in units of linear damping coefficient.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Transition Region Width — Region to smooth spring and damper forces
1e-4 `m` (default) | scalar

Region to smooth the spring and damper forces, specified as a scalar in units of length.

The block applies the full value of the lower-limit force when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Specify Upper Limit — Whether to specify upper position limit
`off` (default) | `on`

Select this parameter to specify the upper limit of the *y* prismatic primitive.

Bound — Upper bound of free region
1 `m` (default) | scalar

Upper bound for the free region of the joint primitive, specified as a scalar in units of length.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Spring Stiffness — Stiffness of spring at upper bound
1e6 `N/m` (default) | scalar

Stiffness of the spring at the upper bound, specified as a scalar in units of linear stiffness.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Damping Coefficient — Damping coefficient at upper bound
1e3 `N/(m/s)` (default) | scalar

Damping coefficient at the upper bound, specified as a scalar in units of linear damping coefficient.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Transition Region Width — Region to smooth spring and damper forces
1e-4 `m` (default) | scalar

Region to smooth the spring and damper forces, specified as a scalar in units of length.

The block applies the full value of the upper-limit force when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of forces 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.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Actuation

Force — Option to provide actuator force
`None` (default) | `Provided by Input` | `Automatically Computed`

Option to provide the actuator force for the joint primitive, specified as one of these values:

Actuation Force Setting	Description
`None`	No actuator force.
`Provided by Input`	Input port **fz** specifies the actuator force for the *z* prismatic primitive.
`Automatically Computed`	The block automatically calculates the amount of force required to satisfy the motion inputs to the mechanism. If you set this parameter to `Automatically Computed`, you do not need to set **Motion** to `Provided by Input` for the same joint primitive. The automatically computed force may to satisfy a motion input elsewhere in the mechanism.

Motion — Option to provide motion
`Automatically Computed` (default) | `Provided by Input`

Option to provide the motion for the joint primitive, specified as one of these values:

Actuation Motion Setting	Description
`Automatically Computed`	The block computes and applies the joint primitive motion based on model dynamics.
`Provided by Input`	Input port **pz** specifies the motion for the joint primitive.

Spherical Primitive (S)

State Targets

Specify Position Target — Whether to specify relative orientation target
`off` (default) | `on`

Select this parameter to specify the target of the relative orientation between the base and follower frames.

Priority — Priority level of relative orientation target
`High (desired)` (default) | `Low (approximate)`

Priority level of the relative orientation target, specified as `High (desired)` or `Low (approximate)`. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, select **Specify Position Target**.

Value > Method — Method to specify relative orientation target
`None` (default) | `Aligned Axes` | `Standard Axis` | `Arbitrary Axis` | `Rotation Sequence` | `Rotation Matrix` | `Quaternion`

Method to use to specify the relative orientation target between the base and follower frames. When specifying the parameter to `None`, the follower and base frames have the same orientation at the beginning of the simulation.

Dependencies

To enable this parameter, select **Specify Position Target**.

Pair 1: Follower — Follower frame axis used to align with specified base frame axis
+X (default) | -X | +Y | -Y | +Z | -Z

Follower frame axis used to align with the base frame axis set by the **Pair 1: Base** parameter, specified as an orthogonal axis of the follower frame. The follower frame rotates with respect to the base frame to enable the alignment between the selected axes of the base and follower frames.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Aligned Axis`.

Pair 1: Base — Base frame axis
+Y (default) | +X | -X | -Y | +Z | -Z

Base frame axis to align with the follower frame specified by the **Pair 1: Follower**, specified as an orthogonal axis of the base frame.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Aligned Axis`.

Pair 2: Follower — Follower frame axis used to align with specified base frame axis
+Y | +X | -X | -Y | +Z | -Z

Base frame axis to align with the follower frame specified by the **Pair 2: Follower**, specified as an orthogonal axis of the follower frame. The follower frame rotates with respect to the base frame to enable the alignment between the selected axes of the base and follower frames.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Aligned Axis`.

Pair2: Base — Base frame axis
+Z (default) | +X | -X | +Y | -Y | -Z

Base frame axis used to let the follower frame axis set in the **Pair2: Follower** parameter to align with, specified as an orthogonal axis of the base frame. The axis choices for **Pair 2** depend on the **Pair 1** axis selections.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Aligned Axis`.

Axis — Standard axis of relative rotation
+Z (default) | +X | -X | +Y | -Y | -Z

Axis of the relative rotation, specified as an orthogonal axis of the base frame.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Standard Axis`.

Angle — Angle of relative rotation
0.0 `deg` (default) | scalar

Angle of the relative rotation, specified as a scalar. The angle indicates the rotation of the follower frame with respect to the base frame about the specified axis.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Standard Axis`.

Axis — Axis of relative rotation
[0 0 1] (default) | 3-by1 vector

Axis of the relative rotation, specified as a 3-by-1 unit vector. The vector is dimensionless and indicates the rotational axis resolved in the base frame.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Arbitrary Axis`.

Angle — Angle of relative rotation
0.0 `deg` (default) | scalar

Angle of the relative rotation, specified as a scalar. The angle indicates the rotation of the follower frame with respect to the base frame about the axis specified by the **Axis** parameter.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Arbitrary Axis`.

Rotation About — Frame whose axes to rotate follower frame about
`Follower Axes` (default) | `Base Axes`

Frame whose axes to rotate the follower frame about, specified as `Follower Axes` or `Base Axes`. If you set the parameter to `Follower Axes`, the follower frame rotates about its own axes, and the follower frame changes the orientation with each successive rotation. If you set the parameter to `Base Axes`, the follower frame rotates about the fixed base frame axes. See “Rotational Measurements” for more information.

Dependencies

To enable this parameter, under **Specify Position Target > Value**, set **Method** parameter to `Rotation Sequence`.

Sequence — Sequence of rotation axis
`X-Y-X` (default) | `X-Y-Z` | `X-Z-X` | `X-Z-Y` | `Y-X-Y` | `Y-X-Z` | `Y-Z-X` | `Y-Z-Y` | `Z-X-Y` | `Z-X-Z` | `Z-Y-X` | `Z-Y-Z`

Sequence of the rotation axis for three successive elementary rotations. See “Rotation Sequence Measurements” for more information.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Rotation Sequence`.

Angles — Angles for rotation sequence parameterization
`[0 0 0] deg` (default) | 1-by-3 vector

Angles for the rotation sequence parameterization, specified as a 1-by-3 vector. See “Rotation Sequence Measurements” for more information.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Rotation Sequence`.

Rotation Matrix — Relative rotation using rotation matrix
`[1 0 0; 0 1 0; 0 0 1]` (default) | 3-by-3 matrix

Relative rotation, specified as a 3-by-3 matrix that maps vectors from the follower frame to the base frame. The matrix must be orthogonal and have determinant 1. See “Rotational Measurements” for more information.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Rotation Matrix`.

Quaternion — Relative rotation using quaternion
`[1 0 0 0]` (default) | unit quaternion vector

Relative rotation, specified as a unit quaternion vector. See “Rotational Measurements” for more information about the quaternion.

Dependencies

To enable this parameter, under **Specify Position Target** > **Value**, set **Method** parameter to `Quaternion`.

Specify Velocity Target — Whether to specify angular velocity target
`off` (default) | `on`

Select this parameter to specify the angular velocity target for the spherical primitive.

Priority — Priority level of angular velocity target
`High (desired)` (default) | `Low (approximate)`

Priority level of the angular velocity target, specified as `High (desired)` or `Low (approximate)`. See “Guiding Assembly” for more information.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Value — Angular velocity target of spherical primitive
[0 0 0] `deg/s` (default) | 1-by-3 vector

Angular velocity target for the spherical primitive, specified as a 1-by-3 vector resoved in resolution frame.

Dependencies

To enable this parameter, select **Specify Velocity Target**.

Resolution Frame — Frame used to resolve specified angular velocity target
`Base` (default) | `Follower`

Frame used to resolve the specified angular velocity target, specified as one of these:

- `Base` — The joint block resolves the angular velocity target in the coordinates of the base frame.
- `Follower` — The joint block resolves the angular velocity target in the coordinates of the follower frame.

Internal Mechanics

Equilibrium Position > Method — Method to specify equilibrium frame
`None` (default) | `Aligned Axes` | `Standard Axis` | `Arbitrary Axis` | `Rotation Sequence` | `Rotation Matrix` | `Quaternion`

Method to use to specify the equilibrium frame with respect to the base frame. The equilibrium frame is fixed during the simulation. If the *z*-axes of the follower and equilibrium frames are aligned, the spring torque of the spherical primitive is zero.

Set the **Equilibrium Position > Method** parameter to `None` to let the equilibrium and base frames be coincident.

Pair 1: Follower — Equilibrium frame axis used to align with specified base frame axis
+X (default) | -X | +Y | -Y | +Z | -Z

Equilibrium frame axis used to align with the base frame axis set by the **Pair 1: Base** parameter, specified as an orthogonal axis of the equilibrium frame. The equilibrium frame rotates with respect to the base frame to enable the alignment between the selected axes of the base and equilibrium frames.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Aligned Axis`.

Pair 1: Base — Base frame axis
+Y (default) | +X | -X | -Y | +Z | -Z

Base frame axis to align with the equilibrium frame specified by the **Pair 1: Follower**, specified as an orthogonal axis of the base frame.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Aligned Axis`.

Pair 2: Follower — Equilibrium frame axis used to align with specified base frame axis
+Y | +X | -X | -Y | +Z | -Z

Base frame axis to align with the equilibrium frame specified by the **Pair 2: Follower**, specified as an orthogonal axis of the equilibrium frame. The equilibrium frame rotates with respect to the base frame to enable the alignment between the selected axes of the base and equilibrium frames.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Aligned Axis`.

Pair2: Base — Base frame axis
+Z (default) | +X | -X | +Y | -Y | -Z

Base frame axis used to let the equilibrium frame axis set in the **Pair2: Follower** parameter to align with, specified as an orthogonal axis of the base frame. The axis choices for **Pair 2** depend on the **Pair 1** axis selections.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Aligned Axis`.

Axis — Standard axis of relative rotation
+Z (default) | +X | -X | +Y | -Y | -Z

Axis of the relative rotation, specified as an orthogonal axis of the base frame.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Standard Axis`.

Angle — Angle of relative rotation
0.0 `deg` (default) | scalar

Angle of the relative rotation, specified as a scalar. The angle indicates the rotation of the equilibrium frame with respect to the base frame about the specified axis.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Standard Axis`.

Axis — Axis of relative rotation
[0 0 1] (default) | 3-by1 vector

Axis of the relative rotation, specified as a 3-by-1 unit vector. The vector is dimensionless and indicates the rotational axis resolved in the base frame.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Arbitrary Axis`.

Angle — Angle of relative rotation
0.0 `deg` (default) | scalar

Angle of the relative rotation, specified as a scalar. The angle indicates the rotation of the equilibrium frame with respect to the base frame about the axis specified by the **Axis** parameter.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Arbitrary Axis`.

Rotation About — Frame whose axes to rotate equilibrium frame about
`Follower Axes` (default) | `Base Axes`

Frame whose axes to rotate the equilibrium frame about, specified as `Follower Axes` or `Base Axes`. If you set the parameter to `Follower Axes`, the equilibrium frame rotates about its own axes, and the equilibrium frame changes the orientation with each successive rotation. If you set the parameter to `Base Axes`, the equilibrium frame rotates about the fixed base frame axes. See "Rotational Measurements" for more information.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Rotation Sequence`.

Sequence — Sequence of rotation axis
`X-Y-X` (default) | `X-Y-Z` | `X-Z-X` | `X-Z-Y` | `Y-X-Y` | `Y-X-Z` | `Y-Z-X` | `Y-Z-Y` | `Z-X-Y` | `Z-X-Z` | `Z-Y-X` | `Z-Y-Z`

Sequence of the rotation axis for three successive elementary rotations. See "Rotation Sequence Measurements" for more information.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Rotation Sequence`.

Angles — Angles for elementary rotations
`[0 0 0] deg` (default) | 1-by-3 vector

Angles for elementary rotations, specified as a 1-by-3 vector. See "Rotation Sequence Measurements" for more information.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Rotation Sequence`.

Matrix — Relative rotation using rotation matrix
`[1 0 0; 0 1 0; 0 0 1]` (default) | 3-by-3 matrix

Relative rotation, specified as a 3-by-3 matrix. The matrix must be orthogonal and have determinant 1. See "Rotational Measurements" for more information.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Rotation Matrix`.

Quaternion — Relative rotation using quaternion
`[1 0 0 0]` (default) | unit quaternion vector

Relative rotation, specified as a unit quaternion vector. See "Rotational Measurements" for more information about the quaternion.

Dependencies

To enable this parameter, under **Equilibrium Position**, set **Method** parameter to `Quaternion`.

Spring Stiffness — Stiffness of force law
`0 N*m/deg` (default) | scalar

Stiffness of the internal spring-damper force law for the spherical primitive, specified as a scalar with a unit of rotational stiffness.

The spring attempts to pull the follower frame so that the follower frame is aligned with the specified equilibrium fame.

Damping Coefficient — Damping coefficient of force law
`0 N*m/(deg/s)` (default) | scalar

Damping coefficient of the internal spring-damper force law for the spherical primitive, specified as a scalar with a unit of rotational damping coefficient.

Limits

Specify Lower Limit — Whether to specify lower position limit
`off` (default) | `on`

Select this parameter to specify the lower limit of the spherical primitive. Joint limits use spring-dampers to resist travel past the bounds of the range.

Bound — Lower bound of free region
`15 deg` (default) | scalar

Lower bound for the free region of the spherical primitive, specified as a scalar with a unit of angle.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Spring Stiffness — Stiffness of spring at lower bound
`1e4 N*m/deg` (default) | scalar

Stiffness of the spring at lower bound, specified as a scalar with a unit of rotational stiffness.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Damping Coefficient — Damping coefficient at lower bound
`10 N*m/(deg/s)` (default) | scalar

Damping coefficient at lower bound, specified as a scalar with a unit of rotational damping coefficient.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Transition Region Width — Region to smooth spring and damper torque
`0.1 deg` (default) | scalar

Region to smooth the spring and damper torques, specified as a scalar with a unit of angle.

The block applies the full value of the lower-limit torque when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of torques 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.

Dependencies

To enable this parameter, select **Specify Lower Limit**.

Specify Upper Limit — Whether to specify upper position limit
`off` (default) | `on`

Select this parameter to specify the upper limit of the spherical primitive. Joint limits use spring-dampers to resist travel past the bounds of the range.

Bound — Upper bound of free region
`45 deg` (default) | scalar

Upper bound for the free region of the spherical primitive, specified as a scalar with a unit of angle.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Spring Stiffness — Stiffness of spring at upper bound
`1e4 N*m/deg` (default) | scalar

Stiffness of the spring at upper bound, specified as a scalar with a unit of stiffness.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Damping Coefficient — Damping coefficient at upper bound
`10 N*m/(deg/s)` (default) | scalar

Damping coefficient at upper bound, specified as a scalar with a unit of damping coefficient.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Transition Region Width — Region to smooth spring and damper torques
`0.1 deg` (default) | scalar

Region to smooth the spring and damper torques, specified as a scalar with a unit of angle.

The block applies the full value of the upper-limit torque when the penetration reaches the width of the transition region. The smaller the region, the sharper the onset of torques 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.

Dependencies

To enable this parameter, select **Specify Upper Limit**.

Actuation

Torque — Option to provide actuation torque
`None` (default) | `Provided by Input`

Option to provide the actuation torque for the spherical primitive, specified as one of these values.

Actuation Torque Setting	Description
`None`	Apply no actuation torque.
`Provided by Input`	Apply actuation torques based on physical signals. The signal specifies the torque acting on the follower frame with respect to the base frame. The signal provides the value of the torque applied equally and oppositely to the base and follower frames. Selecting this option exposes additional parameters that you can use to enable input ports See the **Input** section for details.

Resolution Frame — Frame used to resolve actuation torque
`Base` (default) | `Follower`

Frame used to resolve the input actuation torques, specified as `Base` or `Follower`.

Sensing

Resolution Frame — Frame used to resolve sensing outputs
`Base` (default) | `Follower`

Frame used to resolve the sensing output signals, specified as `Base` or `Follower`. For more information about the output signals, see the **Output** section.

Mode Configuration

Mode — Joint mode
`Normal` (default) | `Disengaged` | `Provided by Input`

Joint mode for the simulation, specified as one of these values:

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.
`Provided by Input`	The port **mode** specifies whether the joint behaves normally or is disengaged.

Composite Force/Torque Sensing

Direction — Measurement direction
`Follower on Base` (default) | `Base on Follower`

Measurement direction, specified as one of these values:

- `Follower on Base` — Sense the force and torque that the follower frame exerts on the base frame.
- `Base on Follower` — Sense the force and torque that the base frame exerts on the follower frame.

Note that this parameter only affects the output signals under the **Composite Force/Torque Sensing** section. Reversing the direction changes the sign of the measurements. For more information see “Force and Torque Measurement Direction”.

Resolution Frame — Frame used to resolve measurements
`Base` (default) | `Follower`

Frame used to resolve the measurements, specified as one of these values:

- `Base` — The joint block resolves the measurements in the coordinates of the base frame.
- `Follower` — The joint block resolves the measurements in the coordinates of the follower frame.

Note that this parameter only affects the output signals under the **Composite Force/Torque Sensing** section.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Bushing Joint | `simscape.multibody.SixDofJoint`

Topics

“Actuating and Sensing with Physical Signals”
“Independent Suspension System Templates”
“Specify Joint Motion in Planar Manipulator Model”
“Motion Sensing”
“Selecting a Measurement Frame”

Telescoping Joint

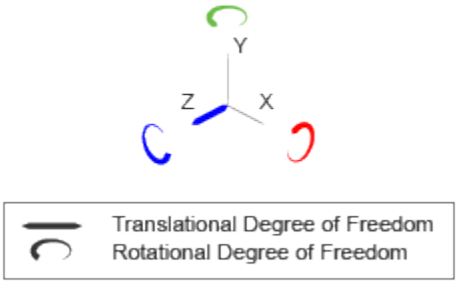
Joint with one prismatic and one spherical joint primitive

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.



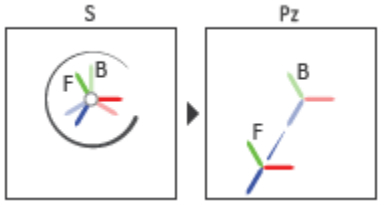
Joint Degrees of Freedom

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 (in the **Apps** gallery, click **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.
`Quaternion`	Specify frame rotation using a unit quaternion vector.

`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]`.

`Quaternion`

Use a unit quaternion vector to specify the rotation of the follower frame with respect to the base frame. The default value is `[1 0 0 0]`. See “Rotational Measurements” for more information about the quaternion.

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.
`Quaternion`	Specify frame rotation using a unit quaternion vector.

`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]`.

`Quaternion`

Use a unit quaternion vector to specify the rotation of the follower frame with respect to the base frame. The default value is `[1 0 0 0]`. See “Rotational Measurements” for more information about the quaternion.

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`.

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.

Mode Configuration

Specify the mode of the joint. The joint mode can be normal or disengaged throughout the simulation, or you can provide an input signal to change the mode during the simulation.

Mode

Select one of the following options to specify the mode of the joint. The default setting is `Normal`.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.
`Provided by Input`	This option exposes the **mode** port that you can connect to an input signal to change the joint mode during the simulation. The joint mode is normal when the input signal is `0` and disengaged when the input signal is `-1`. The joint mode can be changed many times during the simulation.

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 of the Z prismatic joint primitive
- pz — Desired trajectory of the Z prismatic joint primitive

The spherical joint primitive provides the following actuation ports:

- t — Actuation torque vector [*tx*, *ty*, *tz*] acting on the spherical joint primitive
- tx, ty, tz — 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:

- 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
- fllz — Force due to contact with the lower limit of the Z prismatic joint primitive
- fulz — Force due to contact with the upper limit of the Z prismatic joint primitive

The spherical primitive provides the following sensing ports:

- Q — Orientation of the spherical joint primitive in quaternion form

- wx, wy, wz — X, Y, and Z angular velocity components of the spherical joint primitive
- w — Angular velocity [*wx*, *wy*, *wz*] of the spherical joint primitive
- bx, by, bz — X, Y, and Z angular acceleration components of the spherical joint primitive
- b — Angular acceleration [*bx*, *by*, *bz*] of the spherical joint primitive
- tll — Torque due to contact with the lower limit of the spherical joint primitive, given as the signed magnitude of the torque vector
- tul — Torque due to contact with the upper limit of the spherical joint primitive, given as the signed magnitude of the torque vector

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

Mode Port

Mode configuration provides the following port:

- mode — Value of the mode of the joint. If the input is equal to `0`, the joint behaves normally. If the input is equal to `-1`, the joint behaves as disengaged.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Prismatic Joint | Spherical Joint

Transform Sensor

Sensor that measures the relative spatial relationship between two frames

Libraries:
Simscape / Multibody / Frames and Transforms

Description

The Transform Sensor block measures the relative spatial relationship between frames connected to ports **F** and **B** of the block. The measured quantities includes the relative pose, velocity, and acceleration. To measure the absolute translational or rotational quantities of a frame, connect the frame ports **F** and **B** of the block to this frame and the world frame of the model, respectively.

Measurement Frame

The **Measurement Frame** parameter setting affects all the outputs of the block except the ones listed in the table.

Outputs	Ports
Rotation measurements	**q**, **axs**, **Q**, **R**, **seq**
Quaternion derivatives	**Q** and **Qdd**
Rotation-matrix derivatives	**Rd** and **Rdd**
Rotation sequence derivatives	**seqd**
Distance and its derivatives	**dst**, **vdst**, and **adst**

The Transform Sensor block has five different selections for the **Measurement Frame** parameter: `World`, `Base`, `Follower`, `Non-Rotating Base`, and `Non-Rotating Follower`. When you set **Measurement Frame** to `World`, all the measurements are resolved in the world frame. When you set **Measurement Frame** to `Base` or `Follower`, the resolved acceleration measurements include centripetal and Coriolis terms if the corresponding base or follower frame rotates. When you set **Measurement Frame** parameter to `Non-Rotating Base` or `Non-Rotating Follower`, the measurements do not satisfy the standard derivative relationship if the corresponding base or follower frame rotates. For example, the relative linear velocity does not equal the time derivative of the relative translation. See “Selecting a Measurement Frame” for more information.

Rotational and Translational Measurements

The block has four parameterizations to express the measured rotations: angle-axis, quaternion, rotation matrix, and rotation sequence. To enable these parameterizations, in the block dialog box, under **Rotation**, select corresponding parameters. For example, select the **Angle** and **Axis** parameters to use the angle-axis parameterization or select the **Quaternion**, **Transform**, or **Rotation Sequence** parameter to use the quaternion, rotation matrix, or rotation sequence parameterization.

Also, the block has various parameterizations to express rotational velocities and accelerations: *X*-, *Y*-, and *Z*- coordinates; the time derivatives of a quaternion or rotation sequence; or matrix. To enable

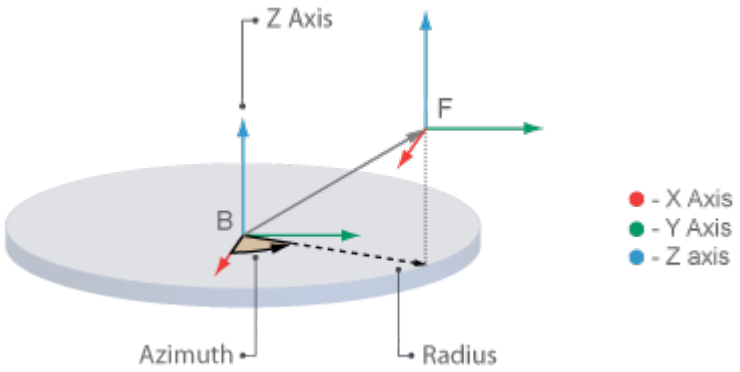
these parameterizations, you can select the corresponding parameters under **Angular Velocity** or **Angular Acceleration**. See “Rotational Measurements” for more information.

The settings of the **Measurement Frame** parameter and coordinate systems affect the translational measurements. The block has three coordinate systems to express the translational measurements: Cartesian, cylindrical, and spherical. You can select one or more of them at the same time. See “Translational Measurements” for more information.

The tables summarize the coordinates of the cylindrical and spherical systems, and the images show the diagrams of these coordinate systems. For simplicity purpose, in the images, specify the **Measurement Frame** parameter as `Base`.

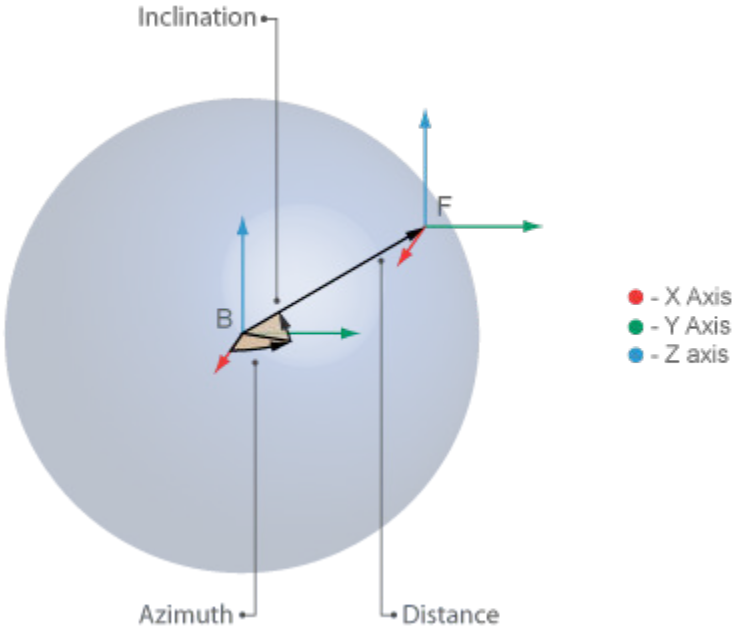
Cylindrical Coordinates

Coordinate	Description
Radius	The length of the projection of the vector BF, in the *X-Y* plane of the measurement frame.
Azimuth	The angle of the `Radius` with respect to the positive *X*-axis of the measurement frame. The angle falls in the range of [-π, π).
Z	The projection of the vector BF on the *Z*-axis of the measurement frame.



Spherical Coordinates

Coordinate	Description
Distance	The distance between origins of the base and follower frames.
Azimuth	The angle of the projection of the vector BF in the *X-Y* plane with respect to the positive *X*-axis. The angle is resolved in the measurement frame and falls in the range of [-π, π).
Inclination	The angle of the vector BF with respect to the *X-Y* plane of the measurement frame. The angle falls in the range of [-π/2, π/2].



To use a specific coordinate system, select the corresponding parameters. For example, if you want to use the Cartesian system to express the measured relative linear velocity of the follower frame, under **Velocity**, select the **X**, **Y**, and **Z** parameters.

The output ports remain hidden until you select their corresponding parameters. Each port outputs a time-varying physical signal. You can use the PS-Simulink Converter block to set the units of the outputs when you connect the Transform Sensor block to Simulink blocks.

Ports

Frame

B — Base frame
frame

Frame port associated with the base frame for the measurements.

F — Follower frame
frame

Frame port associated with the follower frame for the measurements.

Output

q — Angle of relative rotation
scalar

Angle of relative rotation, returned as a scalar. The angle indicates the rotation of the follower frame with respect to the base frame about the axis that is specified by the vector exported from port **axs**. The angle falls in the range [0, π]. Use ports **q** and **axs** to output the rotation signals using the angle-axis parameterization.

Dependencies

To enable this port, under **Rotation**, select **Angle**.

axs — Axis of relative rotation
vector

Axis of relative rotation, returned as a 3-by-1 unit vector. The vector is dimensionless and indicates the rotational axis. Use ports **q** and **axs** to output the rotation signals using the angle-axis parameterization.

Dependencies

To enable this port, under **Rotation**, select **Axis**.

Q — Relative rotation in quaternion parameterization
vector

Relative rotation, returned as a vector in quaternion parameterization. See “Rotational Measurements” and “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Rotation**, select **Quaternion**.

R — Relative rotation matrix
3-by-3 matrix

Relative rotation, returned as a 3-by-3 matrix. See “Rotational Measurements” and “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Rotation**, select **Transform**.

seq — Angles of three successive rotations
3-by-1 vector

Relative rotation, returned as a 3-by-1 vector that contains the angles for the three successive elementary rotations that represent the rotation of the follower frame with respect to the base frame. The elements of the vector are in rad.

The setting of the **Measurement Frame** parameter does not affect the angles. The three rotations are about an intermediate frame and can be one of 12 different rotation sequences. See “Rotation Sequence Measurements” for more information.

Dependencies

To enable this port, under **Rotation**, select **Rotation Sequence**.

wx — *X*-coordinate of relative angular velocity
scalar

X-coordinate of the relative angular velocity, returned as a scalar. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Angular Velocity**, select **Omega X**.

wy — *Y*-coordinate of relative angular velocity
scalar

Y-coordinate of the relative angular velocity, returned as a scalar. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Angular Velocity**, select **Omega Y**.

wz — *Z*-coordinate of relative angular velocity
scalar

Z-coordinate of the relative angular velocity, returned as a scalar. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Angular Velocity**, select **Omega Z**.

Qd — Relative angular velocity expressed as time derivative of quaternion
vector

Relative angular velocity, returned as a 4-by-1 vector that equals the time derivative of the signal from port **Q**.

Dependencies

To enable this port, under **Angular Velocity**, select **Quaternion**.

Rd — Relative angular velocity in matrix format
3-by-3 matrix

Relative angular velocity, returned as a 3-by-3 matrix. The matrix equals the time derivative of the signal from port **R**.

Dependencies

To enable this port, under **Angular Velocity**, select **Transform**.

seqd — Derivatives of rotation sequence angles
3-by-1 vector

Derivatives of the rotation sequence angles, returned as a 3-by-1 vector. The array elements are time derivatives of the output from the port **seq**. The velocities are in rad/s. The setting of the **Measurement Frame** parameter does not affect the measurements.

Dependencies

To enable this port, under **Angular Velocity**, select **Rotation Sequence**.

bx — *X*-coordinate of relative angular acceleration
scalar

X-coordinate of the relative angular acceleration, returned as a scalar. This quantity equals the time derivative of the signal exported from port **wx** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Angular Acceleration**, select **Alpha X**.

by — *Y*-coordinate of relative angular acceleration
scalar

Y-coordinate of the relative angular acceleration, returned as a scalar. This quantity equals the time derivative of the signal exported from port **wy** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Angular Acceleration**, select **Alpha Y**.

bz — *Z*-coordinate of relative angular acceleration
scalar

Z-coordinate of the relative angular acceleration, returned as a scalar. This quantity equals the time derivative of the signal exported from port **wz** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Angular Acceleration**, select **Alpha Z**.

Qdd — Relative angular acceleration expressed as second time derivative of quaternion
vector

Relative angular velocity, returned as a 4-by-1 vector that equals the second time derivative of the signal exported from port **Q**.

Dependencies

To enable this port, under **Angular Acceleration**, select **Quaternion**.

Rdd — Relative angular acceleration in matrix format
3-by-3 matrix

Relative angular acceleration, returned as a 3-by-3 matrix. The matrix equals the second time derivative of the signal exported from port **R**.

Dependencies

To enable this port, under **Angular Acceleration**, select **Transform**.

x — *X*-coordinate of relative translation
scalar

X-coordinate of the relative translation vector resolved in the Cartesian coordinate system, returned as a scalar.

Dependencies

To enable this port, under **Translation**, select **X**.

y — *Y*-coordinate of relative translation
scalar

Y-coordinate of the relative translation vector resolved in the Cartesian coordinate system, returned as a scalar.

Dependencies

To enable this port, under **Translation**, select **Y**.

z — *Z*-coordinate of relative translation
scalar

Z-coordinate of the relative translation vector resolved in the Cartesian coordinate system, returned as a scalar. Because the *Z*-coordinate exists in both the Cartesian and cylindrical coordinate systems, the Transform Sensor block only exposes one output port for both coordinate systems.

Dependencies

To enable this port, under **Translation**, select **Z**.

rad — Cylindrical radius coordinate of relative translation
nonnegative scalar

Radius coordinate of the relative translation vector resolved in the cylindrical coordinate system, returned as a scalar.

Dependencies

To enable this port, under **Translation**, select **Radius**.

azm — Azimuth of relative translation
scalar

Azimuth of the relative translation vector, returned as scalar. The angle falls in the range of [-π, π). The azimuth is undefined if the origins of the base and follower frames coincide with each other. Because the azimuth exists in both the cylindrical and spherical coordinate systems, the Transform Sensor block only exposes one output port for both coordinate systems.

Dependencies

To enable this port, under **Translation**, select **Azimuth**.

dst — Spherical radius coordinate of relative translation
scalar

Radius coordinate of the relative translation vector resolved in the spherical coordinate system, returned as a scalar. The value equals the distance between origins of the base and follower frames.

Dependencies

To enable this port, under **Translation**, select **Distance**.

inc — Inclination of relative translation
scalar

Inclination of the relative translation vector resolved in the spherical coordinate system, returned as a scalar. The angle falls in the range of [-π/2, π/2].

Dependencies

To enable this port, under **Translation**, select **Inclination**.

vx — *X*-coordinate of relative linear velocity
scalar

X-coordinate of the relative linear velocity resolved in the Cartesian coordinate system, returned as a scalar. The value equals the time derivative of the signal exported from port **x** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Velocity**, select **X**.

vy — *Y*-coordinate of relative linear velocity
scalar

Y-coordinate of the relative linear velocity resolved in the Cartesian coordinate system, returned as a scalar. The value equals the time derivative of the signal exported from port **y** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Velocity**, select **Y**.

vz — *Z*-coordinate of relative linear velocity
scalar

Z-coordinate of the relative linear velocity resolved in the Cartesian coordinate system, returned as a scalar. Because the *Z*-coordinate exists in both Cartesian and cylindrical coordinate systems, the Transform Sensor block only exposes one output port for both coordinate systems. The value equals the time derivative of the signal exported from port **z** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Velocity**, select **Z**.

vrad — Cylindrical radius coordinate of relative linear velocity
scalar

Radius coordinate of the relative linear velocity resolved in the cylindrical coordinate system, returned as a scalar. The value equals the time derivative of the signal exported from port **rad** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Velocity**, select **Radius**.

vazm — Azimuth coordinate of relative linear velocity
scalar

The azimuth coordinate of the relative linear velocity, returned as a scalar. Because the azimuth coordinate exists in both the cylindrical and spherical coordinate systems, the Transform Sensor block only exposes one output port for both coordinate systems. The value equals the time derivative of the signal exported from port **azm** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Velocity**, select **Azimuth**.

vdst — Spherical radius coordinate of relative linear velocity
scalar

The radius coordinate of the relative linear velocity resolved in the spherical coordinate system, returned as a scalar. The value equals the time derivative of the signal exported from port **dst**.

Dependencies

To enable this port, under **Velocity**, select **Distance**.

vinc — Inclination coordinate of relative linear velocity
scalar

The inclination coordinate of the relative linear velocity resolved in the spherical coordinate system, returned as a scalar. The value equals the time derivative of the signal exported from port **inc** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Velocity**, select **Inclination**.

ax — *X*-coordinate of relative linear acceleration
scalar

X-coordinate of the relative linear acceleration resolved in the Cartesian coordinate system, returned as a scalar. The value equals the second time derivative of the signal exported from port **x** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Acceleration**, select **X**.

ay — *Y*-coordinate of relative linear acceleration
scalar

Y-coordinate of the relative linear acceleration resolved in the Cartesian coordinate system, returned as a scalar. The value equals the second time derivative of the signal exported from port **y** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Acceleration**, select **Y**.

az — *Z*-coordinate of relative linear acceleration
scalar

Z-coordinate of the relative linear acceleration resolved in the Cartesian coordinate system, returned as a scalar. Because the *Z*-coordinate exists in both Cartesian and cylindrical coordinate systems, the Transform Sensor block only exposes one output port for both coordinate systems. The value equals the second time derivative of the signal exported from port **z** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Acceleration**, select **Z**.

arad — Cylindrical radius coordinate of relative linear acceleration
scalar

Radius coordinate of the relative linear acceleration resolved in the cylindrical coordinate system, returned as a scalar. The value equals the second time derivative of the signal exported from port **rad** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Acceleration**, select **Radius**.

aazm — Azimuth coordinate of relative linear acceleration
scalar

Azimuth coordinate of the relative linear acceleration, returned as a scalar. Because the azimuth coordinate exists in both cylindrical and spherical coordinate systems, the Transform Sensor block only exposes one output port for both coordinate systems. The value equals the second time derivative of the signal exported from port **azm** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Acceleration**, select **Azimuth**.

adst — Spherical radius coordinate of relative linear acceleration
scalar

Radius coordinate of the relative linear acceleration resolved in the spherical coordinate system, returned as a scalar. The value equals the second time derivative of the signal exported from port **dst**.

Dependencies

To enable this port under **Acceleration**, select **Distance**.

ainc — Inclination coordinate of relative linear acceleration
scalar

Inclination coordinate of the relative linear acceleration resolved in the spherical coordinate system, specified as a scalar. The value equals the second time derivative of the signal exported from port **inc** if the **Measurement Frame** parameter is not set to a non-rotating frame. See “Selecting a Measurement Frame” for more information.

Dependencies

To enable this port, under **Acceleration**, select **Inclination**.

Parameters

Measurement Frame — Frame to resolve measurements
`World` (default) | `Base` | `Follower` | `Non-Rotating Base` | `Non-Rotating Follower`

Frame used to resolve the measured vector quantities. See “Selecting a Measurement Frame” for more information.

Measurement Frame Types

Measurement Frame	Type	Description
`World`	Inertial Frame	The block resolves the measured vectors in the coordinates of the world frame.
`Base`	Non-Inertial Frame	The block resolves the measured vectors in the coordinates of the base frame. The base frame is the frame that connects to the frame port **B**. The resolved acceleration measurements involve centripetal and Coriolis terms if the base frame rotates.
`Follower`	Non-Inertial Frame	The block resolves the measured vectors in the coordinates of the follower frame. The follower frame is the frame that connects to the frame port **F**. The resolved acceleration measurements involve centripetal and Coriolis terms if the follower frame rotates.
`Non-Rotating Base`	Instantaneous Frame	The block maps the vectors resolved in the world frame to the non-rotating base frame. The non-rotating base frame is an instantaneous frame that is coincident and aligned with the base frame at the current time. The resolved acceleration measurements do not involve centripetal and Coriolis terms.
`Non-Rotating Follower`	Instantaneous Frame	The block maps the vectors resolved in the world frame to the non-rotating follower frame. The non-rotating follower frame is an instantaneous frame that is coincident and aligned with the follower frame at the current time. The resolved acceleration measurements do not involve centripetal and Coriolis terms.

Sequence — Sequence of rotation axis
`X-Y-X` (default) | `X-Y-Z` | `X-Z-X` | `X-Z-Y` | `Y-X-Y` | `Y-X-Z` | `Y-Z-X` | `Y-Z-Y` | `Z-X-Y` | `Z-X-Z` | `Z-Y-X` | `Z-Y-Z`

Sequence of the rotation axis for three successive elementary rotations. See “Rotation Sequence Measurements” for more information.

Dependencies

To enable this parameter, under **Rotation**, select **Rotation Sequence**.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Rigid Transform

Topics

“Motion Sensing”
“Rotational Measurements”
“Selecting a Measurement Frame”
“Sense Motion Using a Transform Sensor Block”
“Translational Measurements”

Universal Joint

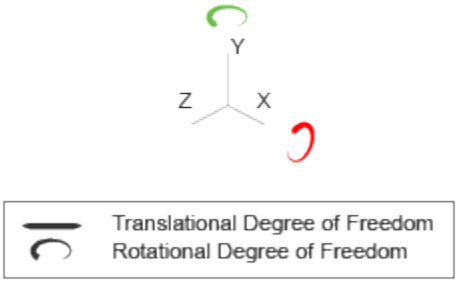
Joint with two revolute primitives

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.



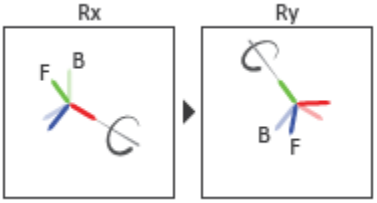
Joint Degrees of Freedom

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.

Mode Configuration

Specify the mode of the joint. The joint mode can be normal or disengaged throughout the simulation, or you can provide an input signal to change the mode during the simulation.

Mode

Select one of the following options to specify the mode of the joint. The default setting is `Normal`.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.

Method	Description
`Provided by Input`	This option exposes the **mode** port that you can connect to an input signal to change the joint mode during the simulation. The joint mode is normal when the input signal is `0` and disengaged when the input signal is `-1`. The joint mode can be changed many times during the simulation.

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:

- tx, ty — Actuation torques acting on the X and Y revolute joint primitives
- qx, qy — 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
- tllx, tlly — Torques due to contact with the lower limits of the X and Y revolute joint primitives
- tulx, tuly — Torques due to contact with the upper limits of 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

Mode Port

Mode configuration provides the following port:

- mode — Value of the mode of the joint. If the input is equal to `0`, the joint behaves normally. If the input is equal to `-1`, the joint behaves as disengaged.

Version History

Introduced in R2012a

See Also

Revolute Joint | Gimbal Joint

Topics
“Actuating and Sensing with Physical Signals”
“Full Vehicle on Four Post Testrig”
“Modeling Constant Velocity Joints - Power Take-Off Shaft”
“Motion Sensing”
“Rotational Measurements”

Variable Brick Solid

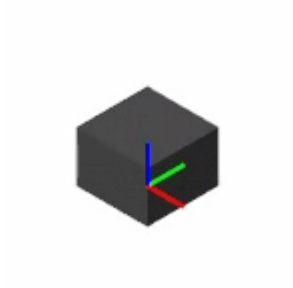
Solid brick with variable mass and size

Libraries:
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 blocks, 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

lx — *x*-dimension of the brick
physical signal

Output port for the *x*-dimension of the brick.

ly — *y*-dimension of the brick
physical signal

Output port for the *y*-dimension of the brick.

lz — *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 **lz**, 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 material
`1000 kg/m^3` (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

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 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 — Graphic to use for visualization
`From Geometry` (default) | `Marker` | `None`

Type of the visual representation of the solid, specified as `From Geometry`, `Marker`, or `None`. Set the parameter to `From Geometry` to show the visual representation of the solid. Set the parameter to `Marker` to represent the solid as a marker. Set the parameter to `None` to hide the solid in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Shape — Shape of marker to represent 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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
10 `pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered solid and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Version History

Introduced in R2017b

Extended Capabilities

C/C++ Code Generation

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

See Also

Brick Solid | Cylindrical Solid | Ellipsoidal Solid | Extruded Solid | Revolved Solid | Spherical Solid | Variable Cylindrical Solid | Variable Spherical Solid | Rigid Transform

Topics

“Modeling Bodies”
“Representing Solid Geometry”
“Manipulate the Color of a Solid”

Variable Cylindrical Solid

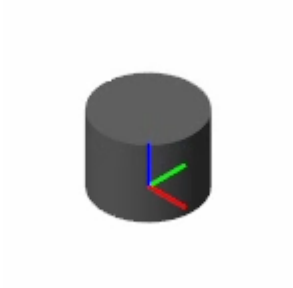
Solid cylinder with variable mass and size

Libraries:
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 blocks, 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/m^3` (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 — Graphic to use for visualization
`From Geometry` (default) | `Marker` | `None`

Type of the visual representation of the solid, specified as `From Geometry`, `Marker`, or `None`. Set the parameter to `From Geometry` to show the visual representation of the solid. Set the parameter to `Marker` to represent the solid as a marker. Set the parameter to `None` to hide the solid in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Shape — Shape of marker to represent 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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
10 `pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered solid and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Simple`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`

2 **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1 **Type** to `From Geometry` or `Marker`
2 **Visual Properties** to `Advanced`

Version History

Introduced in R2017b

Extended Capabilities

C/C++ Code Generation

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

See Also

Brick Solid | Cylindrical Solid | Ellipsoidal Solid | Extruded Solid | Revolved Solid | Spherical Solid | Variable Brick Solid | Variable Spherical Solid | Rigid Transform

Topics

“Representing Solid Inertia”
“Specifying Custom Inertias”

Variable Spherical Solid

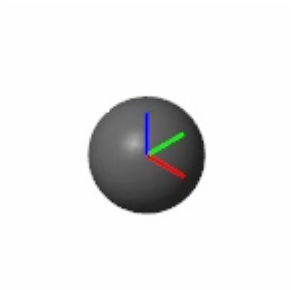
Solid sphere with variable mass and size

Libraries:
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 blocks, 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/m^3` (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.

