SimMechanics™
Reference

R2011b

MATLAB®
&SIMULINK®

MathWorks®

SimMechanics™ Reference

Revision History

October 2008	Online only	New for Version 3.0 (Release 2008b)
March 2009	Online only	Revised for Version 3.1 (Release 2009a)
September 2009	Online only	Revised for Version 3.1.1 (Release 2009b)
March 2010	Online only	Revised for Version 3.2 (Release 2010a)
September 2010	Online only	Revised for Version 3.2.1 (Release 2010b)
April 2011	Online only	Revised for Version 3.2.2 (Release 2011a)
September 2011	Online only	Revised for Version 3.2.3 (Release 2011b)

Contents

1 Block Reference

Block Reference

This page refers to SimMechanics™ First-Generation. For SimMechanics Second-Generation, click here.

Machines, Bodies, and Grounds

Body	Rigid body with frames, inertia and geometry
Ground	Fixed point attached to world
Machine Environment	Mechanical simulation parameters of a machine
Shared Environment	Utility that connects two independent machines in a single mechanical environment

Joints

Assembled Joints (p. 1-3)	Collocated joints
Disassembled Joints (p. 1-4)	Dislocated joints
Massless Connectors (p. 1-4)	Rigidly separated joints

Assembled Joints

Bearing	Joint with three revolute and one prismatic joint primitives
Bushing	Joint with three revolute and three prismatic joint primitives
Custom Joint	Joint with custom combination of prismatic, revolute, and spherical joint primitives
Cylindrical	Joint with one revolute and one prismatic joint primitives
Gimbal	Joint with three revolute joint primitives
In-Plane	Joint with two coplanar prismatic joint primitives
Planar	Joint with one revolute and two prismatic joint primitives
Prismatic	Primitive joint with one translational degree of freedom
Revolute	Primitive joint with one rotational degree of freedom
Screw	Joint with coupled rotational and translational degrees of freedom
Six-DoF	Joint with three revolute and three prismatic joint primitives

Spherical	Primitive joint with three rotational degrees of freedom
Telescoping	Joint with three revolute and one prismatic joint primitives
Universal	Joint with two revolute joint primitives
Weld	Joint with zero degrees of freedom

Disassembled Joints

Disassembled Cylindrical	Joint with misaligned base and follower axes containing one revolute and one prismatic joint primitives
Disassembled Prismatic	Primitive joint with misaligned base and follower axes containing one translational degree of freedom
Disassembled Revolute	Primitive joint with misaligned base and follower axes containing one rotational degree of freedom
Disassembled Spherical	Primitive joint with misaligned base and follower axes containing three rotational degrees of freedom

Massless Connectors

Revolute-Revolute	Constant-length joint connector with two spatially separated revolute axes
Revolute-Spherical	Constant-length joint connector with spatially separated revolute axis and spherical pivot point
Spherical-Spherical	Constant-length joint connector with two spatially separated spherical pivot points

Constraints and Drivers

Angle Driver	Driver specifying a time-dependent angle between two body axis vectors
Distance Driver	Time-dependent distance between two body coordinate systems
Gear Constraint	Constraint that restricts body motion to rotation along tangent circles
Linear Driver	Time-dependent signal of a vector position component between two body coordinate systems
Parallel Constraint	Constant parallel relationship between two body axis vectors
Point-Curve Constraint	Constraint that restricts body motion to a specified path
Velocity Driver	Linear and angular velocity components of base and follower body coordinate systems

Actuators and Sensors

Body Actuator	Time-dependent force and torque used to actuate a body
Body Sensor	Body translation and rotation sensor
Constraint & Driver Sensor	Sensor used to measure the reaction force and torque between two constrained or driven bodies
Driver Actuator	Time-dependent motion input for driver blocks
Joint Actuator	Time-dependent force, torque, or motion input to a joint
Joint Initial Condition Actuator	Initial joint position and velocity
Joint Sensor	Joint force, torque, and motion sensor
Joint Stiction Actuator	Joint static and kinetic friction
Variable Mass & Inertia Actuator	Time-dependent mass and inertia parameters

Force Elements

Body Spring & Damper	Damped linear oscillator force between two bodies
Joint Spring & Damper	Damped linear oscillator force or torque acting on a joint

Interface Elements

Prismatic-Translational Interface	Connection interface between prismatic primitive and Simscape mechanical translational elements
Revolute-Rotational Interface	Connection interface between revolute primitive and Simscape mechanical rotational elements

Utilities

Continuous Angle	Utility that converts a discontinuous bounded angle into a continuous unbounded angle
Mechanical Branching Bar	Utility that maps multiple sensor and actuation signals into a single connection line
RotationMatrix2VR	Utility that transforms 3x3 rotation matrix into rotation axis-angle 4-vector

Blocks — Alphabetical List

Angle Driver

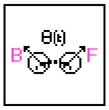
Purpose

Driver specifying a time-dependent angle between two body axis vectors

Library

Constraints & Drivers

Description



The Angle Driver block drives axis vectors defined on two Bodies. You specify fixed base and fixed follower body axis vectors $\boldsymbol{a}_B$, $\boldsymbol{a}_F$ in the Body CS on either side of the Driver on each body, then drive the angle between the body axis vectors as a function of time.

The Angle Driver block specifies the angle θ defined by

$$\cos\theta = |\boldsymbol{a}_B \cdot \boldsymbol{a}_F| / (|\boldsymbol{a}_B|\,|\boldsymbol{a}_F|)$$

as a function of time: $\theta = \theta(t{=}0) + f(t)$. You connect the Angle Driver to a Driver Actuator block.

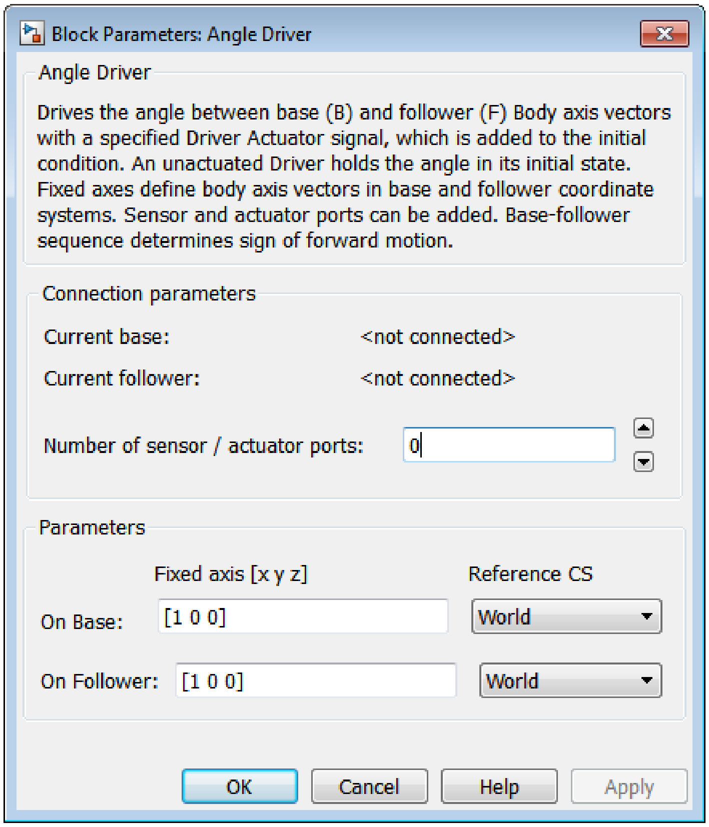
The Simulink® input signal into the Driver Actuator specifies the time-dependent driving function *f(t)* and its first two derivatives, as well as their units. If you do not actuate Angle Driver, this block acts as a time-independent constraint that freezes the angle between the two body axes at its initial value $\theta(t{=}0)$ during the simulation.

Drivers restrict relative degrees of freedom (DoFs) between a pair of bodies as specified functions of time. Locally in a machine, they replace a Joint as the expression of the DoFs. Globally, Driver blocks must occur topologically in closed loops. Like Bodies connected to a Joint, the two Bodies connected to a Drivers are ordered as base and follower, fixing the direction of relative motion.

Caution

If the two axes come close to aligning, that is, if θ approaches zero, the constraint between the two axes becomes singular, and the simulation slows down. See "How SimMechanics Software Works" and "Handling Motion Singularities".

You can also connect a Constraint & Driver Sensor to any Driver measure the reaction forces/torques between the driven bodies.


Block Parameters: Angle Driver

Angle Driver

Drives the angle between base (B) and follower (F) Body axis vectors with a specified Driver Actuator signal, which is added to the initial condition. An unactuated Driver holds the angle in its initial state. Fixed axes define body axis vectors in base and follower coordinate systems. Sensor and actuator ports can be added. Base-follower sequence determines sign of forward motion.

Connection parameters

Current base: <not connected>

Current follower: <not connected>

Number of sensor / actuator ports: 0

Parameters

	Fixed axis [x y z]	Reference CS
On Base:	[1 0 0]	World
On Follower:	[1 0 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive rotation is the follower rotating in the right-handed sense about the rotation axis.

Current base

When you connect the base (B) connector port on the Angle Driver block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Angle Driver Base and Follower Body Connector Ports on page 2-4.

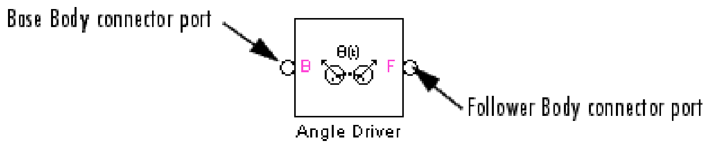
Current follower

When you connect the follower (F) connector port on the Angle Driver block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Angle Driver Base and Follower Body Connector Ports on page 2-4.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Driver Actuator and Constraint & Driver Sensor blocks to this Driver. The default is `0`.

To activate the Driver, connect a Driver Actuator.



Angle Driver Base and Follower Body Connector Ports

Parameters

Fixed axis [x y z]

For the **Base** and **Follower** bodies, respectively, enter the body axis vectors. The defaults are `[1 0 0]`.

Reference CS

Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the **Base** and **Follower** body axis vectors are oriented with respect to. This CS also determines the absolute meaning of reaction forces/torques at this Driver. The defaults are `World`.

See Also

Constraint & Driver Sensor, Driver Actuator, Parallel Constraint, Velocity Driver

See “Constraining and Driving Degrees of Freedom” for more on restricting DoFs with Drivers.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on using drivers in closed loops.

Bearing

Purpose Joint with three revolute and one prismatic joint primitives

Library Joints

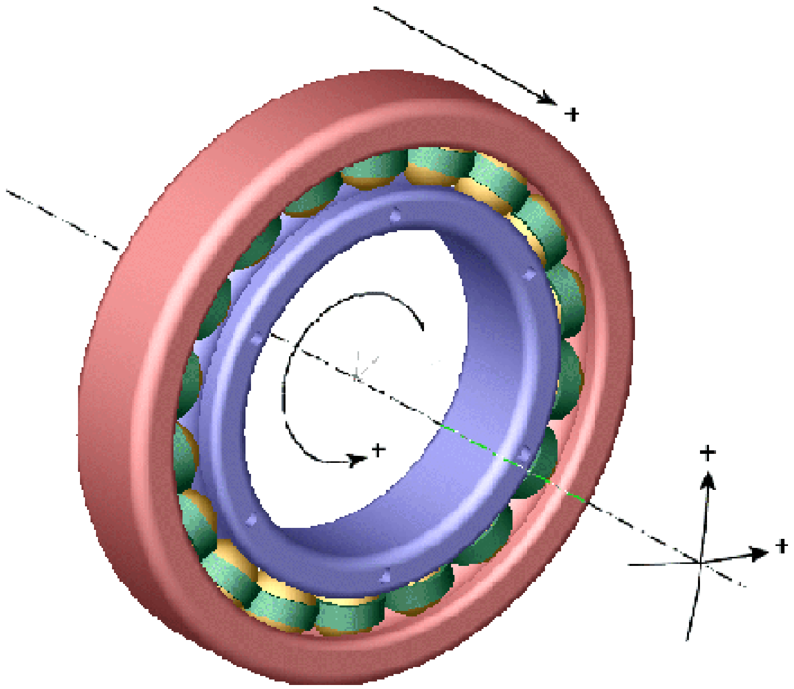
Description The Bearing block represents a composite joint with one translational degree of freedom (DoF) as one prismatic primitive and three rotational DoFs as three revolute primitives. There are no constraints among the primitives. Unlike Telescoping, Bearing represents the rotational DoFs as three revolutes, rather than as one spherical.

Warning

A joint with three revolute primitives becomes singular if two or three of the rotation axes become parallel ("gimbal lock"). The simulation stops with an error in this case.

A joint with three revolute primitives must be configured in the initial state with the three revolute primitive axes mutually orthogonal. There are no restrictions on the primitive axes once the simulation starts, except to prevent any two of the primitive axes from becoming parallel.

Note Bearings are often represented by one translational and one rotational DoF. The Bearing block has *three* rotational degrees of freedom, rather than one, in order to represent transverse "play" in the joint.



Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

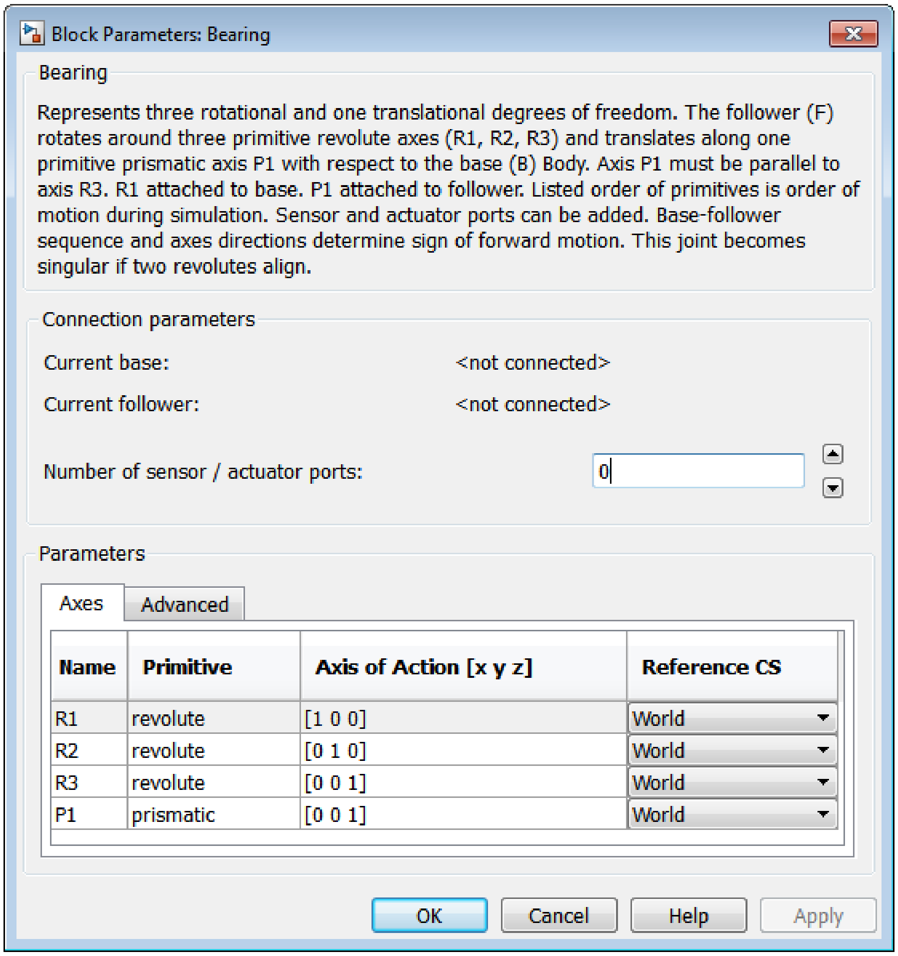
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space


Block Parameters: Bearing

Bearing

Represents three rotational and one translational degrees of freedom. The follower (F) rotates around three primitive revolute axes (R1, R2, R3) and translates along one primitive prismatic axis P1 with respect to the base (B) Body. Axis P1 must be parallel to axis R3. R1 attached to base. P1 attached to follower. Listed order of primitives is order of motion during simulation. Sensor and actuator ports can be added. Base-follower sequence and axes directions determine sign of forward motion. This joint becomes singular if two revolutes align.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>
Number of sensor / actuator ports:	0

Parameters

Axes | Advanced

Name	Primitive	Axis of Action [x y z]	Reference CS
R1	revolute	[1 0 0]	World
R2	revolute	[0 1 0]	World
R3	revolute	[0 0 1]	World
P1	prismatic	[0 0 1]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive translation is the follower moving in the direction of the translation axis. Positive rotation is the follower moving around the rotational axis following the right-hand rule.

Current base

When you connect the base (B) connector port on the Bearing block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Bearing Base and Follower Body Connector Ports on page 2-10.

The base Body is automatically connected to the first joint primitive `R1` in the primitive list in **Parameters**.

Current follower

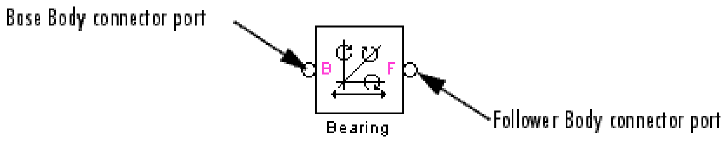
When you connect the follower (F) connector port on the Bearing block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Bearing Base and Follower Body Connector Ports on page 2-10.

The follower Body is automatically connected to the last joint primitive `P1` in the primitive list in **Parameters**.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`.

The motions of prismatic and revolute primitives are specified in linear and angular units, respectively.



Bearing Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

Name - Primitive

The primitive list states the names and types of joint primitives that make up the Bearing block: revolute primitives `R1`, `R2`, `R3`, and prismatic primitive `P1`.

Axis of Action [x y z]

Enter here as a three-component vector the directional axes defining the allowed motions of these primitives and their corresponding DoFs:

- Prismatic: axis of translation
- Revolute: axis of rotation

The default vectors are shown in the dialog above. The axis is a directed vector whose overall sign matters.

To prevent singularities and simulation errors, no two of the revolute axes can be parallel.

Reference CS

Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of action is oriented with respect to. This CS also determines the absolute meaning of forces/torques and motion along/about the joint axis. The default is `World`.

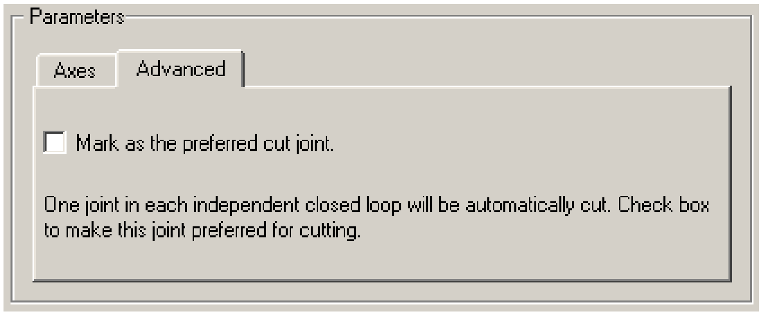
Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

The entries on the **Axes** tab are required. Each DoF primitive in Bearing has an entry line. These lines specify the direction of the axes of action of the DoFs that the Bearing represents.

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint
: In a closed loop, the simulation internally and automatically cuts one and only one joint.

  If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Bushing, Cylindrical, Gimbal, Prismatic, Revolute

See "Modeling Degrees of Freedom" for more on representing DoFs with Joints.

See "Checking Model Topology" and "How SimMechanics Software Works" for more on closed loops and cutting.

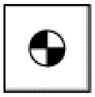
Purpose

Rigid body with frames, inertia and geometry

Library

Bodies

Description



The Body block represents a rigid body with properties you customize. The representation that you specify must include:

- The body's *mass* and *moment of inertia tensor*
- The coordinates for the body's *center of gravity* (CG)
- One or more *Body coordinate systems* (CSs)

The representation can also include optional *body geometry* and *color* information for visualization.

A rigid body is defined in space by the position of its CG (or center of mass) and its orientation in some defined coordinate system.

Setting Body Initial Conditions The initial position and orientation of a body are set by the entries in its Body dialog that define the body's *home configuration*. These initial conditions remain unchanged, unless, with a Joint Initial Condition Actuator, you change the initial conditions of the Joint(s) connected to the Body prior to starting the simulation, or you actuate the Body with a Body Actuator. The imposition of additional initial conditions defines the *initial configuration* of the body.

Defining a Body with Mass, Coordinate System, and Visualization Properties

You must enter properties for a SimMechanics body in two sets, *mass properties* and *coordinate system properties*. A third set, *visualization properties*, is optional.

Mass Properties

The mass properties are defined by the body's mass and inertia tensor.

- The mass is the body’s inertia and controls the translational acceleration of the CG in response to an applied force.
- The inertia tensor measures the distribution of mass density in the body and controls the rotational acceleration of the body about the CG in response to an applied torque.
- The components of the inertia tensor control the initial orientation of the body and are always interpreted as being in the CG CS axes. The orientation of the CG CS axes with respect to another CS external to the body (the World CS, a CS on a Ground, or a CS on another Body) then determines the orientation of the body with respect to other bodies or with respect to World.
- The body’s inertia tensor defines its *principal axes* and *moments* and its *equivalent ellipsoid*, one of the standard shapes available for displaying a body in space.

Coordinate System Properties

The coordinate system properties are defined by the body’s Body CSs.

- The CS with its origin at the CG is required. The CG point specifies both the initial position of the whole body and the origin of the CG CS. You must also orient the CG CS axes.
- You can place one or more additional Body CSs on a body. The Body dialog requires at least one. You must define each Body CS by the position of its origin and the orientation of its CS axes.
- Each connection of a Joint, Constraint/Drive, Actuator, or Sensor block to a Body requires an anchor point on the Body. This anchor point is one of the Body CS origins.
- Body CSs on the block available for connections are shown by Body CS ports ⊞ on the sides of the block. You can show or hide each Body CS on the block sides.
- The set of a body’s Body CS origins (including the CG CS) defines the body’s *convex hull*, one of the standard shapes available for displaying a body in space.

Visualization Properties

The visualization properties of a body include its color and geometry (surface size and shape).

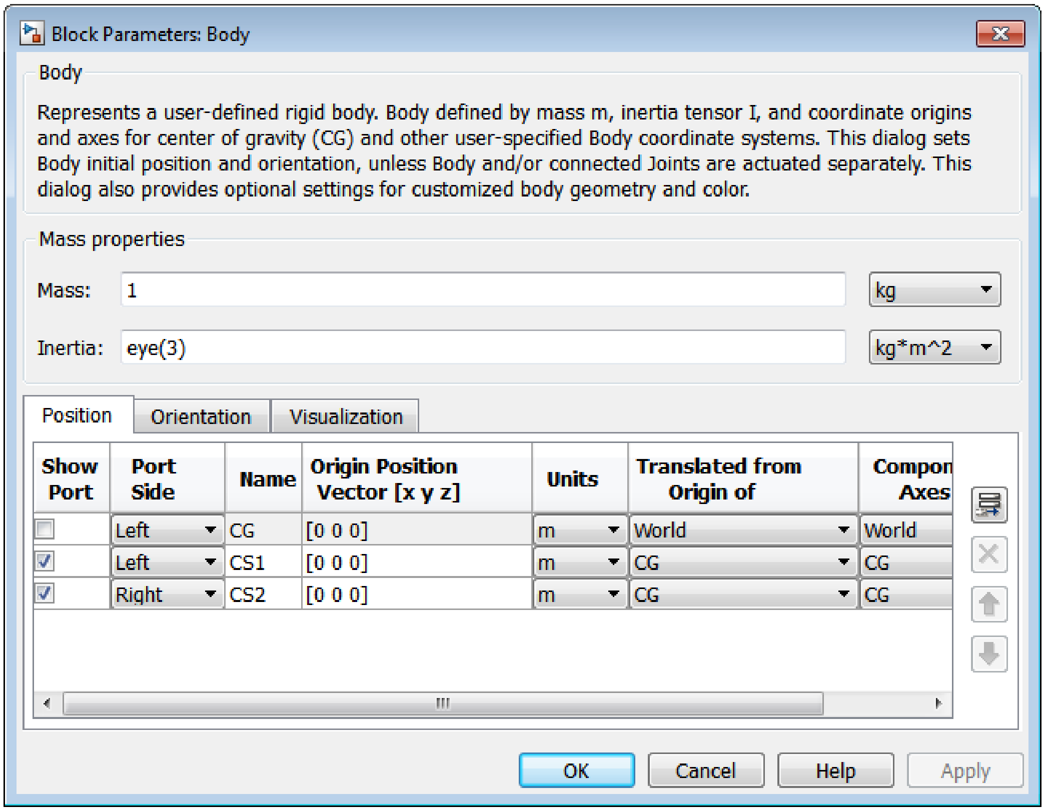
- As a default, the machine of which the Body is a member determines these visualization properties.
- You can partly or completely override these defaults with custom settings for an individual Body.

Default Initial State of a Body

These two properties determine a body's initial position and orientation:

- The position of a body's CG sets its initial position.
- The body's inertia tensor components (in the CG CS) and the orientation of the CG CS axes with respect to other CSs in the machine set its initial orientation.

The initial conditions of a machine can be changed with Joint Initial Condition Actuator blocks before you start a simulation. If you do not change the initial state of a Body before simulation, SimMechanics simulation sets its initial position and orientation to its Body dialog entries, defining the body's home configuration. SimMechanics simulation also sets the Body's initial linear/angular velocities to zero in this case.



Block Parameters: Body

Body

Represents a user-defined rigid body. Body defined by mass m, inertia tensor I, and coordinate origins and axes for center of gravity (CG) and other user-specified Body coordinate systems. This dialog sets Body initial position and orientation, unless Body and/or connected Joints are actuated separately. This dialog also provides optional settings for customized body geometry and color.

Mass properties

Mass: 1 kg

Inertia: eye(3) kg*m^2

Position | Orientation | Visualization

Show Port	Port Side	Name	Origin Position Vector [x y z]	Units	Translated from Origin of	Compon Axes
☐	Left	CG	[0 0 0]	m	World	World
✓	Left	CS1	[0 0 0]	m	CG	CG
✓	Right	CS2	[0 0 0]	m	CG	CG

Dialog Box and Parameters

The dialog has two active areas, **Mass properties** and the set of tabs, **Position** and **Orientation** for Body coordinate systems, as well as **Visualization**.

Restricted Parameters

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

- The **Position** Body coordinate systems tab
- The **Orientation** Body coordinate systems tab

Mass Properties

Mass

Enter the mass of the body in the first field and choose units in the pull-down menu to the right. The mass must be a positive, real number or MATLAB® equivalent expression. The defaults are `1` and `kg` (kilograms).

Inertia

Enter the inertia tensor (with respect to the Body CG CS axes) in the first field and choose units in the pull-down menu to the right. The tensor must be a 3-by-3 real, symmetric matrix. The default tensor is `eye(3)`, the MATLAB 3-by-3 identity matrix. A zero tensor `zeros(3,3)` defines a point mass. The units default is `kg-m`^{2} (kilograms-meters^2).

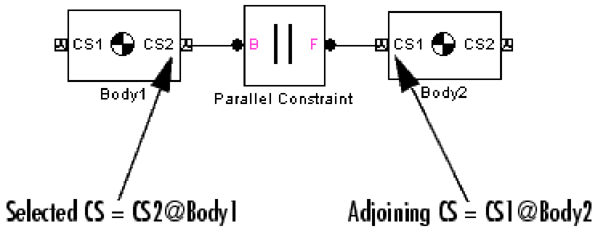
Body Coordinate Systems

Configuring a Body Coordinate System

You set up Body CSs on the **Position** and **Orientation** tabs:

- The default configuration consists of three Body CSs: the required CG CS attached to the body's CG and two other optional Body CSs, called "CS1" and "CS2," for connecting Joints, Constraints, or Drivers.
- You can configure the CG CS but not delete it. You also cannot create additional CG CSs, although you can duplicate the CG CS with a different name. (See more about Body coordinate systems controls following.)
- The other CSs can be configured or deleted as you want, keeping at least one.
- Configuring a Body CSs requires two groups of steps:
  - Positioning the Body CS origin in the **Position** tab
  - Orienting the Body CS axes in the **Orientation** tab
- Defining Body CSs requires referring to other, pre-existing CSs in the model. In a given Body block, you can refer to Body and Grounded CSs in three ways. The references must be to:
  - World

- Other Body CSs on the same body
- The *Adjoining CS*, the coordinate system on a neighboring body or ground directly connected to the selected Body CS by a Joint, Constraint, or Driver



- You must directly or indirectly define all Body CSs by reference to a Ground or to World. With indirect reference, you specify a Body CS relative to another CS and so on, in a chain of references that ultimately ends in a Ground or World. The CS reference chains of the **Position** and **Orientation** tabs can be different. The CS reference chains must not form a closed cycle.
- Switch between the **Position** and **Orientation** tabs.

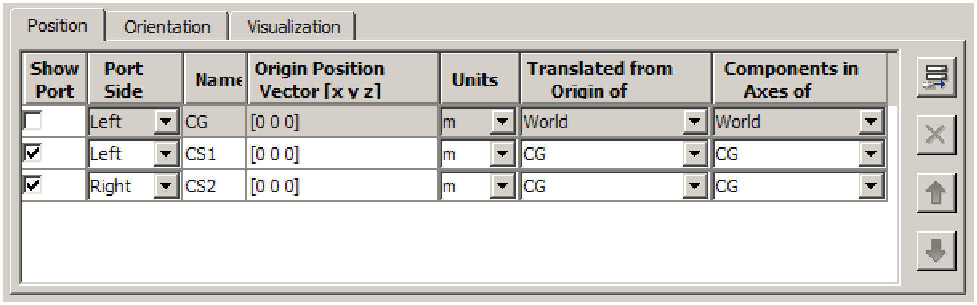
  Each Body CS is labeled with a name, `CG` for the CG CS, and `CS1`, `CS2`, etc., for additional CSs.

Configuring the Position Fields

The **Position** fields for each Body CS specify the position of that CS's origin as a translation vector:

- The numerical components of the vector carry units.
- The translation vector's components are oriented with respect to another set of CS axes.
- The origin is displaced from the origin of another, pre-existing CS in your machine by this translation vector.

Highlight each Body CS to configure it.



Show Port	Port Side	Name	Origin Position Vector [x y z]	Units	Translated from Origin of	Components in Axes of
☐	Left	CG	[0 0 0]	m	World	World
✓	Left	CS1	[0 0 0]	m	CG	CG
✓	Right	CS2	[0 0 0]	m	CG	CG

Origin Position Vector [x y z]

Enter the translation vector that defines the position of the Body CS's origin.

The entry for the CG CS origin positions the entire body.

Units

Choose linear units for the translation vector. The default is m (meters).

Translated from Origin of

In the pull-down menu, choose the other, pre-existing CS in your machine that defines the starting point for the translation vector. The choices are `World`, `Adjoining`, and the other Body CSs on this Body. The ending point of the translation vector is this Body CS's origin.

For the CG CS, the default starting-point CS is `World`. For the additional Body CSs (CS1, CS2, etc.), the default starting point CS is this Body's `CG`.

Components in Axes of

In the pull-down menu, choose the CS whose axes define the orientation of the translation vector's components. The choices are `World`, `Adjoining`, and the other Body CSs on this Body. The translation vector's components are measured relative to the axes of the CS chosen in this column.