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 — Graphic to use for visualization
`From Geometry` (default) | `Marker` | `None`

Type of the visual representation of the solid, specified as `From Geometry`, `Marker`, or `None`. Set the parameter to `From Geometry` to show the visual representation of the solid. Set the parameter to `Marker` to represent the solid as a marker. Set the parameter to `None` to hide the solid in the model visualization.

Visual Properties — Parameterizations for color and opacity
`Simple` (default) | `Advanced`

Parameterizations for specifying visual properties. Select `Simple` to specify **Diffuse Color** and **Opacity**. Select `Advanced` to specify more visual properties, such as **Specular Color**, **Ambient Color**, **Emissive Color**, and **Shininess**.

Dependencies

To enable this parameter, set **Type** to `From Geometry` or `Marker`.

Shape — Shape of marker to represent 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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Size — Width of the marker in pixels
`10 pixels` (default) | scalar

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.

Dependencies

To enable this parameter, set **Type** to `Marker`.

Diffuse Color — Color of light due to diffuse reflection
[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered solid and provides shading that gives the rendered object a three-dimensional appearance.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Opacity — Graphic opacity
1.0 (default) | scalar in the range of 0 to 1

Graphic opacity, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Simple`

Specular Color — Color of light due to specular reflection
[0.5 0.5 0.5 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1. This parameter changes the color of the specular highlight, which is the bright spot on the rendered solid due to the reflection of the light from the light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Ambient Color — Color of ambient light
[0.15 0.15 0.15 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Ambient light refers to a general level of illumination that does not come directly from a light source. The Ambient light consists of light that has been reflected and re-reflected so many times that it is no longer coming from any particular direction. You can adjust this parameter to change the shadow color of the rendered solid.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Emissive Color — Self-illumination color
[0.0 0.0 0.0 1.0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The emission color is color that does not come from any external source, and therefore seems to be emitted by the solid itself. When a solid has a emissive color, the solid can be seen even if there is no external light source.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Shininess — Highlight sharpness
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.

Dependencies

To enable this parameter, set:

1. **Type** to `From Geometry` or `Marker`
2. **Visual Properties** to `Advanced`

Version History

Introduced in R2017b

Extended Capabilities

C/C++ Code Generation

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

See Also

Brick Solid | Cylindrical Solid | Ellipsoidal Solid | Extruded Solid | Revolved Solid | Spherical Solid | Variable Cylindrical Solid | Variable Brick Solid | Rigid Transform

Topics

“Representing Solid Inertia”
“Specifying Custom Inertias”

Weld Joint

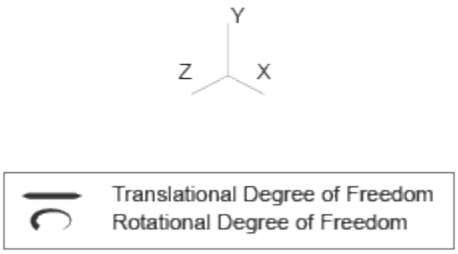
Joint with zero primitives

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

Mode Configuration

Specify the mode of the joint. The joint mode can be normal or disengaged throughout the simulation, or you can provide an input signal to change the mode during the simulation.

Mode

Select one of the following options to specify the mode of the joint. The default setting is `Normal`.

Method	Description
`Normal`	The joint behaves normally throughout the simulation.
`Disengaged`	The joint is disengaged throughout the simulation.

Method	Description
`Provided by Input`	This option exposes the **mode** port that you can connect to an input signal to change the joint mode during the simulation. The joint mode is normal when the input signal is `0` and disengaged when the input signal is `-1`. The joint mode can be changed many times during the simulation.

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

Mode Port

Mode configuration provides the following port:

- mode — Value of the mode of the joint. If the input is equal to `0`, the joint behaves normally. If the input is equal to `-1`, the joint behaves as disengaged.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Rigid Transform

Topics

“Measure Joint Constraint Forces”

World Frame

Inertial reference frame

Libraries:
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.

Version History

Introduced in R2012a

Extended Capabilities

C/C++ Code Generation

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

See Also

Reference Frame | Rigid Transform

Topics

“Working with Frames”
“Creating Connection Frames”

Worm and Gear Constraint

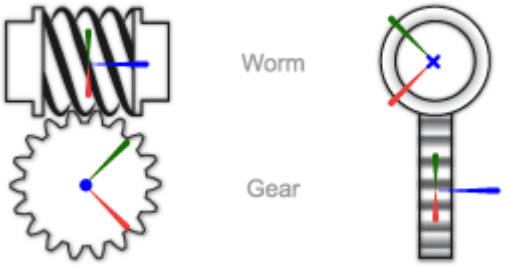
Kinematic constraint between worm and gear bodies with perpendicular non-intersecting rotation axes

Libraries:
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.

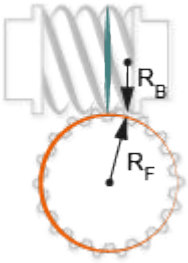
The worm thread direction can follow either right-hand or left-hand conventions. The convention used determines the relative directions of the worm and gear rotational velocities. A right-hand convention causes the worm and gear to rotate in the same direction about the respective z-axes. A left-hand 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.

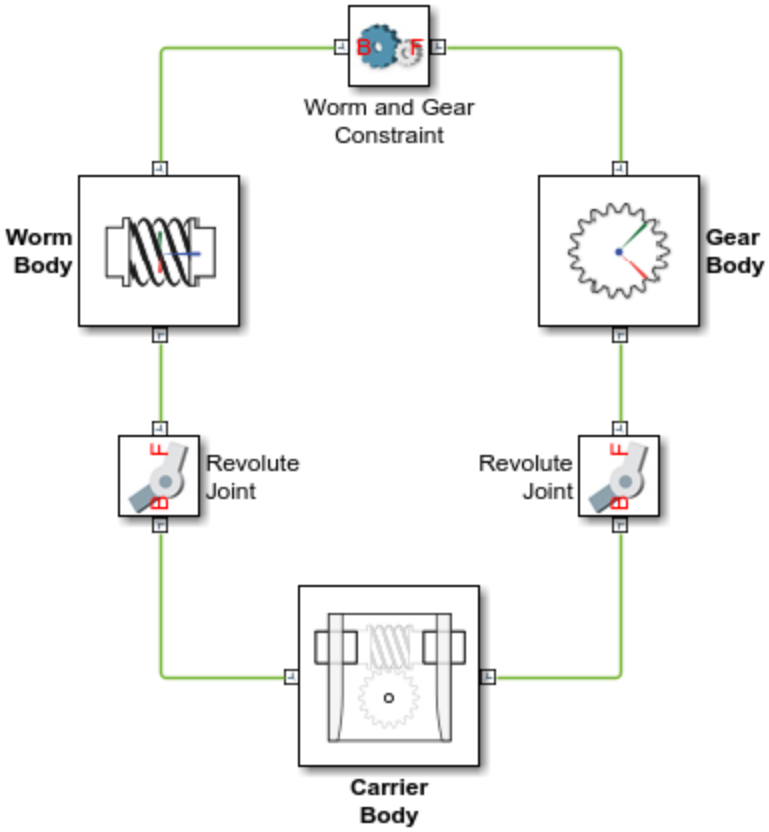
Gear Geometry

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.



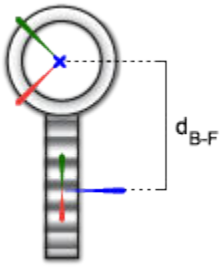
Assembly Requirements

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

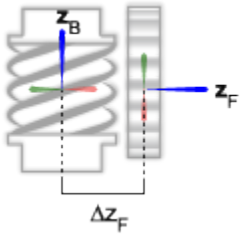
suitable for meshing. The block enforces the restrictions during model assembly, when it first attempts to place the gears in mesh, but relies on the remainder of the model to keep the gears in mesh during simulation.

Position Restrictions

- The distance between the base and follower frame *z*-axes, denoted $d_{B\text{-}F}$ in the figure, must be equal to the distance between the gear centers.

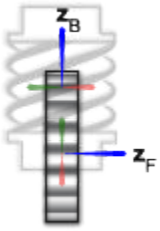


- The translational offset between the base and follower frame origins along the follower frame *z*-axis, denoted ΔZ_F in the figure, must be zero.

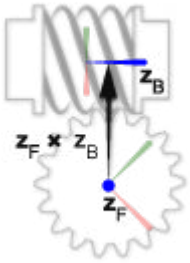


Orientation Restrictions

- 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 $\hat{\mathbf{z}}_F \times \hat{\mathbf{z}}_B$.



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`.

Version History

Introduced in R2016b

Extended Capabilities

C/C++ Code Generation

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

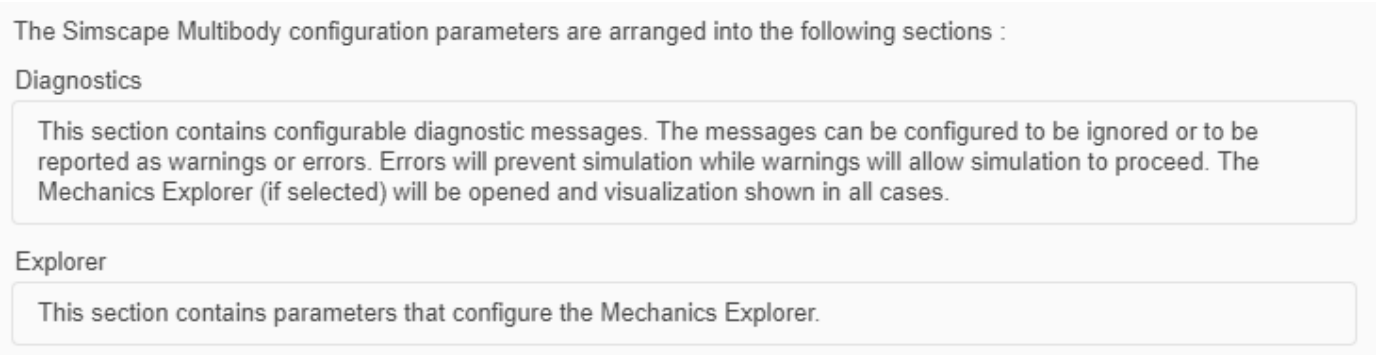
See Also

Rack and Pinion Constraint | Bevel Gear Constraint | Common Gear Constraint

Topics
“Worm and Gear”

Configuration Parameters

Simscape Multibody Pane: General



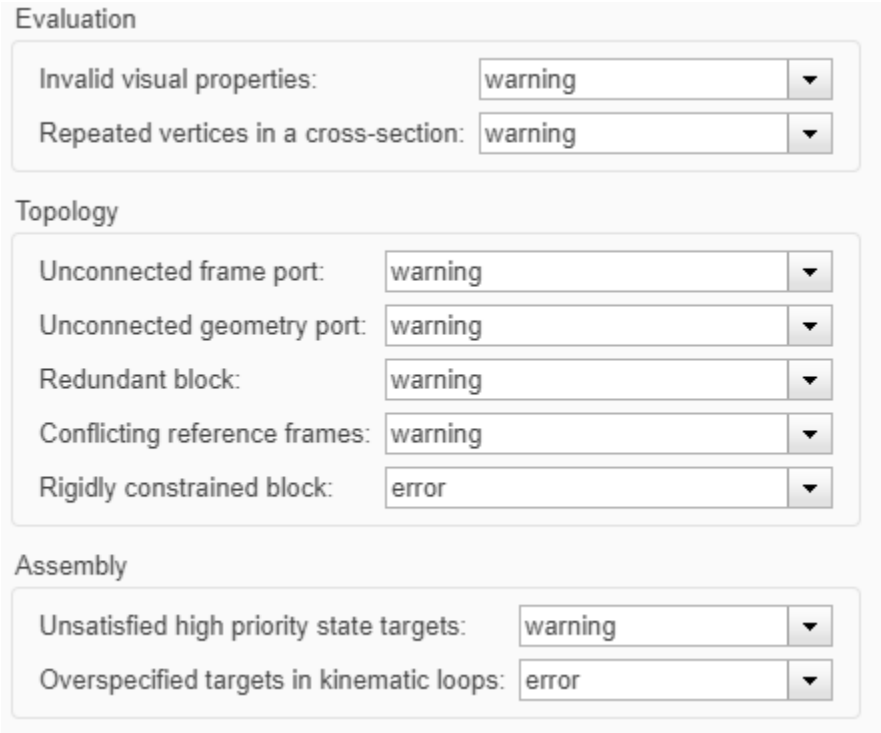
Simscape Multibody Pane Overview

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



In this section...

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`

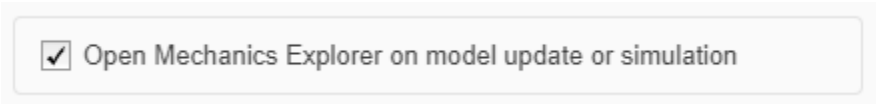
When the application detects this situation, it terminates the simulation and displays an error message.

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

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 Apps

Flexible Body Model Builder

Generate reduced-order model data for flexible bodies

Description

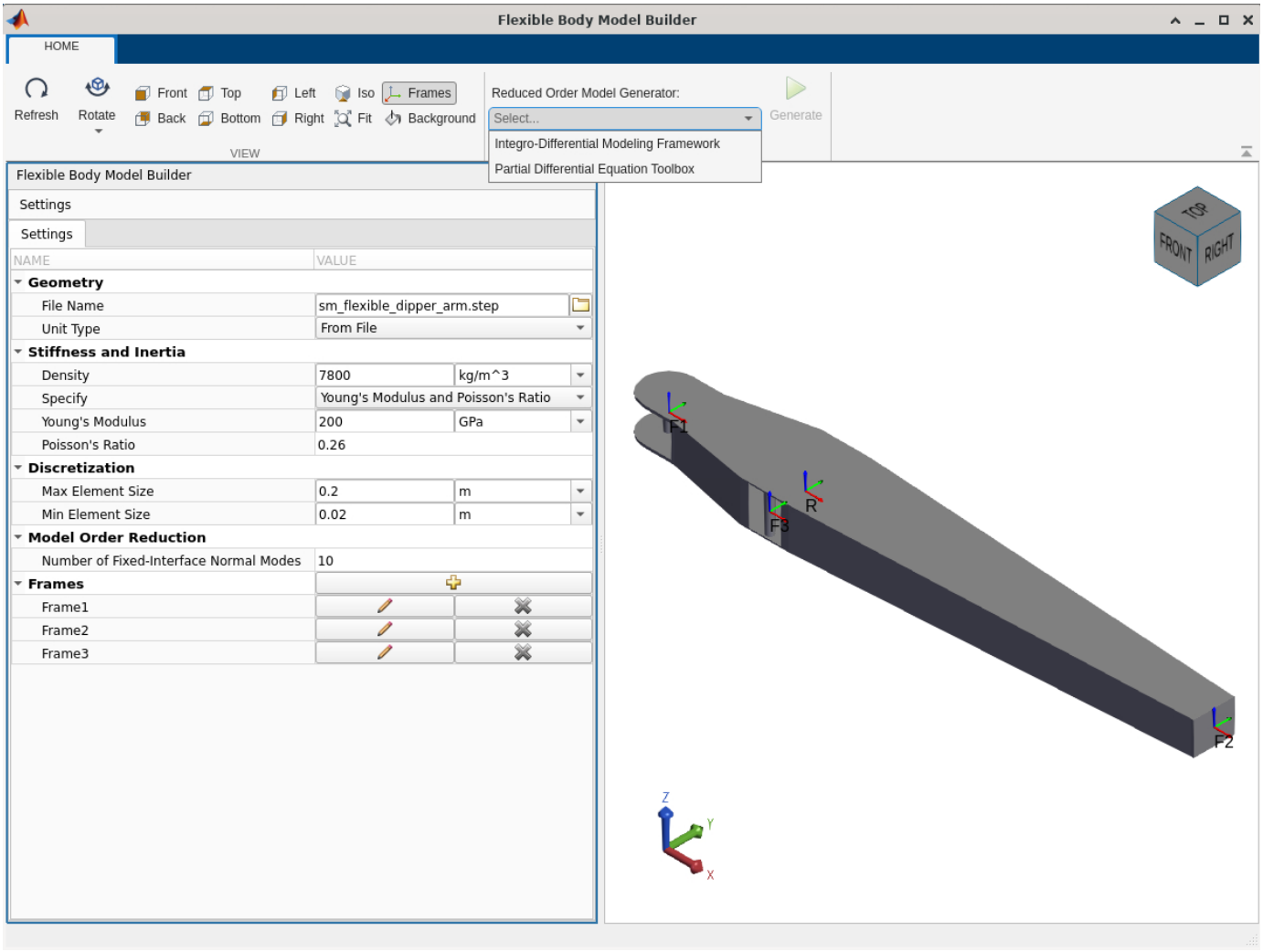
The **Flexible Body Model Builder** app generates the reduced-order model (ROM) data for flexible bodies by using the Craig-Bampton method [1] on page 3-12. You can use the app to specify the geometry, material properties, and interface frames of the flexible body. To control the accuracy of the reduced-order model, you can specify the meshing parameters and model order reduction settings.

The app saves the ROM data as a structure array called `FlexibleBody` in a MAT-file. The ROM data contains the stiffness and mass matrices, the origins and orientations of the interface frames, and graphical information for displaying the deformed shape of the flexible body. You can use the fields of the structure array to specify the corresponding parameters of the Reduced Order Flexible Solid block to model a flexible body.

The **Flexible Body Model Builder** app provides the Partial Differential Equation Toolbox and Integro-Differential Modeling Frameworks (IDMF) options to generate ROM data. The Partial Differential Equation Toolbox option requires the Partial Differential Equation Toolbox. To use the IDMF option, download and install the Integro-Differential Modeling Framework for MATLAB add-on.

The Integro-Differential Modeling Framework for MATLAB add-on supports only Linux® and Windows®. On Linux, the add-on runs without any additional requirement. On Windows, to run the add-on, you must install the Windows Subsystem for Linux (WSL). To install the WSL, see Install Linux on Windows with WSL.

On Windows, every time when using the IDMF to generate a ROM data set, the app imports a Linux distribution called `idmf_hub` to the WSL if the `idmf_hub` does not exist yet. To query the presence of the `idmf_hub` distribution, in the Windows command prompt, type `wsl -l -v`. To remove the `idmf_hub` distribution, type `idmf_container_teardown()` at the MATLAB command line.



Open the Flexible Body Model Builder App

- MATLAB Toolstrip: On the **Apps** tab, under **Simscape**, click Flexible Body Model Builder.
- MATLAB command prompt: Enter `flexibleBodyModelBuilder`.

Examples

Use App to Generate ROM Data

This example shows how to generate the reduced order model, ROM, data by using the **Flexible Body Model Builder** app. The example uses the geometry, material properties, and frames of the dipper arm part in the “Model an Excavator Dipper Arm as a Flexible Body” example.

Launch the App

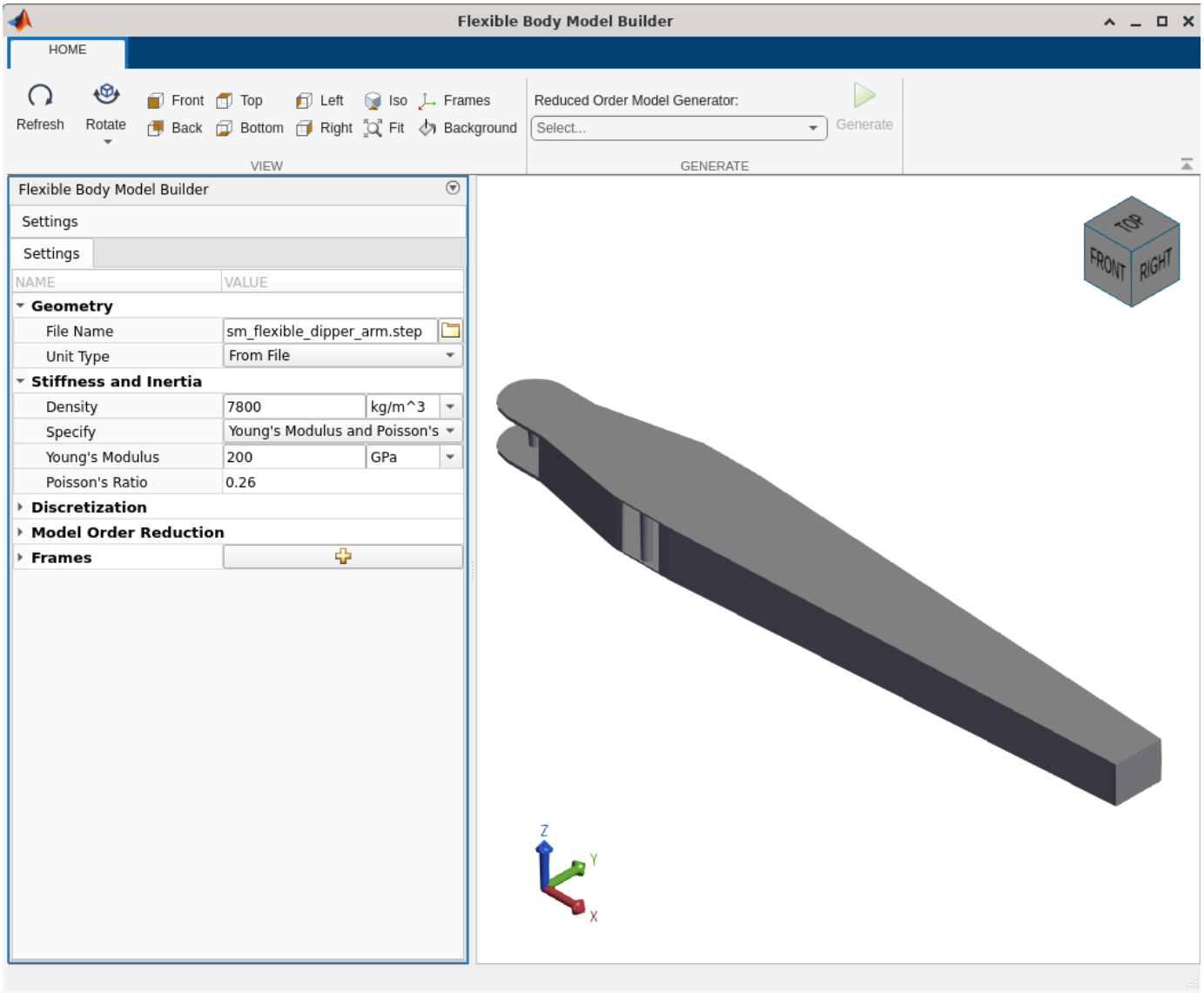
To launch the **Flexible Body Model Builder** app, enter the command at the MATLAB command line:

`flexibleBodyModelBuilder`

Specify the Geometry and Material Properties

To specify the geometry, under **Geometry**, enter `sm_flexible_dipper_arm.step` for the **File Name** parameter. To view the dipper arm, under **Home** tab, click the Refresh button .

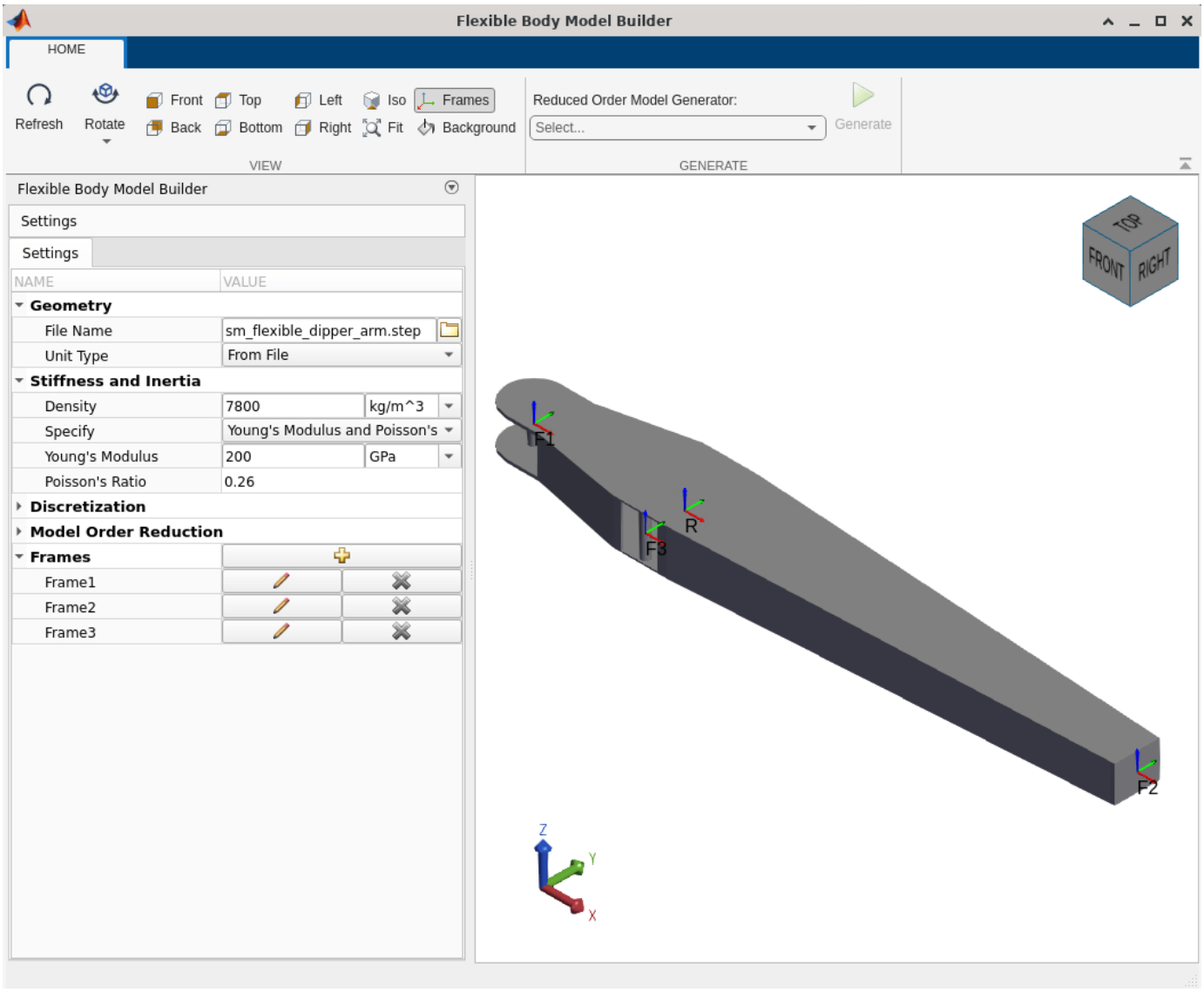
The dipper arm is constructed from steel. To represent the material properties, in the **Stiffness and Inertia** section, specify **Density**, **Young's Modulus**, and **Poisson's Ratio** as shown in the image.



Add the Interface Frames

A flexible body uses the interface frames to connect other Simscape™ Multibody™ elements, such as joints, constraints, forces, and sensors. In this example, add three interface frames on the dipper arm. Use the **Frames** parameter to specify the locations and orientations of the interface frames. As shown in the image, set the locations of the F1, F2, and F3 frames as the centroids of the cylindrical surface

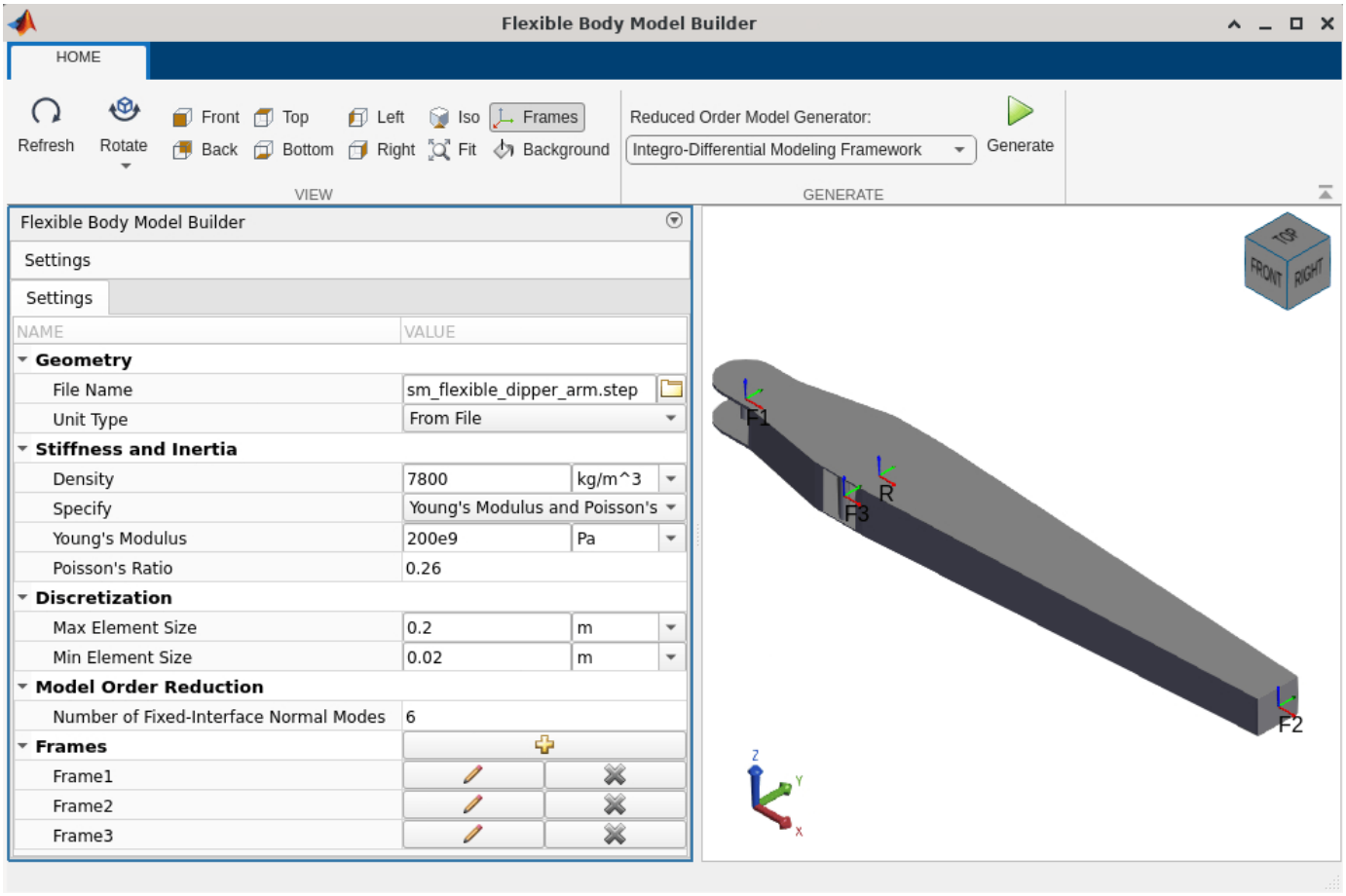
4, planar surface 11, and cylindrical surface 1, respectively. Use the default settings for the orientations of the interface frames.



Specify Mesh and Generate the ROM Data

To control the accuracy of the reduced order model, you can specify the meshing parameters and model order reduction settings. To produce a coarse mesh for body, specify 0.2 m for the **Max Element Size** and 0.02 m for the **Min Element Size**. You can adjust these parameters to refine the mesh for more accurate results. To specify model order reduction settings, under **Model Order Reduction**, specify **Number of Fixed-Interface Normal Modes**. This example retains 10 dynamic Craig-Bampton modes.

To generate the ROM data, under **Home** tab, specify **Reduced Order Model Generator** and click Generate button  . The example uses the `Integro-Differential Modeling Framework` option. The app saves all the ROM data in a MAT-file. In this example, name the MAT-file as `DipperROM.mat`.



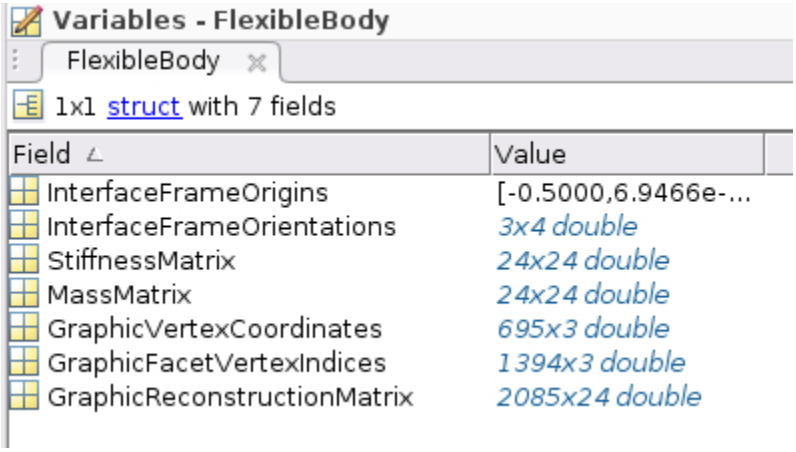
Note that the output ROM data does not save any of the inputs that were provided to the app for generating the ROM data. In other words, if you close the app after generating the ROM data, you cannot access the specifications from the app anymore. It is recommended to record all the inputs before closing the app.

Use ROM Data to Specify the Reduced Order Flexible Solid Block

This example shows how to use the ROM data generated by the **Flexible Body Model Builder** app to specify a Reduced Order Flexible Solid block.

Load the ROM Data

This example uses the ROM data generated in the Generate ROM Data example. Find and load the `DipperROM.mat` file that stores the ROM data. In the MATLAB® workspace, double-click the `FlexibleBody` structure array to see the ROM data categorized in different fields.



Specify the Reduced Order Flexible Solid block

This section uses the “Flexible Dipper Arm” model to show how to specify a Reduced Order Flexible Solid block with the loaded ROM data. This example uses the ROM data to specify the interface frames, stiffness and mass matrices, and graphic properties of the flexible body.

In the Flexible Dipper Arm model, open the Flexible Dipper Arm block. Use the dot notation of the form `FlexibleBody.fieldName` to set these parameters to the corresponding ROM data.

Interface Frames

- Number of Frames
- Origins
- Orientations

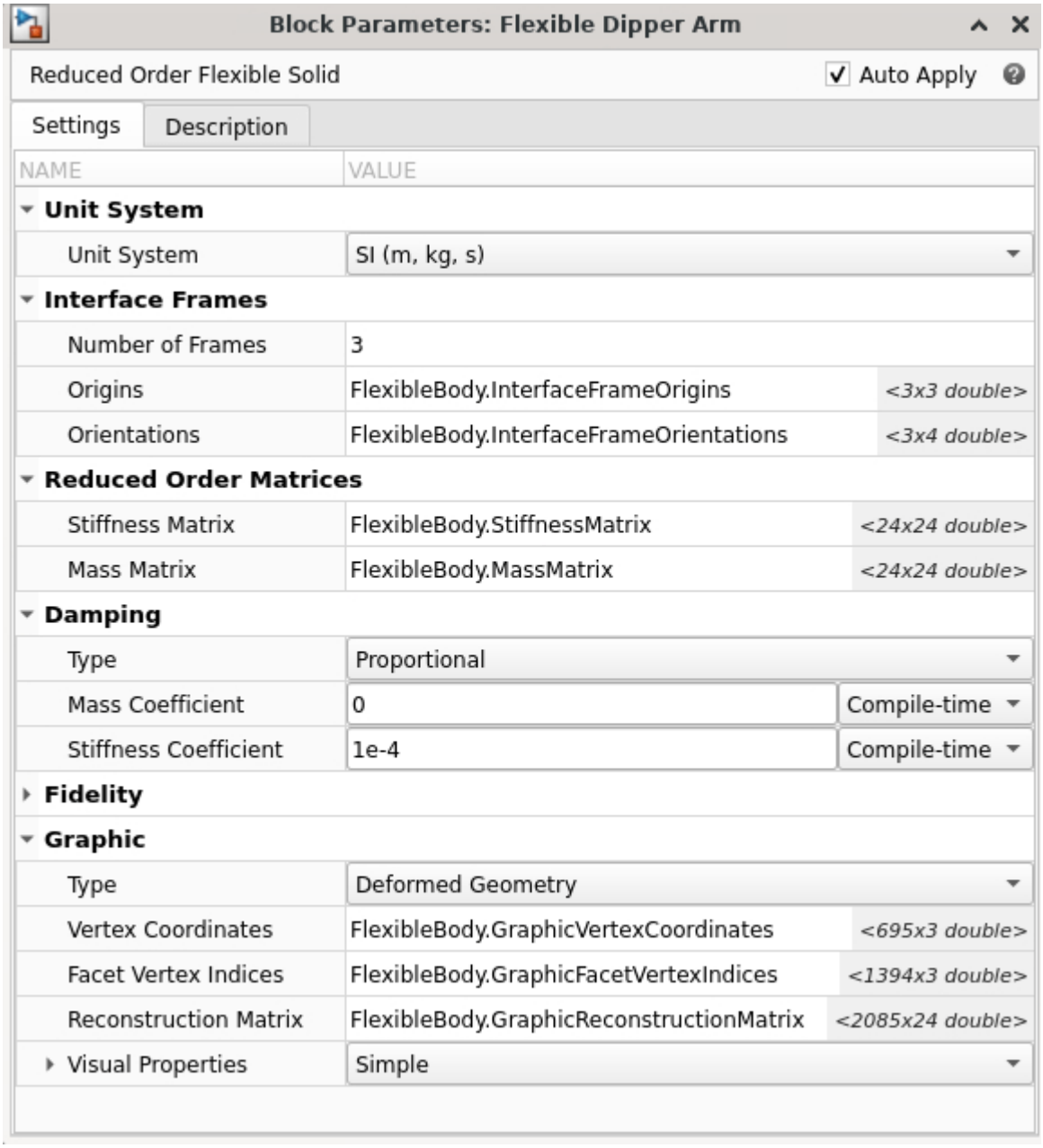
Reduced Order Matrices

- Stiffness Matrix
- Mass Matrix

Graphic

- Vertex Coordinates
- Facet Vertex Indices
- Reconstruction Matrix

For damping properties, use the default settings of the proportional damping. Because the ROM data generated by the **Flexible Body Model Builder** app is in SI units, you must set the **Unit System** to `SI (m, kg, s)`.



- “Model an Excavator Dipper Arm as a Flexible Body”

Parameters

Geometry

`File Name` — Path of CAD geometry file
character vector

Path of the CAD geometry file, specified as a character vector. The file location can the absolute path starting from the root directory of the file system or a relative path starting from a folder on the MATLAB path.

Except the STL, the app supports all the file formats listed in “Supported Software and File Formats” on page 1-124.

Example: `'C:/Users/JDoe/Documents/myShape.STEP'` or `'Documents/myShape.STEP'`

`Unit Type` — Source for solid geometry unit
`From File` (default) | `Custom`

Source of the solid geometry unit, specified as the `From File` or `Custom`. Select `From File` to use the unit specified in the imported file. Select `Custom` to specify your own unit.

`Unit` — Length unit in which to interpret geometry
`m` (default) | `cm` | `mm` | `km` | `um` | `in` | `ft` | `mi` | `yd`

Length unit in which to interpret the geometry. Changing the unit changes the scale of the imported geometry.

Dependencies

To enable this parameter, set **Unit Type** to `Custom`.

Stiffness and Inertia

`Density` — Mass per unit volume of material
`2700 kg/m^3` (default) | positive scalar

Mass per unit volume of material. The default value corresponds to aluminum.

`Specify` — Elastic properties used to parameterize plate
`Young's Modulus and Poisson's Ratio` (default) | `Young's and Shear Modulus`

Elastic properties used to parameterize the plate. You can specify either `Young's Modulus and Poisson's Ratio` or `Young's and Shear Modulus`. These properties are commonly available in materials databases.

`Young's Modulus` — Ratio of normal stress to normal strain
`70 GPa` (default) | positive scalar

Ratio of normal stress to normal strain, specified as a positive scalar. The default value corresponds to aluminum.

The app performs a cross-parameter check between the Young's and shear moduli to ensure the resulting Poisson's ratio is in the range [0, 0.5).

`Poisson's Ratio` — Ratio of transverse to normal strains
0.33 (default) | scalar in the range [0, 0.5)

Poisson's ratio of the plate. The value specified must be greater than or equal to 0 and smaller than 0.5. The default value corresponds to aluminum.

Dependencies

To enable this parameter, set **Specify** to `Young's Modulus and Poisson's Ratio`.

`Shear Modulus` — Ratio of shear stress to engineering shear strain
`26 GPa` (default) | positive scalar

Ratio of shear stress to engineering shear strain, specified as a positive scalar. The default value corresponds to aluminum.

The app performs a cross-parameter check between the Young's and shear moduli to ensure the resulting Poisson's ratio is in the range [0, 0.5).

Dependencies

To enable this parameter, set **Specify** to `Young's and Shear Modulus`.

Discretization

The app creates meshes based on the specified maximum and minimum element sizes. The app provides quadratic tetrahedral meshes whose edge lengths are roughly within the specified element sizes. The values of the **Max Element Size** and **Min Element Size** parameters do not impose strict constraints on the created mesh, but provide soft lower and upper bound targets. Occasionally, the edge length of some elements may be beyond the specified limits.

`Max Element Size` — Target maximum mesh edge length
`1 m` (default) | positive scalar

Target maximum mesh edge length, specified as a positive scalar.

The **Max Element Size** parameter is an approximate upper bound on the mesh edge lengths. The value must be larger than the value of the **Min Element Size** parameter. This parameter controls the overall refinement of the mesh. Small values create finer meshes, but the mesh generation and ROM computation take a longer time.

`Min Element Size` — Target minimum mesh edge length
`1 m` (default) | positive scalar

Target minimum mesh edge length, specified as a positive scalar.

The **Min Element Size** parameter is an approximate lower bound on the mesh edge lengths and controls the mesh refinement in a localized manner, such as at curved edges or on small features. The value must be less than the value of the **Max Element Size** parameter. Small values may better capture details and improve the accuracy of the ROM, but the mesh generation and ROM computation take a longer time.

Model Order Reduction

`Number of Fixed-Interface Normal Modes` — Retained dynamic Craig-Bampton modes
0 (default) | nonnegative integer

Retained dynamic Craig-Bampton modes, specified as an integer in range [0, n].

$n = 3 \times \left(n_n - n_{nf}\right)$,

where:

- n_n is the number of all the nodes in the mesh.
- n_{nf} is the number of nodes on the surfaces to which the interface frames are attached.

For most analyses, it is not necessary to retain high-frequency modes for these reasons:

- High-frequency modes do not contribute much to simulation results.
- Simulations with high-frequency modes are computational expansive.

Frames

`Frames` — Create interface frames
button

Click the Create Frame button  to open a pane for creating a new interface frame on the body. In this pane, you can specify the origin and orientation for the frame.

To define the origin of the frame, under **Frame Origin**, use **Based on Geometric Feature** to make the new frame origin coincident with the canonical point of the selected undeformed feature. Surfaces are the only valid feature. Select a surface from the visualization pane, then click **Select Feature** to confirm the location of the origin. The name of the origin location appears in the field below this option.

After you select a surface to define the origin of the frame, the app rigidly constrains that surface in the flexible body and rigidly attaches the frame to that rigid region. The selected surfaces for the interface frames must not connect to or overlap each other.

To define the orientation of the frame, under the **Frame Axes** section, select the **Primary Axis** and **Secondary Axis** of the frame and then specify their directions.

Use the following methods to select a vector for specifying the directions of the primary and secondary axes. The primary axis is parallel to the selected vector and constrains the remaining two axes to its normal plane. The secondary axis is parallel to the projection of the selected vector onto the normal plane.

- **Along Reference Frame Axis**: Selects an axis of the reference frame of the undeformed geometry.
- **Based on Geometric Feature**: Selects the vector associated with the chosen undeformed geometry feature. Valid features include surfaces and lines. The corresponding vector is indicated by an arrow in the visualization pane. You can select a feature from the visualization pane and then click **Select Feature** to confirm the selection. The name of the select feature appears in the field below this option.

`FrameN` — Edit or delete existing interface frame
frame name

Edit or delete the frames you created. `N` is a unique identifying number for each custom frame.

- Click the Edit button  to edit the interface frame, such as the origin and axes.
- Click the Delete button  to delete the interface frame.

Dependencies

To enable this parameter, create a frame by clicking the Create Frame button  .

Version History

Introduced in R2023a

References

[1] Craig Jr, Roy R., and Andrew J. Kurdila. *Fundamentals of Structural Dynamics*. 2nd ed. Hoboken, N.J: John Wiley, 2006.

See Also

Blocks

Reduced Order Flexible Solid

Topics

“Model an Excavator Dipper Arm as a Flexible Body”

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.

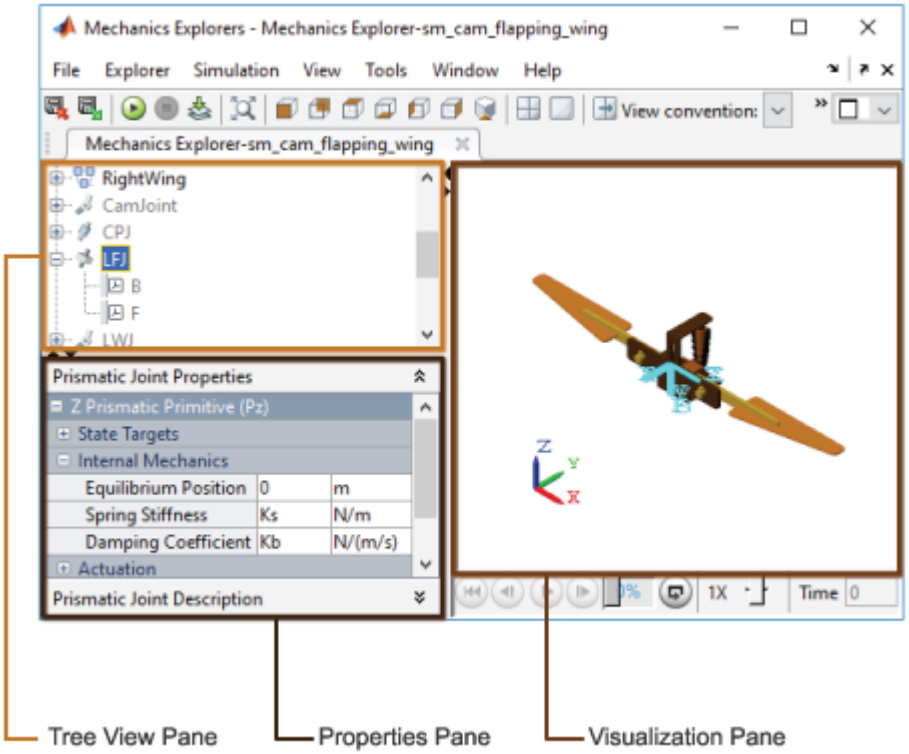
For more information on view manipulation, see “Manipulate the Visualization Viewpoint”.

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 **Modeling** tab, select **Model Settings > Model Settings**.
2. In the left pane of the Configuration Parameters dialog box, 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”

Version History

Introduced in R2012a

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”

Camera Manager

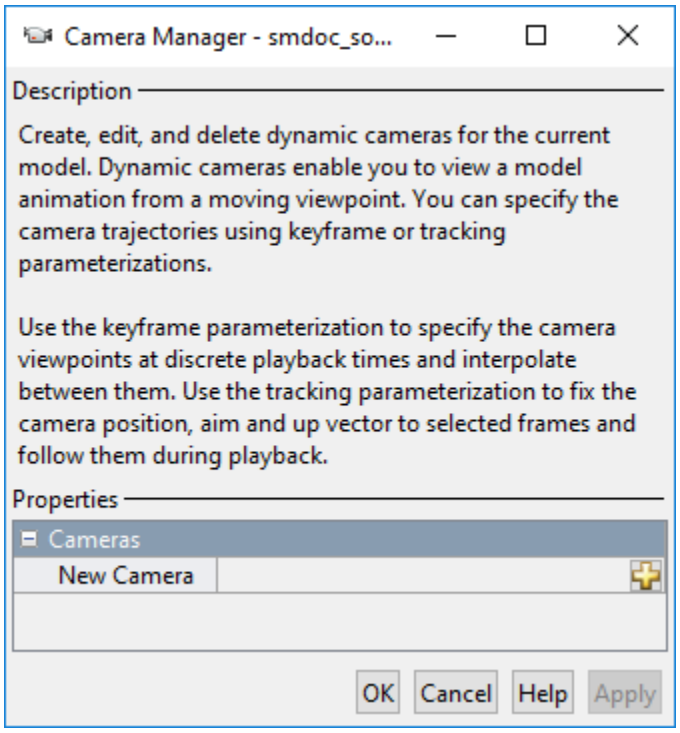
Create, edit, and delete dynamic cameras

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**. Use the camera definition pane to set the camera mode and trajectory constraints. To open the camera definition pane for a new camera, click the button in the **New Camera** field. To open the camera definition pane for an existing camera field, click the button.

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. In `Tracking` mode, the **Status** indicator is red if the camera definition is incomplete or invalid, as determined by the specified values for the "Position" on page 3-0 , "Aim" on page 3-0 , and "Up Vector" on page 3-0 parameters.

Dependencies

When **Mode** is set to:

- `Keyframes` — The "Keyframes" on page 3-0 setting buttons, **Set**, **Remove**, **Previous**, and **Next**, are visible.
- `Tracking` — The tracking status and related parameters, "Position" on page 3-0 , "Aim" on page 3-0 , and "Up Vector" on page 3-0 , are visible.

`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** — Go to the previous or next defined keyframes.

Dependencies

- Before setting keyframes, you must simulate the model and pause playback.
- The **Keyframes** parameter is active only when the "Mode" on page 3-0 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.

Dependencies

The **Position** parameter is active only when the "Mode" on page 3-0 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.

Dependencies

The **Aim** parameter is active only when the "Mode" on page 3-0 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.

Dependencies

The **Up Vector** parameter is active only when the "Mode" on page 3-0 parameter is set to `Tracking`.

Version History

Introduced in R2016a

See Also

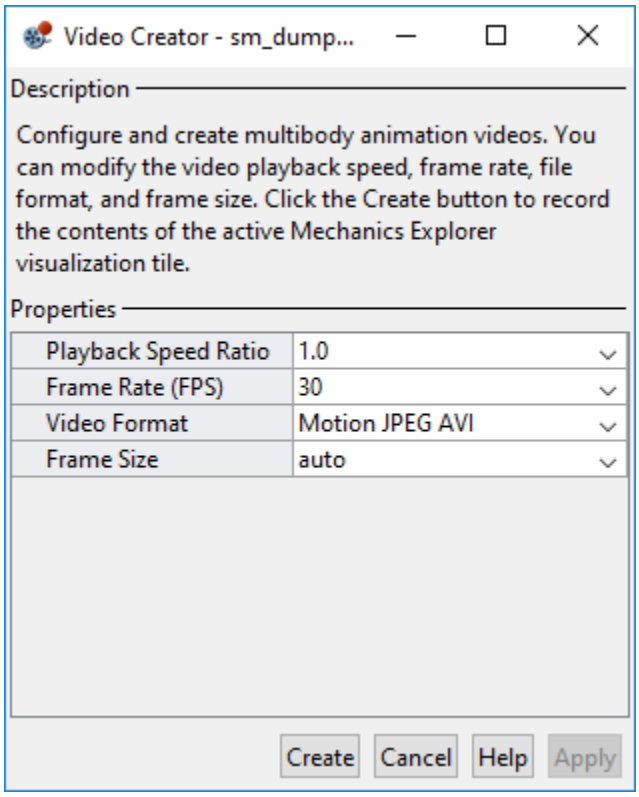
Topics
"Visualization Cameras"
"Create a Dynamic Camera"

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) | `.25` | `.25` | `.5` | `2` | `4` | `8`

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) | `16` | `24` | `60`

Number of video frames per second of playback time, specified as a positive integer. Larger frame rates result in smoother video playback time but also larger file sizes.

`Video Format` — Video file format
`Motion JPEG AVI` (default) | `Archival`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) | `[400 400]` | `[800 600]` | `[1280 720]` | `[1920 1080]`

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. To use the current dimensions of the active visualization tile in **Mechanics Explorer**, select `auto`.

Version History

Introduced in R2016b

See Also

`smwritevideo`

Functions

simscape.multibody.Component class

Package: `simscape.multibody`

Abstract base class for components

Description

`simscape.multibody.Component` is the abstract base class for components. The `Component` class contains subclasses that model the different parts of a multibody system, such as `simscape.multibody.Joint` and `simscape.multibody.Multibody`. To model a robotic arm, for example, you can use `simscape.multibody.RigidBody` objects to construct the physical parts, such as the gripper and wrist, then link the parts with revolute joints constructed by using `simscape.multibody.RevoluteJoint` objects. See the “More About” on page 4-3 section for more details of the `Component` class.

Class Attributes

`Abstract`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Properties

`FrameConnectors` — Names of frame connectors in component object

N-by-1 string array

Names of the frame connectors in a component object, returned as a string array. Every connector has a unique name. See the “Connectors” on page 4-3 section for more information about connectors.

Attributes:

`GetAccess`	`public`
`SetAccess`	`Restricts access`
`NonCopyable`	`true`

`DoVisualize` — Whether to turn on visual representation of component object

`true` | `false`

Whether to turn on the visual representation of a component object, specified as `true` or `false`. A `true` value enables the visualization of the component object. See the “Visualization” on page 4-3 section for more information.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

More About

Connectors

Component objects use connectors to connect to each other. For example, in a `simscape.multibody.Multibody` object, you can attach a `simscape.multibody.Solid` object to the `simscape.multibody.WorldFrame` object by connecting the `R` connector of the `Solid` object to the `W` connector of the `WorldFrame` object.

Component objects only support frame connectors. A frame connector represents a 3-D, right-handed, orthogonal coordinate frame in a component object. Connecting frame connectors from different component objects identifies the frames in a kinematic structure. In other words, the frames become coincident and aligned during the simulation.

Most types of component objects, such as `simscape.multibody.WorldFrame` and `simscape.multibody.RigidTransform`, have a fixed set of frame connectors, and the connectors are not changeable. However, because the `simscape.multibody.RigidBody` and `simscape.multibody.Multibody` objects initially have no connectors by default, you need to explicitly add connectors. Note that, in a component object, every connector must have a unique name. The table shows the frame connectors for component objects:

Component Objects	Connector Name	Definition
`simscape.multibody.WorldFrame`	`"W"`	World frame
`simscape.multibody.RigidTransform`	`"B"`	Base frame
	`"F"`	Follower frame
`simscape.multibody.Joint`	`"B"`	Base frame
	`"F"`	Follower frame
`simscape.multibody.Solid`	`"R"`	Reference frame
`simscape.multibody.RigidBody`	User-defined name	A name that is user defined
`simscape.multibody.Multibody`	User-defined name	A name that is user defined

Visualization

The `visualize` method of a `simscape.multibody.CompiledMultibody` object allows you to visualize a 3-D graphical representation of the corresponding `simscape.multibody.Multibody` object. Component objects have a `DoVisualize` property that specifies whether a component object should be visualized in the 3-D graphical representation. If the property is true, the component object is visualized, otherwise the object is hidden. In a hierarchical `Multibody` or `RigidBody` object, the `DoVisualize` property of the top-level object controls the visualization of the component objects inside the top-level object. If the `DoVisualize` property is false, the component objects contained by the object are hidden.

Only three types of component objects support visualization: `simscape.multibody.Solid`, `simscape.multibody.RigidBody`, and `simscape.multibody.Multibody`. The default value of the `DoVisualize` property is true for these objects. For the other types of component objects, the default value of the `DoVisualize` property is false, and attempting to set the property to true generates an error.

Version History

Introduced in R2022a

See Also

`simscape.multibody.RigidBody` | `simscape.multibody.Multibody` | `simscape.multibody.Solid` | `simscape.multibody.RigidTransform` | `simscape.multibody.WorldFrame` | `simscape.multibody.Joint`

simscape.multibody.Joint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Component`

Abstract base class for joints

Description

`simscape.multibody.Joint` is the abstract base class for joints, such as prismatic joints and revolute joints. To create a joint, use the object of a subclass, such as `simscape.multibody.PrismaticJoint` or `simscape.multibody.RevoluteJoint`.

An object of a joint has two frame connectors called `B` and `F` that correspond to the base and follower frames of the object. The connectors link two arbitrary objects together in a `simscape.multibody.Multibody` object. After being linked, the object connected to the follower frame is able to move in certain ways relative to the object connected to the base frame.

The `Joint` class has a `DegreesOfFreedom` property that indicates how many degrees of freedom are between the base and follower frames of a joint object. The value of the `DegreesOfFreedom` property is the sum of the degrees of freedom of all the joint primitives contained in the joint object.

See the “More About” on page 4-6 section for more information about the `Joint` class.

Class Attributes

`Abstract`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Properties

`DegreesOfFreedom` — Degrees of freedom of joint object

positive integer

Degrees of freedom between the base and follower frames of a joint object, returned as a positive integer. The value is the sum of the degrees of freedom of all the joint primitives contained in the joint object.

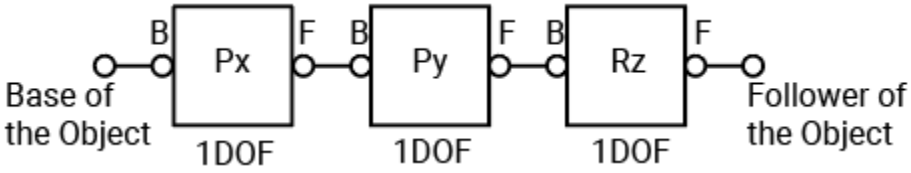
Attributes:

`GetAccess`	`public`
`SetAccess`	`Restricts access`
`Transient`	`true`
`NonCopyable`	`true`

More About

Joint Type

The `simscape.multibody.Joint` class has 17 subclasses that each corresponds to a particular type of joint. A joint object contains zero or more joint primitives arranged in a particular sequence from the base frame to follower frame of the object. For example, the figure shows the joint primitives of a `simscape.multibody.PlanarJoint` object.



A joint object has an associated kinematic map that maps a set of joint variables, such as rotations and translations, to a prescribed transformation between the base and follower frames. The table summarizes the joint primitives and degrees of freedom for different types of joint objects:

Joint Type	Degrees of Freedom	Primitives	Primitive Names
`simscape.multibody.WeldJoint`	0	None	None
`simscape.multibody.RevoluteJoint`	1	1 `simscape.multibody.RevolutePrimitive`	R_z
`simscape.multibody.PrismaticJoint`	1	1 `simscape.multibody.PrismaticPrimitive`	P_z
`simscape.multibody.LeadScrewJoint`	1	1 `simscape.multibody.LeadScrewPrimitive`	LS_z
`simscape.multibody.CylindricalJoint`	2	1 `simscape.multibody.RevolutePrimitive`	R_z
		1 `simscape.multibody.PrismaticPrimitive`	P_z
`simscape.multibody.PinSlotJoint`	2	1 `simscape.multibody.PrismaticPrimitive`	P_x
		2 `simscape.multibody.RevolutePrimitive`	R_z
`simscape.multibody.UniversalJoint`	2	2 `simscape.multibody.RevolutePrimitive`	R_x
			R_y

Joint Type	Degrees of Freedom	Primitives	Primitive Names
simscape.multibody.RectangularJoint	2	2 simscape.multibody.PrismaticPrimitive	P_x
			P_y
simscape.multibody.ConstantVelocityJoint	2	1 simscape.multibody.ConstantVelocityPrimitive	CV
simscape.multibody.GimbalJoint	3	3 simscape.multibody.RevolutePrimitive	R_x
			R_y
			R_z
simscape.multibody.PlanarJoint	3	2 simscape.multibody.PrismaticPrimitive	P_x
			P_y
		1 simscape.multibody.RevolutePrimitive	R_z
simscape.multibody.CartesianJoint	3	3 simscape.multibody.PrismaticPrimitive	P_x
			P_y
			P_z
simscape.multibody.SphericalJoint	3	1 simscape.multibody.SphericalPrimitive	S
simscape.multibody.BearingJoint	4	1 simscape.multibody.PrismaticPrimitive	P_z
		3 simscape.multibody.RevolutePrimitive	R_x
			R_y
			R_z
simscape.multibody.TelescopingJoint	4	1 simscape.multibody.SphericalPrimitive	S
		1 simscape.multibody.PrismaticPrimitive	P_z
simscape.multibody.BushingJoint	6	3 simscape.multibody.PrismaticPrimitive	P_x
			P_y
			P_z
		3 simscape.multibody.RevolutePrimitive	R_x
			R_y
			R_z

Joint Type	Degrees of Freedom	Primitives	Primitive Names
`simscape.multibody.SixDofJoint`	6	3 `simscape.multibody.PrismaticPrimitive`	P_x
			P_y
			P_z
		1 `simscape.multibody.SphericalPrimitive`	S

State Parameters

The state parameters of a joint object include position and velocity. The position can be a translation or rotation, and the velocity is the derivative of the position. It is often necessary to specify a `simscape.multibody.Multibody` object in a particular state for an analysis. For example, you may need to set a multibody system in a certain configuration or simulate the system from a particular initial condition. To specify the configuration or initial condition of a `Multibody` object, use a `simscape.op.OperatingPoint` object to set the targets and the priority level of the targets for the state parameters of some or all the joint objects. See “Examples” on page 4-0 for how to specify the state parameters for a revolute joint object.

If a `Multibody` object has a tree-like structure, the state parameters of all the joint objects can always achieve the specified targets. If a `Multibody` object has a kinematic loop, the state targets of joint objects might be incompatible with each other. In this case, the `computeState` method of `simscape.multibody.CompiledMultibody` class makes compromises to achieve a physically possible state when computing the state of the `Multibody` object.

Version History

Introduced in R2022a

See Also

`simscape.multibody.BearingJoint` | `simscape.multibody.BushingJoint` | `simscape.multibody.CartesianJoint` | `simscape.multibody.ConstantVelocityJoint` | `simscape.multibody.CylindricalJoint` | `simscape.multibody.GimbalJoint` | `simscape.multibody.JointPrimitive` | `simscape.multibody.LeadScrewJoint` | `simscape.multibody.PinSlotJoint` | `simscape.multibody.PlanarJoint` | `simscape.multibody.PrismaticJoint` | `simscape.multibody.RectangularJoint` | `simscape.multibody.RevoluteJoint` | `simscape.multibody.SixDofJoint` | `simscape.multibody.SphericalJoint` | `simscape.multibody.TelescopingJoint` | `simscape.multibody.UniversalJoint` | `simscape.multibody.WeldJoint`

simscape.multibody.BearingJoint class

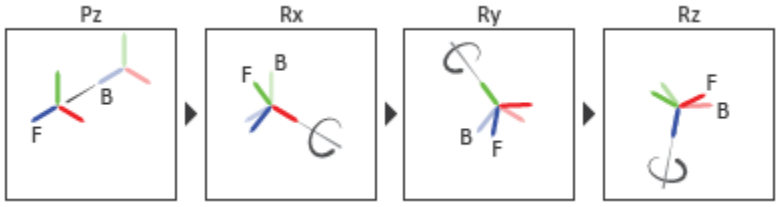
Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct bearing joint

Description

Use an object of the `simscape.multibody.BearingJoint` class to construct a bearing joint. A bearing joint models a compliant bearing that supports a spinning shaft. The bearing can translate along the shaft and allows the rotation about an axis that may not be aligned with the shaft.

A `BearingJoint` object models a transformation of the follower frame with respect to the base frame. The transformation contains a translation and a 3-D rotation. The translation is along the *z*-axis of the base frame. Like the `simscape.multibody.GimbalJoint` object, the `BearingJoint` object has three sequential rotations to achieve a 3-D rotation, as shown in the image.



The properties of the `BearingJoint` object contain one `simscape.multibody.PrismaticPrimitive` object and three `simscape.multibody.RevolutePrimitive` objects.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`bj = simscape.multibody.BearingJoint` constructs a bearing joint with default values.

Properties

Pz — Prismatic primitive along *z*-axis

`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *z*-axis of the base frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to a translation of the follower frame with respect to the base frame along the *z*-axis of the base frame.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Rx — Revolute primitive about *x*-axis

`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *x*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *x*-axis of the follower frame generated after the translation.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Ry — Revolute primitive about *y*-axis

`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *y*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *y*-axis of the follower frame generated after the rotation about the *x*-axis.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Rz — Revolute primitive about *z*-axis

`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *z*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *z*-axis of the follower frame generated after the rotation about the *y*-axis.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.PrismaticPrimitive` | `simscape.multibody.AxialSpringDamper` | `simscape.multibody.TorsionalSpringDamper` | `simscape.multibody.RevolutePrimitive` | `simscape.multibody.RevolutePrimitive`

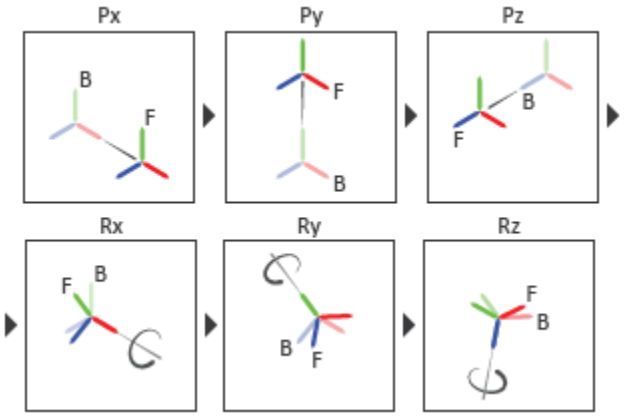
simscape.multibody.BushingJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct bushing joint

Description

Use an object of the `simscape.multibody.BushingJoint` class to construct a bushing joint. You can envision a bushing joint as a Cartesian joint followed by a gimbal joint. The follower frame can have an arbitrary rigid transformation with respect to the base frame. The transformation contains three sequential translations and three sequential rotations, as shown in the image.



The translations are along the *x*, *y*, and *z* axes of the follower frame, respectively. Before the first rotation, the axes of the follower frame are parallel to the corresponding axes of the base frame. The rotations are about the *x*, *y*, and *z* axes of the follower frame.

Similar to the gimbal joint, the bushing joint has a kinematic singularity. See `simscape.multibody.GimbalJoint` for more details.

Class Attributes

`Sealed` `true`

`ConstructOnLoad` `true`

`RestrictsSubclassing` `true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`bj = simscape.multibody.BushingJoint` constructs a bushing joint with default values.

Properties

Px — Prismatic primitive along x-axis

`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *x*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *x*-axis of the follower frame.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Py — Prismatic primitive along y-axis

`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *y*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *y*-axis of the follower frame generated after the translation along the *x*-axis.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Pz — Prismatic primitive along *z*-axis

`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *z*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *z*-axis of the follower frame generated after the translation along the *y*-axis.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Rx — Revolute primitive about x-axis

`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *x*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *x*-axis of the follower frame generated after the translations.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Ry — Revolute primitive about *y*-axis
`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *y*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *y*-axis of the follower frame generated after the rotation about the *x*-axis.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Rz — Revolute primitive about *z*-axis
`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *z*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *z*-axis of the follower frame generated after the rotation about the *y*-axis.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.PrismaticPrimitive` | `simscape.multibody.AxialSpringDamper` | `simscape.multibody.TorsionalSpringDamper` | `simscape.multibody.RevolutePrimitive`

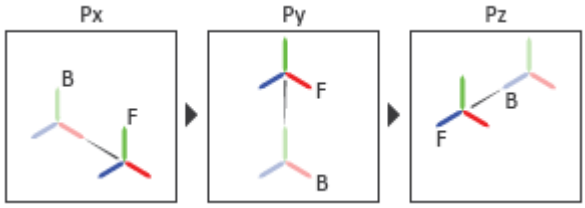
simscape.multibody.CartesianJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct Cartesian joint

Description

Use an object of the `simscape.multibody.CartesianJoint` class to construct a Cartesian joint. The properties of the object contain three `simscape.multibody.PrismaticPrimitive` objects that model a 3-D transformation of the follower frame with respect to the base frame. The transformation contains three sequential translations that are along the *x*, *y*, and *z* axes of the follower frame, respectively, as shown in the image.



The axes of the follower frame are parallel with the corresponding axes of the base frame.

Class Attributes

`Sealed` `true`

`ConstructOnLoad` `true`

`RestrictsSubclassing` `true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`cj = simscape.multibody.CartesianJoint` constructs a Cartesian joint with default values.

Properties

Px — Prismatic primitive along x-axis

`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *x*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *x*-axis of the follower frame.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Py — Prismatic primitive along y-axis
`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *y*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *y*-axis of the follower frame generated after the translation along the *x*-axis.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Pz — Prismatic primitive along *z*-axis
`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *z*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *z*-axis of the follower frame generated after the translation along the *y*-axis.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.PrismaticPrimitive` | `simscape.multibody.AxialSpringDamper`

simscape.multibody.ConstantVelocityJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct constant-velocity joint

Description

Use an object of the `simscape.multibody.ConstantVelocityJoint` class to construct a constant-velocity joint. The `CV` property of the object contains a single `simscape.multibody.ConstantVelocityPrimitive` object that has two degrees of freedom. The `ConstantVelocityJoint` object models a specialized coupling between two shafts such that the spin rates of the shafts are exactly matched even when the shafts are not aligned.

During simulation, the origins of the follower and base frames remain coincident. To specify the position and velocity targets for the bend angle and azimuth of a constant-velocity joint, use a `simscape.op.OperatingPoint` object. Note that you can specify the targets only for the bend angle, but only specifying targets for the azimuth is not allowed.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`cvj = simscape.multibody.ConstantVelocityPrimitive` constructs a constant-velocity joint object with default values.

Properties

`CV` — Constant-velocity primitive

`simscape.multibody.ConstantVelocityPrimitive`

Constant-velocity primitive of a `simscape.multibody.ConstantVelocityJoint` object, returned as a `simscape.multibody.ConstantVelocityPrimitive` object.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Examples

Specify the State Targets for a Constant-Velocity Joint

1. Use a `simscape.multibody.ConstantVelocityJoint` object to create a constant-velocity joint.

   ```
   joint = simscape.multibody.ConstantVelocityJoint;
   ```
2. Before specifying the state targets of the `ConstantVelocityJoint` object, you need to add the object into a `simscape.multibody.Multibody` object using the `addComponent` method. Create a `Multibody` object.

   ```
   mb = simscape.multibody.Multibody;
   addComponent(mb,"MyJoint",joint);
   ```
3. To specify the targets of the object, create an empty operating point.

   ```
   op = simscape.op.OperatingPoint;
   ```
4. Add high-priority position targets for the bend angle and azimuth to the operating point.

   ```
   op("MyJoint/CV/q/b") = simscape.op.Target(simscape.Value(15,"deg"),"High");
   op("MyJoint/CV/q/a") = simscape.op.Target(simscape.Value(90,"deg"),"High");
   ```
5. Add high-priority velocity targets for the bend angle and azimuth to the operating point.

   ```
   op("MyJoint/CV/w/b") = simscape.op.Target(simscape.Value(30,"deg/s"),"High");
   op("MyJoint/CV/w/a") = simscape.op.Target(simscape.Value(50,"deg/s"),"High");
   ```

Version History

Introduced in R2022a

See Also

`simscape.multibody.ConstantVelocityPrimitive` | `simscape.multibody.Joint`

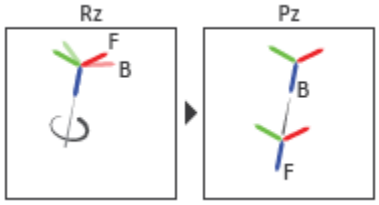
simscape.multibody.CylindricalJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct cylindrical joint

Description

Use an object of the `simscape.multibody.CylindricalJoint` class to construct a cylindrical joint. The *z*-axes of the follower and base frames are aligned, and the follower frame can rotate about and move along the *z*-axis of the base frame, as shown in the image.



The properties of the `CylindricalJoint` object contain a `simscape.multibody.PrismaticPrimitive` object and a `simscape.multibody.RevolutePrimitive` object.

Class Attributes

`Sealed` `true`

`ConstructOnLoad` `true`

`RestrictsSubclassing` `true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`cj = simscape.multibody.CylindricalJoint` constructs a cylindrical joint with default values.

Properties

Rz — Revolute primitive about *z*-axis

`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *z*-axis of the base frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *z*-axis of the base frame.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Pz — Prismatic primitive along *z*-axis
`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *z*-axis of the base frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *z*-axis of the base frame.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint`

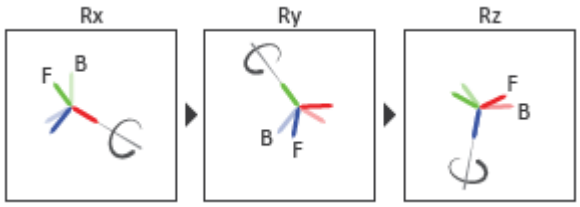
simscape.multibody.GimbalJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct gimbal joint

Description

Use an object of the `simscape.multibody.GimbalJoint` class to construct a gimbal joint. The properties of the object contain three `simscape.multibody.RevolutePrimitive` objects that model a 3-D rotation of the follower frame with respect to the base frame. To achieve the 3-D rotation, the gimbal joint has three sequential rotations, as shown in the image.



The first rotation is about the *x*-axis of the follower frame, the second rotation is about the *y*-axis of the follower frame generated after the first rotation, and the third rotation is about the *z*-axis of the follower frame generated after the second rotation.

Gimbal joints have a kinematic singularity at configurations in which the second rotation is positive or negative 90 degrees. In these configurations, the first and third rotations axes are aligned and the joint looses a degree of freedom.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`gj = simscape.multibody.GimbalJoint` constructs a gimbal joint with default values.

Properties

Rx — Revolute primitive about x-axis

`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *x*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *x*-axis of the follower frame.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Ry — Revolute primitive about *y*-axis
`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *y*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *y*-axis of the follower frame generated after the rotation about the *x*-axis.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Rz — Revolute primitive about *z*-axis
`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *z*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *z*-axis of the follower frame generated after the rotation about the *y*-axis.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.TorsionalSpringDamper` | `simscape.multibody.RevolutePrimitive` | `simscape.multibody.RevolutePrimitive`

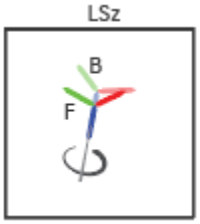
simscape.multibody.LeadScrewJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct lead-screw joint

Description

Use an object of the `simscape.multibody.LeadScrewJoint` class to construct a lead-screw joint. The `LSz` property of the object contains a single `simscape.multibody.LeadScrewPrimitive` object that has one degree of freedom, as shown in the image.



During a simulation, the follower frame translates along the *z*-axis of the base frame while the follower frame rotates about the same *z*-axis. The *z*-axes of the base and follower frames are aligned. The translation is proportional to the rotation based on the value of the `Lead` and `Direction` properties of the `LeadScrewPrimitive` object.

To specify the state targets of a `LeadScrewJoint` object, use a `simscape.op.OperatingPoint` object. Note that you can specify either the linear or angular targets for the object. Specifying both linear and angular targets for one object generates an error.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`lsj = simscape.multibody.LeadScrewJoint` constructs a lead-screw joint with default values.

Properties

LSz — Lead-screw primitive of joint

`simscape.multibody.LeadScrewPrimitive` object

Lead-screw primitive of the `simscape.multibody.LeadScrewJoint` joint, returned as a `simscape.multibody.LeadScrewPrimitive` object.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Examples

Specify the Lead, Direction, and State Targets for a Lead-Screw Joint

1. Use a `simscape.multibody.LeadScrewJoint` object to create a lead-screw joint.

   ```
   joint = simscape.multibody.LeadScrewJoint;
   ```
2. Set the lead and direction for the joint.

   ```
   joint.LSz.Lead = simscape.Value(3,"mm/rev");
   joint.LSz.Direction = simscape.multibody.Handedness.LeftHand;
   ```
3. Before specifying the state targets of the `LeadScrewJoint` object, you need to add the object into a `simscape.multibody.Multibody` object by using the `addComponent` method. Create a `Multibody` object.

   ```
   mb = simscape.multibody.Multibody;
   addComponent(mb,"MyJoint",joint);
   ```
4. To specify the targets of the object, create an empty operating point.

   ```
   op = simscape.op.OperatingPoint;
   ```
5. Add a high-priority angular position target for the lead-screw primitive to the operating point.

   ```
   op("MyJoint/LSz/q") = simscape.op.Target(simscape.Value(30,"deg"),"High");
   ```

   Alternatively, you can add a linear position target. Note that the angular and linear targets must not be added to the same object.

   ```
   op("MyJoint/LSz/p") = simscape.op.Target(simscape.Value(50,"mm"),"High");
   ```
6. Add a high-priority angular velocity target for the lead-screw primitive to the operating point.

   ```
   op("MyJoint/LSz/w") = simscape.op.Target(simscape.Value(3,"deg/s"),"High");
   ```

   Alternatively, you can add a linear velocity target.

   ```
   op("MyJoint/LSz/v") = simscape.op.Target(simscape.Value(5,"mm/s"),"High")
   ```

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.LeadScrewPrimitive`

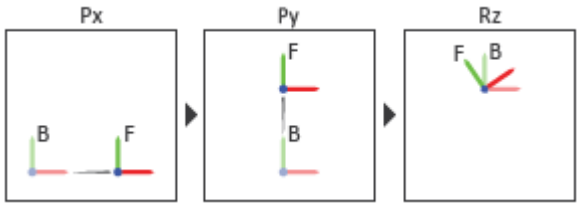
simscape.multibody.PlanarJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct planar joint

Description

Use an object of the `simscape.multibody.PlanarJoint` class to construct a planar joint. The object models a 2-D transformation of the follower frame with respect to the base frame in the *xy*-plane of the base frame. The transformation includes two translations and one rotation that follow the sequence, as shown in the image.



First, the follower frame moves along the *x* and *y* axes of the base frame, respectively, and then rotates about the *z*-axis of the follower frame generated after the translations. The origin of the follower frame lies on the *xy*-plane of the base frame, and the *z*-axes of the base and follower frames remain parallel.

The properties of the `PlanarJoint` object contain two `simscape.multibody.PrismaticPrimitive` objects and one `simscape.multibody.RevolutePrimitive` object.

Class Attributes

```
Sealed                                    true

ConstructOnLoad                                   true

RestrictsSubclassing                                  true
```

For information on class attributes, see “Class Attributes”.

Creation

Description

`pj = simscape.multibody.PlanarJoint` constructs a planar joint with default values.

Properties

Px — Prismatic primitive along x-axis

`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *x*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *x*-axis of the follower frame.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Py — Prismatic primitive along y-axis

`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *y*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *y*-axis of the follower frame generated after the translation along the *x*-axis.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Rz — Revolute primitive about z-axis

`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *z*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *z*-axis of the follower frame generated after the translations.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.PrismaticPrimitive` | `simscape.multibody.AxialSpringDamper` | `simscape.multibody.RevolutePrimitive` | `simscape.multibody.TorsionalSpringDamper`

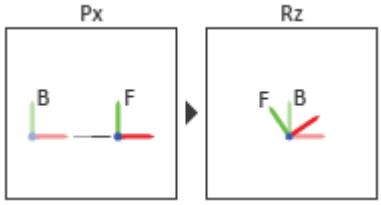
simscape.multibody.PinSlotJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct pin-slot joint

Description

Use an object of the `simscape.multibody.PinSlotJoint` class to construct a pin-slot joint. The `PinSlotJoint` object has two degrees of freedom. The follower frame moves along the *x*-axis of the base frame and then rotates about the *z*-axis of the follower frame, as shown in the image.



The properties of the `PinSlotJoint` object contains a `simscape.multibody.PrismaticPrimitive` object and a `simscape.multibody.RevolutePrimitive` object.

Class Attributes

`Sealed` `true`

`ConstructOnLoad` `true`

`RestrictsSubclassing` `true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`psj = simscape.multibody.PinSlotJoint` constructs a pin-slot joint with default values.

Properties

Px — Prismatic primitive along x-axis

`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *x*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *x*-axis of the follower frame.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Rz — Revolute primitive about z-axis
`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *z*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *z*-axis of the follower frame.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.PrismaticPrimitive` | `simscape.multibody.RevolutePrimitive`

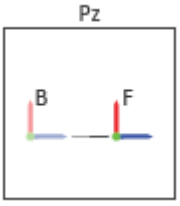
simscape.multibody.PrismaticJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct prismatic joint

Description

Use an object of the `simscape.multibody.PrismaticJoint` class to construct a prismatic joint. The `Pz` property of the object contains a single `simscape.multibody.PrismaticPrimitive` object that has one translational degree of freedom along the *z*-axis of the base frame, as shown in the image.



Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`pj = simscape.multibody.PrismaticJoint` constructs a prismatic joint with default values.

Properties

Pz — Prismatic primitive along *z*-axis

`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *z*-axis of the base frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to a translation of the follower frame with respect to the base frame along the *z*-axis of the base frame.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Examples

Specify the Force Law and State Targets for a Prismatic Joint

1. Use a `simscape.multibody.PrismaticJoint` object to create a prismatic joint.

```
joint = simscape.multibody.PrismaticJoint;
```

2. Use a `simscape.multibody.AxialSpringDamper` object to create an axial force law.

```
asd = simscape.multibody.AxialSpringDamper;
```

3. Specify the equilibrium position, spring stiffness, and damping coefficient of the force law by using `simscape.Value` objects.

```
asd.EquilibriumPosition = simscape.Value(20,"cm");
asd.SpringStiffness = simscape.Value(0.1,"N/cm");
asd.DampingCoefficient = simscape.Value(5e-3,"N/(cm/s)");
```

4. Assign the force law to the `PrismaticJoint` object.

```
joint.Pz.ForceLaws = asd;
```

5. Before specifying the state targets of the `PrismaticJoint` object, you need to add the object into a `simscape.multibody.Multibody` object by using the `addComponent` method. Create a `Multibody` object.

```
mb = simscape.multibody.Multibody;
addComponent(mb,"MyJoint",joint);
```

6. To specify the targets of the object, create an empty operating point.

```
op = simscape.op.OperatingPoint;
```

7. Add a high-priority position target for the prismatic primitive to the operating point. Note that if two targets are incompatible, the priority level determines which of the targets to satisfy.

```
op("MyJoint/Pz/q") = simscape.op.Target(simscape.Value(50,"cm"),"High");
```

8. Add a low-priority velocity target for the prismatic primitive to the operating point.

```
op("MyJoint/Pz/w") = simscape.op.Target(simscape.Value(10,"cm/s"),"Low");
```

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.PrismaticPrimitive` | `simscape.multibody.AxialSpringDamper`

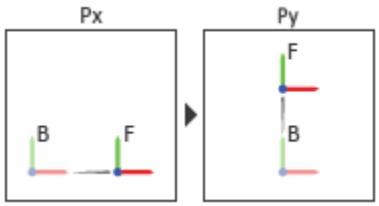
simscape.multibody.RectangularJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct rectangular joint

Description

Use an object of the `simscape.multibody.RectangularJoint` class to construct a rectangular joint. The properties of the object contain two `simscape.multibody.PrismaticPrimitive` objects that model a 2-D transformation of the follower frame with respect to the base frame within the *xy*-plane of the base frame. The transformation contains two sequential translations that are along the *x* and *y* axes of the follower frame, respectively, as shown in the image.



The `RectangularJoint` object constrains the origin of the follower frame within the *xy*-plane of the base frame, and the axes of the follower frame are parallel with the corresponding axes of the base frame.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`rj = simscape.multibody.RectangularJoint` constructs a rectangular joint with default values.

Properties

Px — Prismatic primitive along x-axis

`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *x*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *x*-axis of the follower frame.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Py — Prismatic primitive along y-axis
`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *y*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *y*-axis of the follower frame generated after the translation along the *x*-axis.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.PrismaticPrimitive` | `simscape.multibody.AxialSpringDamper`

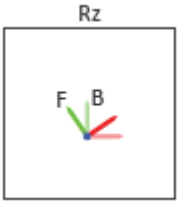
simscape.multibody.RevoluteJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct revolute joint

Description

Use an object of the `simscape.multibody.RevoluteJoint` class to construct a revolute joint. The `Rz` property of the object contains a single `simscape.multibody.RevolutePrimitive` object that has one rotational degree of freedom about the *z*-axis of the base frame, as shown in the image.



Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`rj = simscape.multibody.RevoluteJoint` constructs a revolute joint with default values.

Properties

Rz — Revolute primitive about *z*-axis

`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *z*-axis of the base frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to a rotation of the follower frame with respect to the base frame about the *z*-axis of the base frame.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Examples

Specify the Force Law and State Targets for a Revolute Joint

1. Use a `simscape.multibody.RevoluteJoint` object to create a revolute joint.

   ```
   joint = simscape.multibody.RevoluteJoint;
   ```
2. Use a `simscape.multibody.TorsionalSpringDamper` object to create a force law for the `RevoluteJoint` object.

   ```
   tsd = simscape.multibody.TorsionalSpringDamper;
   ```
3. Specify the equilibrium position, spring stiffness, and damping coefficient of the force law by using `simscape.Value` objects.

   ```
   tsd.EquilibriumPosition = simscape.Value(90,"deg");
   tsd.SpringStiffness = simscape.Value(0.1,"N*cm/deg");
   tsd.DampingCoefficient = simscape.Value(5e-3,"N*cm/(deg/s)");
   ```
4. Assign the specified force law to the `RevoluteJoint` object.

   ```
   joint.Rz.ForceLaws = tsd;
   ```
5. Before specifying the state targets of the `RevoluteJoint` object, you need to add the object into a `simscape.multibody.Multibody` object by using the `addComponent` method. Create a `Multibody` object.

   ```
   mb = simscape.multibody.Multibody;
   addComponent(mb,"MyJoint",joint);
   ```
6. To specify the targets of the object, create an empty operating point.

   ```
   op = simscape.op.OperatingPoint;
   ```
7. Add a high-priority position target for the primitive to the operating point. Note that if two targets are incompatible, the priority level determines which of the targets to satisfy.

   ```
   op("MyJoint/Rz/q") = simscape.op.Target(simscape.Value(30,"deg"),"High");
   ```
8. Add a low-priority velocity target for the primitive to the operating point.

   ```
   op("MyJoint/Rz/w") = simscape.op.Target(simscape.Value(150,"deg/s"),"Low");
   ```

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.RevolutePrimitive` | `simscape.multibody.TorsionalSpringDamper`

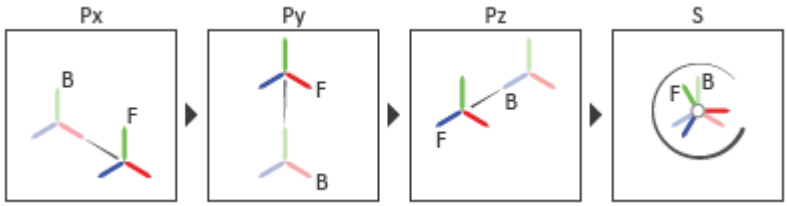
simscape.multibody.SixDofJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct bushing joint

Description

Use an object of the `simscape.multibody.SixDofJoint` class to construct a joint with six degrees of freedom. You can envision a six-DOF joint as a Cartesian joint followed by a spherical joint. The follower frame can have a 3-D transformation with respect to the base frame. The transformation contains three sequential translations and an unconstrained 3-D rotation, as shown in the image.



The translations are along the *x*, *y*, and *z* axes of the base frame, respectively. Before the rotation, the axes of the follower are parallel to the corresponding axes of the base frame. The 3-D rotation is with respect to the follower frame formed after the translations.

The properties of the `SixDofJoint` object contains three `simscape.multibody.PrismaticPrimitive` objects and one `simscape.multibody.SphericalPrimitive` object.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`sj = simscape.multibody.SixDofJoint` constructs a six-DOF joint with default values.