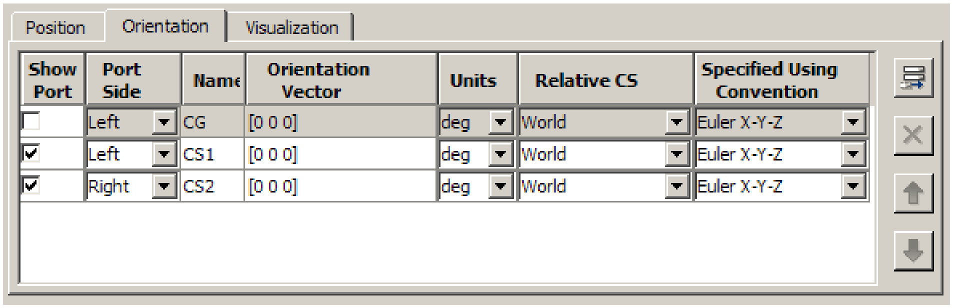
For the CG CS, the default orientation CS is `World`. For the additional Body CSs (CS1, CS2, etc.), the default orientation CS is this Body's `CG`.

Configuring the Orientation Fields

The **Orientation** fields for each Body CS specify the orientation of that CS's triad of axes as a rotation:

- The orientation vector specifying the rotation vector has three components.
- The numerical components of the vector carry units.
- The rotation is oriented with respect to some other, pre-existing set of CS coordinate axes in your machine.
- The orientation vector's components are interpreted in the convention of a rotation representation.

Highlight each Body CS to configure it.



Show Port	Port Side	Name	Orientation Vector	Units	Relative CS	Specified Using Convention
☐	Left	CG	[0 0 0]	deg	World	Euler X-Y-Z
✓	Left	CS1	[0 0 0]	deg	World	Euler X-Y-Z
✓	Right	CS2	[0 0 0]	deg	World	Euler X-Y-Z

Orientation Vector

Enter the components of the rotation that defines the orientation of the Body CS's axes. The geometric meaning of these components is determined by the **Specified Using Convention** column.

The required entry for the CG CS orients the CG CS axes. Together with the **Inertia tensor** entry in **Mass properties**, the CG CS axes orient the whole body with respect to another CS in your machine.

Units

Choose angular units for the rotation, degrees or radians. The default is `deg` (degrees).

Relative CS

In the pull-down menu, choose one of the other pre-existing CSs in your machine to define the starting orientation for the rotation. The choices are `World`, `Adjoining`, and the other Body CSs on this Body.

Specified Using Convention

In the pull-down menu, choose the representation type for the rotation:

Rotation Type	Orientation Vector Components
`Quaternion`	[n_x`*sin(θ/2)` n_y`*sin(θ/2)` n_z`*sin(θ/2) cos(θ/2)]`
`3x3Transform`	3-by-3 orthogonal rotation matrix R
`Euler`	Rotation angles about sequence of three axes defining Euler angle convention [*first-axis second-axis third-axis*]

Rotation Conventions

There are three conventions in a Body block for representing rotations. See "Representations of Body Motion" and "Representations of Body Orientation" to learn more about rotations.

- `Euler`

  The Euler angle convention specifies the rotation of the Body CS axes by rotating about three axes in a sequence. The components of the 1-by-3 row vector are the angles of rotation about those three axes, respectively in sequence, in degrees or radians.

  For example, `Euler X-Y-Z` means rotate about the original X axis, then about the first intermediate Y axis, and then about the second intermediate Z axis. Another example: `Euler X-Z-Y` means rotate about the original X axis, then about the first intermediate Z axis, and then about the second intermediate Y axis.

- `3-by-3 Transform`

  The transform convention specifies the rotation as a dimensionless 3-by-3 orthogonal rotation matrix. The inverse of an orthogonal matrix R is equal to its transpose: $R^{-1} = R^{T}$.

  The columns of R are the *(x,y,z)* unit vectors of the Body CS axes. The units menu is inactive.

- `Quaternion`

  The quaternion convention specifies the rotation in angle-axis form as a dimensionless 1-by-4 row vector:

  $$[n_x\text{*}\sin(\theta/2)\ \ n_y\text{*}\sin(\theta/2)\ \ n_z\text{*}\sin(\theta/2)\ \ \cos(\theta/2)]$$

  $\boldsymbol{n} = (n_x,n_y,n_z)$ is a three-component vector of unit length: $\boldsymbol{n}\cdot\boldsymbol{n} = n_x^2 + n_y^2 + n_z^2 = 1$.

  The unit vector $\boldsymbol{n}$ specifies the axis of rotation. The rotation angle about that axis is θ and follows the right-hand rule.

Managing the Body Coordinate Systems List

The Body coordinate system controls (see the following figure, Body Coordinate Systems Controls on page 2-24) allow you to add, reorder, and delete Body CSs on a Body block.

To add a Body CS to the list:

1. Highlight an existing Body CS in the list.
2. Click the **Add** button (see the following figure, Body Coordinate Systems Controls on page 2-24).

   A new Body CS appears immediately below the Body CS you highlighted. New Body CSs are named in sequence after the current ones: CS3, CS4, etc.

To change the position of a Body CS in the list:

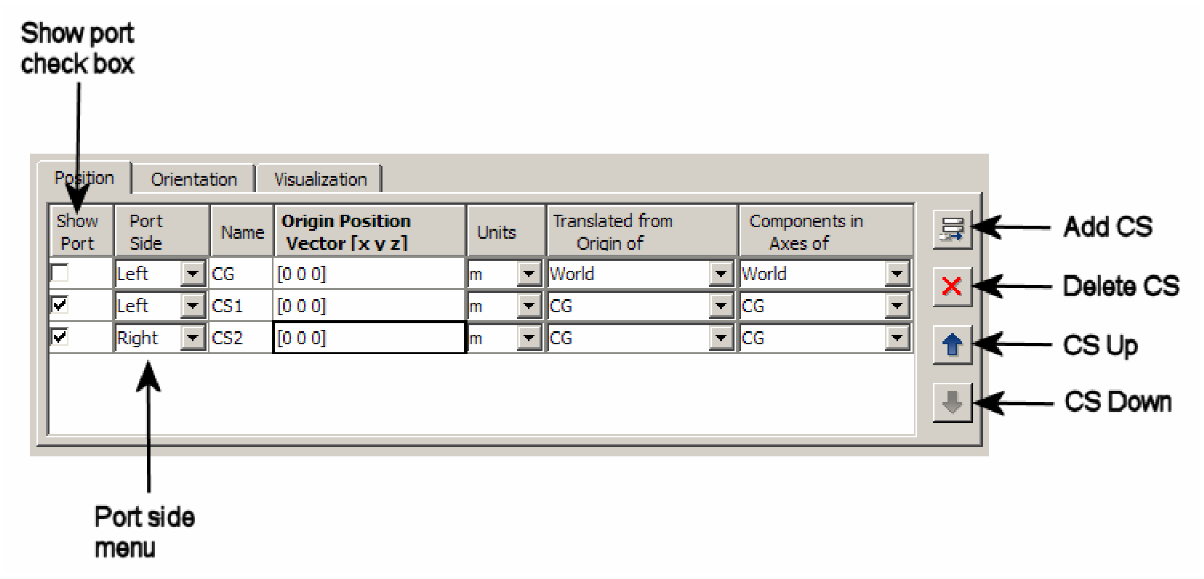
1. Highlight the Body CS whose position you want to change.
2. Click on the **Up** or **Down** button (see the following figure, Body Coordinate Systems Controls on page 2-24) until the Body CS is where you want it.

To delete a Body CS from the list:

1. Highlight the Body CS you want to delete.

   You cannot delete the Body's CG CS or the last one of the non-CG CSs.
2. Click on the **Delete** button (see the following figure, Body Coordinate Systems Controls on page 2-24).

   The Body CS you highlighted disappears.



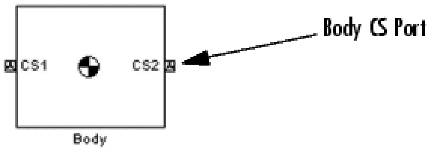
Show Port	Port Side	Name	Origin Position Vector [x y z]	Units	Translated from Origin of	Components in Axes of
☐	Left	CG	[0 0 0]	m	World	World
✓	Left	CS1	[0 0 0]	m	CG	CG
✓	Right	CS2	[0 0 0]	m	CG	CG

Body Coordinate Systems Controls

Managing Body CS Ports on a Body Block

Connecting a Joint, Constraint, Driver, Actuator, or Sensor block to a Body block requires an existing and configured Body CS on that Body:

- These other blocks define, constrain, impart, and measure the motion of bodies with respect to the origin and coordinate axes of Body CSs. Connect each of these blocks to a Body CS with a connection line.
- The actual connection line running from the other block to the Body block must be anchored to a displayed Body CS Port on the side of the Body block in the model window.



- A displayed Body CS Port on a Body block indicates a Body CS with the displayed name configured internally within the Body block.
- Not all the Body CSs configured inside a Body block need to be displayed, however.

  See the preceding figure, Body Coordinate Systems Controls.

Show Port

Select this check box for any Body CS to create a corresponding Body CS Port ⊞ on the side of the Body block. The Body CS on that line in the Body CS list is now accessible for connection to other blocks.

Clear this check box to remove the Body CS Port corresponding to that Body CS on that line in the list.

The defaults are not selected for `CG`, selected for `CS1` and `CS2`.

To apply your choices to the displayed Body block, click **Apply**.

Port Side

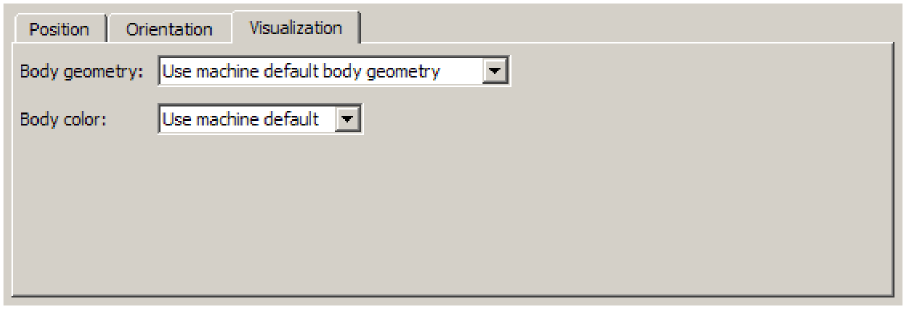
From the pull-down menu, choose which side of the Body block you want the Body CS Port for that Body CS to be placed on, `Left` or `Right`.

The defaults are `Left` for `CG` and `CS1` and `Right` for `CS2`.

To apply your choices to the displayed Body block, click **Apply**.

Visualization Properties

Default Settings



Body geometry: Use machine default body geometry

Body color: Use machine default

Visualization settings for a body consist of its

- Body geometry (surface shape)
- Color

The visualization window uses these settings to display the body. By default, a body inherits the visualization settings of the machine to which it is connected.

Customizing the Body Visualization Settings

You can change the visualization defaults individually with these menus.

Body geometry

From the pull-down menu, choose a body surface shape:

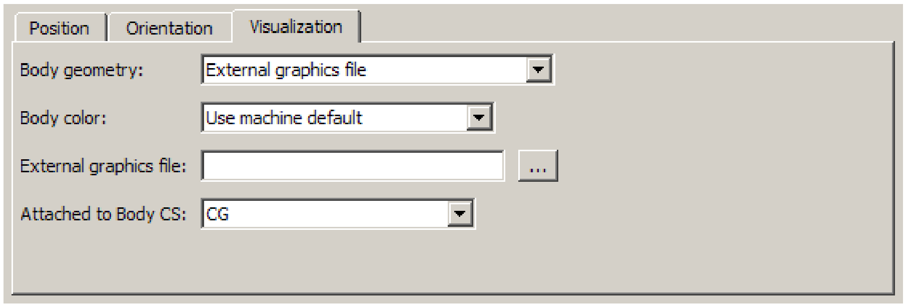
- `Use machine default body geometry` (default)
- `Convex hull from Body CS locations` for convex hulls
- `Equivalent ellipsoid from mass properties` for equivalent ellipsoids
- `External graphics file` (see "Specifying and Configuring an External Graphics File" on page 2-27)

Body color

From the pull-down menu, choose a body color:

- `Use machine default` (default)
- `Use color palette` (see "Specifying a Color from the Color Palette" on page 2-28)

Specifying and Configuring an External Graphics File



Body geometry: External graphics file
Body color: Use machine default
External graphics file:
Attached to Body CS: CG

If you choose `External graphics file` in the **Body geometry** pull-down menu, you must specify some additional information.

External graphics file

In the field, specify the graphics file.

You can search for files on your file system by clicking the browse **...** button.

Attached to Body CS

In the pull-down menu, specify which Body CS to attach the graphics to. Your Body CS list is specified by the **Position** tab.

This Body CS serves as the reference origin and coordinate axes for the body geometry. Geometric measurements in the graphics file are interpreted in the units associated with this Body CS.

Requirements for External Body Geometry Files

Custom body visualization requires a body geometry graphics file in the stereolithographic (STL) format. It supports both binary and ASCII types of the STL format.

See the visualization chapters of the SimMechanics Visualization and Import guide for a complete discussion of graphics files for specifying body geometries.

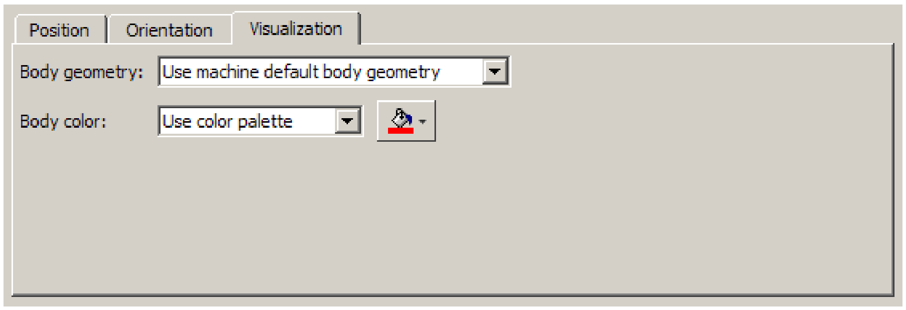
Caution

In order for custom visualization to work, this STL file must be either:

- On your MATLAB path.
- In your MATLAB present working folder.
- Specified with complete path in the Body dialog.

Otherwise, visualization reverts to the default body geometry.

Specifying a Color from the Color Palette



Body geometry: Use machine default body geometry

Body color: Use color palette

If you choose `Use color palette` in the **Body color** pull-down menu, the color palette button appears to the right of the menu. Click the color palette button to select a color for the body.

See Also

Body Actuator, Body Sensor, Ground, Machine Environment, Mechanical Branching Bar

See the relevant entries in the Glossary: adjoining CS, axis-angle rotation, body, Body CS, center of gravity (CG), convex hull, coordinate system (CS), equivalent ellipsoid, Euler angles, inertia tensor, mass, principal axes, principal inertial moments, quaternion, right-hand rule, and rotation matrix.

Setting Up and Configuring Bodies in Models

See “Modeling Grounds and Bodies”, and “Creating Body CS Ports” for more on setting up Bodies in machines.

See “Applying Motions and Forces” for setting general initial conditions (positions and velocities) of DoFs in a machine.

Body Actuator

Purpose

Time-dependent force and torque used to actuate a body

Library

Sensors & Actuators

Description

The Body Actuator block actuates a Body block with a generalized force signal, representing a force/torque applied to the body:

- Force for translational motion
- Torque for rotational motion

The generalized force is a function of time specified by a Simulink input signal. This signal can be any Simulink signal, including a signal feedback from a Sensor block.

The Body Actuator applies the actuation signal in the reference coordinate system (CS) specified in the block dialog.

The inport is the Simulink input signal. The output is the connector port you connect to the Body block you want to actuate.

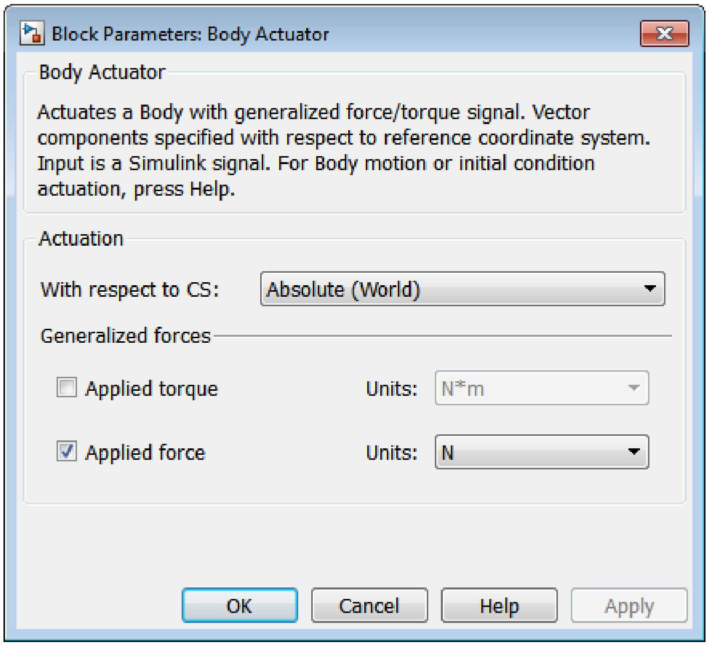
Tip Carefully distinguish the Body Actuator from the Driver blocks:

- The Body Actuator block applies generalized forces to one body in a specified reference CS.
- The Driver blocks drive relative degrees of freedom between pairs of bodies.

Other Ways to Actuate Bodies

The Body Actuator block actuates a Body with force/torque signals only. To actuate a Body with motion signals or initial conditions, or to drive the relative degrees of freedom between a pair of Bodies, see “Actuating a Joint” and “Joint Actuator Example: Body Driver”.

The `mech_body_driver` model from the Demos library shows how to drive the relative DoFs between a pair of bodies. To actuate one body alone, use this model and replace the second Body block with a Ground block. To set body initial conditions, replace the second Body block with a Ground block and the Joint Actuators with Joint Initial Condition Actuators.


Block Parameters: Body Actuator

Body Actuator

Actuates a Body with generalized force/torque signal. Vector components specified with respect to reference coordinate system. Input is a Simulink signal. For Body motion or initial condition actuation, press Help.

Actuation

With respect to CS: Absolute (World)

Generalized forces

[ ] Applied torque Units: N*m

[✓] Applied force Units: N

Dialog Box and Parameters

The dialog has one active area, **Actuation**.

Actuation

With respect to CS

In the pull-down menu, choose the coordinate system (CS) in which the actuating force/torque is interpreted: either the `Local (Body CS)` to which the Actuator is connected or the default `Absolute` (World).

Generalized Forces

You can apply a force, a torque, or both generalized forces to a body.

If you apply both, you need to bundle the torque and force vectors into a 6-component signal, in the order shown in the dialog.

Applied torque

Select the check box if part or all of the actuating signal is a rotational torque. The default is not selected. The Simulink torque input is a 3-component bundled signal.

In the **Units** pull-down menu, choose units for the actuating torque. The default is `N*m` (newton-meters).

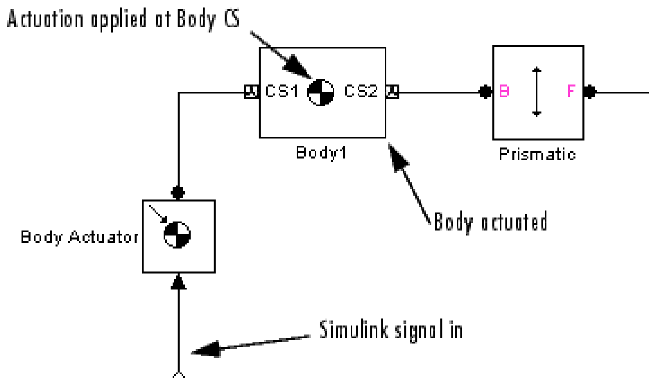
Applied force

Select the check box if part or all of the actuating signal is a translational force. The default is selected. The Simulink force input is a 3-component bundled signal.

In the **Units** pull-down menu, choose units for the actuating force. The default is `N` (newtons).

Example

Here is a Body Actuator connected to a Body:



You must connect the Body Actuator to the Body at one of that Body's attached Body CSs, at the corresponding Body CS Port. The actuation signal acts on the Body at that Body CS's origin.

See Also

Body, Body Sensor, Driver Actuator, Joint Actuator, Joint Initial Condition Actuator, Mechanical Branching Bar

See “Representations of Body Motion” for more details on Body coordinate system rotations.

See “Actuating a Joint” and “Joint Actuator Example: Body Driver”.

Body Sensor

Purpose

Body translation and rotation sensor

Library

Sensors & Actuators

Description

The Body Sensor block senses the motion of a body represented by a Body block. You connect the Body Sensor to a Body coordinate system (CS) on the Body whose motion you want to sense. The sensor specifically measures the motion of the origin of this Body CS.

The Body Sensor measures the components of translational and rotational motion in any combination of:

- Translational position, velocity, and acceleration vectors. The position vector has its tail at the World CS origin.
- Rotational orientation (a 3-by-3 rotation matrix R) and angular velocity and acceleration vectors

In the block dialog, you choose the reference coordinate system (CS) axes in which these components are represented.

The input is the connector port connected to the Body being sensed. The outport is a set of Simulink signals or one bundled Simulink signal of the selected matrix and/or vector components.

Coordinate Representations and Body Orientation

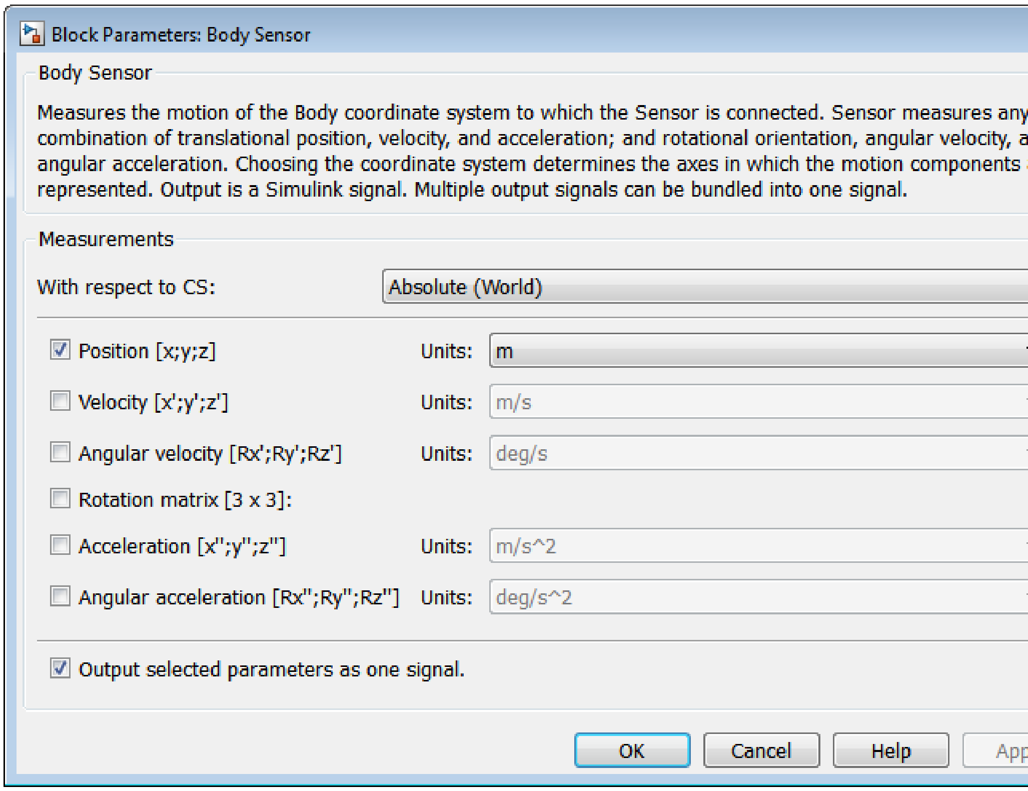
A body's orientation rotation matrix R relates the components of the same vector $\boldsymbol{v}$ as measured in the inertial World CS and in the Body CS by $\boldsymbol{v}_\mathrm{b} = R^\mathrm{T} \cdot \boldsymbol{v}_\mathrm{W}$. The column vector $\boldsymbol{v}_\mathrm{W}$ lists the vector $\boldsymbol{v}$'s three components measured in the World CS. The column vector $\boldsymbol{v}_\mathrm{b}$ lists the vector $\boldsymbol{v}$'s three components measured in the Body CS.

The columns of the rotation matrix R are the components of the Body CS unit basis vectors measured with respect to the World axes.

See "Representations of Body Motion" and "Representations of Body Orientation" for more details on representing body position and orientation, rotation matrices, and angular velocity.

Body Position-Orientation and the Home Configuration

The Body Sensor block can measure the position and/or orientation of a body. It measures these relative to the home configuration of the machine, the machine state *before* the application of initial condition actuators and assembly of disassembled joints. Thus the Body Sensor includes the effect of the latter, which act before the simulation starts.



Block Parameters: Body Sensor

Body Sensor

Measures the motion of the Body coordinate system to which the Sensor is connected. Sensor measures any combination of translational position, velocity, and acceleration; and rotational orientation, angular velocity, a angular acceleration. Choosing the coordinate system determines the axes in which the motion components represented. Output is a Simulink signal. Multiple output signals can be bundled into one signal.

Measurements

With respect to CS: Absolute (World)

- [✓] Position [x;y;z] Units: m
- [ ] Velocity [x';y';z'] Units: m/s
- [ ] Angular velocity [Rx';Ry';Rz'] Units: deg/s
- [ ] Rotation matrix [3 x 3]:
- [ ] Acceleration [x";y";z"] Units: m/s^2
- [ ] Angular acceleration [Rx";Ry";Rz"] Units: deg/s^2
- [✓] Output selected parameters as one signal.

Dialog Box and Parameters

The dialog has one active area, **Measurements**.

Measurements

With respect to CS

In the pull-down menu, choose the coordinate system in which the body motion components are represented: either the `Local (Body CS)` to which the Sensor is connected or the default `Absolute (World)`.

In the `Absolute` case, the rotation matrix R and the motion vectors have components represented in the inertial World CS axes. In the `Local` case, the same body motion components are premultiplied by the body's inverse orientation rotation matrix $R^{-1} = R^T$.

Each vector measurement is a row vector in the Simulink output signal. The selected signals are ordered in the same sequence as the dialog.

Select the check box for each of the possible measurements you want to make:

- Translational motion: **Position**, **Velocity**, and **Acceleration** vectors: $\boldsymbol{r}$, $\boldsymbol{v} = \mathrm{d}\boldsymbol{r}/\mathrm{dt}$, and $\boldsymbol{a} = \mathrm{d}\boldsymbol{v}/\mathrm{dt}$, respectively.
- Rotational motion: **Angular velocity** and **Angular acceleration** vectors and **Rotation matrix**:
  - The **Rotation matrix** is the 3-by-3 orthogonal rotation matrix R:

$$\begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}$$

    representing rotational orientation and satisfying $R^TR = RR^T = I$. The components are output columnwise as a 9-component row vector: $(R_{11}, R_{21}, R_{31}, R_{12}, \ldots)$.
  - If you choose the **With respect to coordinate system** as `Absolute (World)`, the **Rotation matrix** measures the body's rotational orientation with respect to the World CS. Recall

the relationship of vector components in the World and body coordinate axes, $\boldsymbol{v}_W = R \cdot \boldsymbol{v}_b$.

- If you choose the **With respect to coordinate system** as `Local (Body CS)`, the **Rotation matrix** returns the 3-by-3 identity matrix $R^T R = I$.
- The angular velocity is $\omega_j = (1/2)\Sigma_{ik}\ \varepsilon_{ijk}\Omega_{ik}$, where the matrix $\Omega = +(dR/dt)*R^T = -R*(dR^T/dt)$, and ε is the permutation symbol. The angular acceleration is $\alpha = d\omega/dt$.

In the **Units** pull-down menus, choose the units for each of the measurements you want:

- Translation: the defaults are `m` (meters), `m/s` (meters/second), and `m/s`^{2} (meters/second^2), respectively, for **Position**, **Velocity**, and **Acceleration**.
- Rotation: the defaults are `deg/s` (degrees/second) and `deg/s`^{2} (degrees/second^2), respectively, for **Angular velocity** and **Angular acceleration**. The **Rotation matrix** is dimensionless.

Output selected parameters as one signal

Select this check box to convert the output signals into a single bundled signal. The default is selected. If you clear it, the Body Sensor block will grow as many Simulink outports as there are active signals selected, one port for each selected signal.

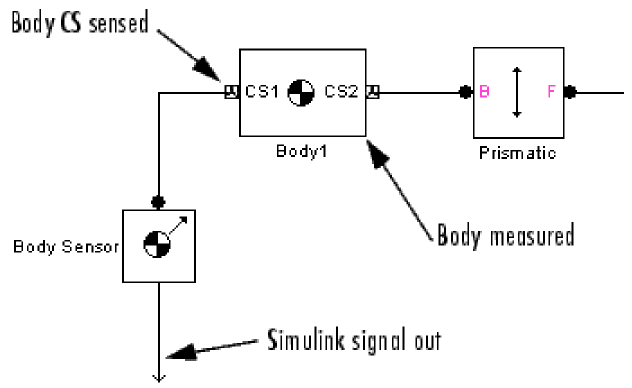
If the check box is selected, the Simulink signal out has all the active (selected) signals ordered into a single row vector, in the same order you see in the dialog. Nonselected components are removed from the vector signal.

The sensor outputs are ordered and labeled as follows.

Body Sensor Output Signal	Label
Position	p
Velocity	v
Angular velocity	av
Rotation matrix	[R]
Acceleration	a
Angular velocity	aa

Example

Here is a Body Sensor connected to a Body:



You must connect the Body Sensor to the Body at one of its Body CS ports. The Sensor measures the motion of that Body CS.

See Also

Body, Body Actuator, Constraint & Driver Sensor, Joint Sensor, Mechanical Branching Bar

See “Kinematics and Machine Motion State”, “Representations of Body Motion”, and “Representations of Body Orientation” for more details on representing body position and orientation.

See “Sensing Motions and Forces”.

See the relevant entries in the Glossary about body orientation: axis-angle rotation, Euler angles, right-hand rule, and rotation matrix.

Body Spring & Damper

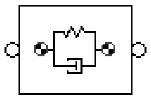
Purpose

Damped linear oscillator force between two bodies

Library

Force Elements

Description



The Body Spring & Damper block models the force of a damped spring acting between two bodies. By Newton's third law, the spring applies equal and opposite forces to the two bodies. You can use this Force Element block to model any linear (Hooke's law) force with constant coefficients that acts between a pair of bodies.

You connect a Body Spring & Damper between two Body coordinate systems (CSs), each on one body. The vector between the Body CSs defines the direction and length of the spring. One of the Bodies can be a Ground.

Caution

The spring and the damper forces act *only* along the axis connecting the two Body CSs.

The Body Spring & Damper has no degrees of freedom (DoFs).

Grounding the Connected Submachines

The Body Spring & Damper block contains a Shared Environment block. The submachines connected to either side of this block constitute a single composite machine that requires exactly one Machine Environment block, but at least one Ground for *each* submachine.

Referencing Coordinate Systems on the Connected Bodies

The Body Spring & Damper block is not a Joint and cannot propagate adjoining coordinate systems from a Body on one side to a Body on the other side.

One Body is connected to one side of the Body Spring & Damper at one of that Body's CSs. If you attempt to define that CS in terms of the adjoining CS (the connected CS of the other Body connected to the other side), the first Body cannot detect the connected CS of the second body.

If you need to define adjoining CSs on either side of a Body Spring & Damper, add a Joint block in parallel with the spring-damper.

Adding Joints in Parallel to the Body Spring & Damper

To represent the DoFs of one body with respect to the other, either

- Connect one or more Joints in series with the Bodies.
- Create additional Body CSs on each body and connect them with a Joint in parallel with the Body Spring & Damper. To create parallel grounds, insert additional Ground blocks.

  You can add more Joint blocks between the Bodies to represent one, two, or three prismatic primitives. Use Prismatic blocks or a Custom Joint block to accomplish this.