Properties

Px — Prismatic primitive along x-axis

`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *x*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *x*-axis of the follower frame.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Py — Prismatic primitive along y-axis
`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *y*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *y*-axis of the follower frame generated after the translation along the *x*-axis.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Pz — Prismatic primitive along *z*-axis
`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *z*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *z*-axis of the follower frame generated after the translation along the *y*-axis.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

S — Spherical primitive
`simscape.multibody.SphercialPrimitive` object

Spherical primitive, returned as a `simscape.multibody.SphercialPrimitive` object.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.PrismaticPrimitive` | `simscape.multibody.AxialSpringDamper` | `simscape.multibody.TorsionalSpringDamper`

| simscape.multibody.RevolutePrimitive | simscape.multibody.RevolutePrimitive |
simscape.multibody.SphericalPrimitive | simscape.multibody.SphericalJoint

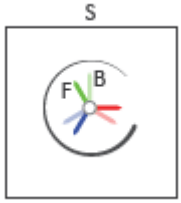
simscape.multibody.SphericalJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct spherical joint

Description

Use an object of the `simscape.multibody.SphericalJoint` class to construct a spherical joint. The `S` property of the object has a single `simscape.multibody.SphericalPrimitive` object that models an arbitrary 3-D rotation of the follower frame with respect to the base frame. During a simulation, the origins of the base and follower frames remain coincident, as shown in the image.



Unlike a gimbal joint, the follower frame of a spherical joint can rotate arbitrarily with respect to the base frame and has no kinematic singularity.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`sj = simscape.multibody.SphericalJoint` constructs a spherical joint with default values.

Properties

S — Spherical joint primitive

`simscape.multibody.SphercialPrimitive` object

Spherical joint primitive of a `simscape.multibody.SphericalJoint` object, returned as a `simscape.multibody.SphercialPrimitive` object.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Examples

Specify the Force Law and State Targets for a Spherical Joint

1. Use a `simscape.multibody.SphericalJoint` object to create a spherical joint.

```
joint = simscape.multibody.SphericalJoint;
```

2. Use a `simscape.multibody.SphericalSpringDamper` object to add a force law for the `SphericalJoint` object.

```
ssd = simscape.multibody.SphericalSpringDamper;
```

3. Specify the equilibrium position, spring stiffness, and damping coefficient of the force law by using `simscape.Value` objects.

```
ssd.EquilibriumPosition = simscape.multibody.ArbitraryAxisRotation(...
            simscape.Value(45,"deg"),[1 1 1]);
ssd.SpringStiffness = simscape.Value(0.1,"N*cm/deg");
ssd.DampingCoefficient = simscape.Value(5e-3,"N*cm/(deg/s)");
```

4. Assign the force law to the `SphericalJoint` object.

```
joint.S.ForceLaws = ssd;
```

5. Before specifying the state targets of the `SphericalJoint` object, you need to add the object into a `simscape.multibody.Multibody` object using the `addComponent` method. Create a `Multibody` object.

```
mb = simscape.multibody.Multibody;
addComponent(mb,"MyJoint",joint);
```

6. To specify the targets of the object, create an empty operating point.

```
op = simscape.op.OperatingPoint;
```

7. Add a high-priority position target for the spherical primitive to the operating point. The rotation is based on the *x*-*y*-*z* rotation sequence.

```
rot = simscape.multibody.RotationSequenceRotation(simscape.multibody.FrameSide.Follower,...
              simscape.multibody.AxisSequence.XYZ,simscape.Value([5 10 60],"deg"));
op("MyJoint/S/angle_axis/ax") = simscape.op.Target(simscape.Value(naturalAxis(rot)),"High");
op("MyJoint/S/angle_axis/q") = simscape.op.Target(naturalAngle(rot),"High");
```

Alternatively, you can use the rotation axis and angle to specify the position target for the spherical primitive.

```
op("MyJoint/S/angle_axis/ax") = simscape.op.Target(simscape.Value([1 -2 3]),"High");
op("MyJoint/S/angle_axis/q")  = simscape.op.Target(simscape.Value(45,"deg"),"High");
```

8. Add a high-priority velocity target for the spherical primitive to the operating point. The angular velocity vector is resolved in the follower frame of the `simscape.multibody.SphericalPrimitive` object.

```
target = simscape.op.Target(simscape.Value([20 -30 50],"deg/s"),"High");
target.Attributes("ResolutionFrame") = "Follower";
op("MyJoint/S/w") = target;
```

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.SphericalPrimitive` | `simscape.multibody.SphericalSpringDamper`

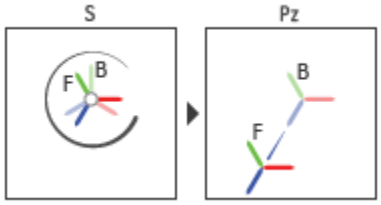
simscape.multibody.TelescopingJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct telescoping joint

Description

Use an object of the `simscape.multibody.TelescoptingJoing` class to construct a telescoping joint. The object models a transformation between the follower frame with respect to the base frame. The transformation includes an unconstrained 3-D rotation, followed by a translation along the *z*-axis of the follower frame generated after the rotation.



The properties of the `TelescoptingJoing` object contain a `simscape.multibody.SphericalPrimitive` object and a `simscape.multibody.PrismaticPrimitive` object.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`tj = simscape.multibody.TelescopingJoint` constructs a telescoping joint with default values.

Properties

S — Spherical primitive

`simscape.multibody.SphericalPrimitive` object

Spherical primitive, returned as a `simscape.multibody.SphericalPrimitive` object. The spherical joint primitive corresponds to an unconstrained 3-D rotation of the follower frame with respect to the base frame.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Pz — Prismatic primitive along z-axis
`simscape.multibody.PrismaticPrimitive` object

Prismatic primitive along the *z*-axis of the follower frame, returned as a `simscape.multibody.PrismaticPrimitive` object. The prismatic primitive corresponds to the translation of the follower frame with respect to the base frame along the *z*-axis of the follower frame generated after the rotation.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.SphericalPrimitive` | `simscape.multibody.SphericalSpringDamper`

simscape.multibody.UniversalJoint class

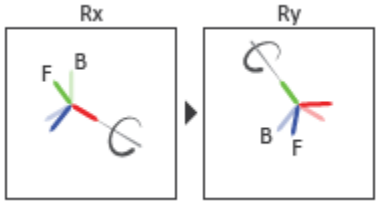
Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct universal joint

Description

Use an object of the `simscape.multibody.UniversalJoint` class to construct a universal joint. Similar to the `simscape.multibody.ConstantVelocityJoint` object, the `UniversalJoint` object models a coupling between two spinning shafts. The two shafts align with the *z*-axes of the base and follower frames and rotate at the same average angular velocity. However, the relative velocities of the shafts fluctuate if the shafts are not aligned. The fluctuation increases as the bend angle increases.

The properties of the `UniversalJoint` object contain two `simscape.multibody.RevolutePrimitive` objects that represent two sequential rotations. The rotations specify the orientation of the follower frame with respect to the base frame. The first rotation is about the *x*-axis of the follower frame, and the second rotation is about the *y*-axis of the follower frame generated after the first rotation, as shown in the image.



Note that the origins of the base and follower frames remain coincident.

Class Attributes

```
Sealed                                          true

ConstructOnLoad                                              true

RestrictsSubclassing                                            true
```

For information on class attributes, see “Class Attributes”.

Creation

Description

`uj = simscape.multibody.UniversalJoint` constructs a universal joint with default values.

Properties

Rx — Revolute primitive about x-axis

`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *x*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *x*-axis of the follower frame.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Ry — Revolute primitive about *y*-axis

`simscape.multibody.RevolutePrimitive` object

Revolute primitive about the *y*-axis of the follower frame, returned as a `simscape.multibody.RevolutePrimitive` object. The revolute primitive corresponds to the rotation of the follower frame with respect to the base frame about the *y*-axis of the follower frame generated after the rotation about the *x*-axis.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint` | `simscape.multibody.ConstantVelocityJoint`

simscape.multibody.WeldJoint class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Joint`

Construct weld joint

Description

Use an object of the `simscape.multibody.WeldJoint` class to construct a weld joint. The object contains no primitives and has zero degrees of freedom. A weld joint constrains the base and follower frames to be coincident and aligned.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`wj = simscape.multibody.WeldJoint` constructs a weld joint.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Joint`

simscape.multibody.Multibody class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Component`

Specify structure of multibody system

Description

Use an object of the `simscape.multibody.Multibody` class to construct a multibody system. A `Multibody` object is a hierarchical container that can have any type of component object, and each component object represents a part or a subsystem of the multibody system. See `simscape.multibody.Component` for more information about different component objects.

By default, a newly created `Multibody` object is empty. You can use the methods of the `Multibody` object to construct a multibody system, prepare the `Multibody` object for analyses, or create a corresponding Simulink model. See the “More About” on page 4-47 section for more information about the `Multibody` class.

The `simscape.multibody.Multibody` class is a `handle` class.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`HandleCompatible`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`mb = simscape.multibody.Multibody` creates an empty `simscape.multibody.Multibody` object.

Properties

`ComponentNames` — Names of top-level component objects in multibody system

[ ] (default) | string array

Names of the component objects at the top level of the `simscape.multibody.Multibody` object, returned as a string array.

Example: `"Base_Bar"`

Attributes:

`GetAccess`	`public`
`SetAccess`	`Restricts access`
`NonCopyable`	`true`
`Transient`	`true`

`Gravity` — Gravitational acceleration

`[0 0 -9.80655]'` : `m/s^2` (default) | `simscape.Value` object

Gravitational acceleration in the multibody system, specified as a `simscape.Value` object that represents a 3-by-1 or 1-by-3 vector with a unit of linear acceleration. The elements of the vector specify the gravity in the *x*, *y*, and *z* directions of the world frame. In a hierarchical `simscape.multibody.Multibody` object, the `Gravity` property of the higher-level `Multibody` object overrides the `Gravity` property of its contained `Multibody` objects.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`
`Transient`	`true`

Methods

Public Methods

addComponent	Add component object to Multibody object
addConnector	Add connector to Multibody object
compile	Compile Multibody object
component	Extract component object from Multibody object
componentPaths	Return paths of component objects in Multibody object
connect	Link two connectors in Multibody object
connectVia	Link two connectors in Multibody object via intermediate object
jointPrimitivePaths	Return paths of joint primitives in Multibody object at all hierarchical levels
makeBlockDiagram	Create Simulink model from Multibody object
removeConnector	Remove existing connector from Multibody object

Specialized Operators and Functions

These methods specialize standard MATLAB operators and functions for objects in this class.

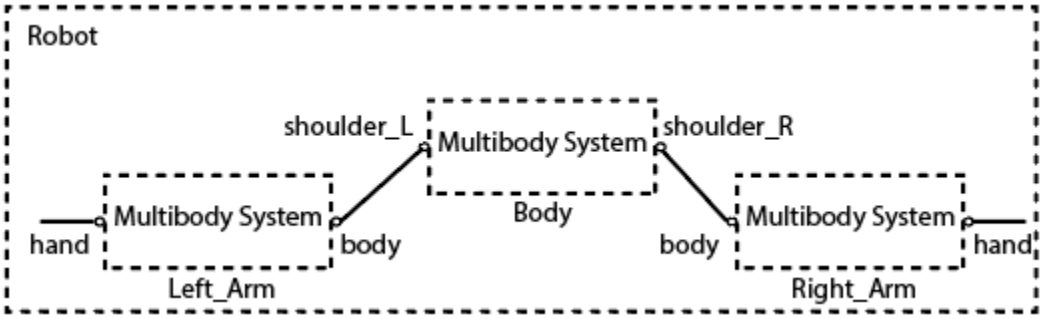
`disp`	Prints the name and type of the connectors and components in the top level of the `simscape.multibody.Multibody` object.

More About

Construction of a Mechanical System

By default, a newly created `simscape.multibody.Multibody` object is empty and has zero connectors. To construct a multibody system, use the `addComponent` method to add objects to the `Multibody` object, then connect the objects using the `connect` or `connectVia` method. Each object represents a part or a subsystem of the multibody system.

To manage the complexity of a large multibody system, you can use several `Multibody` objects to model the subsystems of a system. The image shows an example of a large multibody system, `Robot`.



Note that you must completely construct a `Multibody` object before adding it to a system. For example, in this example, you need to add the connectors `hand` and `body` to the `Left_Arm` object before adding it to the `Robot` object. To add the connectors, use the `addConnector` method.

To establish mechanical relationships between the three subsystems in the `Robot` system, connect the `Body`, `Left_Arm`, `Right_Arm` objects, as shown in the image, by using the `connect` or `connectVia` method.

Compilation

To analyze a multibody system, you need to successfully compile the `simscape.multibody.Multibody` object of the system by using the `compile` method. The compilation performs a thorough error checking for a `Multibody` object. If any error occurs, the `compile` method shows the problem. For example, if a `Multibody` object contains joint objects or `simscape.multibody.RigidTransform` objects that do not connect to the main structure of the `Multibody` object, the compilation fails. See `compile` and `simscape.multibody.CompiledMultibody` for more information about compilation and analysis methods.

Creating Simulink Models

To use the capabilities in Simscape Multibody that you cannot do programmatically, such as simulating a multibody system, create a Simulink model from a `simscape.multibody.Multibody` object by using the `makeBlockDiagram` method.

When you construct a multibody system programmatically, Simscape Multibody has a different approach to specify the state of the joint primitives in the system. Instead of specifying the state through joint targets in joint blocks, the programmatic way uses the `simscape.op.OperatingPoint` object to specify state targets for joint primitives. Therefore, when you create a Simulink model from a `Multibody` object, the `makeBlockDiagram` method must have both the `OperatingPoint` object that specifies the joint states of the system and the `Multibody` object that defines the structure of the system. The `makeBlockDiagram` method uses the data in the `OperatingPoint` object to populate the joint target parameters for the joint blocks of the created Simulink model. Note that if the `OperatingPoint` object targets some joint primitives that do not exist in the `Multibody` object, those targets are ignored.

Tip When setting up the targets of joint primitives, use the `jointPrimitivePaths` method to display the paths of primitives in `Multibody` object.

After you create the Simulink model from a `Multibody` object, the model can be modified just like general Simulink models. However, a best practice is to avoid making changes in the Simulink model that you can make in the `Multibody` object. Make the changes in the code that generates the `Multibody` object, re-generate the object, and then re-create the Simulink model. This process allows you to keep the code as the main source for the model. However, some capabilities are not supported in the programmatic way, and so must be done in Simulink model. For example, you can specify input signals only in the Simulink model.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.CompiledMultibody`

Topics

“Creating a Mobile Robot using a MATLAB App”
“Creating a Robotic Gripper Multibody in MATLAB”
“Creating a Four Bar Multibody Mechanism in MATLAB”
“Creating a Simple Pendulum in MATLAB”
“Create a Mechanism with Different Joints in MATLAB”
“How to Build a Multibody System in MATLAB”

addComponent

Class: `simscape.multibody.Multibody`
Package: `simscape.multibody`

Add component object to `Multibody` object

Syntax

```
addComponent(mb,componentName,C)
```

Description

`addComponent(mb,componentName,C)` adds a component object, `C`, to the top level of the `simscape.multibody.Multibody` object, `mb`. Use the `componentName` argument to specify the name of the new component object in the `mb` object.

Note that the `addComponent` method adds only a copy of a component object to a `Multibody` object, and any subsequent changes to the original component object do not affect the copy. Therefore, you must completely configure a component object before adding the object to a `Multibody` object.

By default, the newly added component object does not connect to other objects in the `Multibody` object. See `addConnector`, `connect`, and `connectVia` to learn how to connect component objects in a `Multibody` object.

Input Arguments

`mb` — Multibody system

`simscape.multibody.Multibody` object

Multibody system, specified as a `simscape.multibody.Multibody` object.

`componentName` — Name of new component object

string scalar | character vector

Name of the new component object in the `Multibody` object, specified as a string scalar or character vector. The name must be unique among the names of component objects at the top level of the `Multibody` object. In addition, the name must be a valid MATLAB identifier. You can use the `isvarname` function to check if the name is a valid identifier.

Example: "NE_Lower_Joint"

`C` — Component object

object of subclass of `simscape.multibody.Compoment` class

Component object to add to the `simscape.multibody.Multibody` object, specified as an object of a subclass of the `simscape.multibody.Component` abstract class, such as `simscape.multibody.Multibody`, `simscape.multibody.RigidBody`, or `simscape.multibody.RigidTransform`.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.Multibody`

addConnector

Class: `simscape.multibody.Multibody`
Package: `simscape.multibody`

Add connector to Multibody object

Syntax

```
addConnector(mb,connectorName,internalPath)
```

Description

`addConnector(mb,connectorName,internalPath)` adds a connector named `connectorName` to the `simscape.multibody.Multibody` object, `mb`. The new connector corresponds to a frame connector of a component object that is at the top level of the `mb` object. The `internalPath` argument is the path for the connector of the internal object. The path has two elements, the first element is the name of the component object, and the second element is the connector name.

Note that you can add multiple connectors that link to the same internal frame connecter when the frame connector serves multiple purposes in the model.

Input Arguments

`mb` — Multibody system

`simscape.multibody.Multibody` object

Multibody system, specified as a `simscape.multibody.Multibody` object.

`connectorName` — Name of new connector

string scalar | character vector

Name of the new connector, specified as a string scalar or character vector. The connector name can be different than the name of the corresponding internal frame connector. The name must be unique among the names of all the connectors on the `Multibody` object. In addition, the name must be a valid MATLAB identifier. You can use the `isvarname` function to check if the name is a valid identifier.

Example: `"elbow"`

`internalPath` — Path of frame connector on internal component object

string scalar | character vector

Path of a frame connector on a component object located at the top level of the `simscape.multibody.Multibody` object, specified as a string scalar or character vector. The path has two elements separated by a forward slash. The first element is the name of the component object, and the second element is the name of the frame connector.

Example: `"UpperArm/elbow"`

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Multibody`

compile

Class: `simscape.multibody.Multibody`
Package: `simscape.multibody`

Compile `Multibody` object

Syntax

```
cmb = compile(mb)
```

Description

`cmb = compile(mb)` compiles the `simscape.multibody.Multibody` object, `mb`, and returns a `simscape.multibody.CompiledMultibody` object if the compilation is successful. The compilation process evaluates all parts of the `mb` object and checks for semantic correctness.

To perform analyses, such as computing the position of a joint primitive, for a multibody system, the corresponding `Multibody` object of the system needs to be successfully compiled.

Input Arguments

`mb` — Multibody system

`simscape.multibody.Multibody` object

Multibody system, specified as a `simscape.multibody.Multibody` object.

Output Arguments

`cmb` — Compiled multibody system

`simscape.multibody.CompiledMultibody` object

Compiled multibody system, returned as a `simscape.multibody.CompiledMultibody` object that you can use for many analyses, such as `visualize` and `transformation`. See `simscape.multibody.CompiledMultibody` for more information.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.CompiledMultibody` | `simscape.multibody.Multibody`

component

Class: `simscape.multibody.Multibody`
Package: `simscape.multibody`

Extract component object from `Multibody` object

Syntax

```
C = component(mb,path)
```

Description

`C = component(mb,path)` extracts a component object, `C`, from the `simscape.multibody.Multibody` object, `mb`. The `path` argument is the path of the component object in the `mb` object and can have more than one elements. In other words, the `component` method can extract an object from any hierarchical level of the `mb` object.

The extracted object is a copy of the original component object. Therefore, any subsequent changes to the extracted copy do not affect the original component object.

Input Arguments

`mb` — Multibody system

`simscape.multibody.Multibody` object

Multibody system, specified as a `simscape.multibody.Multibody` object.

`path` — Path of component object

string scalar | character vector

Path of the component object, specified as a string scalar or character vector. The length of the path depends on the hierarchical level of the component object in the `Multibody` object. If the object is at the top level, the path has one element. If the component object is in an inner object that is at the top level hierarchy, the path has two elements separated by a forward slash. The first element is the name of the inner object, and the second element is the name of desired component object.

Example: "Suspension/Front_Left/Shock/Prismatic_Joint"

Output Arguments

`C` — Copy of component object

object of subclass of `simscape.multibody.Component` class

Copy of the component object, returned as an object of a subclass of `simscape.multibody.Component` abstract class. Modifying the extracted component object does not affect the original component object.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

Topics

`simscape.multibody.Multibody`

componentPaths

Class: `simscape.multibody.Multibody`
Package: `simscape.multibody`

Return paths of component objects in `Multibody` object

Syntax

```
paths = componentPaths(mb)
```

Description

`paths = componentPaths(mb)` returns the paths of the component objects in the entire hierarchy of the `simscape.multibody.Multibody` object, `mb`.

Input Arguments

`mb` — Multibody system

`simscape.multibody.Multibody` object

Multibody system, specified as a `simscape.multibody.Multibody` object.

Output Arguments

`paths` — Paths of component objects

string scalar | character vector

Paths of the component objects in the entire hierarchy of the `Multibody` object, returned as a string array.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Alternatives

You can use the `ComponentNames` property of the `simscape.multibody.Multibody` object to return the names of component objects at the top level of the `Multibody` object. The returned value is a string array.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Multibody`

connect

Class: `simscape.multibody.Multibody`
Package: `simscape.multibody`

Link two connectors in `Multibody` object

Syntax

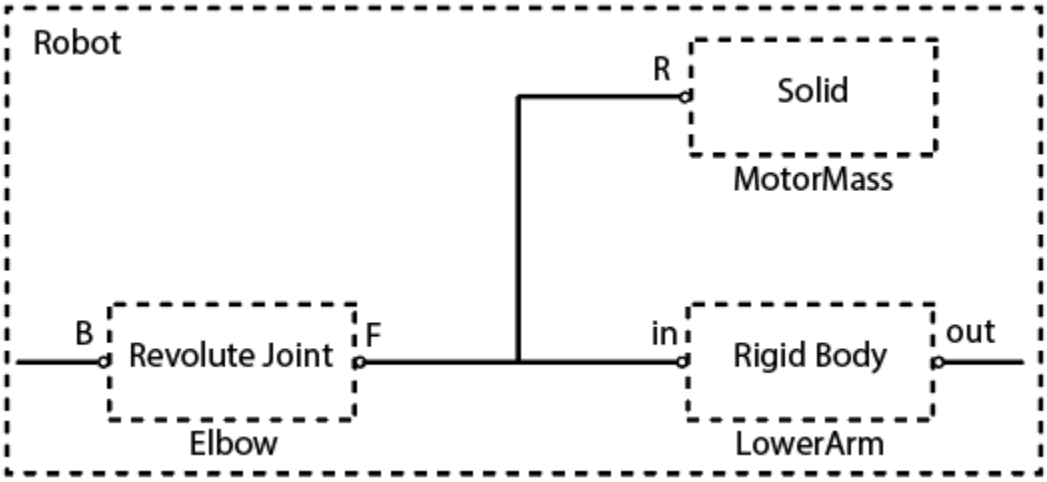
```
connect(mb,connector1,connector2)
```

Description

`connect(mb,connector1,connector2)` links two connectors, `connector1` and `connector2`, of the components at the top level of the `simscape.multibody.Multibody` object, `mb`.

The `connect` method makes connections only for objects at the top level of a `Multibody` object. You cannot link two connectors from the same object.

In general, connections are made between two connectors. However, in some cases, it is necessary to connect three or more connectors together. The figure shows an example of a three-way connection.



The `Elbow`, `LowerArm`, and `MotorMass` objects represent three parts of a `Multibody` object, `Robot`. To connect these objects together, first you connect the `Elbow` and `LowerArm` objects:

```
connect(Robot, "Elbow/F", "LowerArm/in");
```

Then, connect the `MotorMass` object to `Elbow` object:

```
connect(Robot, "MotorMass/R", "Elbow/F");
```

Alternatively, you can connect the `MotorMass` object to `LowerArm` object:

```
connect(Robot, "MotorMass/R", "LowerArm/in");
```

The two options are equivalent. If you need to add a fourth connector to the three-way connection, you can connect the fourth connector to any of the three linked connectors, `F`, `in`, or `R`.

Input Arguments

`mb` — Multibody system

`simscape.multibody.Multibody` object

Multibody system, specified as a `simscape.multibody.Multibody` object.

`connector1` — Path of connector

string scalar | character vector

Path of the first connector, specified as a string scalar or character vector. The path has two elements separated by a forward slash. The first element is the name of the object, and the second element is the connector name.

Example: "Motor_Mass/R"

`connector2` — Path of connector

string scalar | character vector

Path of the second connector, specified as a string scalar or character vector. The path has two elements separated by a forward slash. The first element is the name of the object, and the second element is the connector name.

Example: "Elblow_Joint/F"

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

Topics

`simscape.multibody.Multibody`
`connectVia`

connectVia

Class: `simscape.multibody.Multibody`
Package: `simscape.multibody`

Link two connectors in `Multibody` object via intermediate object

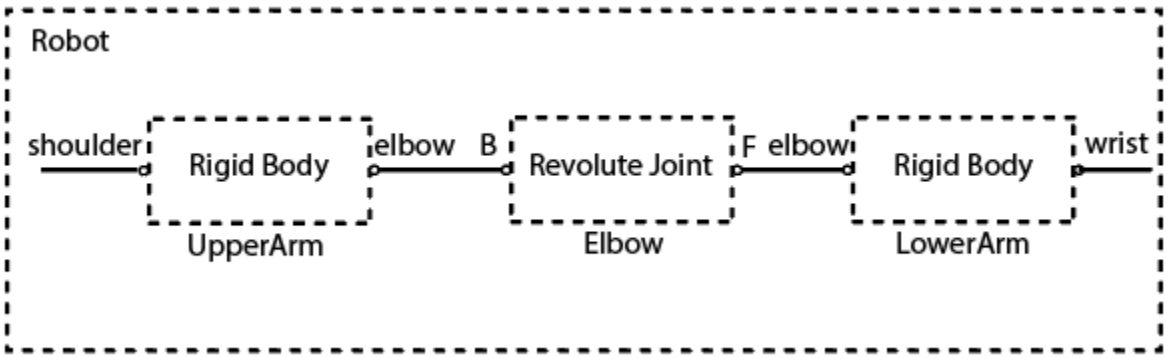
Syntax

```
connectVia(mb,viaComponent,connectorB,connectorF)
```

Description

`connectVia(mb,viaComponent,connectorB,connectorF)` links two connectors, `connectorB` and `connectorF`, of the component objects in a `simscape.multibody.Multibody` object, `mb`, via the intermediate object, `viaComponent`.

The `connectVia` method makes connections only between component objects at the top level of a `Multibody` object. The `viaComponent` object must have exactly two frame connectors, B and F, that refer to the base and follower connectors. The base connector links to the `connectorB` connector, and the follower connector links to the `connectorF` connector. The image shows an example.



Here, the `connectVia` method uses a `simscape.multibody.RevoluteJoint` object, `Elbow`, to connect the `elbow` connectors on the `UpperArm` and `LowerArm` objects.

```
connectVia(Robot, "Elbow", "UpperArm/elbow", "LowerArm/elbow");
```

Input Arguments

`mb` — Multibody system

`simscape.multibody.Multibody` object

Multibody system, specified as a `simscape.multibody.Multibody` object.

`viaComponent` — Path of intermediate object

string scalar | character vector

Path of the intermediate object, specified as a string scalar or character vector. The path has one element. The intermediate object must be a `simscape.multibody.RigidTransform` object or an object of a subclass of the `simscape.multibody.Joint` abstract class.

Example: "Elbow"

`connectorB` — Path of connector
string scalar | character vector

Path of the connector that connects to the base connector of the intermediate object, specified as a string scalar or character vector. The path must have exactly two elements. The first element is the object at the top level of the `Multibody` object, and the second element is the connector name.

Example: "UpperArm/elbow"

`connectorF` — Path of connector
string scalar | character vector

Path of the connector that connects to the follower connector of the intermediate object, specified as a string scalar or character vector. The path must have exactly two elements. The first element is the object at the top level of the `Multibody` object, and the second element is the connector name.

Example: "LowerArm/elbow"

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Multibody` | `connect`

jointPrimitivePaths

Class: `simscape.multibody.Multibody`
Package: `simscape.multibody`

Return paths of joint primitives in `Multibody` object at all hierarchical levels

Syntax

```
primPaths = jointPrimitivePaths(mb)
```

Description

`primPaths = jointPrimitivePaths(mb)` returns the paths of the joint primitives in the entire hierarchy of the `simscape.multibody.Multibody` object, `mb`.

It is useful to display the paths of joint primitives in a `Multibody` object when targeting joint primitives in the `Multibody` object by using a `simscape.op.OperatingPoint` object.

Input Arguments

`mb` — Multibody system

`simscape.multibody.Multibody` object

Multibody system, specified as a `simscape.multibody.Multibody` object.

Output Arguments

`primPaths` — Paths of joint primitives

string scalar | character vector

Paths of the joint primitives in the entire hierarchy of the `Multibody` object, returned as a string array.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Multibody`

makeBlockDiagram

Class: `simscape.multibody.Multibody`
Package: `simscape.multibody`

Create Simulink model from `Multibody` object

Syntax

```
makeBlockDiagram(mb,op,ModelName)
```

Description

`makeBlockDiagram(mb,op,ModelName)` creates a Simulink model named `ModelName` from the `simscape.multibody.Multibody` object, `mb`. The `op` argument is a `simscape.op.OperatingPoint` object that specifies the states of some joint primitives in the `mb` object. The `makeBlockDiagram` method adds a Solver Configuration block to the created Simulink model by default.

The method converts the component objects of the `Multibody` object to mechanical parts or subsystems in the created Simulink model:

- `Multibody` objects at the top level hierarchy become the subsystems.
- `RigidBody` objects at the top level hierarchy become the subsystems constructed by rigidly connected parts.
- Other component objects, such as `simscape.multibody.RigidTransform`, `simscape.multibody.WorldFrame`, `simscape.multibody.Solid`, and objects of subclasses of the `simscape.multibody.Joint` class become the corresponding Simulink blocks.
- Connectors become the ports on the subsystems, and connections between component objects become lines in the block diagram of the Simulink model.

Input Arguments

`mb` — Multibody system

`simscape.multibody.Multibody` object

Multibody system, specified as a `simscape.multibody.Multibody` object.

`op` — Operating point

`simscape.op.OperatingPoint` object

Operating point, specified as the `simscape.op.OperatingPoint` object. The object targets the positions and velocities of some joint primitives in the `simscape.multibody.Multibody` object. If you use an empty `OperatingPoint` object, the `makeBlockDiagram` method creates a Simulink model with no joint primitive targets.

`ModelName` — Name of Simulink model

string scalar | character vector

Name of the Simulink model, specified as a string scalar or character vector. If you do not specify the model name, the name of the model is `untitled` or `untitledN`, where `N` is an unique identifying number.

Example: `"FourBar"`

Data Types: `char` | `string`

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Multibody`

removeConnector

Class: `simscape.multibody.Multibody`
Package: `simscape.multibody`

Remove existing connector from `Multibody` object

Syntax

```
removeConnector(mb,connectorName)
```

Description

`removeConnector(mb,connectorName)` removes the connector, `connectorName`, from the `simscape.multibody.Multibody` object, `mb`.

Input Arguments

`mb` — Multibody system

`simscape.multibody.Multibody` object

Multibody system, specified as a `simscape.multibody.Multibody` object.

`connectorName` — Name of connector

string scalar | character vector

Name of the connector, specified as a string scalar or character vector.

Example: `"endEffectorFrame"`

Attributes

Access	public

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

Topics

`simscape.multibody.Multibody`

simscape.multibody.RigidBody class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Component`

Construct rigid body

Description

Use an object of the `simscape.multibody.RigidBody` class to construct a rigid body. A `RigidBody` object is a hierarchical container and has a tree structure composed of rigidly connected frames and component objects.

By default, a newly created `RigidBody` object contains only one frame called `reference`. The `reference` frame serves as the root of the tree structure. You must explicitly add frames or component objects to a `RigidBody` object, and the new frame or component object must be rigidly connected to the tree structure of the `RigidBody` object. Note that the name of every frame or component object must be unique among the frames and objects at the same hierarchical level of the `RigidBody` object.

In a larger system, to connect to other frames or component objects, a `RigidBody` object must have at least one connector. By default, a newly created `RigidBody` object has zero connectors, and adding frames or component objects to a `RigidBody` object does not automatically create connectors. To add a connector, use the `addConnector` method. Note that you can add connectors only to the frames at the top level of the `RigidBody` object, and each frame can only have one connector.

See “More About” on page 4-73 section for more information about the `RigidBody` class.

The `simscape.multibody.RigidBody` class is a `handle` class.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`HandleCompatible`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`rb = simscape.multibody.RigidBody` creates a `simscape.multibody.RigidBody` object.

Properties

`ComponentNames` — Names of top-level component objects in rigid body

[ ] (default) | string array

Names of the component objects at the top level of a `simscape.multibody.RigidBody` object, returned as a string array.

Example: `"sphere"`

Attributes:

`GetAccess`	`public`
`SetAccess`	`Restricts access`
`NonCopyable`	`true`
`Transient`	`true`

`FrameNames` — Names of top-level frames in rigid body

[ ] (default) | string array

Names of the frames at the top level of a `simscape.multibody.RigidBody` object, returned as a string array.

Example: `"reference"`

Attributes:

`GetAccess`	`public`
`SetAccess`	`Restricts access`
`NonCopyable`	`true`
`Transient`	`true`

Methods

Public Methods

addComponent	Add component object to rigid body
addConnector	Add connector to Rigidbody object
addFrame	Add frame to rigid body
component	Extract component object from RigidBody object
componentPaths	Return paths of component objects in RigidBody object
plotStructure	Plot structure tree of rigid body
removeConnector	Remove connector from RigidBody object

Examples

Add Frames and Component Objects to a Rigid Body

This example shows how to add a frame, solid, and sub-rigid body to a rigid body.

1 To avoid typing the package name for the classes, you can use the `import` function.

```
import simscape.multibody.*;
```

2 Create a `simscape.multibody.RigidBody` object named `rb`. The `rb` object has one frame called `reference` and no connectors.

```
rb = RigidBody
```

```
rb =

  RigidBody:

  No connectors.

  Frames:

  Frame        Parent  Source  Connector?
  ___________  ______  ______  __________

  "reference"  --      --      No

  No components.

  RigidBody with properties:

         FrameNames: "reference"
    ComponentNames: [0×1 string]
       DoVisualize: 1
    FrameConnectors: [0×1 string]
```

3 To add frames to the `reference` frame, create and add frames to a rigid transform object.

First, create a `simscape.multibody.RigidTransform` object named `rt`. The `rt` object represents a translation along the positive *y*-axis of the base frame. The distance of the translation is 50 `cm`.

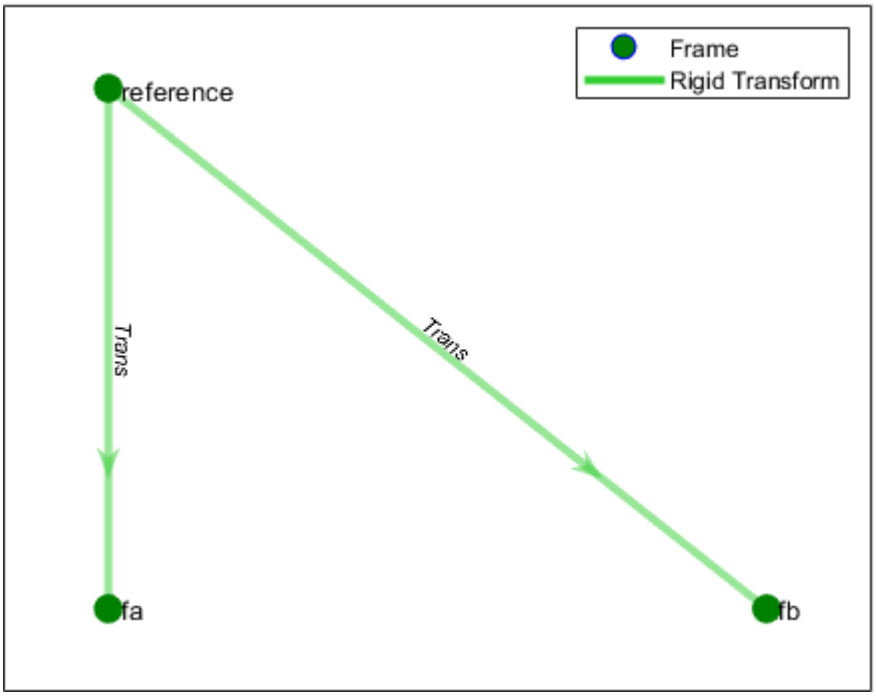
```
rt = RigidTransform(StandardAxisTranslation(simscape.Value(50,"cm"),Axis.PosY));
```

Then, use the `addFrame` method to add two frames, `fa` and `fb`, to the `reference` frame through the `rt` object.

```
addFrame(rb,"fa","reference",rt);
addFrame(rb,"fb","reference",rt);
```

You can view the tree structure of the `rb` object by using the `plotStructure` method.

```
plotStructure(rb);
```



The green vertexes represent the frames, and the green lines represent the rigid transforms between the frames. The arrows indicate the orientations of the rigid transforms. By default, a rigid transform maps vectors from the new frame to the parent frame.

4 Add a `simscape.multibody.Solid` object to the `rb` object.

First, create a `simscape.multibody.Solid` object called `sphere`. The `sphere` object represents a solid with a spherical shape. The sphere has a radius of 50 `cm`.

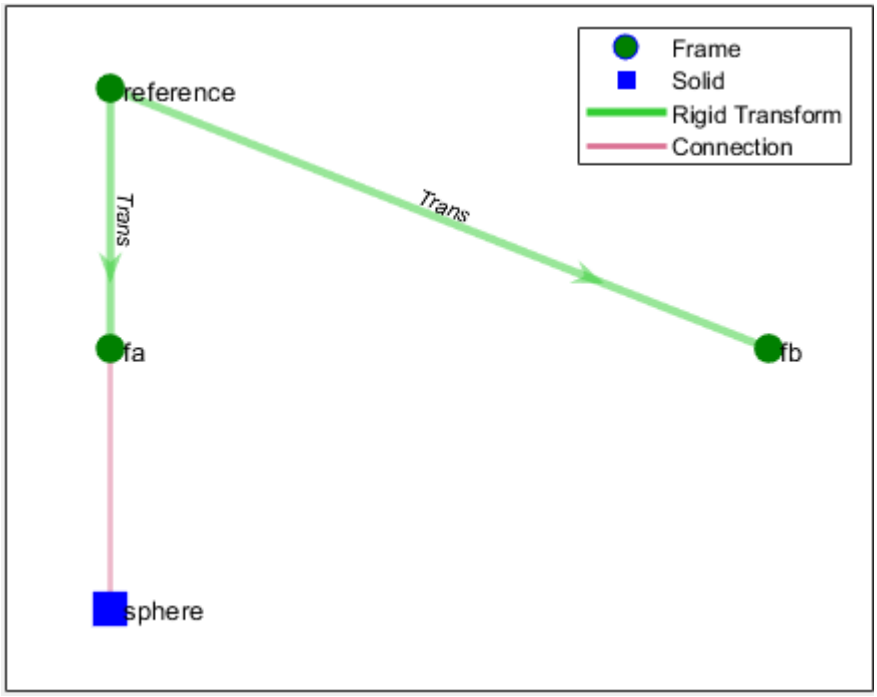
```
sphere = Solid(Sphere(simscape.Value(50,"cm")));
```

Then, use the `addComponent` method to add the `sphere` object to the `fa` frame. By default, a `Solid` object has one connector that you can connect to the `rb` object.

```
addComponent(rb,"sphere","fa",sphere);
```

Use the `plotStructure` method to show the tree structure of the `rb` object in a new figure.

```
figure
plotStructure(rb);
```



The blue vertex represents the `sphere` object, and the pink link indicates the connection between the `fa` frame and the `sphere` object.

5 Add a `RigidBody` object to the `rb` object as a subbody.

Create an empty `RigidBody` object called `subrb` and add two new frames, `f1` and `f2`, to the `subrb` object through the `rt` object.

```
subrb = RigidBody;
addFrame(subrb,"f1","reference",rt);
addFrame(subrb,"f2","reference",rt);
```

Because the `subrb` object has zero connectors, you must add at least one connector to the object. Use the `addConnector` method to add a connector to the `f1` frame of the `subrb` object. The connector and frame have the same name.

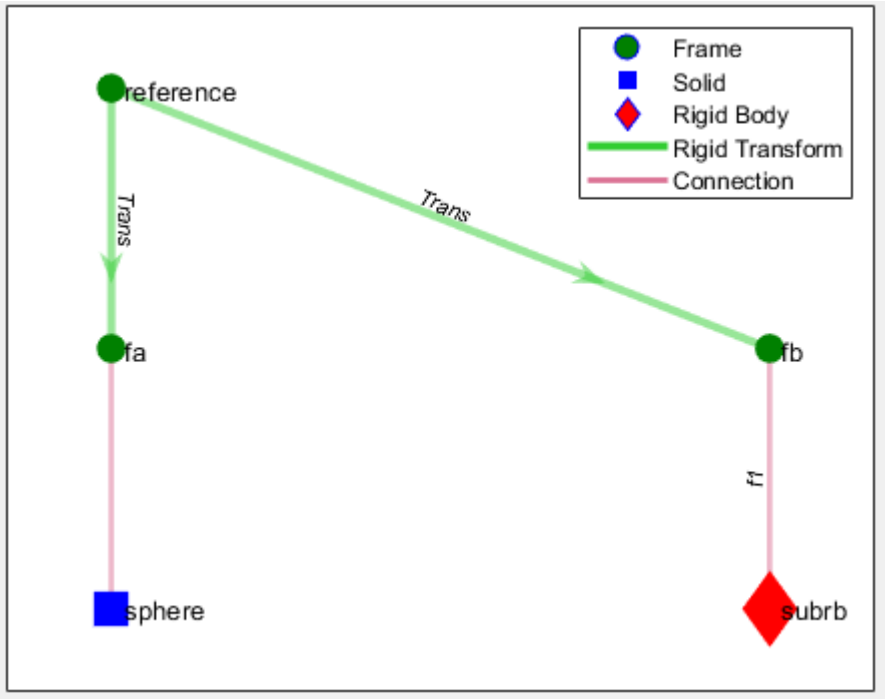
```
addConnector(subrb,"f1");
```

Use the `addComponent` method to attach the `subrb` object to the `fb` frame through the `f1` connector.

```
addComponent(rb,"subrb","fb",subrb,"f1");
```

Use the `plotStructure` method to show the tree structure of the `rb` object in a new figure.

```
figure
plotStructure(rb);
```



The red vertex represents the `subrb` object, and the pink line indicates the connection between the `fb` frame and the `f1` connector of the `subrb` object.

More About

How to Add Frames and Component Objects

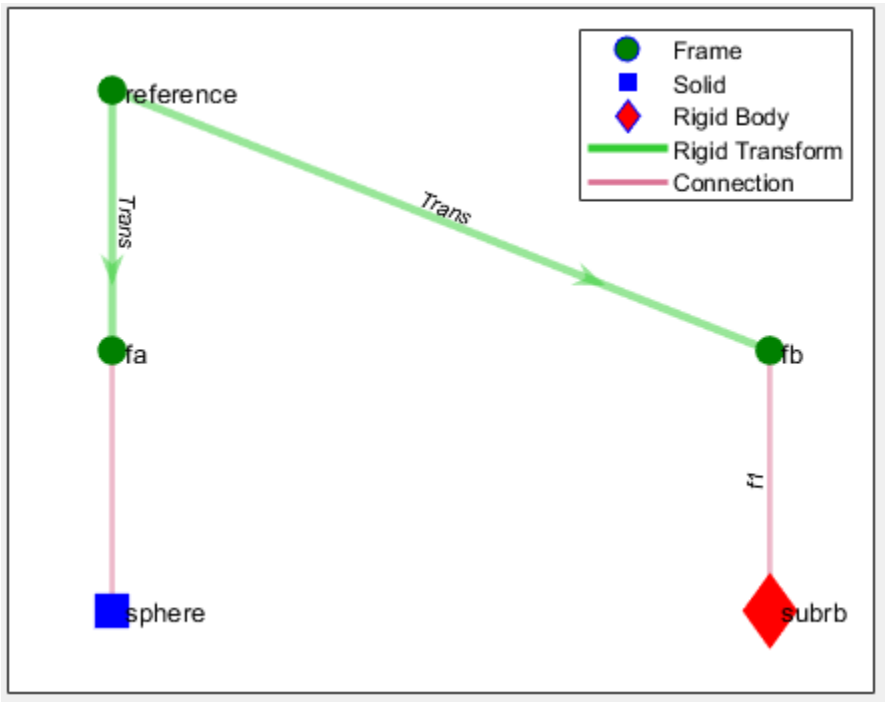
To add a frame to a `simscape.multibody.RigidBody` object, use the `addFrame` method. The frame must connect to an existing frame of the `RigidBody` through a `simscape.multibody.RigidTransform` object. There is no limit to the size of the tree structure.