Body Spring and Damper Force Law

You connect this block to each Body, *A* or *B*, at a Body coordinate system (CS). If $\boldsymbol{r}_A$ and $\boldsymbol{r}_B$ are the positions of these Body CSs, the relative position vector connecting them is $\boldsymbol{r} = \boldsymbol{r}_B - \boldsymbol{r}_A$. The distance of separation is $|\boldsymbol{r}|$. The relative velocity is $\boldsymbol{v} = d\boldsymbol{r}/dt$. Then the vector force that body *A* exerts on body *B* is

$$\boldsymbol{F} = -k(|\boldsymbol{r}| - r_0)(\boldsymbol{r}/|\boldsymbol{r}|) - b(\boldsymbol{v}\cdot\boldsymbol{r})(\boldsymbol{r}/|\boldsymbol{r}|^2)$$

The first term represents the spring or linear displacement force. The second represents the damper or velocity dissipation force, which acts only along the direction of $\boldsymbol{r}$. Thus the damper is equivalent to a dashpot, not a viscous medium.

You specify

- The spring constant k. A stable spring requires $k > 0$.
- The natural spring length (offset) r_0. The natural length is the length of the spring with no forces acting on it and physically must be nonnegative: $r_0 \geq 0$.

- The damping constant b. A damping representing dissipation and respecting the second law of thermodynamics requires $b \geq 0$. You can use a negative b to represent energy pumping.

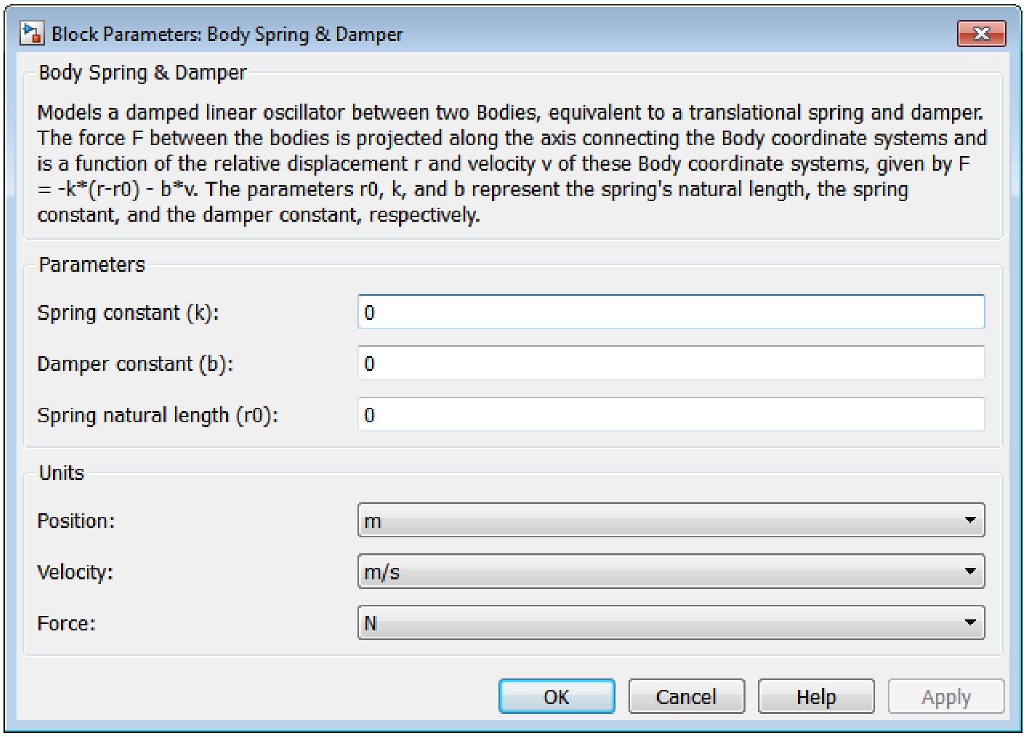
Body Spring and Damper Force in Singular Cases

Caution In certain cases, the force formula breaks down, and the block uses special-case rules to determine the spring-damper force.

To avoid singularities in the initial state of motion, be sure to set the bodies' initial conditions of position and velocity to physically sensible values.

Singular cases include the following:

- If both r_0 and $\boldsymbol{v} \neq 0$, and $\boldsymbol{r} = 0$ at some instant, both terms in the force become singular. The spring force is reprojected along the velocity vector. That is, $\boldsymbol{v}/|\boldsymbol{v}|$ replaces $\boldsymbol{r}/|\boldsymbol{r}|$ in both terms of the force law, once in the first term and twice in the second. If the state $\boldsymbol{r} = 0$ does not persist for more than an instant, this replacement has no effect on the motion.
- If $r_0 \neq 0$, and both $\boldsymbol{r}$ and $\boldsymbol{v} = 0$ at some instant, the force direction is undefined. The simulation stops with an error.


Block Parameters: Body Spring & Damper

Body Spring & Damper

Models a damped linear oscillator between two Bodies, equivalent to a translational spring and damper. The force F between the bodies is projected along the axis connecting the Body coordinate systems and is a function of the relative displacement r and velocity v of these Body coordinate systems, given by F = -k*(r-r0) - b*v. The parameters r0, k, and b represent the spring's natural length, the spring constant, and the damper constant, respectively.

Parameters

Spring constant (k): 0

Damper constant (b): 0

Spring natural length (r0): 0

Units

Position: m

Velocity: m/s

Force: N

Dialog Box and Parameters

The dialog has two active areas, **Parameters** and **Units**.

Tunable Parameters

All of this block's fields are tunable during simulation. See the Simulink documentation for more information about tunable parameters.

Parameters

Spring constant (k)

Enter the linear spring force constant *k*. The default is `0`.

The units for *k* are derived implicitly from your choice of position and force units.

Damper constant (b)

Enter the linear damping force constant b. The default is `0`.

The units for b are derived implicitly from your choice of velocity and force units.

Spring natural length (r0)

Enter the spring's natural length (offset) r_0. The default is `0`.

Units

Position

In the pull-down menu, choose units for the relative position vector $\boldsymbol{r}$. The default is `m` (meters).

Velocity

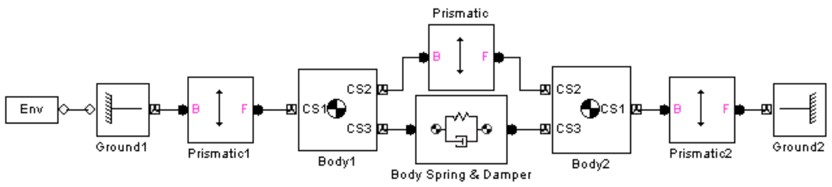
In the pull-down menu, choose units for the relative velocity vector $\boldsymbol{v}$. The default is `m/s` (meters/second).

Force

In the pull-down menu, choose units for the spring-damper force $\boldsymbol{F}$ acting between the bodies. The default is `N` (newtons).

Example

This is a simple but representative use of the Body Spring & Damper.



See Also

Body, Body Actuator, Body Sensor, Custom Joint, Ground, Joint Spring & Damper, Machine Environment, Prismatic, Shared Environment

See "Adding Internal Forces".

Bushing

Purpose

Joint with three revolute and three prismatic joint primitives

Library

Joints

Description

The Bushing block represents a composite joint with three translational degrees of freedom (DoFs) as three prismatic primitives and three rotational DoFs as three revolute primitives. There are no constraints among the primitives. Unlike Six-DoF, Bushing represents the rotational DoFs as three revolutes, rather than as one spherical.

Warning

A joint with three revolute primitives becomes singular if two or three of the rotation axes become parallel ("gimbal lock"). A joint with two or three prismatic primitives becomes singular if two or three of the translation axes become parallel. The simulation stops with errors in these cases.

A joint with three revolute primitives must be configured in the initial state with the three revolute primitive axes mutually orthogonal. There are no restrictions on the primitive axes once the simulation starts, except to prevent any two of the primitive axes from becoming parallel.

Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

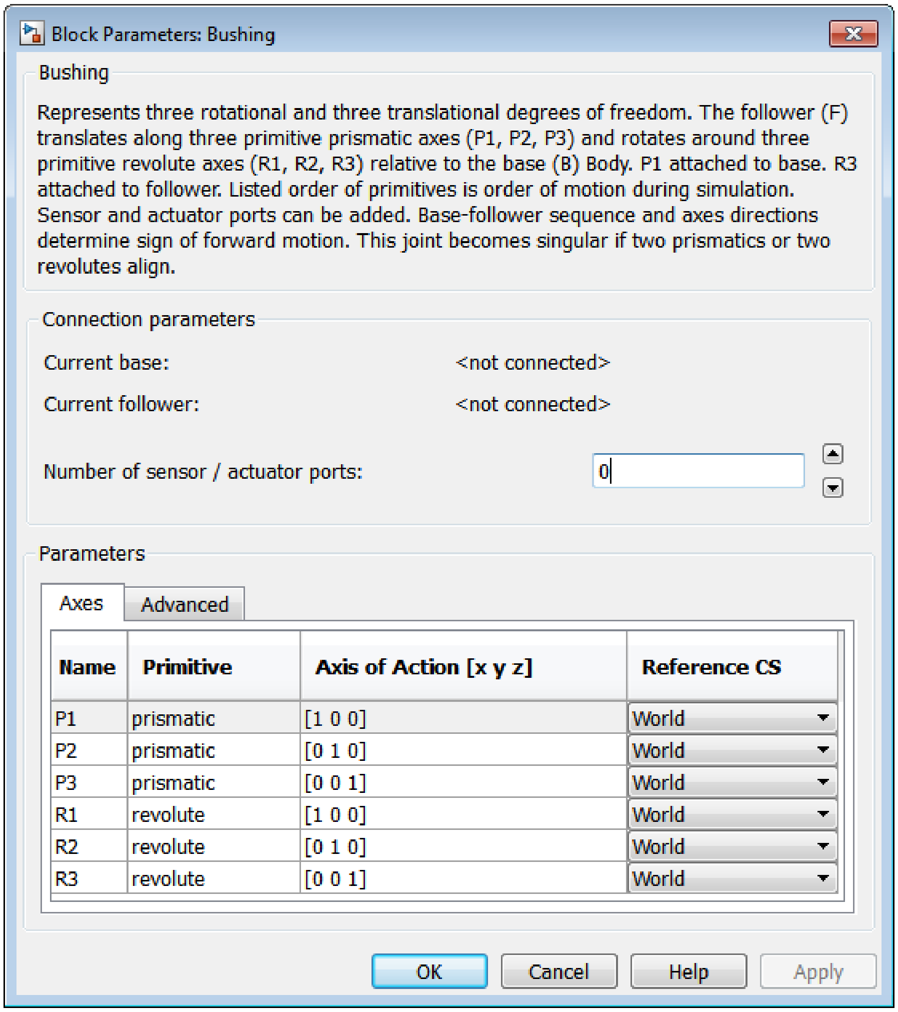
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space



Block Parameters: Bushing

Bushing

Represents three rotational and three translational degrees of freedom. The follower (F) translates along three primitive prismatic axes (P1, P2, P3) and rotates around three primitive revolute axes (R1, R2, R3) relative to the base (B) Body. P1 attached to base. R3 attached to follower. Listed order of primitives is order of motion during simulation. Sensor and actuator ports can be added. Base-follower sequence and axes directions determine sign of forward motion. This joint becomes singular if two prismatics or two revolutes align.

Connection parameters

Current base: <not connected>

Current follower: <not connected>

Number of sensor / actuator ports: 0

Parameters

Axes | Advanced

Name	Primitive	Axis of Action [x y z]	Reference CS
P1	prismatic	[1 0 0]	World
P2	prismatic	[0 1 0]	World
P3	prismatic	[0 0 1]	World
R1	revolute	[1 0 0]	World
R2	revolute	[0 1 0]	World
R3	revolute	[0 0 1]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive translation is the follower moving in the direction of the translation axis. Positive rotation is the follower moving around the rotational axis following the right-hand rule.

Current base

When you connect the base (B) connector port on the Bushing block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Bushing Base and Follower Body Connector Ports on page 2-48.

The base Body is automatically connected to the first joint primitive `P1` in the primitive list in **Parameters**.

Current follower

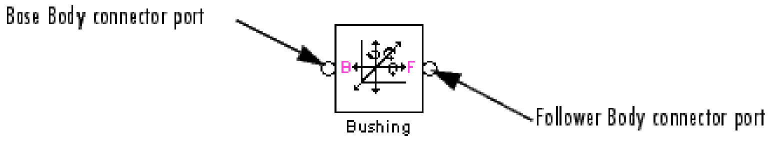
When you connect the follower (F) connector port on the Bushing block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Bushing Base and Follower Body Connector Ports on page 2-48.

The follower Body is automatically connected to the last joint primitive `R3` in the primitive list in **Parameters**.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`.

The motions of prismatic and revolute primitives are specified in linear and angular units, respectively.



Bushing Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. Each DoF primitive in Bushing has an entry line. These lines specify the direction of the axes of action of the DoFs that the Bushing represents.

Name - Primitive
The primitive list states the names and types of joint primitives that make up the Bushing block: prismatic primitives `P1`, `P2`, `P3`, and revolute primitives `R1`, `R2`, `R3`.

Axis of Action [x y z]
Enter here as a three-component vector the directional axes defining the allowed motions of these primitives and their corresponding DoFs:

- Prismatic: axis of translation
- Revolute: axis of rotation

The default vectors are shown in the dialog above. The axis is a directed vector whose overall sign matters.

To prevent singularities and simulation errors, no two of the revolute axes and no two of the prismatic axes can be parallel.

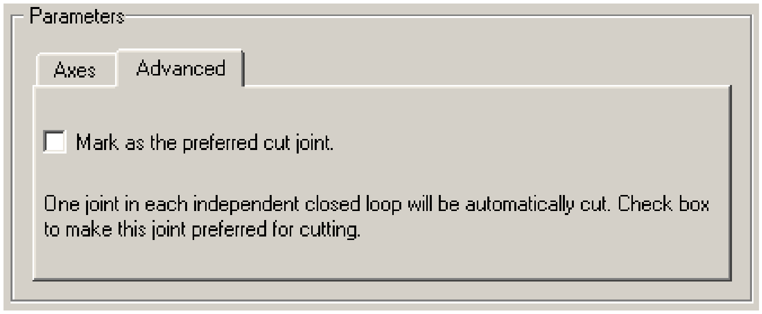
Reference CS
Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of action is oriented with respect to. This CS also determines the absolute meaning of forces/torques and motion along/about the joint axis. The default is `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Bearing, Cylindrical, Gimbal, Prismatic, Revolute, Six-DoF

See "Modeling Degrees of Freedom" for more on representing DoFs with Joints.

See "Checking Model Topology" and "How SimMechanics Software Works" for more on closed loops and cutting.

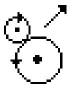
Purpose

Sensor used to measure the reaction force and torque between two constrained or driven bodies

Library

Sensors & Actuators

Description



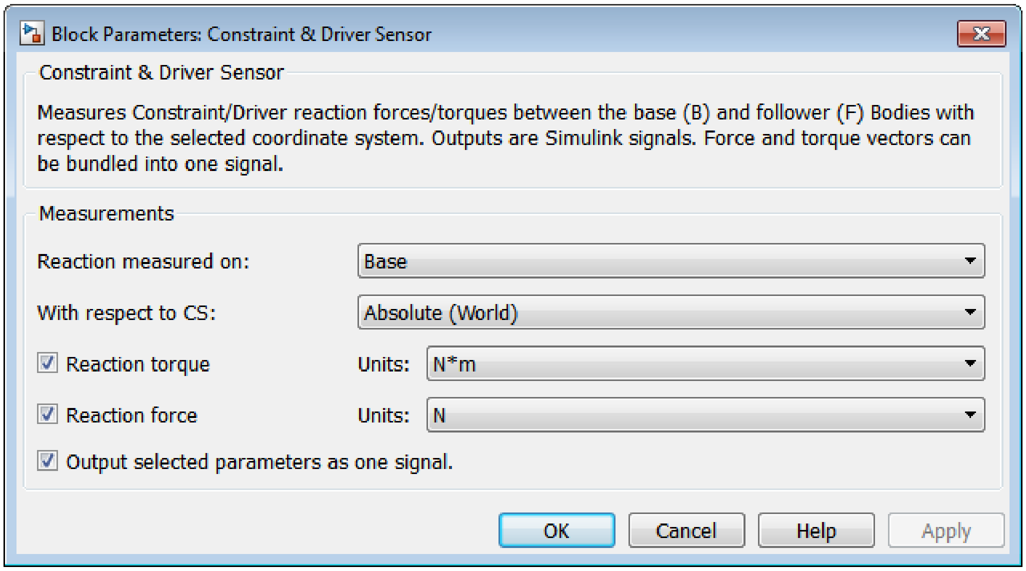
The Constraint & Driver Sensor block measures the force/torque of constraint (reaction force/torque) between a pair of bodies. You connect this block to the Constraint or Driver block connected between the two Bodies. The output signal is the reaction force/torque.

The Constraint & Driver Sensor measures the reaction force/torque in the reference coordinate system (CS) specified in the block dialog. The Constraint or Driver block connects a base and a follower Body. You choose in the dialog to measure the reaction force/torque on either the base or the follower Body.

The input is the connector port connected to the Constraint or Driver block you want to sense. The outport is a set of Simulink signals or one bundled Simulink signal of the reaction force/torque vector(s).

Physical and Unphysical Reaction Forces Not all the components of the output reaction force/torque signal are significant. Only those components projected into the subspace of the degrees of freedom constrained or driven by the connected Constraint or Driver block are physical. Components orthogonal to the constrained or driven degrees of freedom are unphysical.

A body's orientation rotation matrix R relates vector components measured in the body CS and in the inertial World CS by $[R] \cdot \boldsymbol{v}_b = \boldsymbol{v}_s$. The column vector $\boldsymbol{v}_b$ lists the vector $\boldsymbol{v}$'s three components measured in the body CS. The column vector $\boldsymbol{v}_s$ lists the vector $\boldsymbol{v}$'s three components measured in the World CS.



Block Parameters: Constraint & Driver Sensor

Constraint & Driver Sensor

Measures Constraint/Driver reaction forces/torques between the base (B) and follower (F) Bodies with respect to the selected coordinate system. Outputs are Simulink signals. Force and torque vectors can be bundled into one signal.

Measurements

Reaction measured on: Base

With respect to CS: Absolute (World)

☑ Reaction torque — Units: N*m

☑ Reaction force — Units: N

☑ Output selected parameters as one signal.

Dialog Box and Parameters

The dialog has one active area, **Measurements**.

Measurements

Reactions measured on

In the pull-down menu, choose to measure the reaction force/torque on the base (B) or follower (F) Body. The default is `Base`.

With respect to CS

In the pull-down menu, choose the CS in which the reaction force/torque or motion is interpreted. The default is `Absolute (World)`.

In the `Absolute` case, the force vectors have components measured relative to the inertial World CS axes. In the `Local` case, the same force vector signals are premultiplied by the inverse rotation matrix $R^{-1} = R^{T}$ for the Body selected in **Reactions measured on**.

Reaction torque

Select the check box if you want to measure the reaction torque. The default is selected. The torque is a row vector in the Simulink output signal.

In the pull-down menu, choose the units for the reaction torque. The default is `N*m` (newton-meters).

Reaction force

Select the check box if you want to measure the reaction force. The default is selected. The force is a row vector in the Simulink output signal.

In the pull-down menu, choose the units for the reaction force. The default is `N` (newtons).

Output selected parameters as one signal

Select this check box to convert the output signals into a single bundled signal. The default is selected. If you clear it, the Constraint & Driver Sensor block will grow as many Simulink outports as there are active signals selected, one port for each selected signal.

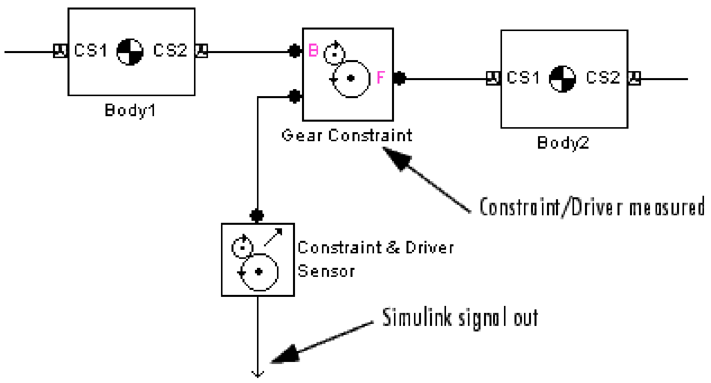
If the check box is selected, the Simulink signal out has all the active signals bundled into a single row vector, ordered in the order shown in the dialog. The type of the signal components depends on which measurements are active (selected).

The sensor outputs are ordered and labeled as follows.

Constraint & Driver Sensor Output Signal	Label
Reaction torque	Tr
Reaction force	Fr

Example

Here is a Constraint & Driver Sensor connected to a Gear Constraint, which connects and constraints two Bodies:



You must add a Sensor port (connector port) to the Constraint/Driver block to connect the Constraint & Driver Sensor to it. The base (B)-follower (F) Body sequence on the two sides of the Joint determines the sense of the Constraint & Driver Sensor data.

See Also

Body Sensor, Driver Actuator, Joint Sensor, Mechanical Branching Bar

See "Representations of Body Motion" and "Sensing Motions and Forces".

Continuous Angle

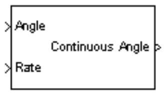
Purpose

Utility that converts a discontinuous bounded angle into a continuous unbounded angle

Library

Utilities

Description



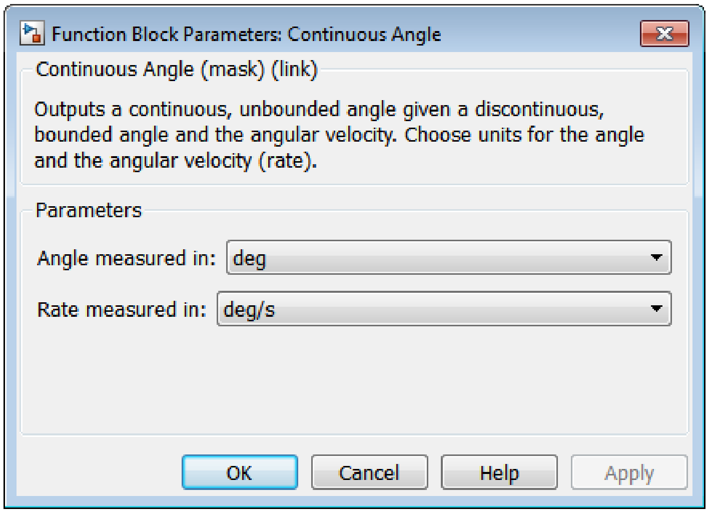
The Continuous Angle block converts a measured angle signal restricted to the semiopen interval (-180^o, +180^o] degrees or (-π,+π] radians to a continuous, unbounded angle not restricted to any interval. This block requires the angle and the angular velocity as input signals. The continuous, unbounded angle is the output signal.

Caution

Each Continuous Angle block in a model adds a normal Simulink state to the model. Use this block with caution if you are trimming or linearizing your model.

The Continuous Angle block does not add any mechanical states to your model.

The Joint Sensor block outputs the absolute rotational measurement of revolute motion as a bounded angle in the interval (-180^o, +180^o] degrees or (-π,+π] radians. Motion that crosses the boundaries of this interval causes discontinuities in the measured angle, from +180^o to -180^o or vice versa. Use the Continuous Angle block if you want to convert this restricted angular measurement to an unbounded measurement.


Function Block Parameters: Continuous Angle

Continuous Angle (mask) (link)

Outputs a continuous, unbounded angle given a discontinuous, bounded angle and the angular velocity. Choose units for the angle and the angular velocity (rate).

Parameters

Angle measured in: deg

Rate measured in: deg/s

Dialog Box and Parameters

The dialog has one active area, **Parameters**.

Parameters

Angle measured in

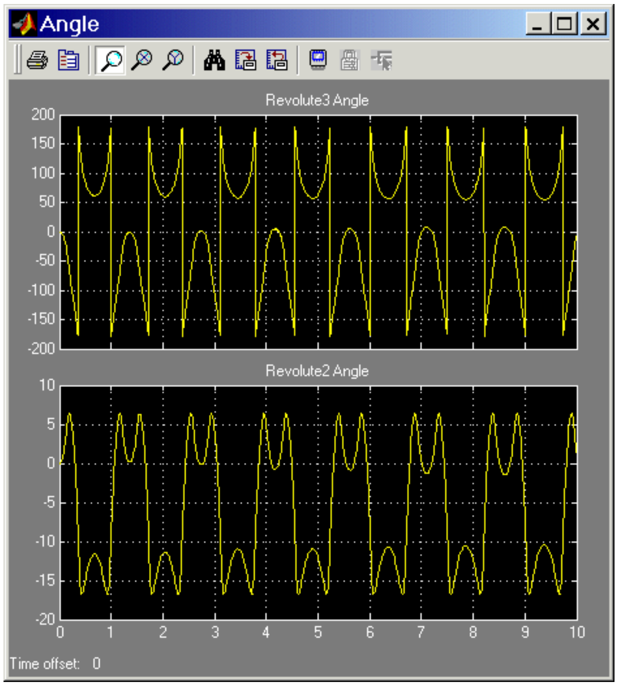
Choose the units for the input angle and the output continuous angle, either `deg` (degrees) or `rad` (radians). The default is `deg`.

Rate measured in

Choose the units for the input rate (angular velocity), either `deg/s` (degrees/second) or `rad/s` (radians/second). The default is `deg/s`

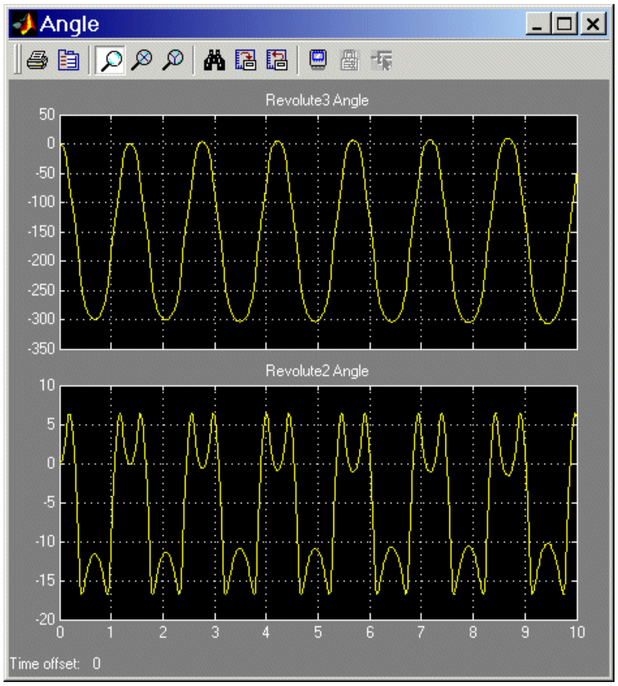
Example

The tutorial "Model and Simulate a Closed-Loop Machine" produces this angular motion output for the Revolute3 and Revolute 2 joints:



The Revolute3 angle is restricted to the interval (-180°, +180°], so values passing either limit of this interval are mapped to the opposite end of the interval. The Revolute2 angle is not restricted, but instead touches genuine turning points in its motion.

After passing the angles and angular velocities through Continuous Angle blocks, the Revolute3 angular motion appears different:



Revolute3's motion is unchanged, but its angle is now continuous, with no interval restriction. Revolute2's angle is unchanged.

See Also

Joint Sensor

See "Trimming Mechanical Models" and "Linearizing Mechanical Models" for more information about states.

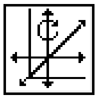
Purpose

Joint with custom combination of prismatic, revolute, and spherical joint primitives

Library

Joints

Description



The Custom Joint block is a composite joint that you can customize with a specified combination of primitives (prismatic, revolute, or spherical) representing the most general and unconstrained degrees of freedom (DoFs) in three dimensions:

- Up to three translational DoFs as three prismatic primitives
- Up to three rotational DoFs:
  - As a single spherical primitive
  - As one, two, or three revolute primitives

  The sense of rotational DoFs is defined by the right-hand rule. One spherical or three revolutes together form a right-handed system.

You can add, configure, and delete these primitives from the Custom Joint, with a minimum and default of one primitive. The properties of each primitive are the same as the individual Joints of the same names.

Warning

A joint with two or three revolute primitives becomes singular if two or three of the rotation axes become parallel ("gimbal lock"). A joint with two or three prismatic primitives becomes singular if two or three of the translation axes become parallel. The simulation stops with errors in these cases.

A joint with three revolute primitives must be configured in the initial state with the three revolute primitive axes mutually orthogonal. There are no restrictions on the primitive axes once the simulation starts, except to prevent any two of the primitive axes from becoming parallel.

You can connect Actuator and Sensor blocks to a Custom Joint, with each Actuator and Sensor connecting to an individual primitive. You cannot connect an Actuator to a spherical primitive.

Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

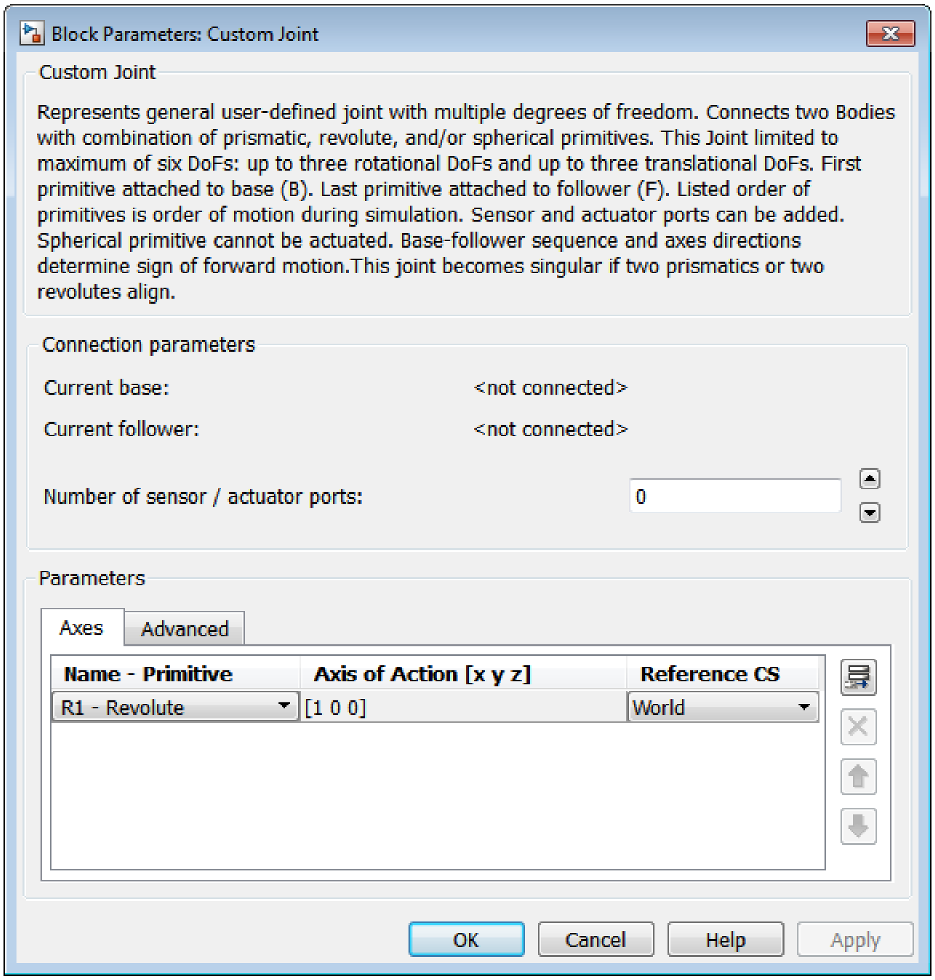
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	**Span of Primitive Axes**
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space



Block Parameters: Custom Joint

Custom Joint

Represents general user-defined joint with multiple degrees of freedom. Connects two Bodies with combination of prismatic, revolute, and/or spherical primitives. This Joint limited to maximum of six DoFs: up to three rotational DoFs and up to three translational DoFs. First primitive attached to base (B). Last primitive attached to follower (F). Listed order of primitives is order of motion during simulation. Sensor and actuator ports can be added. Spherical primitive cannot be actuated. Base-follower sequence and axes directions determine sign of forward motion.This joint becomes singular if two prismatics or two revolutes align.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>
Number of sensor / actuator ports:	0

Parameters

Axes | Advanced

Name - Primitive	Axis of Action [x y z]	Reference CS
R1 - Revolute	[1 0 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion:

- Positive translation is the follower moving in the direction of the translation axis.
- Positive rotation is the follower rotating in the right-hand rule about the rotation axis.
- Positive rotation is the follower rotating in the right-hand rule as shown by the motion figure in the Spherical block reference page.