Similarly, to add a component object to a `RigidBody` object, use the `addComponent` method. Unlike the `simscape.multibody.Multibody` object, a `RigidBody` object can have only two types of component objects: `simscape.multibody.Solid` and `simscape.multibody.RigidBody`. A component object must be connected to an existing frame of the `RigidBody` through a connector. The `addComponent` method adds only a copy of a component object. In other words, after being passed to the `addCompoment` method, any subsequent modifications to the original component object do not affect the added object inside the `RigidBody` object.

Note that you can only add frames or component objects to the top-level of the `RigidBody` object.

Tree Structure of `RigidBody` Object

To display the tree structure of a `RigidBody` object, use the `plotStructure` method. The figure shows the top-level structure of a `RigidBody` object.



The vertexes represent frames and component objects within the `RigidBody` object. A green vertex is the frame, a blue vertex is a `simscape.multibody.Solid` object, and a red vertex is a `simscape.multibody.RigidBody` object. The reference frame serves as the root of the tree structure. See the “Examples” on page 4-0 section to learn how to add a frame or component object to a `RigidBody` object.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component`

addComponent

Class: `simscape.multibody.RigidBody`
Package: `simscape.multibody`

Add component object to rigid body

Syntax

```
addComponent(rb,componentName,attachmentFrame,C)
addComponent( ___ ,connectionFrame)
```

Description

`addComponent(rb,componentName,attachmentFrame,C)` adds a component object to the top level of the `simscape.multibody.RigidBody` object, `rb`. The `componentName` argument specifies the name of the new component object, the `attachmentFrame` argument specifies the frame to which the new component object is attached, and the `C` argument is the object of the new component object. A `RigidBody` object can have only two types of component objects: `simscape.multibody.Solid` and `simscape.multibody.RigidBody`.

Note that the `addCompoment` method adds a copy of a component object. In other words, after being passed to the `addCompoment` method, any subsequent modifications to the original component object do not affect the added object.

`addComponent( ___ ,connectionFrame)` adds a component object to the `rb` object via the specified connector, `connectionFrame`. Use the `connectionFrame` argument when the added component object has more than one connectors.

Input Arguments

`rb` — Rigid body

`simscape.multibody.RigidBody` object

Rigid body, specified as a `simscape.multibody.RigidBody` object. The `rb` object represents the rigid body to which you add the new component object.

`componentName` — Name of new component object

string scalar | character vector

Name of the new component object, specified as a string scalar or character vector. The name must be unique among the names of frames and component objects at the top level of the `RigidBody` object to which you add the component object. In addition, the name must be a valid MATLAB identifier. You can use the `isvarname` function to check if the name is a valid identifier.

Example: "subbody"

Data Types: `char` | `string`

`attachmentFrame` — Name of frame to which new component object is attached

string scalar | character vector

Name of the frame to which the new component object is attached, specified as a string scalar or character vector. The name must correspond to an existing frame at the top level of the `RigidBody` object. In addition, the name must be a valid MATLAB identifier. You can use the `isvarname` function to check if the name is a valid identifier.

Note that the frame does not need to have a connector to connect a component object and can connect to more than one component objects.

Example: "reference"

Data Types: `char` | `string`

`C` — Component object
`simscape.multibody.Solid` object | `simscape.multibody.RigidBody` object

Component object that you add to the `simscape.multibody.RigidBody` object, specified as a `simscape.multibody.Solid` or `simscape.multibody.RigidBody` object. The new component object must have at least one frame connector.

`connectionFrame` — Name of connector on new component
string scalar | character vector

Name of the connector on the new component, specified as a string scalar or character vector. You must use this argument to specify the connector used for the connection when the new component object has multiple connectors.

The name must be a valid MATLAB identifier. You can use the `isvarname` function to check if the name is a valid identifier.

Example: "left"

Data Types: `char` | `string`

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.RigidBody`

addConnector

Class: `simscape.multibody.RigidBody`
Package: `simscape.multibody`

Add connector to `Rigidbody` object

Syntax

```
addConnector(rb,connectorName)
```

Description

`addConnector(rb,connectorName)` adds a connector named `connectorName` to the `simscape.multibody.RigidBody` object, `rb`. The connector corresponds to an existing frame at the top level of the `rb` object, and `connectorName` must have the same name as the frame.

Note that the `addConnector` method can only add a connector to a top-level frame of a `Rigid` object, and a frame can only have one connector.

Input Arguments

`rb` — Rigid body

`simscape.multibody.RigidBody` object

Rigid body, specified as a `simscape.multibody.RigidBody` object. The `rb` object is the `RigidBody` object to which you add the connector.

`connectorName` — Name of connector

string scalar | character vector

Name of the connector, specified as a string scalar or character vector. The names of the connector and the corresponding frame must be the same.

Example: "arm"

Data Types: `char` | `string`

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.RigidBody`

addFrame

Class: `simscape.multibody.RigidBody`
Package: `simscape.multibody`

Add frame to rigid body

Syntax

```
addFrame(rb,newFrame,parentFrame,T)
addFrame( ___ ,side)
addFrame(rb,newFrame,subbodyConnector)
```

Description

`addFrame(rb,newFrame,parentFrame,T)` adds a new frame to a `simscape.multibody.RigidBody` object through a `simscape.multibody.RigidTransform` object. Note that the new frame is rigidly attached to a top-level frame of the `RigidBody` object, and the newly added frame does have a connector by default. To add a connector to a frame, use `addConnector` method.

The `rb` argument is the `RigidBody` object to which you add the new frame. The `newFrame` argument specifies the name of the new frame. The `parentFrame` argument is the name of the top-level frame to which the new frame is attached. The `T` is the `RigidTransform` object used to attach the new frame.

`addFrame( ___ ,side)` adds a new frame to the `rb` object through a `RigidTransfom` object that has the specified orientation. Use a `simscape.multibody.FrameSide` object to specify the `side` argument that specifies the orientation of the `RigidTransform` object.

`addFrame(rb,newFrame,subbodyConnector)` adds a new frame to the `simscape.multibody.RigidBody` object, `rb`, through the subbody connector, `subbodyConnector`. The `subbodyConnector` argument is a two-element path.

Note that the subbody must be a `RigidBody` object that has at least two connectors. The connector used to attach the new frame must not be the connector that has already connected to the structure tree of the `rb` object.

Input Arguments

`rb` — Rigid body

`simscape.multibody.RigidBody` object

Rigid body, specified as a `simscape.multibody.RigidBody` object. The `rb` object represents the rigid body to which you add the new frame.

`newFrame` — Name of new frame

string scalar | character vector

Name of the new frame, specified as a string scalar or character vector. The name must be unique among the names of frames and component objects at the top level of the `RigidBody` object to which you add the frame object. In addition, the name must be valid a MATLAB identifier.

Example: "fnew"

Data Types: `char` | `string`

`parentFrame` — Name of frame to which new frame is connected
string scalar | character vector

Name of the frame to which the new frame is connected, specified as a string scalar or character vector. The name must correspond to an existing frame at the top level of the `RigidBody` object.

Example: "reference"

Data Types: `char` | `string`

`T` — Rigid transform
`simscape.multibody.RigidTransfrom` object

Rigid transform, specified as a `simscape.multibody.RigidTransform` object. The `RigidTransform` object represents the rigid transform between the new frame and the parent frame.

`side` — Which side of `T` is connected to parent frame
`simscape.multibody.FrameSide.Base` object (default) |
`simscape.multibody.FrameSide.Follower` object

Which side of the `RigidTransform` object is connected to the parent frame, specified as a `simscape.multibody.FrameSide.Base` or `simscape.multibody.FrameSide.Follower` object. Omitting the `side` argument connects the base frame of the`RigidTransfom` object to the parent frame.

This argument specifies the orientation of the `RigidTransfom` object used for the `T` argument. Use the `simscape.multibody.FrameSide.Base` object to connect the base frame of the `RigidTransfom` object to the parent frame, or use the `simscape.multibody.FrameSide.Follower` object to connect the follower frame of the `RigidTransform` object to the parent frame.

`subbodyConnector` — Name of connector of subbody
string scalar | character vector

Name of a connector of a subbody of the `RigidBody` object, specified as a string scalar or character vector. The name is a two-element path, and the path elements are separated by a forward slash. The first element is the name of the subbody, and the second element is the name of the connector.

Note that the subbody object must have at least two connectors. The specified connector must not be a connector that is already connected to the tree structure of the parent `RigidBody` object.

Example: "subrb/reference"

Data Types: `char` | `string`

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.RigidBody` | `simscape.multibody.RigidTransform` | `addConnector`

component

Class: `simscape.multibody.RigidBody`
Package: `simscape.multibody`

Extract component object from `RigidBody` object

Syntax

```
C = component(rb,path)
```

Description

`C = component(rb,path)` extracts a component object, `C`, from the `simscape.multibody.RigidBody` object, `rb`. The `path` argument is the path of the component object in the `rb`object and can have more than one elements. In other words, the `component` method can extract an object from any hierarchical level of the `rb` object.

The extracted object is a copy of the original component object. Therefore, any subsequent changes for the extracted copy do not affect the original object.

Input Arguments

`rb` — Rigid body

`simscape.multibody.RigidBody` object

Rigid body, specified as a `simscape.multibody.RigidBody` object. The `rb` object represents the rigid body from which you add the component object.

`path` — Path of component object

string scalar | character vector

Path of the component object, specified as a string scalar or character vector. The length of the path depends on the hierarchical level of the component object. If the object is at the top level, the path has one element. If the component object is in a subbody, the path has two elements, and the path elements are separated by forward slashes. The first element is the name of the subbody, and the last element is the name of desired component object.

Example: "subrb/sphere"

Output Arguments

`C` — Copy of component object

`simscape.multibody.Solid` | `simscape.multibody.RigidBody`

Copy of the component object, returned as a `simscape.multibody.Solid` or `simscape.multibody.RigidBody` object. Modifying the extracted component object does not affect the original component object inside the `RigidBody` object.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.RigidBody` | `simscape.multibody.Solid`

componentPaths

Class: `simscape.multibody.RigidBody`
Package: `simscape.multibody`

Return paths of component objects in `RigidBody` object

Syntax

```
paths = componentPaths(rb)
```

Description

`paths = componentPaths(rb)` returns the paths of the component objects in the entire hierarchy of the `simscape.multibody.RigidBody` object, `rb`.

Input Arguments

`rb` — Rigid body

`simscape.multibody.RigidBody` object

Rigid body, specified as a `simscape.multibody.RigidBody` object.

Output Arguments

`paths` — Paths of component objects

string scalar | character vector

Paths of the component objects in the entire hierarchy of the `RigidBody` object, returned as a string array.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.RigidBody`

plotStructure

Class: `simscape.multibody.RigidBody`
Package: `simscape.multibody`

Plot structure tree of rigid body

Syntax

```
h = plotStructure(rb)
```

Description

`h = plotStructure(rb)` plots the structure tree of the `simscape.multibody.RigidBody` object, `rb`. The vertices on the structure tree indicate the frames and component objects of the `rb` object, and the lines between the vertices show relationships between the frames and component objects. See “Tree Structure of RigidBody Object” on page 4-73 section for more information.

Input Arguments

`rb` — Rigid body

`simscape.multibody.RigidBody` object

Rigid body, specified as a `simscape.multibody.RigidBody` object.

Output Arguments

`h` — Handle of plot

`matlab.graphics.chart.primitive.GraphPlot` object

Handle of the plot, returned as a `matlab.graphics.chart.primitive.GraphPlot` object. See `GraphPlot` class for more information.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.RigidBody`

removeConnector

Class: `simscape.multibody.RigidBody`
Package: `simscape.multibody`

Remove connector from `RigidBody` object

Syntax

```
removeConnector(rb,connectorName)
```

Description

`removeConnector(rb,connectorName)` removes a connector, `connectorName`, from the `simscape.multibody.RigidBody` object, `rb`.

Input Arguments

`rb` — Rigid body

`simscape.multibody.RigidBody` object

Rigid body, specified as a `simscape.multibody.RigidBody` object. The `rb` object represents the rigid body from which you remove the connector.

`connectorName` — Name of connector

string scalar | character vector

Name of the connector, specified as a string scalar or character vector.

Example: "arm"

Attributes

Access	public

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.RigidBody`

simscape.multibody.RigidTransform class

Package: simscape.multibody
Superclasses: simscape.multibody.Component

Construct rigid transform

Description

Use an object of the simscape.multibody.RigidTransform class to construct a rigid transform. A RigidTransform object specifies and maintains a fixed spatial relationship between two arbitrary frames. The spatial relationship includes a translation and a rotation.

To specify the Rotation property, use an object of a subclass of the simscape.multibody.Rotation class. To specify the Translation property, use an object of a subclass of the simscape.multibody.Translation class.

A RigidTransform object has two frame connectors called B and F that refer to the base and follower frames of the object.

Class Attributes

Sealed	true
ConstructOnLoad	true
RestrictsSubclassing	true

For information on class attributes, see “Class Attributes”.

Creation

Description

RT = simscape.multibody.RigidTransform creates a rigid transform with default values.

RT = simscape.multibody.RigidTransform(rot) creates a rigid transform with the specified rotation and default translation.

RT = simscape.multibody.RigidTransform(trans) creates a rigid transform with the specified translation and default rotation.

RT = simscape.multibody.RigidTransform(rot,trans) creates a rigid transform with the specified rotation and translation.

Properties

Rotation — Rotation of rigid transform

simscape.multibody.ZeroRotation object (default) | object of subclass of simscape.multibody.Rotation class

Rotation of the rigid transform, specified as an object of a subclass of the `simscape.multibody.Rotation` class.

Example: `simscape.multibody.StandardAxisRotation`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`Translation` — Translation of rigid transform
`simscape.multibody.ZeroTranslation` object (default) | object of subclass of `simscape.multibody.Translation` class

Translation of the rigid transform, specified as an object of a subclass of the `simscape.multibody.Translation` class.

Example: `simscape.multibody.StandardAxisTranslation`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Methods

Public Methods

`transformation`

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.Rotation` | `simscape.multibody.Transformation` | `simscape.multibody.Translation`

transformation

Class: `simscape.multibody.RigidTransform`
Package: `simscape.multibody`

Compute transformation of rigid transform

Syntax

```
X = transformation(T)
```

Description

`X = transformation(T)` computes the transformation of the rigid transform, `T`.

Input Arguments

T — Rigid transform

`simscape.multibody.RigidTransform` object

Rigid transform, specified as a `simscape.multibody.RigidTransform` object.

Output Arguments

X — Transformation of rigid transform

`simscape.multibody.Transformation` object

Transformation of the rigid transform, returned as a `simscape.multibody.Transformation` object.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.RigidTransform` | `simscape.multibody.Transformation`

simscape.multibody.Solid class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Component`

Construct rigid solid

Description

Use an object of the `simscape.multibody.Solid` class to construct a rigid solid. `Solid` objects model the physical parts of a multibody system, such as hinges or shafts. By default, each `Solid` object has a reference frame that has a corresponding connector called `R`.

To specify a `Solid` object, use the `Geometry`, `Inertia`, and `VisualProperties` properties. The geometry and inertia of a solid are relative to the reference frame of the `Solid` object.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`solid = simscape.multibody.Solid` constructs a solid with default values.

`solid = simscape.multibody.Solid(geometry)` constructs a solid with the specified geometry.

`solid = simscape.multibody.Solid(geometry,inertia)` constructs a solid with the specified geometry and inertia.

`solid = simscape.multibody.Solid(geometry,visualProperties)` constructs a solid with the specified geometry and visual properties.

`solid = simscape.multibody.Solid(geometry,inertia,visualProperties)` constructs a solid with the specified geometry, inertia, and visual properties.

Properties

`Geometry` — Geometry of solid

`simscape.multibody.Brick` object (default) | object of subclass of `simscape.multibody.Geometry` class

Geometry of the solid, specified as an object of a subclass of the `simscape.multibody.Geometry` class.

Example: `simscape.multibody.Sphere`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`Inertia` — Mass distribution of solid
`simscape.multibody.UniformDensity` object (default) | object of subclass of `simscape.multibody.Inertia` class

Mass distribution of the solid, specified as an object of a subclass of the `simscape.multibody.Inertia` class.

Example: `simscape.multibody.MassDistribution`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`VisualProperties` — Visual properties of solid
`simscape.multibody.SimpleVisualProperties` object (default) | `simscape.multibody.AdvancedVisualProperties` object

Visual properties of the solid, specified as an object of a subclass of the `simscape.multibody.VisualProperties` class.

Example: `simscape.multibody.AdvancedVisualProperties`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Methods

Public Methods

massDistribution	Extract mass distribution of solid

Examples

Create a Spherical Solid

The example shows how to create a solid that has a radius of 10 `cm`, a mass of 1 `kg`, and red color.

1. Use a `simscape.multibody.Sphere` object to create a spherical geometry with radius of 10 `cm`.

```
geom = simscape.multibody.Sphere(simscape.Value(10,"cm"));
```

2. Use a `simscape.multibody.UniformMass` object to create a uniform mass distribution with a total mass of 1 `kg`.

```
inert = simscape.multibody.UniformMass(simscape.Value(1,'kg'));
```

3 Use a `simscape.multibody.SimpleVisualProperties` object to create a visual property with red color.

```
visProps = simscape.multibody.SimpleVisualProperties([1 0 0]);
```

4 Use a `simscape.multibody.Solid` object to create a spherical solid that has a radius of 10 `cm`, a mass of 1 `kg`, and red color.

```
solid = simscape.multibody.Solid(geom,inert,visProps);
```

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component`

massDistribution

Class: `simscape.multibody.Solid`
Package: `simscape.multibody`

Extract mass distribution of solid

Syntax

```
MD = massDistribution(solid)
```

Description

`MD = massDistribution(solid)` extracts the mass distribution of the specified solid and returns the result as a `simscape.multibody.MassDistribution` object.

Input Arguments

`solid` — Solid

`simscape.multibody.Solid` object

Solid from which to extract the mass distribution, specified as a `simscape.multibody.Solid` object.

Output Arguments

`MD` — Mass distribution

`simscape.multibody.MassDistribution` object

Mass distribution of the solid, returned as a `simscape.multibody.MassDistribution` object.

Attributes

Access	public

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.MassDistribution` | `simscape.multibody.Solid`

simscape.multibody.WorldFrame class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Component`

Construct world frame

Description

Use an object of the `simscape.multibody.WorldFrame` class to construct a world frame. In a `simscape.multibody.Multibody` object, the world frame is the global reference frame that is fixed and impervious to any applied forces and moments.

The `WorldFrame` object has one frame whose axes are orthogonal and arranged based on the right-hand rule. The frame has a corresponding connector called `W`. Directly connecting another object to the `W` connector makes that object motionless. A `Multibody` object may have multiple `WorldFrame` objects, but all the `WorldFrame` objects refer to the same world frame.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`wf = simscape.multibody.WorldFrame` constructs a world frame.

Examples

Connect a Solid to the World Frame

1. Use a `simscape.multibody.WorldFrame` object to create a world frame.

   ```
   world = simscape.multibody.WorldFrame;
   ```
2. Add the `WorldFrame` object to a `simscape.multibody.Multibody` object.

   First, use a `Multibody` object to construct a multibody system. Then, add the `WorldFrame` object to the multibody system by using the `addComponent` method.

   ```
   mb = simscape.multibody.Multibody;
   addComponent(mb,"World",world);
   ```
3. Use a `simscape.multibody.Solid` object to construct a solid with default values. Then, add the `Solid` object to the multibody system.

```
solid = simscape.multibody.Solid;
addComponent(mb,"Solid",solid);
```

4 Connect the `Solid` object to the world frame by using the `connect` method.

```
connect(mb,"World/W","Solid/R")
```

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component`

simscape.multibody.CompiledMultibody class

Package: `simscape.multibody`

Compiled multibody system

Description

Use an object of the `simscape.multibody.CompiledMultibody` class to represent a compiled multibody system. To perform analyses on a `simscape.multibody.Multibody` object, you need a corresponding `CompiledMultibody` object. To create a `CompiledMultibody` object, use the `compile` method.

The compilation process checks the `Multibody` object for errors. If an error occurs, the `compile` method shows the problem and does not return a `CompiledMultibody` object.

After you create a `CompiledMultibody` object, subsequent changes to the corresponding `Multibody` object do not affect the `CompiledMultibody` object. To capture those changes, create a new `CompiledMultibody` object.

Class Attributes

```
Sealed                                          true

ConstructOnLoad                                      true

RestrictsSubclassing                                    true
```

For information on class attributes, see “Class Attributes”.

Creation

To create a `simscape.multibody.CompiledMultibody` object for a `simscape.multibody.Multibody` object, use the `compile` method.

Methods

Public Methods

computeState	Compute state of compiled multibody system
jointPrimitivePosition	Return position of specified joint primitive
jointPrimitiveVelocity	Return velocity of provided joint primitive
plotKinematicGraph	Plot kinematic graph of compiled multibody system
transformation	Compute 3-D transformation between two multibody frames in provided state
visualize	Visualize compiled multibody system in given state

Version History

Introduced in R2022a

See Also

`simscape.multibody.Multibody` | `compile`

Topics

“Creating a Mobile Robot using a MATLAB App”
“Creating a Robotic Gripper Multibody in MATLAB”
“Creating a Four Bar Multibody Mechanism in MATLAB”
“Creating a Simple Pendulum in MATLAB”
“Create a Mechanism with Different Joints in MATLAB”
“How to Build a Multibody System in MATLAB”

computeState

Class: `simscape.multibody.CompiledMultibody`
Package: `simscape.multibody`

Compute state of compiled multibody system

Syntax

```
state = computeState(cmb,op)
```

Description

`state = computeState(cmb,op)` computes the state of the `simscape.multibody.CompiledMultibody` object, `cmb`, based on the joint targets specified by the `simscape.op.OperatingPoint` object, `op`.

The `computeState` method attempts to satisfy the targeted joint primitives specified by the `OperatingPoint` object as much as possible. However, in some cases, to find a complete state that satisfies all of the kinematic constraints, some targets may not be met.

Input Arguments

`cmb` — Compiled multibody system

`simscape.multibody.CompiledMultibody` object

Compiled multibody system, specified as a `simscape.multibody.CompiledMultibody` object.

`op` — Targeted joint primitives

`simscape.op.OperatingPoint` object

Targeted joint primitives, specified as a `simscape.op.OperatingPoint` object. Use an `OperatingPoint` object to target the positions and velocities of desired joint primitives in the compiled multibody system.

You can specify the priority for each individual target. The `computeState` method searches for a solution that is compatible with all the high-priority targets. However, in some cases, some targets may not be met. When multiple solutions exist, low-priority targets can help bias the method to converge on the desired one.

Output Arguments

`state` — State of compiled multibody system

`simscape.multibody.State` object

State of the compiled multibody system with specified targets, returned as a `simscape.multibody.State` object. The `State` object includes the position and velocity of each joint primitive in the compiled multibody system.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.CompiledMultibody` | `simscape.multibody.State`

jointPrimitivePosition

Class: `simscape.multibody.CompiledMultibody`
Package: `simscape.multibody`

Return position of specified joint primitive

Syntax

```
[pos,status] = jointPrimitivePosition(cmb,jointPrimitive,state)
```

Description

`[pos,status] = jointPrimitivePosition(cmb,jointPrimitive,state)` returns the position of the specified joint primitive in a `simscape.multibody.CompiledMultibody` object, `cmb`, in the state, `state`.

Input Arguments

`cmb` — Compiled multibody system

`simscape.multibody.CompiledMultibody` object

Compiled multibody system, specified as a `simscape.multibody.CompiledMultibody` object.

`jointPrimitive` — Path of joint primitive

string scalar | character vector

Path of the joint primitive, specified as a string scalar or character vector. Separate the path elements by using forward slashes.

Example: 'Bottom_Right_Joint/Rz'

`state` — State of compiled multibody system

`simscape.multibody.State` object

State of the compiled multibody system, specified as a `simscape.multibody.State` object. The `state` object includes the position and velocity of every joint primitive in the compiled multibody system.

Output Arguments

`pos` — Position of joint primitive

`simscape.Value` object | `simscape.multibody.QuaternionRotation` object

Position of the joint primitive, returned as a `simscape.Value` or `simscape.multibody.QuaternionRotation` object. The form of the value depends on the type of the joint primitive:

Primitive Type	Type of Returned Position
Revolute	`simscape.Value` object that represents a scalar with a unit of angle
Prismatic	`simscape.Value` object that represents a scalar with a unit of length
Spherical	`simscape.multibody.QuaternionRotation` object
Lead Screw	`simscape.Value` object that represents a scalar with a unit of angle
Constant Velocity	`simscape.Value` object that represents a 2-by-1 vector with a unit of angle, where the first element is the bend angle and second element is the azimuth

`status` — Status of joint primitive in provided state
member of `simscape.multibody.StateStatus` class

Status of the joint primitive in the given state, returned as a member of the `simscape.multibody.StateStatus` class that indicates whether the returned value is valid:

- `simscape.multibody.StateStatus.Valid`: All kinematic constraints of the joint primitive are satisfied. Note that a valid status does not indicate that all the specified targets of the joint primitive have been met.
- `simscape.multibody.StateStatus.PositionViolation`: Not all position constraints of the joint primitive are satisfied.
- `simscape.multibody.StateStatus.VelocityViolation`: All position constraints of the joint primitive are satisfied, but not all velocity constraints of the joint primitive are satisfied.
- `simscape.multibody.StateStatus.SingularityViolation`: All kinematic constraints of the joint primitive are satisfied, but the joint primitive is in a kinematic singularity.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

jointPrimitiveVelocity

Class: `simscape.multibody.CompiledMultibody`
Package: `simscape.multibody`

Return velocity of provided joint primitive

Syntax

```
[vel,status] = jointPrimitiveVelocity(cmb,jointPrimitive,state)
```

Description

`[vel,status] = jointPrimitiveVelocity(cmb,jointPrimitive,state)` returns the velocity of the specified joint primitive in a `simscape.multibody.CompiledMultibody` object, `cmb`, in the state, `state`.

Input Arguments

`cmb` — Compiled multibody system

`simscape.multibody.CompiledMultibody` object

Compiled multibody system, specified as a `simscape.multibody.CompiledMultibody` object.

`jointPrimitive` — Path of joint primitive

string scalar | character vector

Path of the joint primitive, specified as a string scalar or character vector. Separate the path elements by using forward slashes.

Example: 'Bottom_Right_Joint/Rz'

`state` — State of compiled multibody system

`simscape.multibody.State` object

State of the compiled multibody system, specified as a `simscape.multibody.State` object. The `state` object includes the position and velocity of every joint primitive in the compiled multibody system.

Output Arguments

`vel` — Velocity of joint primitive

`simscape.Value` object

Velocity of the joint primitive, returned as a `simscape.Value` object. The form of the value depends on the type of the joint primitive:

Primitive Type	Type of Returned Position
Revolute	`simscape.Value` object that represents a scalar with a unit of angular velocity
Prismatic	`simscape.Value` object that represents a scalar with a unit of linear velocity
Spherical	`simscape.Value` object that represents a 3-by-1 vector with a unit of angular velocity. The vector is resolved in the base frame.
Lead Screw	`simscape.Value` object that represents a scalar with a unit of angular velocity
Constant Velocity	`simscape.Value` object that represents a 2-by-1 vector with a unit of angular velocity, where the first element is for the bend angle and the second element is for the azimuth

`status` — Status of joint primitive in provided state
member of `simscape.multibody.StateStatus` class

Status of the joint primitive in the given state, returned as a member of the `simscape.multibody.StateStatus` class that indicates whether the returned value is valid:

- `simscape.multibody.StateStatus.Valid`: All kinematic constraints of the joint primitive are satisfied. Note that a valid status does not indicate that all the specified targets of the joint primitive have been met.
- `simscape.multibody.StateStatus.PositionViolation`: Not all position constraints of the joint primitive are satisfied.
- `simscape.multibody.StateStatus.VelocityViolation`: All position constraints of the joint primitive are satisfied, but not all velocity constraints of the joint primitive are satisfied.
- `simscape.multibody.StateStatus.SingularityViolation`: All kinematic constraints of the joint primitive are satisfied, but the joint primitive is in a kinematic singularity.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.CompiledMultibody` | `computeState`

plotKinematicGraph

Class: `simscape.multibody.CompiledMultibody`
Package: `simscape.multibody`

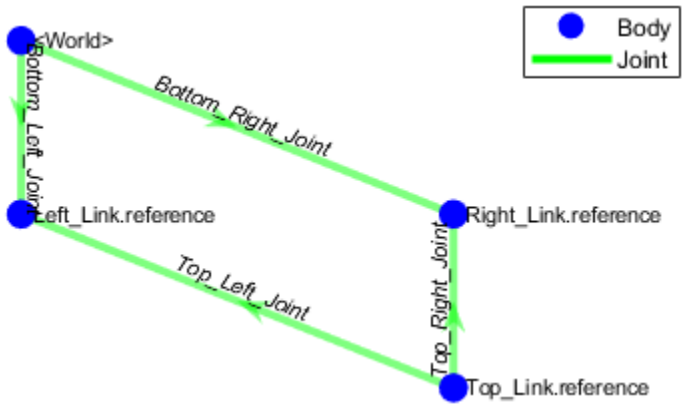
Plot kinematic graph of compiled multibody system

Syntax

```
h = plotKinematicGraph(cmb)
h = plotKinematicGraph(cmb,Name=Value)
```

Description

`h = plotKinematicGraph(cmb)` plots the kinematic graph of a `simscape.multibody.CompiledMultibody` object, `cmb`, in a tree layout graph. The dots in the plot indicate the component objects, such as `simscape.multibody.Multibody`, `simscape.multibody.RigidBody`, `simscape.multibody.Solid`, and `simscape.multibody.WorldFrame`, in the `cmb` object. The lines between the dots indicate joint objects between component objects, and the arrow on each line points toward the follower frame of each joint object. For example, the image shows the kinematic graph of a four bar system.



The blue dots indicate the `Solid` objects that represent the four bars, the green lines indicate the `simscape.multibody.RevoluteJoint` objects that represent the joints between each pair of bars, and the arrow on each line points toward the follower frame of each `RevoluteJoint` object. See the “Creating a Four Bar Multibody Mechanism in MATLAB” example for more information about the four bar system.

`h = plotKinematicGraph(cmb,Name=Value)` plots the kinematic graph using `Name,Value` arguments to specify the layout of the graph.

Input Arguments

`cmb` — Compiled multibody system

`simscape.multibody.CompiledMultibody` object

Compiled multibody system, specified as a `simscape.multibody.CompiledMultibody` object.

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1,...,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Before R2021a, use commas to separate each name and value, and enclose `Name` *in quotes.*

Example: `layout="tree"`

`layout` — Layout of kinematic graph

`"tree"` (default) | `"circle"` | `"snake"` | `"layered"`

Layout of the kinematic graph, specified as `"tree"`, `"circle"`, `"snake"`, or `"layered"`.

- `tree` — A layout that emphasizes the spanning tree of the graph. The distinguished inertial body attached to the world frame is the root of the tree, and other bodies are in rows below the world frame:
- `circle` — A layout that arranges the distinguished inertial body at the center and other bodies in concentric circles. This is a good option for mechanisms with radial symmetry, such as a Stewart platform.
- `snake` — A layout that arranges the bodies in a snake-like pattern that tends to use all of the available 2-D area of the plot. This layout is a good choice for kinematic trees that are long and chain-like, such as a hanging chain with many links.
- `layered` — A layout that arranges the bodies into a set of layers, which emphasizes hierarchical structures. By default, the layers progress downwards.

The best layout option depends on the details of a `CompiledMultibody` object. Different layout options may preferable for different `CompiledMultibody` objects.

Example: `layout="circle"`

Output Arguments

`h` — Handle of plot

`matlab.graphics.chart.primitive.GraphPlot` object

Handle of the plot, returned as a `matlab.graphics.chart.primitive.GraphPlot` object. For more information about the `GraphPlot` object, see `GraphPlot`.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.CompiledMultibody` | `plotStructure`

transformation

Class: `simscape.multibody.CompiledMultibody`
Package: `simscape.multibody`

Compute 3-D transformation between two multibody frames in provided state

Syntax

```
T = transformation(cmb,baseFrame,followerFrame,state)
```

Description

`T = transformation(cmb,baseFrame,followerFrame,state)` computes the 3-D transformation between the specified base and follower frames of the multibody system, `cmb`, in the given state, `state`.

Input Arguments

`cmb` — Compiled multibody system

`simscape.multibody.CompiledMultibody` object

Compiled multibody system, specified as a `simscape.multibody.CompiledMultibody` object.

`baseFrame` — Path of base frame

string scalar | character vector

Path of the base frame, specified as a string scalar or character vector. Separate the path elements by using forward slashes.

Example: "engine/CrankShaft/reference"

`followerFrame` — Path of follower frame

string scalar | character vector

Path of the follower frame, specified as a string scalar or character vector. Separate the path elements by using forward slashes.

Example: "engine/Rod_0/Bar/R"

`state` — State of compiled multibody system

`simscape.multibody.State` object

State of the compiled multibody system, specified as a `simscape.multibody.State` object. The `state` object includes the position and velocity of every joint primitive in the compiled multibody system.

Output Arguments

`T` — 3-D transformation between base and follower frames

`simscape.multibody.Transformation` object

3-D transformation between the base and follower frames, returned as a `simscape.multibody.Transformation` object that maps follower-frame coordinates to base-frame coordinates.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.CompiledMultibody` | `simscape.multibody.Transformation`

visualize

Class: `simscape.multibody.CompiledMultibody`
Package: `simscape.multibody`

Visualize compiled multibody system in given state

Syntax

```
visualize(cmb,state,modelName)
```

Description

`visualize(cmb,state,modelName)` opens the Mechanics Explorer to display the 3-D graphical representation of the compiled multibody system, `cmb`, in the given state, `state`. If the Mechanics Explorer is already open, calling the `visualize` method adds a new tab to the visualization pane to display the compiled multibody system.

The Mechanics Explorer comprises a visualization pane, which displays the system, and a tree view pane that you can use to explore the system hierarchy. In the visualization pane, you can rotate, roll, pan, and zoom the model to more clearly view its components.

If the tab specified by `modelName` is already in use, the Mechanics Explorer replace the existing visualization with the new visualization.

If the given state has a status other than `simscape.multibody.StateStatus.Valid`, you can still use the `visualize` method to view the compiled multibody system, although some of the kinematic loops in the system may not be closed.

Input Arguments

`cmb` — Compiled multibody system

`simscape.multibody.CompiledMultibody` object

Compiled multibody system, specified as a `simscape.multibody.CompiledMultibody` object.

`state` — State of compiled multibody system

`simscape.multibody.State` object

State of the compiled multibody system, specified as a `simscape.multibody.State` object. The `state` object includes the position and velocity of every joint primitive in the compiled multibody system.

`modelName` — Name of tabbed pane

string scalar | character vector

Name of the tabbed pane in which to display the compiled multibody system, specified as a string scalar or character vector. The name must be a valid MATLAB identifier. You can use the `isvarname` function to check if the name is a valid identifier.

Example: 'Rocker_Down'

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.CompiledMultibody`

simscape.multibody.Geometry class

Package: `simscape.multibody`

Abstract base class for geometries

Description

`simscape.multibody.Geometry` class is the abstract base class that represents 3-D geometries, such as bricks, spheres, and ellipsoids. To create geometries, use the subclasses of the `simscape.multibody.Geometry` class, such as `simscape.multibody.Brick` and `simscape.multibody.Cylinder`.

Class Attributes

`Abstract`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Brick` | `simscape.multibody.Cylinder` | `simscape.multibody.Ellipsoid` | `simscape.multibody.GeneralExtrusion` | `simscape.multibody.RegularExtrusion` | `simscape.multibody.Revolution` | `simscape.multibody.Sphere`

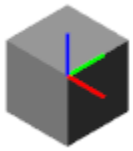
simscape.multibody.Brick class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Geometry`

Create brick geometry

Description

Use an object of the `simscape.multibody.Brick` class to represent a brick geometry. The centroid of the brick geometry is coincident with the origin of the reference frame whose axes are normal to the surfaces of the brick.



To specify the dimensions of the brick, use the `Dimensions` property.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`brick = simscape.multibody.Brick` creates a brick geometry with default values.

`brick = simscape.multibody.Brick(dimensions)` creates a brick geometry with the specified dimensions along the *x*-, *y*-, and *z*-axes of the reference frame.

Properties

`Dimensions` — Dimensions of brick

`simscape.Value([1 1 1], "m")` (default) | `simscape.Value([x y z], "Length unit")` | `simscape.Value([x y z]', "Length unit")`

Dimensions of the brick geometry, specified as a `simscape.Value` object that represents a 3-by-1 or 1-by-3 array with a unit of length.

Example: `simscape.Value([240 112 70], "mm")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

Brick Solid | `simscape.multibody.Geometry`

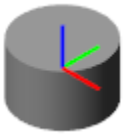
simscape.multibody.Cylinder class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Geometry`

Create cylindrical geometry

Description

Use an object of the `simscape.multibody.Cylinder` class to represent cylindrical geometry. To specify the dimensions of the geometry, use the `Radius` and `Length` properties. The reference frame of the cylinder is located at the centroid of the cylinder, and the axis of the cylinder is aligned with the *z*-axis of the reference frame.



Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`cylinder = simscape.multibody.Cylinder` creates a cylindrical geometry with default values.

`cylinder = simscape.multibody.Cylinder(radius,length)` creates a cylindrical geometry with the specified radius and length.

Properties

`Radius` — Radius of cylinder

`simscape.Value(1,"m")` (default) | `simscape.Value(R,"Length unit")`

Radius of the cylindrical geometry, specified as a `simscape.Value` object that represents a scalar with a unit of length.

Example: `simscape.Value(10, "mm")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Length — Length of cylinder

`simscape.Value(1,"m")` (default) | `simscape.Value(L,"Length unit")`

Length of the cylindrical geometry, specified as a `simscape.Value` object that represents a scalar with a unit of length. The length of the cylinder is aligned with the *z*-axis of the reference frame.

Example: `simscape.Value(30,"mm")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

Cylindrical Solid | `simscape.multibody.Geometry`

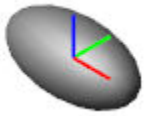
simscape.multibody.Ellipsoid class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Geometry`

Create ellipsoidal geometry

Description

Use an object of the `simscape.multibody.Ellipsoid` class to represent ellipsoidal geometry. The reference frame of the geometry is located at the centroid of the ellipsoid and the semi-axes are aligned with the axes of the reference frame. To specify the dimensions of the geometry, use the `Radii` property.



Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`ellipsoid = simscape.multibody.ellipsoid` creates an ellipsoidal geometry with default values.

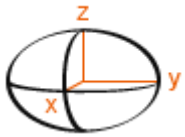
`ellipsoid = simscape.multibody.ellipsoid(radii)` creates an ellipsoidal geometry with the specified radii.

Properties

Radii — Lengths of three semi-axes of ellipsoid

`simscape.Value([1 1 2],"m")` (default) | `simscape.Value([x y z],"Length unit")` | `simscape.Value([x y z]',"Length unit")`

Lengths of the three semi-axes of the ellipsoidal geometry, specified as a `simscape.Value` object that represents a 3-by-1 or 1-by-3 array with a unit of length. The array defines the intercepts of the ellipsoidal surface with the axes of the reference frame.



Example: `simscape.Value([240 112 70], "mm")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

Ellipsoidal Solid

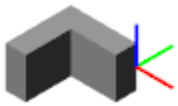
simscape.multibody.GeneralExtrusion class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Geometry`

Create extruded geometry with arbitrary polygonal cross section

Description

Use an object of the `simscape.multibody.GeneralExtrusion` class to represent extruded geometry with an arbitrary polygonal cross-section. The cross section is in the *xy*-plane of the reference frame and can have a convex or concave shape. You can also specify a cross section with holes. The number of sides of the cross section must be greater than two, but has no upper limit. The direction of the extrusion is along the *z*-axis of the reference frame.



To specify the dimensions of the geometry, use the `CrossSection` and `Length` properties.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`extrusion = simscape.multibody.GeneralExtrusion` creates an extruded geometry with default values.

`extrusion = simscape.multibody.GeneralExtrusion(crossSection,length)` creates an extruded geometry with specified cross section and length.

Properties

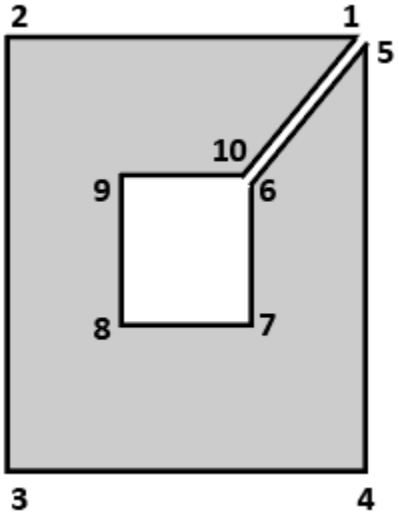
`CrossSection` — Cross section of extruded geometry

`simscape.Value([1 1; -1 1; -1 -1; 1 -1],"m")` (default) | `simscape.Value` object

Cross section of the extruded geometry, specified as a `simscape.Value` object that represents an *N*-by-2 matrix with a unit of length. *N* must be greater than two.

The matrix must define a closed loop in the *xy*-plane. The closed loop must have no self-intersecting segments. Each row of the matrix corresponds to the coordinates of a vertex of the cross-section polygon. The vertexes must be specified in a counter-clockwise order when viewed from the positive *z*-axis. The polygon does not have to be centered at the origin of the reference frame or even enclose the origin, but the polygon must not be self-intersecting.

To specify a cross section with holes, you can traverse the outer boundary and all inner boundaries in a single path. The traversal of the outer boundary is in the counter-clockwise direction and the traversal of the inner boundaries is in the clockwise direction.



Example: `simscape.Value([0 0;12 0;12 4;4 4;4 16;0 16],"cm");`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

`Length` — Extruded distance of geometry
`simscape.Value(1,"m")` (default) | `simscape.Value(L,"Length unit")`

Extruded distance of the geometry, specified as a `simscape.Value` object that represents a positive scalar with a unit of length.

Example: `simscape.Value(10,"cm")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

Extruded Solid | `simscape.multibody.Geometry`

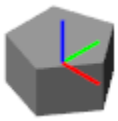
simscape.multibody.RegularExtrusion class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Geometry`

Create extruded geometry with regular polygonal cross section

Description

Use an object of the `simscape.multibody.RegularExtrusion` class to represent extruded geometry with a regular polygonal cross-section. The cross-section is in the *xy*-plane of the reference frame whose origin is coincident with the centroid of the extruded geometry. The *x*-axis of the reference frame is always an axis of symmetry of the cross section, and the *y*-axis of the reference frame is an axis of symmetry of the cross section when the value of `NumberOfSides` property is even. The direction of the extrusion is along the *z*-axis of the reference frame.



To specify the dimensions of the geometry, use the `NumberOfSides`, `OuterRadius`, and `Length` properties.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`extrusion = simscape.multibody.RegularExtrusion` creates a regular extruded geometry with default values.

`extrusion = simscape.multibody.RegularExtrusion(ns,radius,length)` creates a regular extruded geometry with the specified number of sides, radius, and length.

Properties

`NumberOfSides` — Number of sides of cross section

3 (default) | integer in the range of 3 to 64

Number of sides of the polygonal cross section, specified as an integer in the range of 3 to 64.

Example: 5

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Data Types: `Double`

`OuterRadius` — Radius of circle that circumscribes the cross section

`simscape.Value(1,"m")` (default) | `simscape.Value(R,"Length unit")`

Radius of the circle that circumscribes the cross section, specified as a `simscape.Value` object that represents a positive scalar with a unit of length.

Example: `simscape.Value(50,"mm")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`Length` — Extruded length of geometry

`simscape.Value(1,"m")` (default) | `simscape.Value(L,"Length unit")`

Extruded distance of the geometry, specified as a `simscape.Value` object that represents a positive scalar with a unit of length.

Example: `simscape.Value(10,"cm")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

Extruded Solid | `simscape.multibody.Geometry`

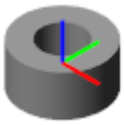
simscape.multibody.Revolution class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Geometry`

Create revolved geometry

Description

Use an object of the `simscape.multibody.Revolution` class to represent revolved geometry. The geometry is a rotational sweep of a polygonal cross-section specified in the *xz*-plane. The rotational axis is along the *z*-axis of the reference frame. To specify the geometry, you can use the `CrossSection` and `RevolutionAngle` properties.



Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`rev = simscape.multibody.Revolution` creates a revolved geometry with default values.

`rev = simscape.multibody.Revolution(CrossSection)` creates a fully revolved geometry with the specified cross-section.

`rev = simscape.multibody.Revolution(crosssection, angle)` creates a partially revolved geometry with the specified cross-section and revolution angle.

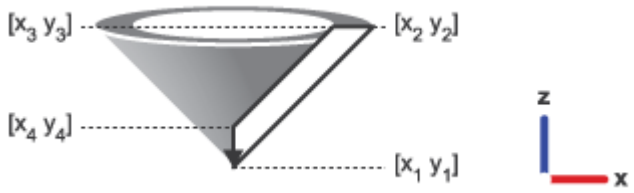
Properties

`CrossSection` — Cross-section of revolved geometry

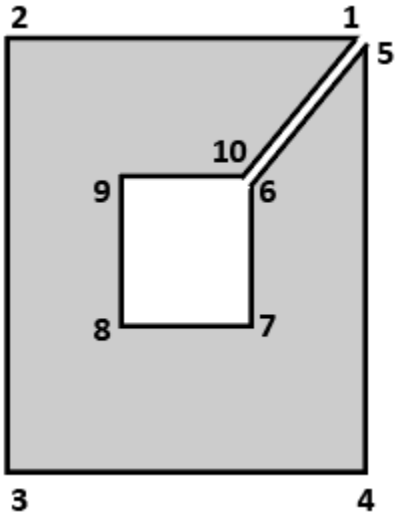
`simscape.Value([1 1;1 -1;2 -1;2 1],"m")` (default) | `simscape.Value` object

Cross-section of the revolved geometry, specified as a `simscape.Value` object that presents a *N*-by-2 matrix with a unit of length. *N* must be greater than two.

The matrix must define a closed loop in the *xz*-plane. The closed loop must have no self-intersecting segments, each row of the matrix must correspond to coordinates of a vertex of the cross-section polygon, and the vertexes must be specified in a counter-clockwise order when viewed from the negative *y*-axis. The *x* coordinate of each vertex must be nonnegative.



To specify a cross section with holes, you can traverse the outer boundary and all inner boundaries in a single path. The traversal of the outer boundary is in the counter-clockwise direction and the traversal of the inner boundary is in the clockwise direction.



Example: `simscape.Value([10 1;5 1;5 -1;10 -1],"mm")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

`FullExtent` — Flag that indicates if geometry has full revolution

`true` (default) | `false`

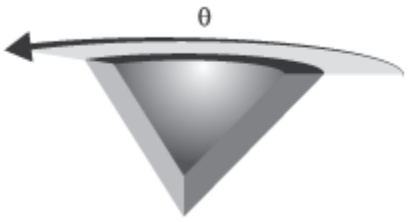
Flag that indicates if the geometry has a full revolution, returned as `true` or `false`. A `true` value means that the geometry has a full revolution. If the `FullExtent` property is true, the value of the `RevolutionAngle` property equals 360 degrees or 2 pi.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

RevolutionAngle — Angle of rotational sweep
`simscape.Value(360,"deg")` (default) | `simscape.Value` object

Angle of rotational sweep, specified as a `simscape.Value` object that represents a scalar with a unit of angle. The angle must be positive and have a maximum value of 360 degrees or 2 pi. Specifying the `RevolutionAngle` automatically sets the `FullExtent` property to `false`.



Example: `simscape.Value(90,"deg")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

Revolved Solid | `simscape.multibody.Geometry`

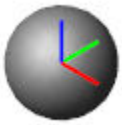
simscape.multibody.Sphere class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Geometry`

Create spherical geometry

Description

Use an object of the `simscape.multibody.Sphere` class to represent spherical geometry. The reference frame of the sphere is located at the centroid of the sphere. To specify the dimensions of the geometry, use the `Radius` property.



Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`sphere = simscape.multibody.sphere` creates a spherical geometry with default values.

`sphere = simscape.multibody.Sphere(radius)` creates a spherical geometry with the specified radius.

Properties

`Radius` — Radius of sphere

`simscape.Value(1,"m")` (default) | `simscape.Value(R,"Length unit")`

Radius of the spherical geometry, specified a `simscape.Value` object that represents a positive scalar with a unit of length.

Example: `simscape.Value(85,"mm")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

Spherical Solid | `simscape.multibody.Geometry`

simscape.multibody.Inertia class

Package: `simscape.multibody`

Abstract base class for specifying mass distributions

Description

`simscape.multibody.Inertia` class is the abstract base class for specifying mass distributions. To construct a mass distribution, use a subclass of the `simscape.multibody.Inertia` class, such as `simscape.multibody.PointMass`, `simscape.multibody.UniformDensity`, or `simscape.multibody.MassDistribution`.

Class Attributes

`Abstract`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Version History

Introduced in R2022a

See Also

`simscape.multibody.MassDistribution` | `simscape.multibody.PointMass` | `simscape.multibody.UniformDensity` | `simscape.multibody.UniformMass`

simscape.multibody.MassDistribution class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Inertia`

Construct custom mass distribution

Description

Use an object of the `simscape.multibody.MassDistribution` class to construct a custom mass distribution that explicitly specifies the mass, center of mass, moments of inertia, and products of inertia. The mass distribution has an implicit reference frame. If you use a `simscape.multibody.MassDistribution` object to specify the mass distribution of a solid, the implicit reference frame is coincident with the reference frame of the solid.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`mdmass = simscape.multibody.Massdistribution` constructs a custom mass distribution using default values.

`mdmass = simscape.multibody.Massdistribution(mass)` constructs a custom mass distribution and sets the total mass to `mass`.

`mdmass = simscape.multibody.Massdistribution(mass,com)` constructs a custom mass distribution and also sets the specified center of mass to `com`.

`mdmass = simscape.multibody.Massdistribution(mass,com,moi)` constructs a custom mass distribution and also sets the moments of inertia to `moi`.

`mdmass = simscape.multibody.Massdistribution(mass,com,moi,poi)` constructs a custom mass distribution and also sets the products of inertia to `poi`.

Properties

`Mass` — Total mass

`simscape.Value(1,"kg")` (default) | `simscape.Value(M,"Mass unit")`

Total mass, specified as a `simscape.Value` object that represents a scalar with a unit of mass. The scalar can be positive or negative. Use a negative value to capture the effect of a void or cavity in a

compound body that comprises multiple solids. The total mass of the compound body must be positive.

Example: `simscape.Value(16,"lbm")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`CenterOfMass` — Center-of-mass coordinates

`simscape.Value([0 0 0]',"m")` (default) | `simscape.Value([x y z],"Length unit")` | `simscape.Value([x y z]',"Length unit")`

Center-of-mass coordinates, specified as a `simscape.Value` object that represents a 3-by-1 or 1-by-3 array with a unit of length. The coordinates are resolved in the implicit reference frame of the mass distribution.

Example: `simscape.Value([0 0 5],"mm")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`MomentsOfInertia` — Diagonal elements of inertia matrix

`simscape.Value([0 0 0]',"kg*m^2")` (default) | `simscape.Value([`I_{xx} I_{yy} I_{zz}`],"mass*length^2")` | `simscape.Value([`I_{xx} I_{yy} I_{zz}`]',"mass*length^2")`

I_{xx}, I_{yy}, and I_{zz} elements of the inertia matrix, specified as a `simscape.Value` object that represents a 3-by-1 or 1-by-3 array with a unit of mass*length^2.

The quantities, I_{xx}, I_{yy}, and I_{zz}, are the moments of inertia with respect to the axes of a frame whose origin is coincident with the center of mass and whose axes are parallel to the implicit reference frame. The moments of inertia are the diagonal elements of the inertia matrix

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

where:

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

Example: `simscape.Value([12 15 10],"kg*cm^2")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

ProductsOfInertia — Off-diagonal elements of inertia matrix

`simscape.Value([0 0 0]',"kg*m^2")` (default) | `simscape.Value([`I_{yz} I_{zx} I_{xy}`],"mass*length^2")` | `simscape.Value([`I_{yz} I_{zx} I_{xy}`]',"mass*length^2")`

I_{yz}, I_{zx}, and I_{xy} elements of the inertia matrix, specified as a `simscape.Value` object that represents a 3-by-1 or 1-by-3 array with a unit of mass*length^2.

The quantities, I_{xy}, I_{yz}, and I_{zx} are the products of inertia with respect to the axes of a frame whose origin is coincident with the center of mass and whose axes are parallel to the implicit reference frame. 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:

- $I_{yz} = -\int_m yz\,dm$
- $I_{zx} = -\int_m zx\,dm$
- $I_{xy} = -\int_m xy\,dm$

Example: `simscape.Value([-2 3 1],"kg*cm^2")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Methods

Public Methods

inertiaMatrix	Compute inertia matrix
principalMomentsOfInertia	Compute principal moments of inertia
principalToReferenceTransformation	Compute 3-D rigid transformation between principal and reference frames
transform	Transform inertia to different frame

Version History

Introduced in R2022a

See Also

`simscape.multibody.Geometry` | `simscape.multibody.Inertia`

simscape.multibody.PointMass class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Inertia`

Construct point mass

Description

Use an object of the `simscape.multibody.PointMass` class to construct a point mass. The moments of inertia and products of inertia of a point mass are zero.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`pmass = simscape.multibody.PointMass` constructs a point mass with default values.

`pmass = simscape.multibody.PointMass(mass)` constructs a point mass with the specified total mass, `mass`.

Properties

`Mass` — Mass of point mass

`simscape.Value(1,"kg")` (default) | `simscape.Value(M,"Mass unit")`

Mass of the point mass, specified as a `simscape.Value` object that represents a scalar with a unit of mass.

Example: `simscape.Value(10,"kg")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

`simscape.multibody.Geometry` | `simscape.multibody.Inertia`

simscape.multibody.UniformDensity class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Inertia`

Construct uniform mass distribution with given density

Description

Use an object of the `simscape.multibody.UniformDensity` class to construct a uniform mass distribution with a given density. If you define the mass distribution of a solid by using a `UniformDensity` object, the solid has a constant density over the entire geometry. To compute the properties of a solid, such as the center of mass, moments of inertia, and products of inertia, the `UniformDensity` object uses the geometry and mass of the solid.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`udmass = simscape.multibody.UniformDensity` constructs a uniform mass distribution with default values.

`udmass = simscape.multibody.UniformDensity(density)` constructs a uniform mass distribution with the specified density.

Properties

`Density` — Density

`simscape.Value(1000,"kg/m^3")` (default) | `simscape.Value(Rho,"Density unit")`

Density, specified as a `simscape.Value` object that represents a scalar with a unit of density.

Example: `simscape.Value(2700,"kg/m^3")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

`simscape.multibody.Geometry` | `simscape.multibody.Inertia` | `simscape.multibody.UniformMass`

simscape.multibody.UniformMass class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Inertia`

Construct uniform mass distribution with given total mass

Description

Use an object of the `simscape.multibody.UniformMass` class to construct a uniform mass distribution with a given total mass. If you use a `UniformMass` object to define the mass distribution of a solid, the solid has a constant density over the entire geometry. The density is equal to the given total mass divided by the volume of the solid. To compute the properties of a solid, such as center of mass, moments of inertia, and products of inertia, the `UniformMass` object uses the geometry and mass of the solid.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`umass = simscape.multibody.UniformMass` constructs a uniform mass distribution with default values.

`umass = simscape.multibody.UniformMass(mass)` constructs a uniform mass distribution with the specified total mass.

Properties

`Mass` — Total mass

`simscape.Value(1,"kg")` (default) | `simscape.Value(M,"Mass unit")`

Total mass, specified as a `simscape.Value` object that represents a scalar with a unit of mass. The scalar can be positive or negative. Use a negative value to capture the effect of a void or cavity in a compound body that comprises multiple solids. The total mass of the compound body must be positive.

Example: `simscape.Value(16,"lbm")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

simscape.multibody.Geometry | simscape.multibody.Inertia | simscape.multibody.UniformDensity

inertiaMatrix

Class: `simscape.multibody.MassDistribution`
Package: `simscape.multibody`

Compute inertia matrix

Syntax

```
IM = inertiaMatrix(md)
```

Description

`IM = inertiaMatrix(md)` computes the inertia matrix of the provided `simscape.multibody.MassDistribution` object, `md`.

Input Arguments

`md` — Mass distribution

`simscape.multibody.MassDistribution` object

Mass distribution, specified as a `simscape.multibody.MassDistribution` object.

Output Arguments

`IM` — Inertia matrix

`simscape.Value([`I_{xx} I_{xy} I_{zx}; I_{xy} I_{yy} I_{yz}; I_{zx} I_{yz} I_{zz}`],"mass*length^2")`

Inertia matrix of the mass distribution, returned as a `simscape.Value` object that represents a 3-by-3 matrix with a unit of mass*length^2. The diagonal elements of the inertia matrix are the moments of inertia, and the off-diagonal elements of the inertia matrix are the products of inertia.

Attributes

Access	public

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.MassDistribution`

principalMomentsOfInertia

Class: `simscape.multibody.MassDistribution`
Package: `simscape.multibody`

Compute principal moments of inertia

Syntax

```
pmoi = principalMomentsOfInertia(md)
```

Description

`pmoi = principalMomentsOfInertia(md)` computes the principal moments of inertia of the provided `simscape.multibody.MassDistribution` object, `md`.

Input Arguments

`md` — Mass distribution

`simscape.multibody.MassDistribution` object

Mass distribution, specified as a `simscape.multibody.MassDistribution` object.

Output Arguments

`pmoi` — Principal moments of inertia

`simscape.Value([`I_{xx} I_{yy} I_{zz}`]',"mass*length^2")`

Principal moments of a inertia, returned as a `simscape.Value` object that represents a 3-by-1 array with a unit of mass*length^2.

Attributes

Access	public

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.MassDistribution`

principalToReferenceTransformation

Class: `simscape.multibody.MassDistribution`
Package: `simscape.multibody`

Compute 3-D rigid transformation between principal and reference frames

Syntax

```
T = principalToReferenceTransformation(md)
```

Description

`T = principalToReferenceTransformation(md)` computes the 3-D rigid transformation that maps the coordinates from the principal frame to the implicit reference frame of the `simscape.multibody.MassDistribution` object, `md`.

Input Arguments

`md` — Mass distribution

`simscape.multibody.MassDistribution` object

Mass distribution, specified as a `simscape.multibody.MassDistribution` object.

Output Arguments

`T` — 3-D rigid transformation between principal and reference frames

`simscape.multibody.Transformation` object

3-D rigid transformation between the principal and implicit reference frames of the mass distribution, returned as a `simscape.multibody.Transformation` object.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.MassDistribution` | `principalMomentsOfInertia`

transform

Class: `simscape.multibody.MassDistribution`
Package: `simscape.multibody`

Transform inertia to different frame

Syntax

```
nmd = transform(md,T)
```

Description

`nmd = transform(md,T)` transforms the provided `simscape.multibody.MassDistribution` object, `md`, by using the 3-D rigid transformation, `T`. If `T` is a transformation from the implicit reference frame of the mass distribution to a new frame, the method returns the same `MassDistribution` object, but expressed in the new frame.

Input Arguments

`md` — Mass distribution

`simscape.multibody.MassDistribution` object

Mass distribution, specified as a `simscape.multibody.MassDistribution` object.

`T` — 3-D rigid transformation

`simscape.multibody.Transformation` object

3-D rigid transformation from the implicit reference frame of a mass distribution to a new frame, specified as a `simscape.multibody.Transformation` object.

Output Arguments

`nmd` — New mass distribution

`simscape.multibody.MassDistribution` object

New mass distribution after the specified 3-D rigid transformation, returned as a `simscape.multibody.MassDistribution` object. The returned value is the same mass distribution but expressed in a new frame.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.MassDistribution`

simscape.multibody.JointPrimitive class

Package: `simscape.multibody`

Abstract base class to construct joint primitives

Description

`simscape.multibody.JointPrimitive` is the abstract base class for joint primitives, such as prismatic and revolute primitives. Joint primitives are the building blocks for of joints. To construct a joint primitive, use a subclass of the `simscape.multibody.JointPrimitive` class, such as `simscape.multibody.PrismaticPrimitive` or `simscape.multibody.RevolutePrimitive`.

The `JointPrimitive` class has two properties. The `DegreesOfFreedom` property indicates the degrees of freedom between the base and follower frames of the joint primitive. The `ForceLaws` property shows the force law applied to the joint primitive. See `simscape.multibody.JointForceLaw` class for more information about how to specify a force or torque for a joint primitive.

Class Attributes

`Abstract`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Properties

`DegreesOfFreedom` — Degrees of freedom of joint primitive

positive integer

Degrees of freedom between the base and follower frames of the joint primitive, returned as a positive integer.

Attributes:

`GetAccess`	`public`
`NonCopyable`	`true`

Data Types: `double`

`ForceLaws` — Force law applied to joint primitive

`JointForceLaw` array (default) | object of subclass of `simscape.multibody.JointForceLaw` class

Force law applied to the joint primitive, specified as an object of one subclass of the `simscape.multibody.JointForceLaw` class. Only objects of the `simscape.multibody.PrismaticPrimitive`, `simscape.multibody.RevolutePrimitive`, and `simscape.multibody.SphericalPrimitive` classes support the `ForceLaws` property.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

simscape.multibody.ConstantVelocityPrimitive | simscape.multibody.JointForceLaw | simscape.multibody.Joint | simscape.multibody.LeadScrewPrimitive | simscape.multibody.PrismaticPrimitive | simscape.multibody.RevolutePrimitive | simscape.multibody.SphericalPrimitive

simscape.multibody.ConstantVelocityPrimitive class

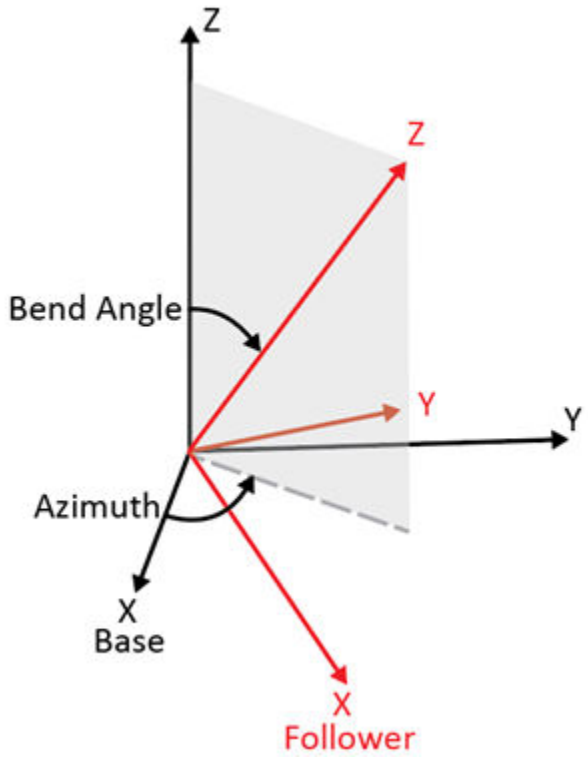
Package: `simscape.multibody`
Superclasses: `simscape.multibody.JointPrimitive`

Construct constant-velocity joint primitive

Description

Use an object of the `simscape.multibody.ConstantVelocityPrimitive` class to construct a constant-velocity joint primitive. A constant-velocity joint primitive models a specialized coupling between two shafts whose spin rates are exactly matched even when the shafts are not aligned.

A constant-velocity primitive is parameterized by two angles: bend angle and azimuth. These angles define the relative orientation between the base and follower frames of the primitive. The azimuth specifies the rotation of the follower frame about the *z*-axis of the base frame and locates the plane in which the bend angle occurs. In the figure, the dashed line indicates the intersection between the plane of the bend angle and the *xy*-plane of the base frame. The bend angle specifies the orientation of the *z*-axis of the follower frame with respect to the *z*-axis of the base frame.



The `InternalState` property of a `ConstantVelocityPrimitive` object specifies how to represent the internal state of the primitive. You can specify the property as either a *z*-*y*-*z* rotation sequence or a quaternion.

Class Attributes

Sealed	true
ConstructOnLoad	true
RestrictsSubclassing	true

For information on class attributes, see “Class Attributes”.

Creation

Description

`cvp = simscape.multibody.ConstantVelocityPrimitive` constructs a constant-velocity joint primitive with default values.

Properties

`InternalState` — Internal state representation
`simscape.multibody.ConstantVelocityInternalState.RotationSequence` member (default) | `simscape.multibody.ConstantVelocityInternalState.Quaternion` member

Internal state representation, specified as a member of the `simscape.multibody.ConstantVelocityInternalState` class. The choice does not affect the kinematics of the primitive. In common cases, using the `RotationSequence` member results in faster simulations but induces a kinematic singularity when the bend angle is zero. Using the `Quaternion` member does not suffer from a kinematic singularity when the bend angle is zero.

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

`simscape.multibody.JointPrimitive` | `simscape.multibody.ConstantVelocityJoint`

simscape.multibody.LeadScrewPrimitive class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.JointPrimitive`

Construct lead-screw primitive

Description

Use an object of the `simscape.multibody.LeadScrewPrimitive` class to construct a lead-screw joint primitive. A lead-screw joint primitive has one degree of freedom and models a pair of coupled rotation and translation. The rotation is about the *z*-axis of the `LeadScrewPrimitive` object, and the translation is along the *z*-axis. A helpful mental model is how a nut moves along a threaded screw.

To fully characterize the kinematics of a lead-screw joint primitive, use the `Lead` and `Direction` properties of the primitive to specify the relationship between the rotation and translation.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`lsp = simscape.multibody.LeadScrewPrimitive` constructs a lead-screw joint primitive with default values.

Properties

Lead — Lead of lead-screw primitive

`simscape.Value(1,"mm/rev")` (default) | `simscape.Value(ratio,"Length/angle unit")`

Lead of the lead-screw primitive, specified as a `simscape.Value` object that represents a scalar with a unit of length/angle. The scalar must be positive and specifies the axial translation the follower frame makes with respect to the base frame per unit of revolution of the follower frame.

Example: `simscape.Value(0.5,"mm/rev")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

`Direction` — Handedness of lead-screw primitive
`simscape.multibody.Handedness.RightHand` member (default) | `simscape.multibody.Handedness.LeftHand` member

Handedness of the lead-screw primitive, specified as a member of the `simscape.multibody.Handedness` class. When you set the property to the `RightHand` member, the angular position of the primitive increases as the translational position increases. When you set this property to the `LeftHand` member, the angular position of the primitive increases as the translational position decreases.

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

Topics
`simscape.multibody.JointPrimitive`
`simscape.multibody.LeadScrewJoint`

simscape.multibody.PrismaticPrimitive class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.JointPrimitive`

Construct prismatic joint primitive

Description

Use an object of the `simscape.multibody.PrismaticPrimitive` class to construct a prismatic joint primitive. The prismatic joint primitive has one translational degree of freedom along an axis of the base frame. The prismatic primitive appears in many joint objects, such as `simscape.multibody.PrismaticJoint` and `simscape.multibody.PlanarJoint`.

Set the `ForceLaws` property of a `PrismaticPrimitive` object to a `simscape.multibody.AxialSpringDamper` object to apply an axial force to the primitive.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`rp = simscape.multibody.PrismaticPrimitive` constructs a prismatic joint primitive with default values.

Version History

Introduced in R2022a

See Also

`simscape.multibody.JointPrimitive` | `simscape.multibody.AxialSpringDamper` | `simscape.multibody.PrismaticJoint`

simscape.multibody.RevolutePrimitive class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.JointPrimitive`

Construct revolute joint primitive

Description

Use an object of the `simscape.multibody.RevolutePrimitive` class to construct a revolute joint primitive. A revolute joint primitive has one rotational degree of freedom about an axis of the base frame. The revolute primitive appears in many joint objects, such as `simscape.multibody.RevoluteJoint` and `simscape.multibody.UniversalJoint`.

Set the `ForceLaws` property of a revolute joint primitive to a `simscape.multibody.TorsionalSpringDamper` object to apply a torque to the primitive.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`rp = simscape.multibody.RevolutePrimitive` constructs a revolute joint primitive with default values.

Version History

Introduced in R2022a

See Also

`simscape.multibody.TorsionalSpringDamper` | `simscape.multibody.JointPrimitive` | `simscape.multibody.RevoluteJoint`

simscape.multibody.SphericalPrimitive class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.JointPrimitive`

Construct spherical joint primitive

Description

Use an object of the `simscape.multibody.SphericalPrimitive` class to construct a spherical joint primitive. A spherical joint primitive has three rotational degrees of freedom and can model an arbitrary 3-D rotation. The spherical primitive appears in many joint objects, such as `simscape.multibody.TelescopingJoint` and `simscape.multibody.SphericalJoint`.

Set the `ForceLaws` property of a spherical joint primitive to a `simscape.multibody.SphericalSpringDamper` object to apply a torque to the primitive.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`sp = simscape.multibody.SphericalPrimitive` constructs a spherical joint primitive with default values.

Version History

Introduced in R2022a

See Also

`simscape.multibody.JointPrimitive` | `simscape.multibody.SphericalSpringDamper` | `simscape.multibody.SphericalJoint`

simscape.multibody.JointForceLaw class

Package: `simscape.multibody`

Abstract base class to construct joint force laws

Description

The `simscape.multibody.JointForceLaw` is the abstract base class for joint force laws. The joint force laws describe the forces and torques that are applied to joint primitives. The forces and torques are functions of the dynamic state of joint primitives. To apply a force or torque to a joint primitive, use a subclass of the `simscape.multibody.JointForceLaw` class, such as `simscape.multibody.AxialSpringDamper` or `simscape.multibody.TorsionalSpringDamper`.

Class Attributes

`Abstract`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Version History

Introduced in R2022a

See Also

`simscape.multibody.AxialSpringDamper` | `simscape.multibody.SphericalSpringDamper` | `simscape.multibody.TorsionalSpringDamper`

simscape.multibody.AxialSpringDamper class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.JointForceLaw`

Construct axial spring-damper force law

Description

Use an object of the `simscape.multibody.AxialSpringDamper` class to construct an axial spring-damper force law. You can use the force law to apply a pure axial force to a `simscape.multibody.PrismaticPrimitive` object. The force is based on the current position and velocity of the primitive

$F = -k_s \cdot (p - p_0) - k_d \dot{p},$

where:

Symbol	Description
F	Applied axial force
p_0	Equilibrium position of the spring
k_s	Spring stiffness
k_d	Damping coefficient
p	Prismatic primitive position
$\dot{p}$	Prismatic primitive velocity

Class Attributes

`Sealed` `true`

`ConstructOnLoad` `true`

`RestrictsSubclassing` `true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`asd = simscape.multibody.AxialSpringDamper` constructs an axial spring-damper force law and sets the parameters to `simscape.Value` objects with a value of zero.

Properties

EquilibriumPosition — Equilibrium position of spring

`simscape.Value(0,"m")` (default) | `simscape.Value(`p_0`,"Length unit")`

Equilibrium position of the spring, specified as a `simscape.Value` object that represents a scalar with a unit of length. The scalar can be positive, negative, or zero. The axial spring force is zero when the primitive position is at the equilibrium position.

Example: `simscape.Value(1,"mm")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`SpringStiffness` — Stiffness of spring
`simscape.Value(0,"N/m")` (default) | `simscape.Value(`k_s`,"Force/length unit")`

Stiffness of the spring, specified as a `simscape.Value` object that represents a scalar with a unit of force/length. The scalar is a measure of how hard the spring pushes the primitive position toward the equilibrium position and must be nonnegative.

Example: `simscape.Value(1,"N/m")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`DampingCoefficient` — Damping coefficient of damper
`simscape.Value(0,"N/(m/s)")` (default) | `simscape.Value(`k_d`,"Force/linear velocity unit")`

Damping coefficient of the damper, specified as a `simscape.Value` object that represents a scalar with a unit of force/linear velocity. The scalar is a measure of how hard the damper resists the motion of a primitive and must be nonnegative.

Example: `simscape.Value(1,"N/(m/s)")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

`simscape.multibody.JointForceLaw`

simscape.multibody.SphericalSpringDamper class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.JointForceLaw`

Construct spherical spring-damper force law

Description

Use an object of the `simscape.multibody.SphericalSpringDamper` class to construct a spherical spring-damper force law. You can use the force law to apply a pure torque to a `simscape.multibody.SphericalPrimitive` object.

The spherical spring-damper force law is a vector force law. The deviation between the current and equilibrium position is expressed as

$R_{rel} = R_e^{-1} \cdot R,$

where:

Symbol	Description
R_{rel}	Relative rotation, specified as a vector.
R_e	Equilibrium position of a spherical primitive, specified as a vector.
R	Current position of the spherical primitive, specified as a vector that represents a 3-D rotation from the follower to the base frame of the primitive.

The relative rotation can be parameterized by a natural angle-axis pair. The angle, θ, is a scalar angle in the range of [0,180] degree, and the axis, u, is a unit vector along the axis. Therefore, the torque is expressed as

$T = -k_s\theta u - k_d\omega,$

where:

Symbol	Description
T	Applied torque, specified as a vector.
k_s	Spring stiffness
k_d	Damping coefficient
u	Natural axis of the rotation, specified as a vector. For more information about the natural axis, see `naturalAxis` method.

Symbol	Description
θ	Natural angle of the rotation. For more information about the natural angle, see `naturalAngle` method.
ω	Spherical primitive 3-D angular velocity, specified as a vector.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`ssd = simscape.multibody.SphericalSpringDamper` constructs a spherical spring-damper force law with default values.

Properties

`EquilibriumPosition` — Equilibrium position of spring
`simscape.multibody.ZeroRotation` object (default) | objects of the subclasses of the `simscape.multibody.Rotation` class

Equilibrium position of the spherical spring, specified as an object of a subclass of the `simscape.multibody.Rotation` class. The torque is zero when the primitive position is at the equilibrium position.

Example: `simscape.multibody.AlignedAxesRotation(simscape.multibody.Axis.PosX, simscape.multibody.Axis.NegZ,simscape.multibody.Axis.NegY,simscape.multibody.Axis.PosY)`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

`SpringStiffness` — Stiffness of spring
`simscape.Value(0,"N*m/deg")` (default) | `simscape.Value(`k_s`,"Torque/angle unit")`

Stiffness of the spherical spring, specified as a `simscape.Value` object that represents a scalar with a unit of torque/angle. The scalar is a measure of how hard the spring pushes the primitive position toward the equilibrium position and must be nonnegative.

Example: `simscape.Value(5,"N*m/deg")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`DampingCoefficient` — Damping coefficient of spring
`simscape.Value(0,"N*m/(deg/s)")` (default) | `simscape.Value(`k_d`,"Torque/angular velocity unit")`

Damping coefficient of the spherical damper, specified as a `simscape.Value` object that represents a scalar with a unit of torque/angular velocity. The scalar is a measure of how hard the damper resists the motion of a primitive and must be nonnegative.

Example: `simscape.Value(10,"N*m/(deg/s))`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

`simscape.multibody.JointForceLaw`

simscape.multibody.TorsionalSpringDamper class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.JointForceLaw`

Construct torsional spring-damper force law

Description

Use an object of the `simscape.multibody.TorsionalSpringDamper` class to construct a torsional spring-damper force law. You can use the force law to apply a pure torque to a `simscape.multibody.RevolutePrimitive` object. The torque is based on the current position and velocity of the primitive

$T = -k_s \cdot (q - q_0) - k_d \dot{q},$

where:

Symbol	Description
T	Applied torque
q_0	Equilibrium position of the spring
k_s	Spring stiffness
k_d	Damping coefficient
q	Revolute primitive position
$\dot{q}$	Revolute primitive velocity

Class Attributes

`Sealed` `true`

`ConstructOnLoad` `true`

`RestrictsSubclassing` `true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`tsd = simscape.multibody.TorsionalSpringDamper` constructs a torsional spring-damper force law and sets the parameters to `simscape.Value` objects with a value of zero.

Properties

EquilibriumPosition — Equilibrium position of spring

`simscape.Value(0,"deg")` (default) | `simscape.Value(`q_0`,"Angle unit")`

Equilibrium position of the torsional spring, specified as a `simscape.Value` object that represents a scalar with a unit of angle. The scalar can be positive, negative, or zero. The torque is zero when the primitive position is at the equilibrium position.

Example: `simscape.Value(10,"deg")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`SpringStiffness` — Stiffness of spring

`simscape.Value(0,"N*m/deg")` (default) | `simscape.Value(`k_s`,"Torque/angle unit")`

Stiffness of the torsional spring, specified as a `simscape.Value` object that represents a scalar with a unit of torque/angle. The scalar is a measure of how hard the spring pushes the primitive position toward the equilibrium position and must be nonnegative.

Example: `simscape.Value(5,"N*m/deg")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`DampingCoefficient` — Damping coefficient of damper

`simscape.Value(0,"N*m/(deg/s)")` (default) | `simscape.Value(`k_d`,"Torque/angular velocity unit")`

Damping coefficient of the damper, specified as a `simscape.Value` object that represents a scalar with a unit of torque/angular velocity. The scalar is a measure of how hard the damper resists the motion of a primitive and must be nonnegative.

Example: `simscape.Value(10,"N*m/(deg/s))`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

`simscape.multibody.JointForceLaw`

simscape.multibody.Rotation class

Package: `simscape.multibody`

Abstract base class for 3-D rotations

Description

`simscape.multibody.Rotation` is the abstract base class for 3-D rotations. A rotation affects only the relative orientation between two arbitrary frames. To construct a rotation, use an object of a subclass of the `Rotation` class, such as `simscape.multibody.ArbitraryAxisRotation` or `simscape.multibody.RotationMatrixRotation`.

Class Attributes

`Abstract`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Methods

Public Methods

matrix	Compute rotation matrix
naturalAngle	Compute natural angle of rotation
naturalAxis	Compute natural axis of rotation
quaternion	Compute unit quaternion of rotation
sequenceAngles	Compute equivalent rotation sequence angles
transform	Transform 3-by-N matrix by specified rotation

Version History

Introduced in R2022a

See Also

`simscape.multibody.AlignedAxesRotation` | `simscape.multibody.ArbitraryAxisRotation` | `simscape.multibody.QuaternionRotation` | `simscape.multibody.RotationMatrixRotation` | `simscape.multibody.RotationSequenceRotation` | `simscape.multibody.StandardAxisRotation` | `simscape.multibody.ZeroRotation`

matrix

Class: `simscape.multibody.Rotation`
Package: `simscape.multibody`

Compute rotation matrix

Syntax

```
M = matrix(R)
```

Description

`M = matrix(R)` computes the rotation matrix of the rotation `R`.

Input Arguments

`R` — Rotation

object of subclass of `simscape.multibody.Rotation` class

Rotation, specified as an object of a subclass of the `simscape.multibody.Rotation` class.

Output Arguments

`M` — Rotation matrix

3-by-3 matrix

Rotation matrix, returned as a 3-by-3 matrix. The rotation matrix is orthogonal and has determinate of 1.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Rotation`

naturalAngle

Class: `simscape.multibody.Rotation`
Package: `simscape.multibody`

Compute natural angle of rotation

Syntax

```
q = naturalAngle(R)
```

Description

`q = naturalAngle(R)` computes the natural angle of the provided rotation, `R`. A 3-D rotation has a unique nonnegative angle of rotation in the range [0,pi] radians called the natural angle. Note that the value of this angle can be expressed in any angular unit.

The returned natural angle, `q`, has the same unit as the provided rotation. If the rotation does not have a unit, the natural angle is in degrees.

Input Arguments

R — Rotation

object of subclass of `simscape.multibody.Rotation` class

Rotation, specified as an object of a subclass of the `simscape.multibody.Rotation` class.

Output Arguments

q — Natural angle of rotation

`simscape.Value` object

Natural angle of the rotation, returned as a `simscape.Value` object that represents a scalar. `q` has the same units as the input argument `R`. If `R` does not have a unit, `q` is in degrees.

Attributes

Access	public

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Rotation` | `naturalAxis`

naturalAxis

Class: `simscape.multibody.Rotation`
Package: `simscape.multibody`

Compute natural axis of rotation

Syntax

```
ax = naturalAxis(R)
```

Description

`ax = naturalAxis(R)` computes the natural axis of the provided rotation, `R`. The natural axis is the axis of rotation that corresponds to the natural angle of the rotation. See `naturalAngle` for more information.

The returned natural axis, `ax`, is a 3-by-1 unit vector. If the natural angle is zero, the natural axis is the zero vector.

Input Arguments

R — Rotation

object of subclass of `simscape.multibody.Rotation` class

Rotation, specified as an object of a subclass of the `simscape.multibody.Rotation` class.

Output Arguments

ax — Natural axis of rotation

3-by-1 vector

Natural axis of the rotation, returned as a 3-by-1 unit vector.

Attributes

Access	public

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Rotation` | `naturalAngle`

quaternion

Class: `simscape.multibody.Rotation`
Package: `simscape.multibody`

Compute unit quaternion of rotation

Syntax

```
Q = quaternion(R)
```

Description

`Q = quaternion(R)` computes the unit quaternion of the desired rotation, `R` and returns a 4-by-1 unit vector,

$Q = (S\ V)$,

where:

$S = \cos\left(\frac{\theta}{2}\right)$,

and

$V = [U_x\, U_y\, U_z]\sin\left(\frac{\theta}{2}\right)$.

θ is the angle of rotation and $[U_x, U_y, U_z]$ is the unit vector of the rotational axis. Note that for any given rotation, there are two unit quaternions that are negatives of each other, but represent the same rotation. For example, the quaternions `[1 0 0 0]` and `[-1 0 0 0]` both represent the identity rotation.

Input Arguments

R — Rotation

object of subclass of `simscape.multibody.Rotation` class

Rotation, specified as an object of a subclass of the `simscape.multibody.Rotation` class.

Output Arguments

Q — Unit quaternion of rotation

`quaternion` object

Unit quaternion of the rotation, returned as a `quaternion` object that represents a four-element unit row vector.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Rotation` | `quaternion`

sequenceAngles

Class: `simscape.multibody.Rotation`
Package: `simscape.multibody`

Compute equivalent rotation sequence angles

Syntax

```
seq = sequenceAngles(R,axes,sequence)
```

Description

`seq = sequenceAngles(R,axes,sequence)` computes a set of rotation sequence angles to represent the provided rotation, `R`. The rotation sequence angles correspond to three successive elementary rotations. The `sequenceAngles` method computes the angles based on the specification of the `axes` and `sequence`.

Input Arguments

R — Rotation

object of subclass of `simscape.multibody.Rotation` class

Rotation, specified as an object of a subclass of the `simscape.multibody.Rotation` class.

axes — Selection of rotation axes

`simscape.multibody.FrameSide.Base` | `simscape.multibody.FrameSide.Follower`

Selection of rotation axes, specified as either the `simscape.multibody.FrameSide.Follower` or `simscape.multibody.FrameSide.Base` member. To compute angles for intrinsic or extrinsic rotations, use the `simscape.multibody.FrameSide.Follower` or `simscape.multibody.FrameSide.Base` member, respectively. For more information about the intrinsic or extrinsic rotations, see “Rotation Sequence Measurements”.

sequence — Sequence of three axes to rotate about

member of `simscape.multibody.AxisSequence` class

Sequence of three axes to rotate about, specified as a member of the `simscape.multibody.AxisSequence` class. Use one of the following members:

- `simscape.multibody.AxisSequence.XYX`
- `simscape.multibody.AxisSequence.XYZ`
- `simscape.multibody.AxisSequence.XZX`
- `simscape.multibody.AxisSequence.XZY`
- `simscape.multibody.AxisSequence.YXY`
- `simscape.multibody.AxisSequence.YXZ`
- `simscape.multibody.AxisSequence.YZX`
- `simscape.multibody.AxisSequence.YZY`

- `simscape.multibody.AxisSequence.ZXY`
- `simscape.multibody.AxisSequence.ZXZ`
- `simscape.multibody.AxisSequence.ZYX`
- `simscape.multibody.AxisSequence.ZYZ`

Output Arguments

seq — rotation sequence angles
`simscape.Value` object

Rotation sequence angles, specified as a `simscape.Value` object that represents a 3-by-1 column vector. Each element of the array is an angle.

The unit of the angles depends on which rotation subclass is used to create the object `R`. If the subclass, such as the `simscape.multibody.StandardAxisRotation` class, has an angular input argument, the unit of the rotation sequence angles matches with the angular unit specified in the subclass. If the subclass, such as the `simscape.multibody.RotationMatrixRotation` class, does not have an angular input argument, the rotation sequence angles are in degrees.

The `sequenceAngles` method and the **Sequence** parameter of the Transform Sensor block work similarly when computing rotation sequence angles. For more information about the computed angles and how the `sequenceAngles` method deals with locking issues, such the gimbal lock problem, see “Rotation Sequence Measurements”.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Examples

Compute Rotation Sequence Angles for a Rotation Object

This example shows how to compute equivalent rotation sequence angles for a rotation object constructed by the `simscape.multibody.StandardAxisRotation` class.

Use the `StandardAxisRotation` class to construct a rotation object `R` that represents a rotation about the x-axis of the base frame. The rotation angle is `1.05` rad.

```
angle = simscape.Value(1.05,"rad");
axis = simscape.multibody.Axis.PosX;
R = simscape.multibody.StandardAxisRotation(angle,axis);
```

Use the `sequenceAngles` method to compute the rotation sequence angles of the object `R`. Specify the base frame as the reference fame and X-Y-Z as the rotation sequence.

```
axes = simscape.multibody.FrameSide.Base;
sequence = simscape.multibody.AxisSequence.XYZ;
seq = sequenceAngles(R,axes,sequence)
```

```
seq =
    1.0500
         0
         0

    : rad
```

Version History

Introduced in R2023a

See Also

`simscape.multibody.Rotation` | `simscape.multibody.RotationSequenceRotation`

Topics

“Rotation Sequence Measurements”

transform

Class: `simscape.multibody.Rotation`
Package: `simscape.multibody`

Transform 3-by-N matrix by specified rotation

Syntax

```
xv = transform(R,v)
```

Description

`xv = transform(R,v)` transforms the value, `v`, by a specified rotation, `R`.

Input Arguments

R — Rotation

object of subclass of `simscape.multibody.Rotation` class

Rotation, specified as an object of a subclass of the `simscape.multibody.Rotation` class.

v — Value

matrix | `simscape.Value` object

Value, specified as either a 3-by-*N* matrix or a `simscape.Value` object that represents a 3-by-*N* matrix with an arbitrary unit, where *N* is a positive integer.

Output Arguments

xv — Transformed value

matrix | `simscape.Value` object

Transformed value, returned as either a 3-by-*N* matrix or a `simscape.Value` object that represents a matrix. This value has the same format as the value of `v` argument.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Rotation` | `simscape.multibody.Transformation`

simscape.multibody.AlignedAxesRotation class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Rotation`

Construct rotation by using aligned-axes parameterization

Description

Use an object of the `simscape.multibody.AlignedAxesRotation` class to construct a 3-D rotation by setting the two axes of the follower frame align with two axes of the base frame. Use the properties of the `AlignedAxesRotation` class to specify the desired axes with members of the `simscape.multibody.Axis` class:

- `simscape.multibody.Axis.PosX`: *x*-axis
- `simscape.multibody.Axis.NegX`: -*x*-axis
- `simscape.multibody.Axis.PosY`: *y*-axis
- `simscape.multibody.Axis.NegY`: -*y*-axis
- `simscape.multibody.Axis.PosZ`: *z*-axis
- `simscape.multibody.Axis.NegZ`: -*z*-axis

The axis specified by the `FollowerAxis1` property aligns with the axis specified by the `BaseAxis1` property, and the axis specified by the `FollowerAxis2` property aligns with the axis specified by the `BaseAxis2` property.

Note that the axes specified by the `BaseAxis1` and `BaseAxis2` properties must not be parallel, and the axes specified by the `FollowerAxis1` and `FollowerAxis2` must not be parallel.

Class Attributes

```
Sealed                                          true

ConstructOnLoad                                      true

RestrictsSubclassing                                    true
```

For information on class attributes, see “Class Attributes”.

Creation

Description

`R = simscape.multibody.AlignedAxesRotation` creates an aligned-axes rotation with default values.

`R = simscape.multibody.AlignedAxesRotation(followeraxis1,base1,followeraxis2, base2)` creates an aligned-axes rotation with the specified axes.

Properties

`FollowerAxis1` — First axis of follower frame

`simscape.multibody.Axis.PosX` (default) | member of `simscape.multibody.Axis` class

First axis of the follower frame, specified as a member of the `simscape.multibody.Axis` class.