Current base

When you connect the base (B) connector port on the Custom Joint block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Custom Joint Base and Follower Body Connector Ports on page 2-63.

The base Body is automatically connected to the first joint primitive in the primitive list in **Parameters**.

Current follower

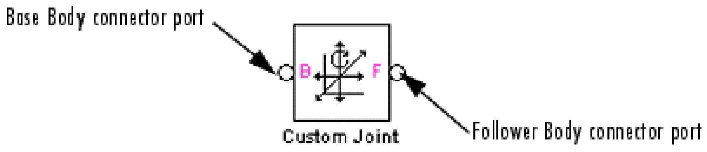
When you connect the follower (F) connector port on the Custom Joint block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Custom Joint Base and Follower Body Connector Ports on page 2-63.

The follower Body is automatically connected to the last joint primitive in the primitive list in **Parameters**.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`. A spherical primitive cannot be connected to an Actuator.

The motion of a prismatic primitive is specified in linear units. The motion of a revolute primitive is specified in angular units. The motion of a spherical primitive is three DoFs specified in quaternion form.



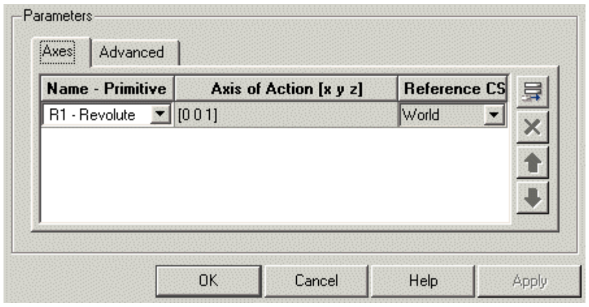
Custom Joint Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. Each DoF primitive in Custom Joint has an entry line. These lines specify the direction of the axes of action of the DoFs that the Custom Joint represents.



Name - Primitive	Axis of Action [x y z]	Reference CS
R1 - Revolute	[0 0 1]	World

Name - Primitive

In the pull-down menu, select a label and primitive type for this DoF. Up to three prismatic primitives P1, P2, P3 are allowed. The rotational DoFs are represented by up to three revolute primitives: R1, R2, R3. A spherical primitive S can take all three allowed rotational DoFs instead.

The default value is `R1 - Revolute`.

Axis of Action [x y z]

Enter here as a three-component vector the directional axis defining the allowed motion of this primitive and its corresponding DoF:

- Prismatic: axis of translation
- Revolute: axis of rotation
- Spherical: field is not active

The default vector is `[0 0 1]`. The axis is a directed vector whose overall sign matters.

To prevent singularities and simulation errors, no two of the revolute axes and no two of the prismatic axes can be parallel.

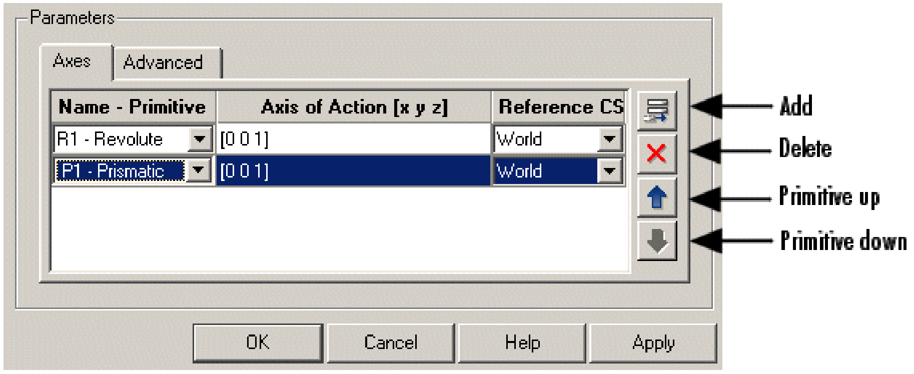
Reference CS

Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of action is oriented with respect to. This CS also determines the absolute meaning of forces/torques and motion along/about the joint axis. The default is `World`.

The field is not active for a spherical primitive.

Managing the Joint Primitives List in a Custom Joint

The Custom Joint primitives list controls, in the figure, allow you to add, reorder, and delete joint primitives in a Custom Joint block.



Name - Primitive	Axis of Action [x y z]	Reference CS
R1 - Revolute	[0 0 1]	World
P1 - Prismatic	[0 0 1]	World

Custom Joint Primitives List Controls

- To add a joint primitive to the primitives list:
  - Highlight an existing primitive name in the list.
  - Click on the **Add** button (see the preceding figure, Custom Joint Primitives List Controls on page 2-65).

    A new primitive will appear immediately below the primitive you highlighted.
- To change the position of a joint primitive in the list:
  - Highlight the primitive whose position you want to change.
  - Click on the **Up** or **Down** button (see the preceding figure, Custom Joint Primitives List Controls on page 2-65) until the primitive is where you want it.
- To delete a joint primitive from the list:
  - Highlight the primitive you want to delete.
  - Click on the **Delete** button (see the preceding figure, Custom Joint Primitives List Controls on page 2-65).

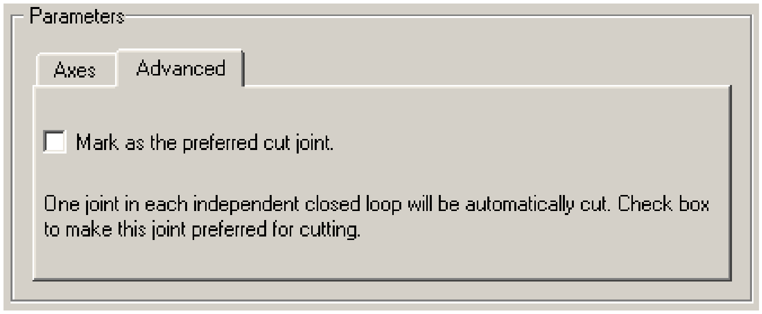
    The primitive you highlighted disappears.
- Custom Joint requires at least one primitive, which you cannot delete.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Bushing, Gimbal, Joint Actuator, Joint Initial Condition Actuator, Joint Sensor, Joint Stiction Actuator, Prismatic, Revolute, Six-DoF, Spherical

See “Modeling Degrees of Freedom” for more on representing DoFs with Joints.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on closed loops and cutting.

Purpose

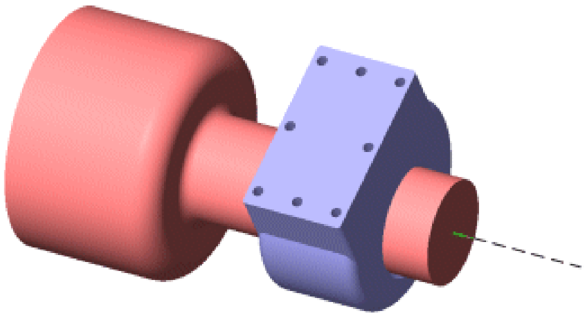
Joint with one revolute and one prismatic joint primitives

Library

Joints

Description

The Cylindrical block represents a composite joint with one translational degrees of freedom (DoF) as one prismatic primitive and one rotational DoF as one revolute primitive. The translation and rotation axes must be parallel.



Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

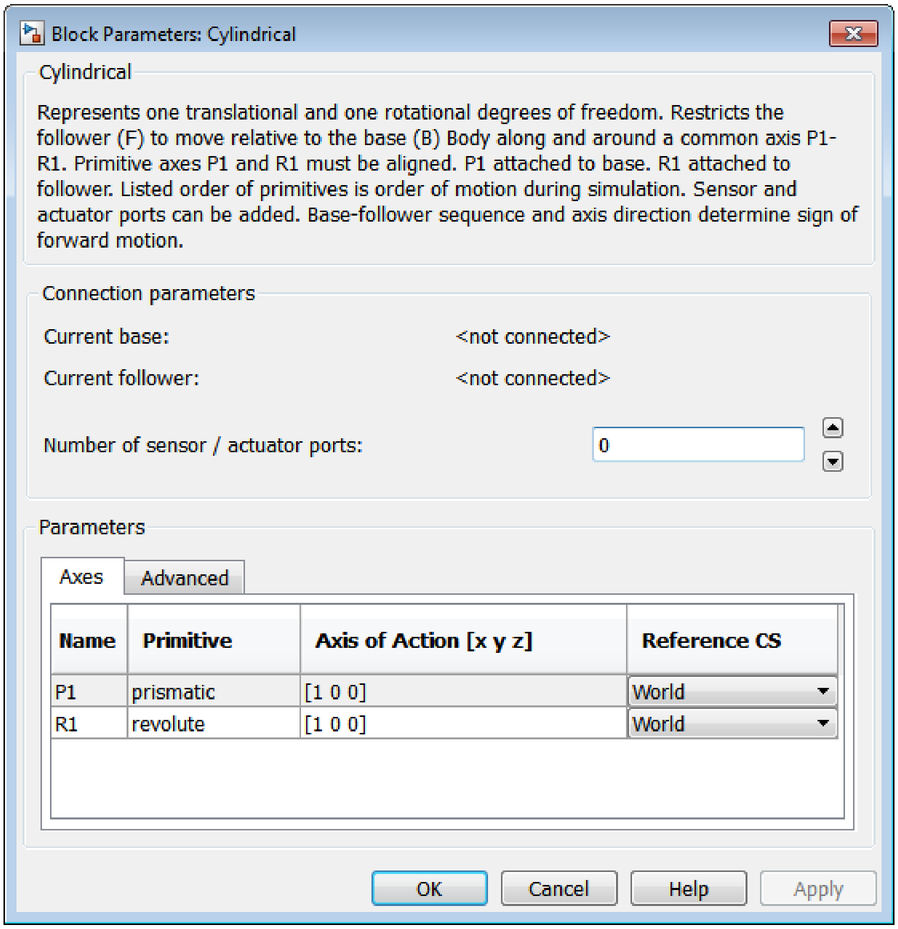
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space



Block Parameters: Cylindrical

Cylindrical

Represents one translational and one rotational degrees of freedom. Restricts the follower (F) to move relative to the base (B) Body along and around a common axis P1-R1. Primitive axes P1 and R1 must be aligned. P1 attached to base. R1 attached to follower. Listed order of primitives is order of motion during simulation. Sensor and actuator ports can be added. Base-follower sequence and axis direction determine sign of forward motion.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>
Number of sensor / actuator ports:	0

Parameters

Axes | Advanced

Name	Primitive	Axis of Action [x y z]	Reference CS
P1	prismatic	[1 0 0]	World
R1	revolute	[1 0 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive translation is the follower moving in the direction of the translation axis. Positive rotation is the follower moving around the rotational axis following the right-hand rule.

Current base

When you connect the base (B) connector port on the Cylindrical block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Cylindrical Base and Follower Body Connector Ports on page 2-70.

The base Body is automatically connected to the first joint primitive `P1` in the primitive list in **Parameters**.

Current follower

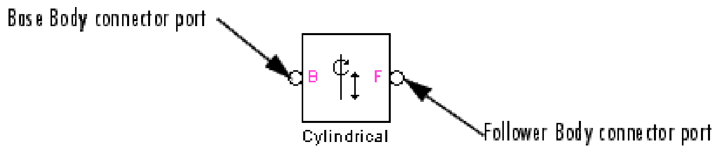
When you connect the follower (F) connector port on the Cylindrical block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Cylindrical Base and Follower Body Connector Ports on page 2-70.

The follower Body is automatically connected to the last joint primitive `R1` in the primitive list in **Parameters**.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`.

The motions of prismatic and revolute primitives are specified in linear and angular units, respectively.



Cylindrical Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. Each DoF primitive in Cylindrical has an entry line. These lines specify the direction of the axes of action of the DoFs that the Cylindrical represents.

Name - Primitive

The primitive list states the names and types of joint primitives that make up the Cylindrical block: prismatic revolute `P1` and revolute primitive `R1`.

Axis of Action [x y z]

Enter here as a three-component vector the directional axes defining the allowed motions of these primitives and their corresponding DoFs:

- Prismatic: axis of translation
- Revolute: axis of rotation

The default vectors are shown in the dialog above. The axes are directed vectors whose overall sign matters.

The two axes `P1` and `R1` in Cylindrical must be aligned.

Reference CS

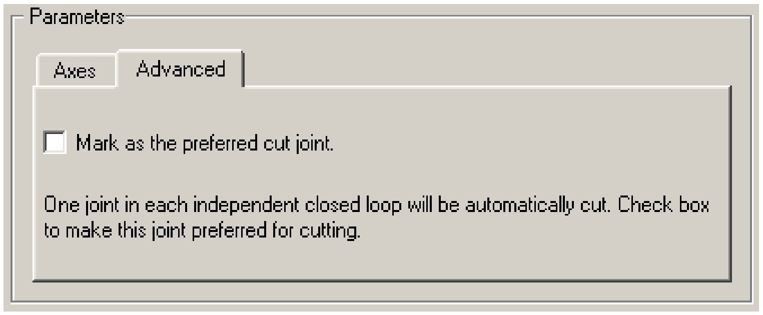
Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of action is oriented with respect to. This CS also determines the absolute meaning of forces/torques and motion along/about the joint axis. The default is `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Disassembled Cylindrical, Prismatic, Revolute, Screw

See "Modeling Degrees of Freedom" for more on representing DoFs with Joints.

See "Checking Model Topology" and "How SimMechanics Software Works" for more on closed loops and cutting.

Purpose

Joint with misaligned base and follower axes containing one revolute and one prismatic joint primitives

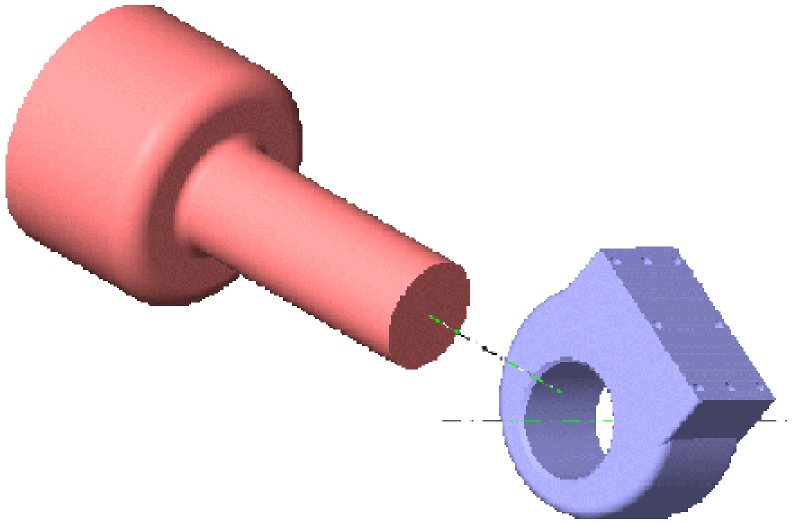
Library

Joints/Disassembled Joints

Description

The Disassembled Cylindrical block represents a composite joint, one translational and one rotational degrees of freedom (DoF) along and about a pair of specified misaligned axes between two bodies. SimMechanics simulation automatically assembles (aligns) the two axes at simulation start when it defines a machine's assembled configuration.