Example: `simscape.multibody.Axis.NegY`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`BaseAxis1` — First axis of base frame

`simscape.multibody.Axis.PosX` (default) | member of `simscape.multibody.Axis` class

First axis of the base frame, specified as a member of the `simscape.multibody.Axis` class.

Example: `simscape.multibody.Axis.NegY`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`FollowerAxis2` — Second axis of follower frame

`simscape.multibody.Axis.PosY` (default) | member of `simscape.multibody.Axis` class

Second axis of the follower frame, specified as a member of the `simscape.multibody.Axis` class.

Example: `simscape.multibody.Axis.NegZ`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

`BaseAxis2` — Second axis of base frame

`simscape.multibody.Axis.PosY` (default) | member of `simscape.multibody.Axis` class

Second axis of the base frame, specified as a member of the `simscape.multibody.Axis` class.

Example: `simscape.multibody.Axis.NegZ`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

`simscape.multibody.Rotation`

simscape.multibody.ArbitraryAxisRotation class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Rotation`

Construct rotation by specifying custom axis-angle pair

Description

Use an object of the `simscape.multibody.ArbitraryAxisRotation` class to construct a 3-D rotation by specifying a rotation angle and the corresponding axis of rotation. The `Axis` property specifies the axis of rotation by using a 1-by-3 or 3-by-1 nonzero vector. The `Angle` property specifies the angle of the rotation.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`R = simscape.multibody.ArbitraryAxisRotation` creates a rotation by specifying a custom axis-angle pair with default values.

`R = simscape.multibody.ArbitraryAxisRotation(axis,angle)` creates a rotation by specifying a custom axis-angle pair with the specified axis of rotation and rotation angle.

Properties

`Axis` — Axis of rotation

`[0 0 1]'` (default) | 1-by-3 or 3-by-1 nonzero vector

Axis of rotation, specified as a 1-by-3 or 3-by-1 nonzero vector

Example: `[2 -1 5]`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

`Angle` — Angle of rotation

`simscape.Value(0,"deg")` (default) | `simscape.Value` object

Angle of rotation, specified as a `simscape.Value` object that represents a scalar in units of angle.

Example: `simscape.Value(2.5,"rad")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

`simscape.multibody.Rotation`

simscape.multibody.QuaternionRotation class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Rotation`

Construct rotation by using unit quaternion

Description

Use an object of the `simscape.multibody.QuaternionRotation` class to construct a 3-D rotation by using a `quaternion` object. A `quaternion` object is a four-element unit vector,

$Q = (S\ V)$,

where:

$S = \cos\left(\frac{\theta}{2}\right)$,

and

$V = [U_x\, U_y\, U_z]\sin\left(\frac{\theta}{2}\right)$.

θ is the angle of rotation and $[U_x, U_y, U_z]$ is the unit vector of the rotational axis. Note that for any given rotation, there are two quaternions that are negatives of each other, but represent the same rotation. For example, the quaternions `[1 0 0 0]` and `[-1 0 0 0]` both represent the identity rotation.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`R = simscape.multibody.QuaternionRotation` creates a rotation with default quaternion.

`R = simscape.multibody.QuaternionRotation(Q)` creates a rotation with the specified quaternion.

Properties

`Q` — Unit quaternion

`1 + 0i + 0j + 0k` (default) | `quaternion` object

Unit quaternion, specified as a `quaternion` object that represents a four-element unit vector.

Example: `0.70711 + 0.40825i + 0.40825j + 0.40825k`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

`simscape.multibody.Rotation` | `quaternion`

simscape.multibody.RotationMatrixRotation class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Rotation`

Construct rotation by using rotation matrix

Description

Use an object of the `simscape.multibody.RotationMatrixRotation` class to construct a 3-D rotation by using a rotation matrix specified by the `Matrix` property. The rotation matrix is a 3-by-3 matrix that is orthogonal and has the determinant of 1.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`R = simscape.multibody.RotationMatrixRotation` creates a rotation with a 3-by-3 identity matrix.

`R = simscape.multibody.RotationMatrixRotation(M)` creates a rotation with the specified rotation matrix.

Properties

`M` — Rotation matrix

3-by-3 identity matrix (default) | 3-by-3 matrix

Rotation matrix, specified as a 3-by-3 matrix that is orthogonal and has the determinant of 1.

Example: `[-0.3651 0.8165 0.4472;0.1826 -0.4082 0.8944;0.9129 0.4082 0]`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

`simscape.multibody.Rotation`

simscape.multibody.RotationSequenceRotation class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Rotation`

Construct rotation by using rotation-sequence parameterization

Description

Use an object of the `simscape.multibody.RotationSequenceRotation` class to construct a 3-D rotation by using a rotation-sequence parameterization. Any 3-D rotation can be described by three successive elementary rotations. The elementary rotations correspond to the axes of a frame that can be either the base or follower frame. Note that the base frame is fixed while the follower frame rotates after each elementary rotation.

The `Axes` property specifies which frame's axes the elementary rotations are performed about. The `Sequence` property specifies the sequence of the elementary rotations. The `Angles` property specifies the angles for the three elementary rotations. Note that consecutive axes are distinct.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`R = simscape.multibody.RotationSequenceRotation` creates a rotation-sequence rotation with default values.

`R = simscape.multibody.RotationSequenceRotation(axes,sequence,angles)` creates a rotation-sequence rotation with the specified axes of rotation, rotation sequence, and rotation angles.

Properties

`Axes` — Selection of rotation axes

`simscape.multibody.FrameSide.Follower` (default) | `simscape.multibody.FrameSide.Base`

Selection of rotation axes, specified as either the `simscape.multibody.FrameSide.Follower` or `simscape.multibody.FrameSide.Base` member. Use the `simscape.multibody.FrameSide.Base` member to set the axes of the base frame as rotation axes

for elementary rotations, or use `simscape.multibody.FrameSide.Follower` to set the axes of the follower frame as rotation axes for elementary rotations.

Example: `simscape.multibody.FrameSide.Base`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Sequence — Sequence of three axes to rotate about

`simscape.multibody.AxisSequence.XYX` (default) | member of `simscape.multibody.AxisSequence` class

Sequence of three axes to rotate about, specified as a member of the `simscape.multibody.AxisSequence` class. Use one of the following members to set the `Sequence` property:

- `simscape.multibody.AxisSequence.XYX`
- `simscape.multibody.AxisSequence.XYZ`
- `simscape.multibody.AxisSequence.XZX`
- `simscape.multibody.AxisSequence.XZY`
- `simscape.multibody.AxisSequence.YXY`
- `simscape.multibody.AxisSequence.YXZ`
- `simscape.multibody.AxisSequence.YZX`
- `simscape.multibody.AxisSequence.YZY`
- `simscape.multibody.AxisSequence.ZXY`
- `simscape.multibody.AxisSequence.ZXZ`
- `simscape.multibody.AxisSequence.ZYX`
- `simscape.multibody.AxisSequence.ZYZ`

Example: `simscape.multibody.AxisSequence.XZY`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Angles — Angles of elementary rotations

`simscape.Value([0 0 0]',"deg")` (default) | `simscape.Value` object

Angles of elementary rotations, specified as a `simscape.Value` object that represents a 3-by-1 or 1-by-3 array in units of angle.

Example: `simscape.Value([30 10 40],"deg")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

simscape.multibody.Rotation

simscape.multibody.StandardAxisRotation class

Package: simscape.multibody
Superclasses: simscape.multibody.Rotation

Construct rotation by using standard-axis parameterization

Description

Use an object of the simscape.multibody.StandardAxisRotation class to construct a 3-D rotation by specifying a rotation angle and the corresponding axis of rotation. The axis of rotation is one axis of the base frame. The Axis property specifies the axis of rotation by using a member of the simscape.multibody.Axis class. The Angle property specifies the angle of the rotation.

Class Attributes

Sealed	true
ConstructOnLoad	true
RestrictsSubclassing	true

For information on class attributes, see “Class Attributes”.

Creation

Description

R = simscape.multibody.StandardAxisRotation creates a standard-axis rotation with default values.

R = simscape.multibody.StandardAxisRotation(angle,axis) creates a standard-axis rotation with the specified rotation angle and axis of rotation.

Properties

Angle — Angle of rotation

simscape.Value(0,"deg") (default) | simscape.Value object

Angle of rotation, specified as a simscape.Value object that represents a scalar in units of angle.

Example: simscape.Value(2.5,"rad")

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Axis — Axis of rotation

simscape.multibody.Axis.PosZ (default) | member of simscape.multibody.Axis class

Axis of the rotation, specified as a member of the `simscape.multibody.Axis` class. To specify the axis, use one of the following members:

- `simscape.multibody.Axis.PosX`: *x*-axis of the base frame
- `simscape.multibody.Axis.NegX`: -*x*-axis of the base frame
- `simscape.multibody.Axis.PosY`: *y*-axis of the base frame
- `simscape.multibody.Axis.NegY`: -*y*-axis of the base frame
- `simscape.multibody.Axis.PosZ`: *z*-axis of the base frame
- `simscape.multibody.Axis.NegZ`: -*z*-axis of the base frame

Example: `simscape.multibody.Axis.NegY`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

`simscape.multibody.Rotation`

simscape.multibody.ZeroRotation class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Rotation`

Construct zero rotation

Description

Use an object of the `simscape.multibody.ZeroRotation` class to construct a zero rotation. A zero rotation requires the base and follower frames to be aligned.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`R = simscape.multibody.ZeroRotation` creates a zero rotation.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Rotation`

simscape.multibody.State class

Package: `simscape.multibody`

State of compiled multibody system

Description

Use an object of the `simscape.multibody.State` class to include the position and velocity of every joint primitive in a compiled multibody system. You use the `State` object in may analyses, such as computing the position or velocity of a joint primitive in the system.

To compute the state of a compiled multibody system, use the `computeState` method. The `Status` property shows whether the state is valid. If the state is not valid, the `Status` property shows what type of problem occurred.

You can only use a `State` object with the `simscape.multibody.CompiledMultibody` object that was used to create the `State` object. Attempting to pass a `State` object to a method of a different `CompiledMultibody` object generates an error.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

To create the `simscape.multibody.State` object of a compiled multibody system, use the `computeState` method.

Properties

`Status` — State status

member of `simscape.multibody.StateStatus` class

State status, returned as a member of the `simscape.multibody.StateStatus` class. The status indicates the high-level result of attempt at computing the state of a compiled multibody system:

- `simscape.multibody.StateStatus.Valid`: All kinematic constraints of the multibody system are satisfied. Note that a valid status does not indicate that all the specified targets have been met.
- `simscape.multibody.StateStatus.PositionViolation`: Not all position constraints of the multibody system are satisfied.
- `simscape.multibody.StateStatus.VelocityViolation`: All position constraints are satisfied, but at least one velocity constraint is not satisfied.

- `simscape.multibody.StateStatus.SingularityViolation`: All kinematic constraints of the multibody system are satisfied, but at least one joint primitive is in kinematic singularity.

To see the status of individual targets, enter the `state` object at the command line without a trailing semicolon.

Attributes:

`GetAccess`	`public`
`SetAccess`	`Restricts access`
`NonCopyable`	`true`
`Transient`	`true`

Version History

Introduced in R2022a

See Also

`simscape.multibody.CompiledMultibody` | `computeState`

simscape.multibody.Transformation class

Package: `simscape.multibody`

Construct transformation

Description

Use an object of the `simscape.multibody.Transformation` class to construct a 3-D transformation. Like a `simscape.multibody.RigidTransform` object, a `Transformation` object includes a translation and a rotation. However, a `Transformation` object is a mathematical object that is used only in 3-D computations and analyses.

To specify the `Rotation` property, use an object of a subclass of the `simscape.multibody.Rotation` class. To specify the `Translation` property, use an object of a subclass of the `simscape.multibody.Translation` class.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`T = simscape.multibody.Transformation` creates a 3-D transformation with default values.

`T = simscape.multibody.Transformation(rot)` creates a 3-D transformation with the specified rotation and default translation.

`T = simscape.multibody.Transformation(trans)` creates a 3-D transformation with the specified translation and default rotation.

`RT = simscape.multibody.Transformation(rot,trans)` creates a 3-D transformation with the specified rotation and translation.

Properties

`Rotation` — Rotation of transformation

`simscape.multibody.ZeroRotation` object (default) | object of subclass of `simscape.multibody.Rotation` class

Rotation of the transformation, specified as an object of a subclass of the `simscape.multibody.Rotation` class.

Example: `simscape.multibody.StandardAxisRotation`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

`Translation` — Translation of transformation
`simscape.multibody.ZeroTranslation` object (default) | object of subclass of `simscape.multibody.Translation` class

Translation of the transformation, specified as an object of a subclass of the `simscape.multibody.Translation` class.

Example: `simscape.multibody.StandardAxisTranslation`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Methods

Public Methods

transformDirection	Apply rotation to 3-D column vectors
transformPoint	Transform 3-D column vectors

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.RigidTransform` | `simscape.multibody.Rotation` | `simscape.multibody.Translation`

transformDirection

Class: `simscape.multibody.Transformation`
Package: `simscape.multibody`

Apply rotation to 3-D column vectors

Syntax

```
xv = transformDirection(T,v)
```

Description

`xv = transformDirection(T,v)` transforms a set of 3-D column vectors that represent directions. The rotational part of the transformation changes the orientations for the directions. The translational part does not affect the 3-D column vectors because directions do not have offset.

`v` argument specifies the vectors of the directions, `T` argument specifies a desired transformation used to change the directions. `xv` argument returns the computed 3-D column vectors, and each column vector represents a transformed direction. The returned vectors, `xv`, have the same format as `v`.

If the `T` argument is the transformation from follower frame to base frame, and the *i*th column of the `v` argument is a direction resolved in follower frame, then the *i*th column of the `xv` argument is the same direction resolved in base frame.

Input Arguments

T — Transformation

`simscape.multibody.Transformation` object

Transformation, specified as a `simscape.multibody.Transformation` object.

v — Directions

3-D column vectors | `simscape.Value` | object

Directions, specified as either 3-D column vectors or a `simscape.Value` object that represents 3-D column vectors with arbitrary units. Each vector represents a direction.

Output Arguments

xv — Transformed directions

3-D column vectors | `simscape.Value` | object

Transformed directions, returned as either 3-D column vectors or a `simscape.Value` object that represents 3-D column vectors with arbitrary units. Each vector represents a direction. This argument and the vectors specified for the `v` argument always have the same format.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.Transformation` | `transformPoint`

transformPoint

Class: `simscape.multibody.Transformation`
Package: `simscape.multibody`

Transform 3-D column vectors

Syntax

```
xp = transformPoint(T,p)
```

Description

`xp = transformPoint(T,p)` transforms a set of 3-D column vectors that represent points. Only the translational part of the provided transformation affects the points specified in the `p` argument because points do not have orientations.

If the `T` argument is a transformation from the follower frame to base frame and the *i*th column of the `p` argument represents the position vector of a point resolved in the follower frame, the *i*th column of the `xp` argument represents the same point resolved in the base frame.

Input Arguments

T — Transformation

`simscape.multibody.Transformation` object

Transformation, specified as a `simscape.multibody.Transformation` object.

p — Coordinates of points

`simscape.Value` object

Coordinates of the points, specified as a `simscape.Value` object that represents one or more 3-D column vectors with units of length.

Output Arguments

xp — Coordinates of transformed points

`simscape.Value` object

Coordinates of the transformed points, returned as a `simscape.Value` object that represents one or more 3-D column vectors with units of length. The returned vectors and the vectors specified for the `p` argument have the same size and units.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Component` | `simscape.multibody.Transformation` | `transformDirection`

simscape.multibody.Translation class

Package: `simscape.multibody`

Abstract base class for 3-D translations

Description

`simscape.multibody.Translation` is the abstract base class for 3-D translations. A translation affects only the position of the follower frame with respect to the base frame. To construct a translation, use an object of a subclass of the `Translation` class, such as `simscape.multibody.StandardAxisTranslation` or `simscape.multibody.CylindricalTranslation`.

Class Attributes

`Abstract`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Methods

Public Methods

cartesianOffset Extract offset from translation

Version History

Introduced in R2022a

See Also

`simscape.multibody.CartesianTranslation` | `simscape.multibody.CylindricalTranslation` | `simscape.multibody.ZeroTranslation` | `simscape.multibody.StandardAxisTranslation`

cartesianOffset

Class: `simscape.multibody.Translation`
Package: `simscape.multibody`

Extract offset from translation

Syntax

```
d = cartesianOffset(T)
```

Description

`d = cartesianOffset(T)` extracts the offset of the specified translation, `T`, and returns the offset in terms of Cartesian coordinates. The offset is resolved in the base frame of the translation.

Input Arguments

T — Translation to extract

object of subclass of `simscape.multibody.Translation` class

Translation to extract, specified as an object of a subclass of the `simscape.multibody.Translation` class.

Output Arguments

d — Offset of translation

`simscape.Value([x y z],"Length unit")` object

Offset of the translation, returned as a `simscape.Value` object that represents a 1-by-3 array with units of length. The units of the offset match the length unit used in the object specified by the `T` input argument. If `T` uses multiple length units, the offset uses the coarsest length unit. If `T` does not have a unit, the offset uses meters.

Attributes

`Access`	`public`

To learn about attributes of methods, see Method Attributes.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Translation`

simscape.multibody.CartesianTranslation class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Translation`

Construct Cartesian translation

Description

Use an object of the `simscape.multibody.CartesianTranslation` class to construct a 3-D translation by using Cartesian coordinates.

The `offset` property specifies the coordinates of the translation in the *x*, *y*, and *z* directions of the base frame.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`T = simscape.multibody.CartesianTranslation` creates a Cartesian translation with default values.

`T = simscape.multibody.CartesianTranslation(offset)` creates a Cartesian translation with the specified offset.

Properties

`offset` — Offset of translation

`simscape.Value([0 0 1],"m")` (default) | `simscape.Value([x y z],"Length unit")` | `simscape.Value([x y z]',"Length unit")`

Offset of the translation, specified as a `simscape.Value` object that represents a 3-by-1 or 1-by-3 array with a unit of length. The array specifies the coordinates of the translation in the *x*, *y*, and *z* directions of the base frame.

Example: `simscape.Value([3 4 5],"cm")`

Attributes:

GetAccess	public
SetAccess	public
NonCopyable	true

Version History

Introduced in R2022a

See Also

simscape.multibody.Translation

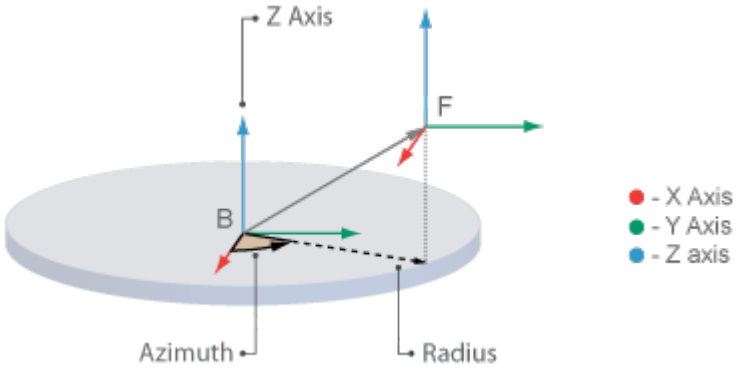
simscape.multibody.CylindricalTranslation class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Translation`

Construct cylindrical-axis translation

Description

Use an object of the `simscape.multibody.CylindricalTranslation` class to construct a 3-D translation by using cylindrical coordinates. The image shows an example.



The `Radius` property specifies the length of the projection of the translation, `BF`, in the *xy*-plane of the base frame. The `Angle` property specifies the azimuth, which is the angle of the radius with respect to the *x*-axis of the base frame. The angle falls in the range of [-π, π). The `ZOffset` property specifies the offset of the translation along the *z*-axis of the base frame.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`T = simscape.multibody.CylindricalTranslation` creates a cylindrical coordinate translation with default values.

`T = simscape.multibody.CylindricalTranslation(Radius,Angle,ZOffset)` creates a cylindrical coordinate translation with the specified radius, azimuth, and *z* offset.

Properties

`Radius` — Radius of translation
`simscape.Value(0,"m")` (default) | `simscape.Value(r,"Length unit")`

Radius of the translation, specified as a `simscape.Value` object that represents a scalar with a unit of length.

Example: `simscape.Value(3,"cm")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

`Angle` — Azimuth of translation
`simscape.Value(0,"deg")` (default) | `simscape.Value(angle,"anglar unit")` class

Azimuth of the translation, specified as a `simscape.Value` object that represents a scalar with a unit of angle.

Example: `simscape.Value(10,"deg")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

`ZOffset` — *z* offset of translation
`simscape.Value(0,"m")` (default) | `simscape.Value(z,"Length unit")`

z offset of the translation, specified as a `simscape.Value` object that represents a scalar with a unit of length.

Example: `simscape.Value(3,"m")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History

Introduced in R2022a

See Also

`simscape.multibody.Translation`

simscape.multibody.StandardAxisTranslation class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Translation`

Construct standard-axis translation

Description

Use an object of the `simscape.multibody.StandardAxisTranslation` class to construct a translation along an axis of the base frame.

The `offset` property specifies the distance of the translation. The `Axis` property specifies the axis of the translation. The axis can be any axis of the base frame.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`T = simscape.multibody.StandardAxisTranslation` creates a standard-axis translation with default values.

`T = simscape.multibody.StandardAxisTranslation(offset,axis)` creates a standard-axis translation with the specified offset and axis.

Properties

`Offset` — Offset of translation

`simscape.Value(0,"m")` (default) | `simscape.Value(d,"Length unit")`

Offset of the translation, specified as a `simscape.Value` object that represents a scalar with a unit of length.

Example: `simscape.Value(3,"cm")`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

`Axis` — Axis of translation
`simscape.multibody.Axis.PosZ` (default) | member of `simscape.multibody.Axis` class

Axis of the translation, specified as a member of the `simscape.multibody.Axis` class. To specify the axis, use the following members:

- `simscape.multibody.Axis.PosX`: *x*-axis of the base frame
- `simscape.multibody.Axis.NegX`: -*x*-axis of the base frame
- `simscape.multibody.Axis.PosY`: *y*-axis of the base frame
- `simscape.multibody.Axis.NegY`: -*y*-axis of the base frame
- `simscape.multibody.Axis.PosZ`: *z*-axis of the base frame
- `simscape.multibody.Axis.NegZ`: -*z*-axis of the base frame

Example: `simscape.multibody.Axis.NegY`

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Version History
Introduced in R2022a

See Also
`simscape.multibody.Translation`

simscape.multibody.ZeroTranslation class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.Translation`

Construct zero translation

Description

Use an object of the `simscape.multibody.ZeroTranslation` class to construct a zero translation. A zero translation has zero offsets and allows the follower and base frames to be coincident and aligned.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`T = simscape.multibody.ZeroTranslation` creates a zero translation.

Version History

Introduced in R2022a

See Also

`simscape.multibody.Translation`

simscape.multibody.VisualProperties class

Package: `simscape.multibody`

Abstract base class for visual properties

Description

`simscape.multibody.VisualProperties` is the abstract base class for visual properties, such as color, transparency, and shininess. To specify the visual properties of the rendered object, use a subclass of the `VisualProperties` class, such as `simscape.multibody.SimpleVisualProperties` or `simscape.multibody.AdvancedVisualProperties`.

Class Attributes

`Abstract`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Version History

Introduced in R2022a

See Also

`simscape.multibody.SimpleVisualProperties` | `simscape.multibody.AdvancedVisualProperties`

simscape.multibody.AdvancedVisualProperties class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.VisualProperties`

Specify advanced visual properties

Description

Use an object of the `simscape.multibody.AdvancedVisualProperties` class to specify advanced visual properties, including shininess and ambient, diffuse, emissive, and specular light on the rendered object. You can specify each of the four color properties by using a 1-by-3 or 3-by-1 vector that specifies the red, green, and blue color components. You can also add an optional fourth element to the RGB vector to specify the opacity of each color.

Each element of the RGB vector of the `AmbientColor`, `DiffuseColor`, and `SpecularColor` properties specifies the reflectivity of the corresponding color component. For example, the first element of the `SpecularColor` property gives the proportion of red light in the specular reflection on the rendered object surface. The `EmissiveColor` property specifies the color of the light emitted by the rendered object itself.

The `Shininess` property specifies the size and sharpness of a specular highlight. A rendered object with a small value for the `Shininess` property has duller surface and produces a big and fuzzy specular highlight. In contrast, an object with a large value for the `Shininess` property has a shiny surface that produces a small and sharp specular highlight.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`avis = simscape.multibody.AdvancedVisualProperties` creates an advanced visual properties object with default values.

Properties

`AmbientColor` — Color of ambient light

[0.15 0.15 0.15] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the ambient light, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

This property specifies a general level of illumination that does not come directly from a light source. To change the shadow color of the rendered object, use the `AmbientColor` property.

Example: [0.5 0.5 0.5]

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Data Types: `double`

`DiffuseColor` — Color of light due to diffuse reflection

[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to diffuse reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

The diffuse color reflects the main color of the rendered object and provides shading that gives the rendered object a three-dimensional appearance.

Example: [0 1 0 1]

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Data Types: `double`

`EmissiveColor` — Self-illumination color

[0 0 0] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color due to self illumination, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

Example: [0 0 1 1]

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Data Types: `double`

`Shininess` — Shininess of rendered object

75 (default) | scalar in the range of 0 to 128

Shininess of the rendered object, specified as a scalar in the range of 0 to 128. This property affects the sharpness of the specular reflections of the rendered object. An object with high shininess has a mirror-like appearance, and an object with low shininess has a more low-gloss or satin appearance.

Example: 90

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Data Types: `double`

`SpecularColor` — Color of light due to specular reflection

[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1 | 4-by-1 or 1-by-4 vector with values in the range of 0 to 1

Color of the light due to specular reflection, specified as an [R,G,B] or [R,G,B,A] vector with values in the range of 0 to 1. The vector can be a row or column vector. The optional fourth element specifies the color opacity. Omitting the opacity element is equivalent to specifying a value of 1.

To change the color of the specular highlight, which is the bright spot on the rendered object due to the reflection of the light from the light source, use the `SpecularColor` property.

Example: [1 0 0 1]

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Data Types: `double`

Version History

Introduced in R2022a

See Also

`simscape.multibody.SimpleVisualProperties` | `simscape.multibody.VisualProperties`

simscape.multibody.SimpleVisualProperties class

Package: `simscape.multibody`
Superclasses: `simscape.multibody.VisualProperties`

Specify simple visual properties

Description

Use an object of the `simscape.multibody.SimpleVisualProperties` class to specify only the color and opacity of a rendered object. To specify additional visual properties, use the `simscape.multibody.AdvancedVisualProperties` class.

Class Attributes

`Sealed`	`true`
`ConstructOnLoad`	`true`
`RestrictsSubclassing`	`true`

For information on class attributes, see “Class Attributes”.

Creation

Description

`svis = simscape.multibody.SimpleVisualProperties` creates a simple visual properties object with default values.

`svis = simscape.multibody.SimpleVisualProperties(color)` creates a simple visual properties object with the specified color.

`svis = simscape.multibody.SimpleVisualProperties(color,opacity)` creates a simple visual properties object with the specified color and opacity.

Properties

`Color` — RGB color of rendered object

[0.5 0.5 0.5] (default) | 3-by-1 or 1-by-3 vector with values in the range of 0 to 1

RGB color of the rendered object, specified as a 3-by-1 or 1-by-3 vector with values in the range of 0 to 1.

Example: [0.50 1.00 0.83]

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Data Types: `double`

`Opacity` — Opacity of rendered object
1 (default) | scalar in the range of 0 to 1

Opacity of the rendered object, specified as a scalar in the range of 0 to 1. A scalar of 0 corresponds to completely transparent, and a scalar of 1 corresponds to completely opaque.

Example: 0.5

Attributes:

`GetAccess`	`public`
`SetAccess`	`public`
`NonCopyable`	`true`

Data Types: `double`

Version History

Introduced in R2022a

See Also

Topics

`simscape.multibody.AdvancedVisualProperties`
`simscape.multibody.VisualProperties`

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)` creates a set of frame variables for a pair of frames. One of the frames serves as the follower and the other serves as the base. These frame variables share a group name, which must be a valid MATLAB variable name, and correspond to a position or velocity relationship between the base and follower frames.

Use the `type` argument to specify the relationship for frame variables. There are four types: `Translation`, `Rotation`, `LinearVelocity`, and `AngularVelocity`. The `base` and `follower` arguments must be full paths to frame ports in the model.

The function outputs a table that includes all frame variables. Each row of the table includes ID, path from the root to the base and follower frames, and unit for its numerical value of a frame variable. The IDs of the frame variables have the form `groupName.type.primitiveComponent`.

`addFrameVariables(ks,groupName,type,base,follower,Name,Value)` creates a set of frame variables for a pair of frames that represents a position-based or velocity-based relationship. Use the “Name-Value Arguments” on page 4-211 to specify the unit of the frame variables.

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

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. If adding variables to an existing group, use the name of that group.

Example: 'wrist_position'

Data Types: `char` | `string`

type — Type of relationship to represent between frames

'Translation' | 'Rotation' | 'LinearVelocity' | 'AngularVelocity'

Type of relationship that the new frame variables are to represent, specified as one of the following options:

- The `Translation` variables represent to the *x*-, *y*-, *z*-components of the linear position of the follower frame with respect to the base frame.
- The `Rotation` variables represent the three angles of the *xyz* intrinsic rotation sequence specifying the angular position of the follower frame with respect to the base frame.
- The `LinearVelocity` variables represent to the *x*-, *y*-, *z*-components of the linear velocity of the follower frame with respect to the base frame.
- The `AngularVelocity` variables represent to the *x*-, *y*-, *z*-components of the angular velocity of the follower frame with respect to the base frame.

Note The `Translation`, `Rotation`, and `LinearVelocity` variables are resolved in the base frame coordinates, but the `AngularVelocity` variables are resolved in the follower frame coordinates.

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.

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.

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','Link','Rotation','sm_four_bar/'World Frame/W','sm_four_bar/Connector Link/Left End Cap/R','angleUnit','rad');

`angleUnit` — Unit to replace default angular unit
string scalar | character array

Unit to replace the default angular unit, specified as the comma-separated pair consisting of '`angleUnit`' and a string scalar or character array. Use this argument when the `type` is '`Rotation`'.

Example: `rad`

Data Types: `char` | `string`

`lengthUnit` — Unit to replace default length unit
string scalar | character array

Unit to replace the default length unit, specified as the comma-separated pair consisting of '`lengthUnit`' and a string scalar or character array. Use this argument when the `type` is '`Translation`'.

Example: `in`

Data Types: `char` | `string`

`LinearVelocityUnit` — Unit to replace default linear velocity unit
string scalar | character array

Unit to replace the default linear velocity unit, specified as the comma-separated pair consisting of '`LinearVelocityUnit`' and a string scalar or character array. Use this argument when the `type` is '`LinearVelocity`'.

Example: `in/s`

Data Types: `char` | `string`

`AngularVelocityUnit` — Unit to replace default angular velocity unit
string scalar | character array

Unit to replace the default angular velocity unit, specified as the comma-separated pair consisting of '`AngularVelocityUnit`' and a string scalar or character array. Use this argument when the `type` is '`AngularVelocity`'.

Example: `rad/s`

Data Types: `char` | `string`

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `jointPositionVariables` | `jointVelocityVariables` | `frameVariables` | `clearFrameVariables` | `removeFrameVariables`

addInitialGuessVariables

Package: `simscape.multibody`

Assign kinematic variables from KinematicsSolver object as guesses

Syntax

```
addInitialGuessVariables(ks,ids)
```

Description

`addInitialGuessVariables(ks,ids)` assigns the kinematic variables, `ids`, of the `KinematicsSolver` object, `ks`, as initial guesses.

The output of the `addInitialGuessVariables` object function is a table that lists the variables that have been assigned as guesses. Each row lists the details of a variable, such as the ID, joint type, block path, and unit.

You can assign only joint variables as initial guesses. When assigning joint variables, note that:

- Since R2022a, you cannot specify the frame variables of the `KinematicsSolver` as initial guesses.
- You must assign the four joint position variables of a spherical primitive as initial guesses simultaneously.
- You must assign the three joint velocity variables of a spherical primitive as initial guesses simultaneously.
- You cannot assign the joint position variable for the azimuth angle of a constant velocity primitive as a guess unless you also assign the corresponding bend angle.
- You cannot assign the joint velocity variable for the azimuth velocity of a constant velocity primitive as a guess unless you also assign the corresponding bend angle velocity.

The solver uses the guess variables as initial conditions for the search to find a solution that satisfies the targets. When multiple solutions exist, the initial guess variables can help bias the solver to converge on the desired solution. Guess variables are optional but important solver guides in some kinematics problems.

Input Arguments

`ks` — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

`ids` — Identifiers of kinematic variables

cell array of characters | string vector

Identifiers of the kinematic variables, specified as either a cell array of characters or string vector. The cell array of character and string vector can be 1-by-*N* or *N*-by-1, where *N* is a positive integer. Use the `jointPositionVariables` or `jointVelocityVariables` object function to show the IDs for joint variables. Use the `frameVariables` object function to show the IDs for frame variables.

Example: "j1.Rz.q", ["j1.Rz.q", "j2.Rz.q"], {'j1.Rz.q'}, or {'j1.Rz.q'; 'j2.Rz.q'};

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `initialGuessVariables` | `clearInitialGuessVariables` | `removeInitialGuessVariables`

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.)

Input Arguments

`ks` — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

`ids` — Identifiers of kinematic variables

cell array of characters | string vector

Identifiers of the kinematic variables, specified as either a cell array of characters or string vector. The cell array of character and string vector can be 1-by-*N* or *N*-by-1, where *N* is a positive integer. Use the `jointPositionVariables` or `jointVelocityVariables` object function to show the IDs for joint variables. Use the `frameVariables` object function to show the IDs for frame variables.

Example: "j1.Rz.q", ["j1.Rz.q", "j2.Rz.q"], {'j1.Rz.q'}, or {'j1.Rz.q'; 'j2.Rz.q'};

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `outputVariables` | `clearOutputVariables` | `removeOutputVariables`

addTargetVariables

Package: `simscape.multibody`

Assign kinematic variables from KinematicsSolver object as targets

Syntax

```
addTargetVariables(ks,ids)
```

Description

`addTargetVariables(ks,ids)` assigns the kinematic variables, `ids`, of the `KinematicsSolver` object, `ks`, as targets.

The output of the `addTargetVariables` object function is a table that lists the targeted variables in rows. Each row shows the details of a variable. For a joint variable, the table shows the ID, joint type, block path, and unit. For a frame variable, the table shows the ID, base and follower frames, and unit.

You can assign both frame and joint variables as target variables and add them in any order. When adding target variables, note that:

- You cannot specify `LinearVelocity` frame variables as targets.
- You cannot specify `AngularVelocity` frame variables as targets.
- When using a `Rotation` type frame variable as target, you must specify the *x* and *y* components of the frame variable simultaneously.
- You must target the four joint position variables of a spherical primitive simultaneously.
- You must target three joint velocity variables of a spherical primitive simultaneously.
- You cannot target the joint position variable for the azimuth angle of a constant velocity primitive unless you also target the corresponding bend angle.
- You cannot target the joint velocity variable for the azimuth velocity of a constant velocity primitive unless you also target the corresponding bend angle velocity.

During a search, the target variables serve as constraints for the system, and the solver searches for a solution that is compatible with the targeted joint and frame variables. Do not overconstrain the system with target variables. A system is overconstrained if a kinematic loop in the system has a target for every joint. One way of avoiding overconstraining is to reassign one of the joint variables from a target to an initial guess by using the `addInitialGuessVariables` object function.

A variable can serve as both target and output. However, a variable cannot serve as both target and guess.

Input Arguments

`ks` — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

`ids` — Identifiers of kinematic variables
cell array of characters | string vector

Identifiers of the kinematic variables, specified as either a cell array of characters or string vector. The cell array of character and string vector can be 1-by-*N* or *N*-by-1, where *N* is a positive integer. Use the `jointPositionVariables` or `jointVelocityVariables` object function to show the IDs for joint variables. Use the `frameVariables` object function to show the IDs for frame variables.

Example: "j1.Rz.q", ["j1.Rz.q", "j2.Rz.q"], {'j1.Rz.q'}, or {'j1.Rz.q'; 'j2.Rz.q'};

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `targetVariables` | `clearTargetVariables` | `removeTargetVariables`

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.

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `frameVariables` | `addFrameVariables` | `removeFrameVariables`

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.

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `initialGuessVariables` | `addInitialGuessVariables` | `removeInitialGuessVariables`

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.

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `outputVariables` | `addOutputVariables` | `removeOutputVariables`

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.

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `targetVariables` | `addTargetVariables` | `removeTargetVariables`

closeViewer

Package: `simscape.multibody`

Close the Kinematics Solver Viewer window

Syntax

```
closeViewer(ks)
```

Description

`closeViewer(ks)` closes the Kinematics Solver Viewer window.

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2020a

See Also

`KinematicsSolver` | `solve` | `viewSolution`

frameVariables

Package: `simscape.multibody`

List kinematic variables associated with frame pairs

Syntax

```
frameVariables(ks)
```

Description

`frameVariables(ks)` outputs a table that lists the frame variables currently defined in the `KinematicsSolver` object `ks`. Each row of the table shows the ID, base and follower frames, and unit of a frame variable.

Use this function to identify the IDs of frame variables that you want to assign as targets and outputs. Note that you cannot specify the `LinearVelocity` and `AngularVelocity` types of frame variables as targets, and you must specify the *x* and *y* components of the variable simultaneously when using a `Rotation` type frame variable as a target.

By default, a newly created `KinematicsSolver` object does not have any frame variables. To create frame variables, use the `addFrameVariables` object function. To remove frame variables, use the `removeFrameVariables` object function to drop frame variables that are no longer needed for the analysis, or use the `clearFrameVariables` object function to drop all frame variables in one call.

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `addFrameVariables` | `clearFrameVariables` | `removeFrameVariables`

generateCode

Package: `simscape.multibody`

Generate C code to run kinematic analysis on KinematicsSolver object

Syntax

```
generateCode(ks)
```

Description

`generateCode(ks)` creates a standalone MATLAB function that is equivalent to the `solve` function, but supports code generation. It also creates a directory called `ModelName`_codegen_kinematics that contains all the source files for the code generation in the current directory, where `ModelName` is the output string of `ks.ModelName`.

The created MATLAB function is called `ModelName`_solveKinematics and has the same signature as the `solve` function:

```
[outputs,statusFlag,targetFlags,targets] = ModelName_solveKinematics(targets, initialGuesses)
```

Once generated, the function is completely independent from the original object and will not reflect any changes to the object. You can generate MEX functions, static libraries (LIB), and dynamics libraries (DLL) from MATLAB code that contains the `ModelName`_solveKinematics function by using the `codegen` function, which requires a MATLAB Coder license.

Note `ModelName`_solveKinematics is not meant to be invoked from MATLAB and an error occurs when calling it directly from the MATLAB command line or a MATLAB file. However, you can call this function directly from a MATLAB Function block in a Simulink model.

Examples

Generate a MEX Function to Solve an Inverse Kinematics Problem for a Double Pendulum Model

1 Set up the inverse kinematics problem for the double pendulum model.

```
mdl = 'sm_double_pendulum';
open_system(mdl);
ks = simscape.multibody.KinematicsSolver(mdl);
addFrameVariables(ks, 'LowerLinkPeg', 'translation',...
                  'sm_double_pendulum/World Frame/W',...
                  'sm_double_pendulum/Lower Link/Right Peg/R');
targetIds = ["LowerLinkPeg.Translation.x";...
             "LowerLinkPeg.Translation.z"];
addTargetVariables(ks, targetIds);
initialGuessIds = "j2.Rz.q";
addInitialGuessVariables(ks, initialGuessIds);
```

```
outputIds = ["j2.Rz.q"; "j1.Rz.q"];
addOutputVariables(ks, outputIds);
```

2 Create a standalone solve function and a directory with source files.

```
generateCode(ks);
```

3 Create a MEX function for the MATLAB function.

```
codegen -config:mex sm_double_pendulum_solveKinematics
```

4 Solve the inverse kinematics problem using the MEX function.

```
[outputVals, status, targetSuccess, actTargetVals] =...
 sm_double_pendulum_solveKinematics_mex([0.3, 0], 120)
```

5 Outputs

```
outputVals =
  124.0477
  -57.1217

status =
  1

targetSuccess =
  2×1 logical array
  1
  1

actTargetVals =
  0.3000
  0
```

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `solve`

initialGuessVariables

Package: `simscape.multibody`

List all kinematic variables assigned as initial guesses

Syntax

```
initialGuessVariables(ks)
```

Description

`initialGuessVariables(ks)` outputs a table that lists the guess variables in the `KinematicsSolver` object `ks`. Each row lists the ID, joint type, block path, and unit of a variable.

Only joint variables can serve as guesses. To assign a variable as guess, use the `addInitialGuessVariables` object function. To remove guesses, use the `removeInitialGuessVariables` object function to drop guess variables that are no longer needed for the analysis, or use the `clearInitialGuessVariables` object function to drop all guess variables in one call.

When multiple solutions exist, initial guess variables can help bias the solver to converge on the desired solution. Guess variables are optional but important solver guides in some kinematics problems.

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `addInitialGuessVariables` | `clearInitialGuessVariables` | `removeInitialGuessVariables`

jointPositionVariables

Package: `simscape.multibody`

List all kinematic variables associated with joint positions

Syntax

```
jointPositionVariables(ks)
```

Description

`jointPositionVariables(ks)` outputs a table showing all kinematic variables corresponding to joint positions. Each row of the table shows ID, type of joint, path from the root, and unit for the numerical value of a joint position variable. Use this function to identify the IDs of position-based variables that you want to assign as targets, initial guesses, and outputs using the `addTargetVariables`, `addInitialGuessVariables`, and `addOutputVariables` object functions, respectively.

The IDs of position-based variables have the form: `jointName.primitiveType.primitiveComponent`. The `jointName` string is based on the index of the joint block associated with the variable. The `primitiveType` string corresponds to the primitive associated with the variable (`Px`, `Py`, or `Pz` for prismatic; `Rx`, `Ry`, or `Rz` for revolute; `S` for spherical; `CV` for constant velocity; `LSz` for lead screw). The `primitiveComponent` string identifies a particular scalar value associated with the primitive (`p` for position of a prismatic primitive; `q` for angle of a revolute primitive; `q`, `ax_x`, `ax_y` and `ax_z` for the for the angle and three axis components of a spherical primitive; `q_a` and `q_b` for the azimuth and bend angles of a `CV` primitive; `q` for the rotation angle of a lead screw primitive).

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2019a

See Also

Blocks

6-DOF Joint | Bearing Joint | Bushing Joint | Cartesian Joint | Constant Velocity Joint | Cylindrical Joint | Gimbal Joint | Lead Screw Joint | Pin Slot Joint | Planar Joint | Prismatic Joint | Rectangular Joint | Revolute Joint | Spherical Joint | Universal Joint | Telescoping Joint

Functions
`KinematicsSolver`

jointVelocityVariables

List all kinematic variables associated with joint velocities

Syntax

```
jointVelocityVariables(ks)
```

Description

`jointVelocityVariables(ks)` outputs a table showing all kinematic variables corresponding to joint velocities. Each row of the table shows ID, type of joint, path from the root, and unit for the numerical value of a joint velocity variable. Use this function to identify the IDs of velocity-based variables that you want to assign as targets, initial guesses, and outputs using the `addTargetVariables`, `addInitialGuessVariables`, and `addOutputVariables` object functions, respectively.

The IDs of velocity-based variables have the form: `jointName.primitiveType.primitiveComponent`. The `jointName` string is based on the index of the joint block associated with the variable. The `primitiveType` string corresponds to the primitive associated with the variable (`Px`, `Py`, or `Pz` for prismatic; `Rx`, `Ry`, or `Rz` for revolute; `S` for spherical; `CV` for constant velocity; `LSz` for lead screw). The `primitiveComponent` string identifies a particular scalar value associated with the primitive (`v` for linear velocity of a prismatic primitive; `w` for angular velocity of a revolute primitive; `w_x`, `w_y` and `w_z` for the angular velocity components of a spherical primitive; `w_a` and `w_b` for the azimuth and bend velocities of a `CV` primitive; `w` for angular velocity of a lead screw primitive).

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2020a

See Also

Blocks

6-DOF Joint | Bearing Joint | Bushing Joint | Cartesian Joint | Constant Velocity Joint | Cylindrical Joint | Gimbal Joint | Lead Screw Joint | Pin Slot Joint | Planar Joint | Prismatic Joint | Rectangular Joint | Revolute Joint | Spherical Joint | Universal Joint | Telescoping Joint

Functions
`KinematicsSolver` | `jointPositionVariables`

KinematicsSolver

Solve kinematics problems for a multibody model

Description

`KinematicsSolver` objects allow users to formulate and numerically solve kinematics problems for their Simscape Multibody models. You can use the object to solve standard forward and inverse kinematics problems, as well as more general problems with closed-loop kinematic systems and multiple targets.

A kinematics problem is formulated using kinematic variables. These variables have scalar values that specify the relationships between frames in the corresponding Simscape Multibody model. There are two types of kinematic variables: joint and frame. Joint variables correspond to joint position and velocity states and are created automatically when the object is constructed. You can view the joint variables using the `jointPositionVariables` and `jointVelocityVariables` object functions. Frame variables correspond to position and velocity relationships between arbitrary frames in the model and must be defined using the `addFrameVariables` object function. Once defined, they can be viewed using the `frameVariables` object function.

To formulate a kinematics problem, you must assign roles for the relevant kinematic variables. There are three roles: targets, initial guesses, and outputs. Variables are assigned to these roles using the `addTargetVariables` , `addInitialGuessVariables`, and `addOutputVariables` object functions. To solve the problem with the assigned variables, use the `solve` object function. Starting from an initial state, the solver attempts to find a final state of the system consistent with the values of the target variables. The initial state is synthesized using the values of the initial guess variables. The initial states that are not specified by initial guess variables are initialized to zero. The values of the output variables are derived from the final state returned by the solver. If the solver is unable to find a final state that satisfies all the targets, it tries to at least return a state that is kinematically feasible.

Creation

Syntax

```
ks = simscape.multibody.KinematicsSolver(modelName)
ks = simscape.multibody.KinematicsSolver( ___ ,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, and you need to load the model into memory before creating its `KinematicsSolver` object. If blocks of the model have MATLAB variables, you

need to numerically define those variables in the model workspace or MATLAB workspace. The `KinematicsSolver` object ignores any contacts and several parameters of joint blocks, like **State Targets**, **Limits**, **Actuation**, and **Mode Configuration**. For example, during an analysis, two bodies can penetrate each other even though there is a Spatial Contact Force block that connects them. 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.

`ks = simscape.multibody.KinematicsSolver( ___ ,Name,Value)` creates a `KinematicsSolver` object with additional options specified by one or more `Name,Value` pair arguments.

Properties

`MaxIterations` — Maximum number of solver iterations

100 (default) | positive integer

Maximum number of solver iterations, specified as a positive integer. You can specify this property after creating a `KinematicsSolver` object.

Example: 50

Data Types: `double` | `int`

`ModelName` — Model Name

string scalar | character vector

Simscape Multibody model name from which the object derives. This property is read-only.

Example: 'sm_four_bar'

Data Types: `char` | `string`

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1,...,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Before R2021a, use commas to separate each name and value, and enclose `Name` *in quotes.*

Example: `'DefaultAngleUnit','rad'`

`'DefaultAngleUnit'` — Default angle unit of new kinematic variables

`"deg"` (default) | string scalar | character vector

Default angle unit of new kinematic variables, specified as the comma-separated pair consisting of `'DefaultAngleUnit'` and a character vector or string scalar. When you change the `'DefaultAngleUnit'` property, the change applies only to new kinematic variables. Existing variables are not changed.

Example: 'DefaultAngleUnit','rad'

Data Types: `char` | `string`

`'DefaultAngularVelocityUnit'` — Default angular velocity unit of new kinematic variables
`"deg/s"` (default) | string scalar | character vector

Default angle unit of new kinematic variables, specified as the comma-separated pair consisting of `'DefaultAngularVelocityUnit'` and a character vector or string scalar. When you change the `'DefaultAngularVelocityUnit'` property, the change applies only to new kinematic variables. Existing variables are not changed.

Example: 'DefaultAngularVelocityUnit','rad/s'

Data Types: `char` | `string`

`'DefaultLengthUnit'` — Default length unit of new kinematic variables
`"m"` (default) | string scalar | character vector

Default angle unit of new kinematic variables, specified as the comma-separated pair consisting of `'DefaultLengthUnit'` and a character vector or string scalar. When you change the `'DefaultLengthUnit'` property, the change applies only to new kinematic variables. Existing variables are not changed.

Example: 'DefaultLengthUnit','in'

Data Types: `char` | `string`

`'DefaultLinearVelocityUnit'` — Default linear velocity unit of new kinematic variables
`"m/s"` (default) | string scalar | character vector

Default angle unit of new kinematic variables, specified as the comma-separated pair consisting of `'DefaultLinearVelocityUnit'` and a character vector or string scalar. When you change the `'DefaultLinearVelocityUnit'` property, the change applies only to new kinematic variables. Existing variables are not changed.

Example: 'DefaultLinearVelocityUnit','in/s'

Data Types: `char` | `string`

Object Functions

Listing Variables

frameVariables	List kinematic variables associated with frame pairs
initialGuessVariables	List all kinematic variables assigned as initial guesses
jointVelocityVariables	List all kinematic variables associated with joint velocities
jointPositionVariables	List all kinematic variables associated with joint positions
outputVariables	List all kinematic variables assigned as outputs
targetVariables	List kinematic variables assigned as targets

Adding and Configuring Variables

addFrameVariables	Create kinematic variables from select frame pair in KinematicsSolver object
addInitialGuessVariables	Assign kinematic variables from KinematicsSolver object as guesses
addOutputVariables	Assign kinematic variables from the KinematicsSolver object as outputs

addTargetVariables	Assign kinematic variables from KinematicsSolver object as targets
setVariableUnit	Change physical unit of kinematic variable

Removing 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

Clearing 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

Solving

solve	Run kinematic analysis for KinematicsSolver object

Code Generation

generateCode	Generate C code to run kinematic analysis on KinematicsSolver object

Viewing Solution

viewSolution	Open Kinematics Solver Viewer window to visualize KinematicsSolver solution
closeViewer	Close the Kinematics Solver Viewer window

Examples

Run Forward Kinematics on Humanoid Robot Arm

This example shows how to compute forward kinematics for the `sm_import_humanoid_urdf` model. Specifically, it computes the position of the robot wrist for specified angles of the shoulder and elbow joints.

1. Load the humanoid robot model into memory and create a `KinematicsSolver` object for the model. The object contains a kinematic representation of the model and a list of all the joint variables that it contains.

   ```
   mdl = 'sm_import_humanoid_urdf';
   load_system(mdl);
   fk = simscape.multibody.KinematicsSolver(mdl);
   ```

2. Add to the object, `fk`, a group of frame variables for the left wrist. Specify the **B** frame of the `left_wrist` joint as follower and the world frame as base. Name the frame variable group `Wrist`. The object now has six frame variables—three for the *x*, *y*, and *z* translation components and three for the *x*, *y*, and *z* rotation components.

   ```
   base = 'sm_import_humanoid_urdf/World/W';
   follower = 'sm_import_humanoid_urdf/left_wrist/B';
   addFrameVariables(fk,'Wrist','translation',base,follower);
   addFrameVariables(fk,'Wrist','rotation',base,follower);
   ```

Note The paths in `base` and `follower` are the full block paths from the root of the model to the selected port of a desired block. This example selects the **W** port of the World Frame block as the base and the **B** port of the `left_wrist` joint block as the follower.

3. Use `jointPositionVariables(fk)` to list all joint variables. Assign as targets the joint variables for the elbow (j2.Rz.q), shoulder frontal (j6.Rz.q), and shoulder sagittal (j7.Rz.q).

```
jointPositionVariables(fk)
targetIDs = ["j2.Rz.q";"j6.Rz.q";"j7.Rz.q"];
addTargetVariables(fk,targetIDs);
```

4. Use the `frameVariables(fk)` to list all frame variables and assign them in the `Wrist` group as outputs.

```
frameVariables(fk)
outputIDs = ["Wrist.Translation.x";"Wrist.Translation.y";...
"Wrist.Translation.z";"Wrist.Rotation.x";"Wrist.Rotation.y";"Wrist.Rotation.z"];
addOutputVariables(fk,outputIDs);
```

5. Solve the forward kinematics problem given the elbow, shoulder frontal, and shoulder sagittal joint angles of `30`, `45`, and `45` degrees.

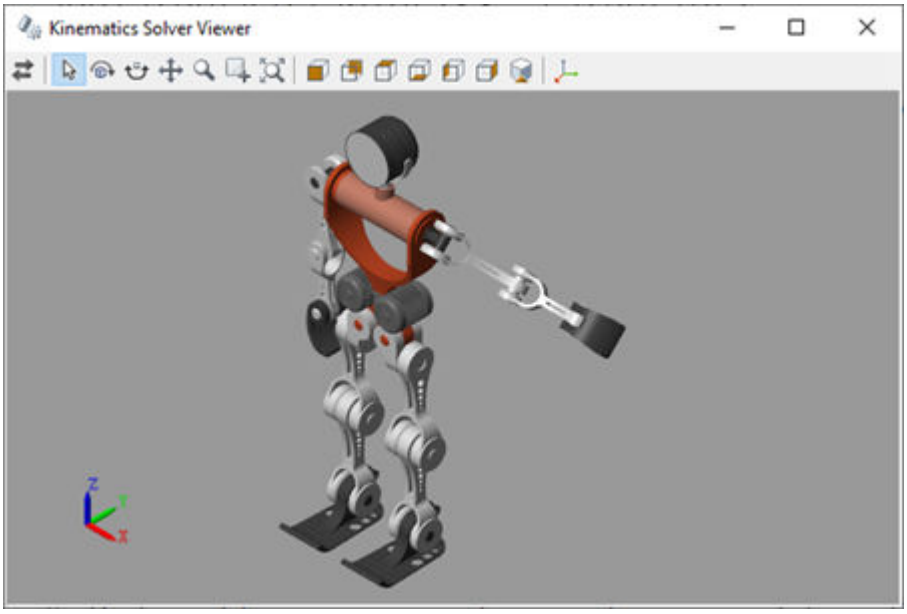
```
targets = [30,45,45];
[outputVec,statusFlag] = solve(fk,targets)
```

```
outputVec =

    0.2196
    0.0584
   -0.0983
  135.0000
    0.0027
  -15.0000

statusFlag =

     1
```

The `solve` function returns the values of the output variables. The values are sorted in the same order as the output variables. The units are the defaults of `m` for translation components, and `deg` for rotation components. The `statusFlag` shows that all model constraints and target variables are satisfied.

6. View the Solution.

```
viewSolution(fk);
```



7 Close the viewer.

```
closeViewer(fk);
```

Run Inverse Kinematics on Humanoid Robot Arm

This example shows how to compute inverse kinematics for the `sm_import_humanoid_urdf` model. Specifically, it computes the angles of the elbow and shoulder joints corresponding to a desired wrist position. Since this problem has multiple solutions, initial guesses for the shoulder joint angles are used to guide the solver towards a desirable solution.

1 Load the humanoid robot model into memory and create a `KinematicsSolver` object for the model. The object contains a kinematic representation of the model and a list of all the joint variables that it contains.

```
mdl = 'sm_import_humanoid_urdf';
load_system(mdl);
ik = simscape.multibody.KinematicsSolver(mdl);
```

2 Add to the object, `ik`, a group of frame variables for the right wrist. Specify the **B** frame of the `right_wrist` joint as follower and the world frame as base. Name the frame variable group `Wrist`. The object now has three frame variables for *x*, *y*, and *z* translation components.

```
base = 'sm_import_humanoid_urdf/World/W';
follower = 'sm_import_humanoid_urdf/right_wrist/B';
addFrameVariables(ik,'Wrist','translation',base,follower);
```

Note The paths in `base` and `follower` are the full block paths from the root of the model to the selected port of a desired block. This example selects the **W** port of the World Frame block as the base and the **B** port of the `right_wrist` joint block as the follower.

3 Use the `frameVariables(ik)` to list all frame variables and assign them in the `Wrist` group as targets.

```
frameVariables(ik)
targetIDs = ["Wrist.Translation.x";"Wrist.Translation.y";"Wrist.Translation.z"];
addTargetVariables(ik,targetIDs);
```

Note Not all frame variables are needed in an analysis. You can use the `frameVariables(ik)` to list all frame variables and then select desired variables for your analysis.

4. Use `jointPositionVariables(ik)` to list all joint variables and assign as outputs the joint variables for the elbow (j10.Rz.q), shoulder frontal (j14.Rz.q), and shoulder sagittal (j15.Rz.q).

```
jointPositionVariables(ik)
outputIDs = ["j10.Rz.q";"j14.Rz.q";"j15.Rz.q"];
addOutputVariables(ik,outputIDs);
```

5. Compute the joint angles for the elbow and shoulder corresponding to a wrist position of `[-0.16,-0.12,0] m`.

```
targets = [-0.16,-0.12,0];
[outputVec,statusFlag] = solve(ik,targets)
```

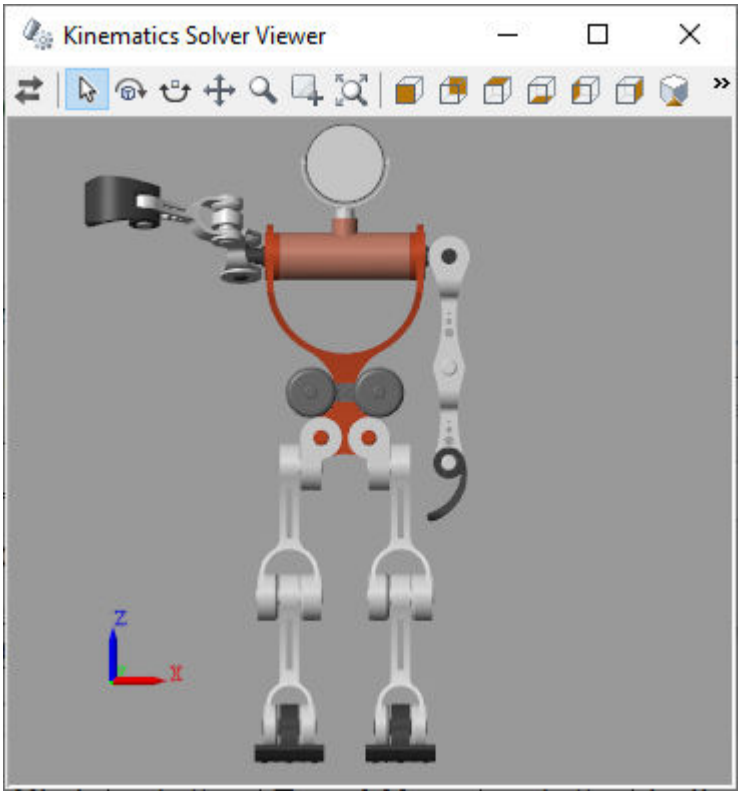
The `solve` function returns the values of the output variables—the rotation angles of the elbow, shoulder frontal, and shoulder sagittal, each in the default units of `deg`. The `statusFlag` shows that all model constraints and target variables are satisfied.

```
outputVec =

  -52.8384
  -71.6077
  172.9586


statusFlag =

     1
```

6. Visualize the solution in the Kinematics Solver Viewer and determine if it is reasonable.

```
viewSolution(ik);
```

Click **Front View** button on the toolstrip to view the result.



The right wrist is in the right place, but the right arm has an unnatural pose. Note that this solution is one of the possible solutions for this problem. You can specify the joints of the shoulder as guess variables to have a better solution.

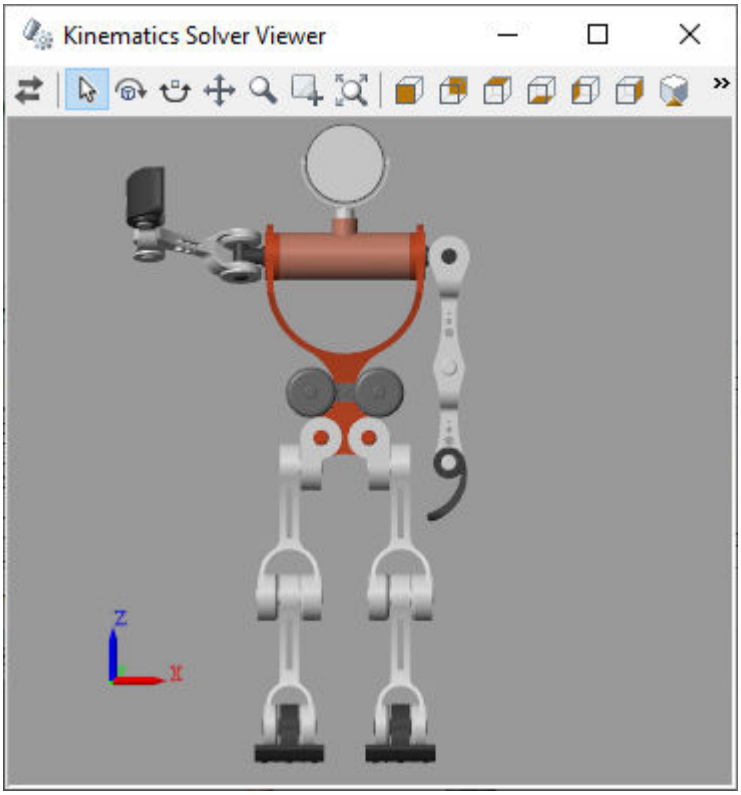
7 Set the shoulder frontal and shoulder sagittal joint variables as guess variables and run the analysis once again for rotations of `[90,90] deg`.

```
guessesIDs=["j14.Rz.q","j15.Rz.q"];
guesses = [90,90];
addInitialGuessVariables(ik,guessesIDs);
[outputVec,statusFlag] = solve(ik,targets,guesses)
```

The `solve` function returns a new solution for the joint angles.

```
outputVec =

  -52.8384
  108.3891
   55.5025

statusFlag =

     1
```

8 Click the **Update Visualization** button to update the Kinematics Solver Viewer to visualize the results.



9 Close the viewer.

```
closeViewer(ik);
```

Version History

Introduced in R2019a

See Also

Functions

`frameVariables` | `initialGuessVariables` | `jointPositionVariables` | `jointVelocityVariables` | `outputVariables` | `targetVariables` | `addFrameVariables` | `addInitialGuessVariables` | `addOutputVariables` | `addTargetVariables` | `removeFrameVariables` | `removeInitialGuessVariables` | `removeOutputVariables` | `removeTargetVariables` | `clearFrameVariables` | `clearInitialGuessVariables` | `clearOutputVariables` | `clearTargetVariables` | `generateCode` | `setVariableUnit` | `solve` | `viewSolution` | `closeViewer`

Topics

“Pick and Place Robot Using Forward and Inverse Kinematics”
“Perform Forward and Inverse Kinematics on a Five-Bar Robot”

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.

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `addOutputVariables` | `clearOutputVariables` | `removeOutputVariables`

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.

Input Arguments

`ks` — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

`ids` — Identifiers of kinematic variables

cell array of characters | string vector

Identifiers of the kinematic variables, specified as either a cell array of characters or string vector. The cell array of character and string vector can be 1-by-*N* or *N*-by-1, where *N* is a positive integer. Use the `jointPositionVariables` or `jointVelocityVariables` object function to show the IDs for joint variables. Use the `frameVariables` object function to show the IDs for frame variables.

Example: "j1.Rz.q", ["j1.Rz.q", "j2.Rz.q"], {'j1.Rz.q'}, or {'j1.Rz.q'; 'j2.Rz.q'};

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `frameVariables` | `addFrameVariables` | `clearFrameVariables`

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.)

Input Arguments

`ks` — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

`ids` — Identifiers of kinematic variables

cell array of characters | string vector

Identifiers of the kinematic variables, specified as either a cell array of characters or string vector. The cell array of character and string vector can be 1-by-*N* or *N*-by-1, where *N* is a positive integer. Use the `jointPositionVariables` or `jointVelocityVariables` object function to show the IDs for joint variables. Use the `frameVariables` object function to show the IDs for frame variables.

Example: "j1.Rz.q", ["j1.Rz.q", "j2.Rz.q"], {'j1.Rz.q'}, or {'j1.Rz.q'; 'j2.Rz.q'};

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `initialGuessVariables` | `addInitialGuessVariables` | `clearInitialGuessVariables`

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.)

Input Arguments

`ks` — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

`ids` — Identifiers of kinematic variables

cell array of characters | string vector

Identifiers of the kinematic variables, specified as either a cell array of characters or string vector. The cell array of character and string vector can be 1-by-*N* or *N*-by-1, where *N* is a positive integer. Use the `jointPositionVariables` or `jointVelocityVariables` object function to show the IDs for joint variables. Use the `frameVariables` object function to show the IDs for frame variables.

Example: "j1.Rz.q", ["j1.Rz.q", "j2.Rz.q"], {'j1.Rz.q'}, or {'j1.Rz.q'; 'j2.Rz.q'};

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `outputVariables` | `addOutputVariables` | `clearOutputVariables`

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.)

Input Arguments

`ks` — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

`ids` — Identifiers of kinematic variables

cell array of characters | string vector

Identifiers of the kinematic variables, specified as either a cell array of characters or string vector. The cell array of character and string vector can be 1-by-*N* or *N*-by-1, where *N* is a positive integer. Use the `jointPositionVariables` or `jointVelocityVariables` object function to show the IDs for joint variables. Use the `frameVariables` object function to show the IDs for frame variables.

Example: "j1.Rz.q", ["j1.Rz.q", "j2.Rz.q"], {'j1.Rz.q'}, or {'j1.Rz.q'; 'j2.Rz.q'};

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `targetVariables` | `addTargetVariables` | `clearTargetVariables`

setVariableUnit

Package: `simscape.multibody`

Change physical unit of kinematic variable

Syntax

```
setVariableUnit(ks,ids,unit)
```

Description

`setVariableUnit(ks,ids,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` — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

`ids` — Identifiers of kinematic variables

cell array of characters | string vector

Identifiers of the kinematic variables, specified as either a cell array of characters or string vector. The cell array of character and string vector can be 1-by-*N* or *N*-by-1, where *N* is a positive integer. Use the `jointPositionVariables` or `jointVelocityVariables` object function to show the IDs for joint variables. Use the `frameVariables` object function to show the IDs for frame variables.

Example: "j1.Rz.q", ["j1.Rz.q", "j2.Rz.q"], {'j1.Rz.q'}, or {'j1.Rz.q'; 'j2.Rz.q'};

`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`

Version History

Introduced in R2019a

See Also

`KinematicsSolver`

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.

Version History

Introduced in R2012a

See Also

`smnew`

Topics

“Start a Model from a Template”

smexportonshape

Export a CAD assembly model from Onshape cloud software

Syntax

```
multibodyDescriptionFile = smexportonshape(assemblyURL)
multibodyDescriptionFile = smexportonshape(assemblyURL,Name,Value)
```

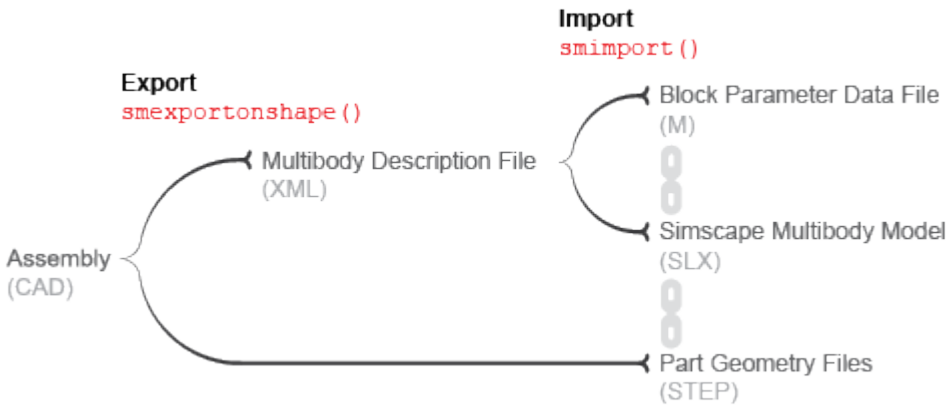
Description

`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/15ab3bfb58cacbf427d77ff3/
   ```
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/15ab3bfb58cacbf427d77ff3/
   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 pairs of arguments as `Name1=Value1,...,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Before R2021a, use commas to separate each name and value, and enclose `Name` *in quotes.*

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`

Version History

Introduced in R2017a

See Also

`smimport`

smimport

Import a CAD, URDF, or Robotics System Toolbox model

Syntax

```
[H,dataFileName] = smimport(modelSource)
___ = smimport( ___ ,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 for import. Use XML files for CAD models, URDF files for URDF models, and `rigidBodyTree` objects for Robotics System Toolbox™ models. XML files must conform to the Simscape Multibody XML schema, and URDF files must conform to the "Supported URDF Elements and Attributes". 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. For example, the `smexportonshape` function converts Onshape CAD models into XML files. The Simscape Multibody Link plug-in is able to convert Autodesk Inventor®, PTC®, and SolidWorks® CAD models into XML files. For other CAD applications and multibody modeling tools, 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. 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 and `rigidBodyJoint` objects. In the created Simscape Multibody model, joints become siblings to rigid bodies and features that not inside rigid body subsystems but alongside them. Joint limits and home positions persist, the later as position state targets, in the appropriate joint blocks.

Note Joint limits are imported from URDF and `rigidBodyTree` models but not from CAD models. Reproduce the joint limits of CAD models manually if you must—by enabling joint limits in the joint blocks and setting the limit positions to appropriate values.

`___ = smimport( ___ ,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.

Examples

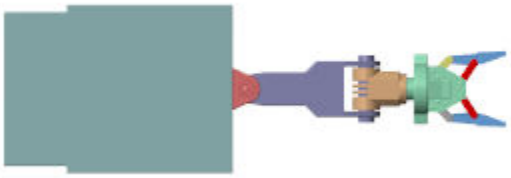
Import a CAD Model with Default Name

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.

Import the model and store it in memory as `Untitled`. You can change the name later. As the model is in XML format, you can omit the extension in its name.

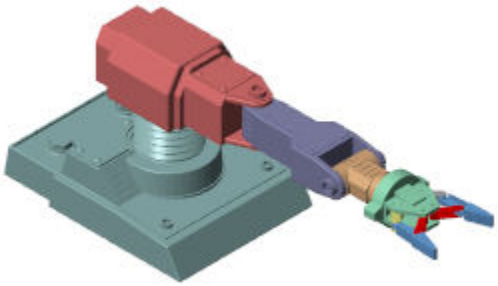
```
smimport('sm_robot');
```

Update the diagram to assemble the model and visualize it in Mechanics Explorer. In the **Modeling** tab, click **Update Model**.



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 the bottom view of the robot on the screen. To change the view convention, use the **View convention** drop-down list in Mechanics Explorer.

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.



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 = 'robottos_new_data_file';
hws.reload
```

To do the same using Model Explorer, first, select **MODELING** > **DESIGN** > **Model Workspace** to open the Model Explorer. Next, in the Model Hierarchy of the Model Explorer, left click **Model Workspace**. In the **Model Workspace** pane, enter `robottos_new_data_file.m` in the **File Name** parameter and click **Reinitialize from Source** button to apply the change.

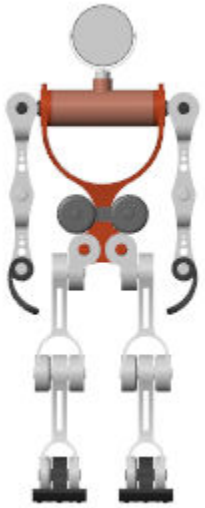
Import a URDF Model

Import a URDF model of a humanoid robot. The model is named `sm_humanoid.urdf`, and it is part of your Simscape Multibody installation.

Import the model and store it in memory as `Untitled`. You can change the name later. As the model is in URDF format, the file extension is required.

```
smimport('sm_humanoid.urdf');
```

Update the diagram to visualize the model in its initial configuration using Mechanics Explorer. In the **Modeling** tab, click **Update Model**.



Simulate the model. 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.

You can modify the model and program the motions of the robot. The “Humanoid Robot” shows a simple example. 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. The model is a part of the Robotics System Toolbox installation and a URDF model named `iiwa14.urdf`.

Convert the URDF model into a `ribidBodyTree` object.

```
iiwaRBT = importrobot('iiwa14.urdf');
```

Note A Robotics System Toolbox license is required to run `importrobot`.

Import the `iiwaRBT` object into Simscape Multibody.

```
iiwaSM = smimport(iiwaRBT);
```

Update the diagram to visualize the model in its initial configuration using Mechanics Explorer. In the **Modeling** tab, click **Update Model**.

Input Arguments

`modelSource` — Name of import file or object

string scalar | character vector

Name of import file or object, specified as a string scalar or character vector. Use XML files for CAD models, URDF files for URDF models, and `rigidBodyTree` objects for Robotics System Toolbox models. 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 pairs of arguments as `Name1=Value1,...,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Before R2021a, use commas to separate each name and value, and enclose `Name` *in quotes.*

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: `'ModelSimplification','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: `'ImportMode','dataFile'`

Data Types: `char` | `string`

`ModelName` — Name of the multibody model to generate
string scalar | character vector

Name of the Simscape Multibody model, specified as the comma-separated pair consisting of `'ModelName'` and a character vector or string scalar. 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: `'ModelName','robotto'`

Data Types: `char` | `string`

`DataFileName` — Name of the parameter data file to generate
string scalar | character vector

Name of the supporting parameter data file, specified as the comma-separated pair consisting of `'DataFileName'` and a character vector or string scalar. 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: `'DataFileName','robottos_new_data'`

Data Types: `char` | `string`

`PriorDataFile` — Name of the last used parameter data file
string scalar | character vector

Name of the last parameter data file, specified as the comma-separated pair consisting of `'PriorDataFile'` and a character vector or string scalar. 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: `'PriorDataFile','robottos_original_data'`

Data Types: `char` | `string`

`VariableName` — Name of the MATLAB structure provided in the parameter data file
string scalar | character vector

Name of the MATLAB data structure provided in the parameter data file, specified as the comma-separated pair consisting of `'VariableName'` and a character vector or string scalar. 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: `'VariableName','robottosData'`

Data Types: `char` | `string`

Output Arguments

`H` — Simscape Multibody model
model handle

Simscape Multibody model, returned as a model handle. 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, returned as a character vector. The 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`

Version History

Introduced in R2012b

See Also

`smexportonshape` | `importrobot`

Topics
“Import URDF Models”
“URDF Primer”
“Import a URDF Humanoid Model”

smnew

Open Simscape Multibody model template

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`

Version History

Introduced in R2012a

See Also

`sm_lib`

Topics

“Start a Model from a Template”

smwritevideo

Configure and create multibody animation videos

Syntax

```
smwritevideo(modelIdentifier,videoName)
smwritevideo(modelIdentifier,videoName,Name,Value)
```

Description

`smwritevideo(modelIdentifier,videoName)` creates a video from the visualization of a multibody model. `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”.

By default, if the model visualization pane is split into tiles, the function captures only the active tile. A colored outline identifies the active tile. You can click a tile to make it the active tile—or use the `tile` name-value pair argument to specify the number of the tile to record.

`smwritevideo(modelIdentifier,videoName,Name,Value)` adds options for specifying the video properties. Use the `Name,Value` pair arguments to set the visualization tile to record, the video file format, the frame refresh rate, the frame width and height, and the 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
```

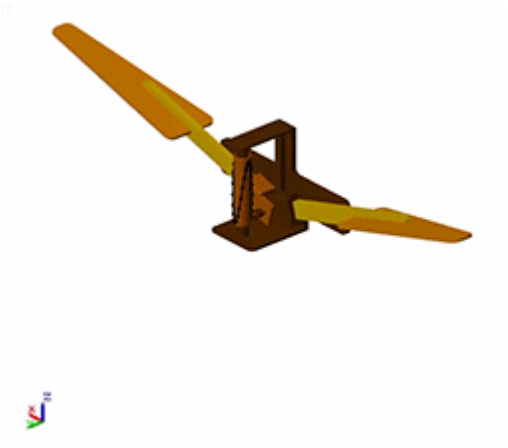
2 Simulate the model.

```
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 from Tile of Split Screen

Split the visualization pane for the `sm_cam_flapping_wing` model and record a video named `flapping_wing_video` from a specified tile.

1. Open the flapping wing featured example.

   ```
   sm_cam_flapping_wing
   ```
2. Simulate the model.

   ```
   sim('sm_cam_flapping_wing')
   ```
3. In Mechanics Explorer, click the quad view button or select **View > Layout > Four Standard Views**. The top left tile (numbered 1) is by default the active tile.
4. Record a video from the bottom right tile (numbered 4 in a counting scheme that runs first top to bottom and then left to right).

   ```
   smwritevideo('sm_cam_flapping_wing','flapping_wing_video','tile','4');
   ```

   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`

Property	Argument	Setting
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(s'm_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 pairs of arguments as `Name1=Value1,...,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Before R2021a, use commas to separate each name and value, and enclose `Name` *in quotes.*

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`

`Tile` — Number of visualization tile to record
unitless scalar

Number of the visualization tile to record. Use this parameter when the visualization pane is split and you want to record a tile other than the active tile (that enveloped in red highlight). Tiles are numbered first top to bottom and then left to right. In a quad view, the top left tile is numbered 1, the bottom left tile is 2, the top right tile is 3, and the bottom right tile is 4.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Version History

Introduced in R2016b

See Also

Video Creator

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` — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

`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.

- **1** — All model constraints and target variables are satisfied.
- **2** — All model constraints and target variables are satisfied, but the mechanism is in a singular configuration.
- **-1** — All model constraints are satisfied, but one or more target variables are not satisfied.
- **-2** — All model constraints are satisfied, but one or more target variables are not satisfied, and the mechanism is in a singular configuration.
- **-3** — The solution is invalid because one or more model constraints cannot be satisfied. The output variables are set to `NaN`.
- **-4** — The solution is invalid because the mechanism is in a singular configuration. The output variables are set to `NaN`.

Note In a singular configuration, the mechanism's motion is restricted and certain velocities cannot be computed. The most common type of kinematic singularity is due to gimbal lock in Bearing Joint, Bushing Joint, or Gimbal Joint. In addition, mechanisms with a Constant Velocity Joint or a belt-cable network may also have kinematic singularities.

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`

Version History

Introduced in R2019a

See Also

`KinematicsSolver`

targetVariables

Package: `simscape.multibody`

List kinematic variables assigned as targets

Syntax

```
targetVariables(ks)
```

Description

`targetVariables(ks)` outputs a table that lists the target variables in the `KinematicsSolver` object `ks`. Each row shows the details of a variable. For a joint variable, the table shows the ID, joint type, block path, and unit. For a frame variable, the table shows the ID, base and follower frames, and unit.

Both joint and frame variables can serve as targets. Note that there are a few limitations for variables to be assigned as targets. See `addTargetVariables` for more details. To assign a variable as target, use the `addTargetVariables` object function. To remove target variables, use the `removeTargetVariables` object function to drop target variables that are no longer needed for the analysis, or use the `clearTargetVariables` object function to drop all target variables in one call.

During a search, the target variables serve as constraints of the system, and the solver searches for a solution that is compatible with the targeted joint and frame variables. Do not overconstrain the system with target variables. A system is overconstrained if a kinematic loop in the system has a target for every joint. One way of avoiding overconstraining is to reassign one of the joint variables from a target to an initial guess by using the `addInitialGuessVariables` object function.

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2019a

See Also

`KinematicsSolver` | `addTargetVariables` | `clearTargetVariables` | `removeTargetVariables`

simscape.multibody.tirread

Read TIR file

Syntax

```
tireParams = simscape.multibody.tirread(filename)
```

Description

`tireParams = simscape.multibody.tirread(filename)` reads a TIR file and returns a structure array that contains the parameters defined in the TIR file.

Input Arguments

`filename` — TIR file name

character vector | string scalar

TIR file name, specified as a character vector or string scalar. The file name must end with the `.tir` or `.TIR` extension.

Example: `'myFile.tir'`

Data Types: `char` | `string`

Output Arguments

`tireParams` — Tire parameters

scalar structure array

Tire parameters, returned as a scalar structure array. You can use the structure array in the **Tire Parameters** parameter of the Magic Formula Tire Force and Torque block.

Version History

Introduced in R2021b

See Also

Magic Formula Tire Force and Torque

viewSolution

Open Kinematics Solver Viewer window to visualize KinematicsSolver solution

Syntax

```
viewSolution(ks)
```

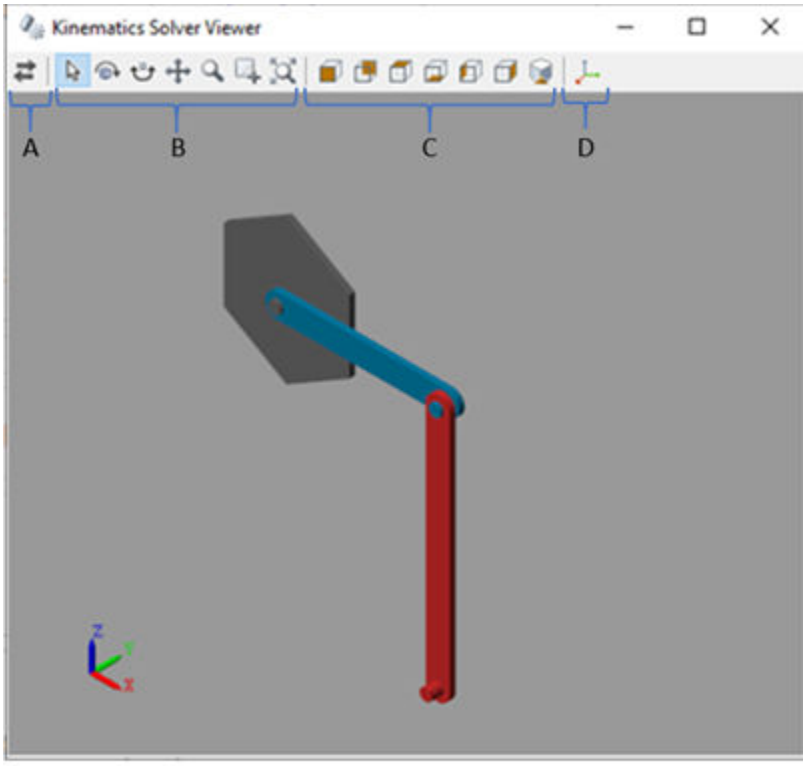
Description

`viewSolution(ks)` opens a Kinematics Solver Viewer window to visualize the mechanism corresponding to the latest solution computed by the `solve` function for the `KinematicsSolver` object `ks`.

Note

- An error occurs if the `solve` function was not invoked before calling the `viewSolution` function.
- If the `statusFlag` corresponding to the last call to solve is -3, the state returned by the solver is not kinematically feasible. In this case, the mechanism rendered in the viewer is not physically realizable.

The Kinematics Solver Viewer window is shown in the following figure. This pane provides visual feedback on the mechanism that you are analysing. Select the Update Visualization button  or press **F5** to view the mechanism that corresponds to the latest solution of `ks`. Use the viewer to examine the mechanism from different perspectives by selecting the standard view or by rotating, panning, and zooming the mechanism. Right-click the window to access a context-sensitive menu. This menu provides additional options that allow you to change the background color, split the visualization window into multiple tiles, and modify the view convention from the default **+Z up (XY Top)** setting.



Kinematics Solver Viewer Window

Examples

Visualize the Solution of a Forward Kinematics Problem of a Double Pendulum Model

Create a KinematicsSolver object for the sm_double_pendulum model.

```
mdl = 'sm_double_pendulum';
open_system(mdl);
ks = simscape.multibody.KinematicsSolver(mdl);
```

List all of the joint position variables.

```
jointPositionVariables(ks)
```

```
ans =
2×4 table

    ID        JointType                  BlockPath                 Unit
 _________ _______________ ____________________________________ _____

 "j1.Rz.q" "Revolute Joint" "sm_double_pendulum/Lower Revolute" "deg"
 "j2.Rz.q" "Revolute Joint" "sm_double_pendulum/Upper Revolute" "deg"
```

Assign the upper and lower revolute joint angles as target variables.

```
targetIDs = ["j1.Rz.q";"j2.Rz.q"];
addTargetVariables(ks,targetIDs);
```

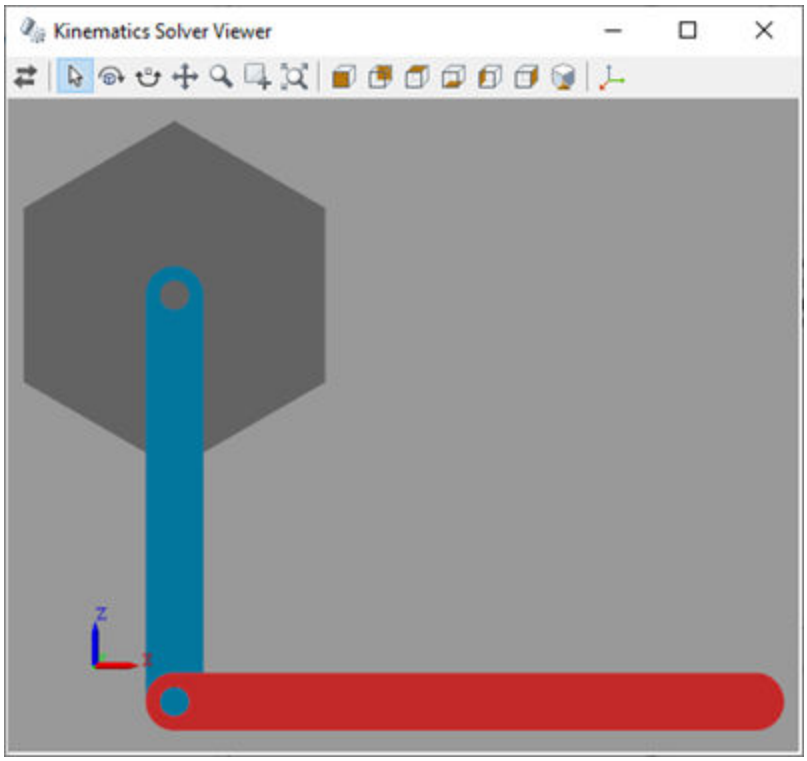
Solve the forward kinematics problem with given joint angles.

```
targets = [90,0];
outputVec = solve(ks,targets);
```

Open the Kinematics Solver Viewer.

```
viewSolution(ks);
```

Click the **Front view** button to view the solution.



Close the viewer.

```
closeViewer(ks);
```

Input Arguments

ks — Kinematics solver object

`KinematicsSolver` object

Kinematics solver object, specified as a `KinematicsSolver` object that is the representation of the Simscape Multibody model used for kinematic analysis.

Example: ks = simscape.multibody.KinematicsSolver('sm_double_pendulum')

Version History

Introduced in R2020a

See Also

`KinematicsSolver` | `solve` | `closeViewer`

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.

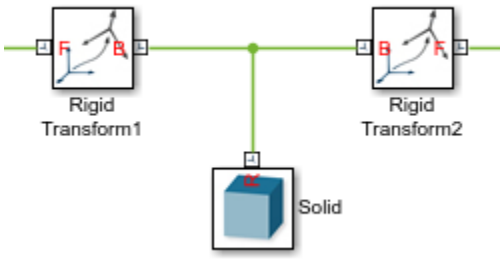
The input and output ports of blocks are now compatible only with Simscape physical signals. In first-generation blocks, the same types of ports were compatible only with Simulink signals. The switch to physical signal inputs and outputs allows for tighter integration with other Simscape physical domains. 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, which is not supported anymore, with inertia, geometry, and frames—originally referred to as coordinate systems, or CSs—represents each body. In the second-generation environment, the solid blocks replaces the Body block. Rigid Transform blocks often complement a 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	🟩	🟩
Derived inertia parameters	🔴	🟩
Variable inertia parameters	🟩	🟩
Body visualization	🟡	🟩
Interactive frame creation	🔴	🟩

Geometry, Inertia, and Frames

The solid blocks provide a selection of native geometries parameterized in terms of key dimensions, such as Spherical Solid (parameterized in terms of radius) and Cylindrical Solid (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 blocks also provide 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 block, which is not supported anymore, 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		
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, which are not supported anymore, 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, which is not supported anymore. 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, which is not supported anymore, 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	■	■
URDF assembly import	●	■
Onshape assembly export	●	■

For more information, see:

- “CAD Translation”
- “Import URDF Models”
- “Onshape Import”