A cylindrical joint is composite, with two DoFs and a single specified axis: a prismatic primitive translating along the axis, and a revolute primitive rotating about the axis. There are no constraints between the two primitives.



Disassembled Cylindrical Axes of Follower (blue) and Base (red)

Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have a default of two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) point.

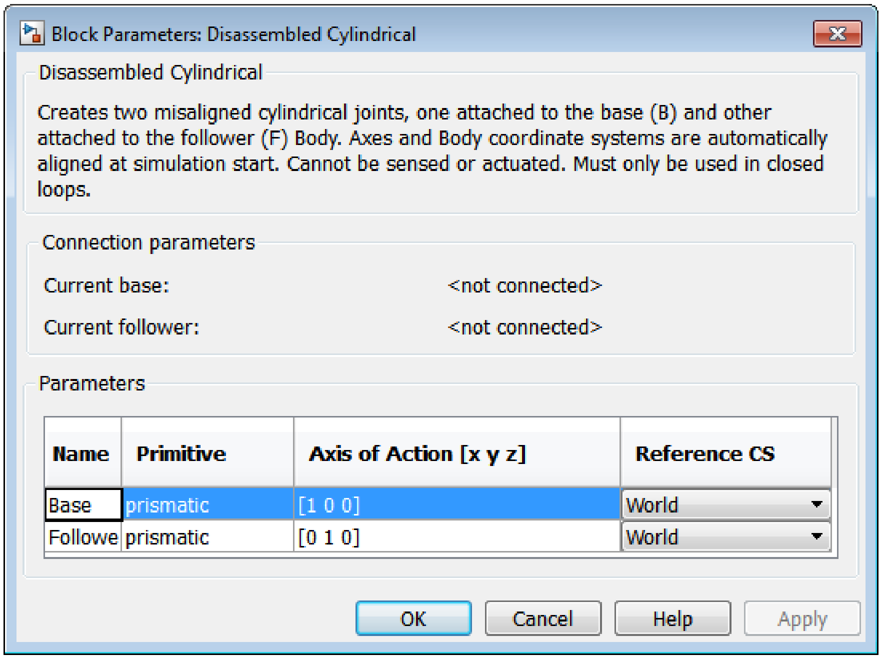
You specify the joint primitive axes, if any, in the Joint dialog. The two disassembled primitive joint axes are associated, in order, with the base and follower Bodies, respectively.

You cannot connect an Actuator or Sensor to a Disassembled Joint.

Assembly Restrictions on Disassembled Joints

This Joint block is disassembled: the connected Body CS origins do not need to be spatially collocated points or lie within the span of prismatic primitives, if any.

You can only use a Disassembled Joint block within a closed loop. One loop can have no more than one disassembled joint.


Block Parameters: Disassembled Cylindrical

Disassembled Cylindrical

Creates two misaligned cylindrical joints, one attached to the base (B) and other attached to the follower (F) Body. Axes and Body coordinate systems are automatically aligned at simulation start. Cannot be sensed or actuated. Must only be used in closed loops.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>

Parameters

Name	Primitive	Axis of Action [x y z]	Reference CS
Base	prismatic	[1 0 0]	World
Followe	prismatic	[0 1 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

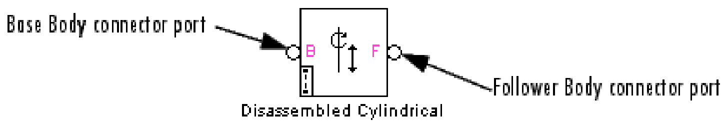
Connection Parameters

Current base

When you connect the base (B) connector port on the Disassembled Cylindrical block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Disassembled Cylindrical Base and Follower Body Connector Ports on page 2-76.

Current follower

When you connect the follower (F) connector port on the Disassembled Cylindrical block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Disassembled Cylindrical Base and Follower Body Connector Ports on page 2-76.

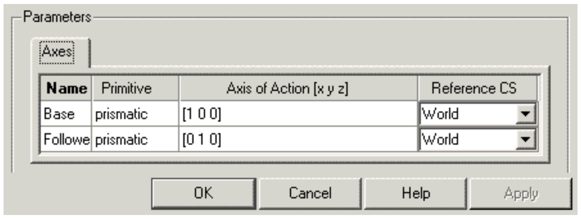


Disassembled Cylindrical Base and Follower Body Connector Ports

Parameters

There is one **Axes** tab.

The entries on the **Axes** tab are required. They specify the directions of the two misaligned axes of the translational-rotational DoFs that the Disassembled Cylindrical represents.



Name	Primitive	Axis of Action [x y z]	Reference CS
Base	prismatic	[1 0 0]	World
Followe	prismatic	[0 1 0]	World

Name
: This column automatically displays the names of the two misaligned rotation axes attached to base and follower bodies, respectively.

Axis of Action [x y z]
: Enter here as two three-component vectors the two misaligned directional axes along and about which the base and follower bodies respectively can translate and rotate. The default vectors are `[1 0 0]` and `[0 1 0]`, respectively. The axes are directed vectors whose overall signs matter.

Reference CS
: Using the pull-down menu, choose the coordinate systems (World, the base Body CS, or the follower Body CS) whose coordinate axes the two vector axes of translation-rotation are oriented with respect to. The defaults are `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

See Also

Cylindrical, Disassembled Prismatic, Disassembled Revolute, Disassembled Spherical

See “Modeling Degrees of Freedom” for more on representing DoFs with Disassembled Joints.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on closed loops and cutting disassembled joints.

Disassembled Prismatic

Purpose

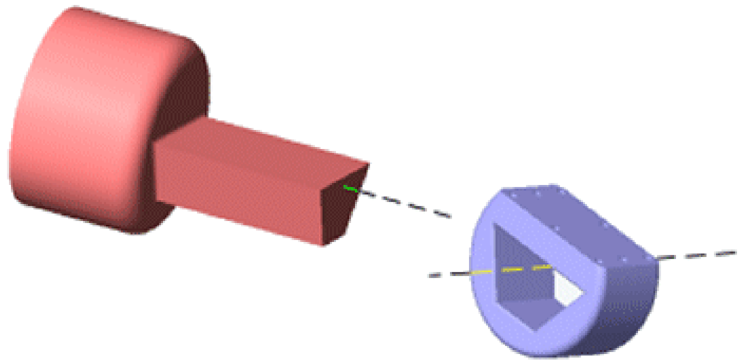
Primitive joint with misaligned base and follower axes containing one translational degree of freedom

Library

Joints/Disassembled Joints

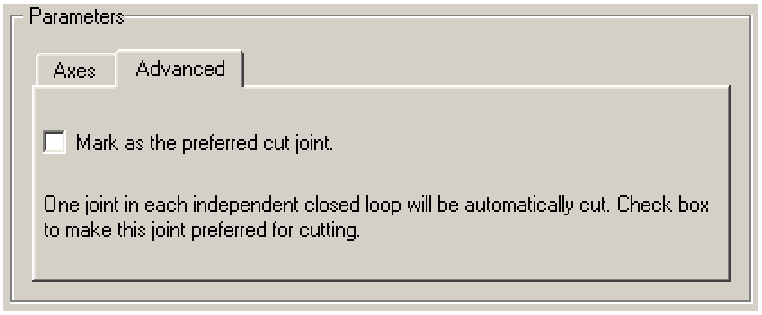
Description

The Disassembled Prismatic block represents a single translational degrees of freedom (DoF) along a pair of specified misaligned axes between two bodies. SimMechanics simulation automatically assembles (aligns) the translation axes at simulation start when it defines a machine's assembled configuration.



Disassembled Prismatic Axes of Follower (blue) and Base (red)

Advanced Tab



☐ Mark as the preferred cut joint.

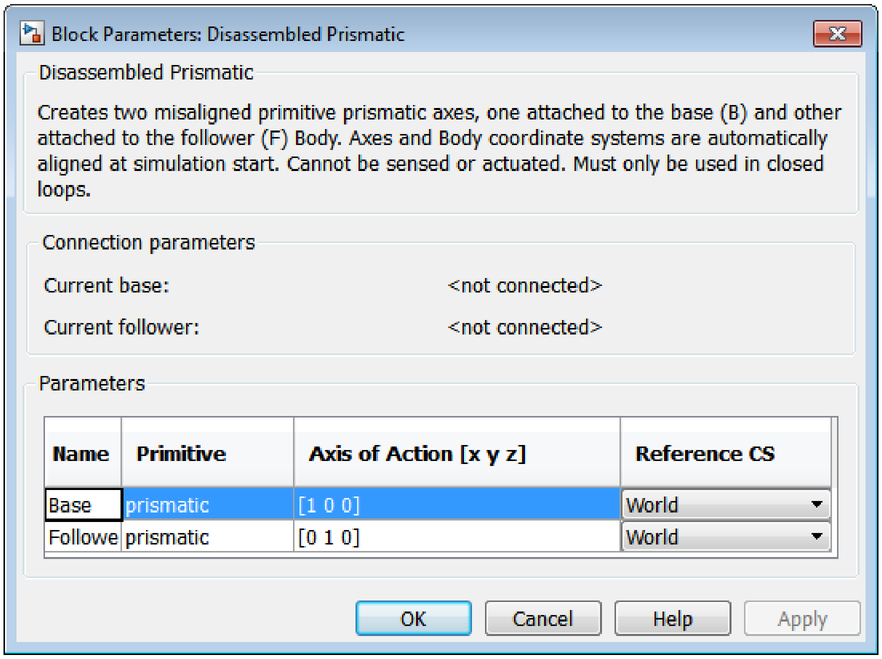
One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.



Disassembled Prismatic

Creates two misaligned primitive prismatic axes, one attached to the base (B) and other attached to the follower (F) Body. Axes and Body coordinate systems are automatically aligned at simulation start. Cannot be sensed or actuated. Must only be used in closed loops.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>

Parameters

Name	Primitive	Axis of Action [x y z]	Reference CS
Base	prismatic	[1 0 0]	World
Followe	prismatic	[0 1 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

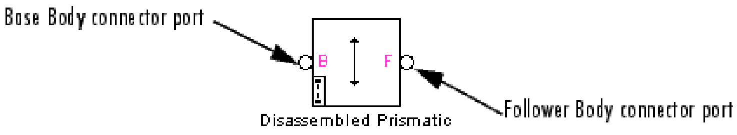
Connection Parameters

Current base

When you connect the base (B) connector port on the Disassembled Prismatic block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Disassembled Prismatic Base and Follower Body Connector Ports on page 2-81.

Current follower

When you connect the follower (F) connector port on the Disassembled Prismatic block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Disassembled Prismatic Base and Follower Body Connector Ports on page 2-81.

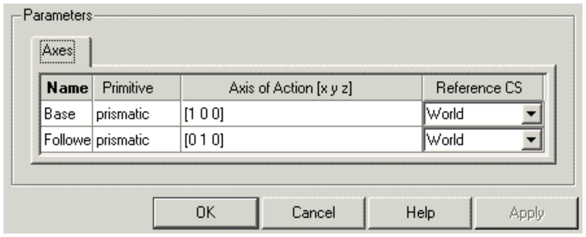


Disassembled Prismatic Base and Follower Body Connector Ports

Parameters

There is one **Axes** tab.

The entries on the **Axes** tab are required. They specify the directions of the two misaligned axes of the translational DoF that the Disassembled Prismatic represents.



Name	Primitive	Axis of Action [x y z]	Reference CS
Base	prismatic	[1 0 0]	World
Followe	prismatic	[0 1 0]	World

Name

This column automatically displays the names of the two misaligned translation axes attached to base and follower bodies, respectively.

Axis of translation [x y z]

Enter here as two three-component vectors the two misaligned directional axes along which the base and follower bodies respectively can translate. The default vectors are `[1 0 0]` and `[0 1 0]`, respectively. The axes are directed vectors whose overall signs matter.

Reference CS

Using the pull-down menu, choose the coordinate systems (World, the base Body CS, or the follower Body CS) whose coordinate axes the two vector axes of translation are oriented with respect to. The defaults are `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

See Also

Disassembled Cylindrical, Disassembled Revolute, Disassembled Spherical, Prismatic

See “Modeling Degrees of Freedom” for more on representing DoFs with Disassembled Joints.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on closing loops with disassembled joints.

Purpose

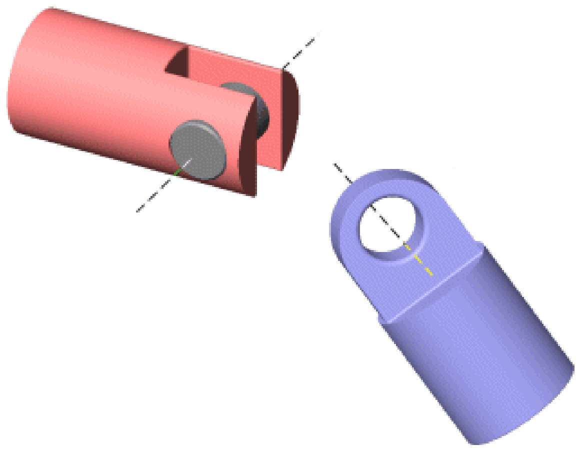
Primitive joint with misaligned base and follower axes containing one rotational degree of freedom

Library

Joints/Disassembled Joints

Description

The Disassembled Revolute block represents a single rotational degrees of freedom (DoF) along a pair of specified misaligned axes between two bodies. SimMechanics simulation automatically assembles (aligns) the rotation axes at simulation start when it defines a machine's assembled configuration.



Disassembled Revolute Axes of Follower (blue) and Base (red)

Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have a default of two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) point.

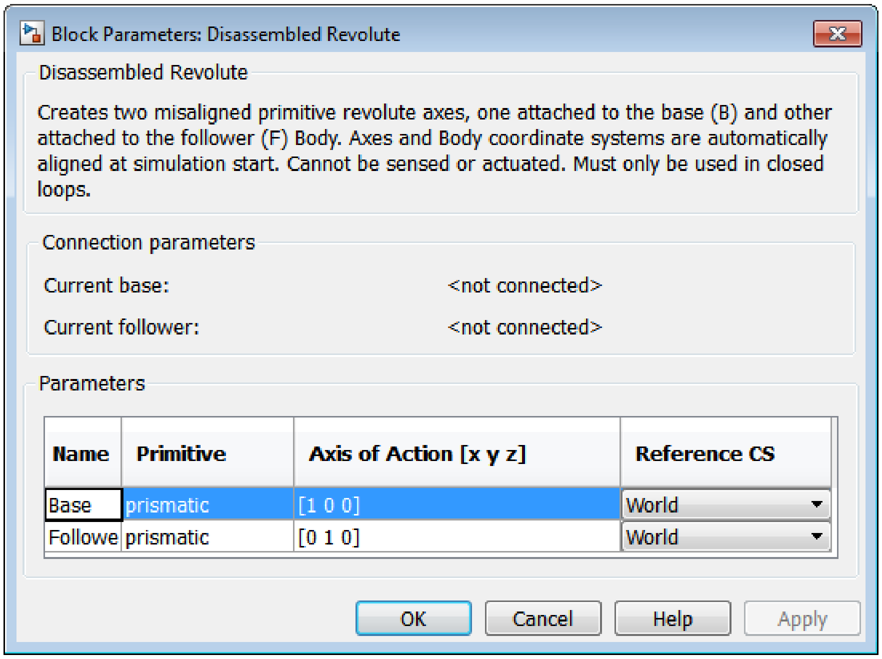
You specify the joint primitive axes, if any, in the Joint dialog. The two disassembled primitive joint axes are associated, in order, with the base and follower Bodies, respectively.

You cannot connect an Actuator or Sensor to a Disassembled Joint.

Assembly Restrictions on Disassembled Joints

This Joint block is disassembled: the connected Body CS origins do not need to be spatially collocated points or lie within the span of prismatic primitives, if any.

You can only use a Disassembled Joint block within a closed loop. One loop can have no more than one disassembled joint.


Block Parameters: Disassembled Revolute

Disassembled Revolute

Creates two misaligned primitive revolute axes, one attached to the base (B) and other attached to the follower (F) Body. Axes and Body coordinate systems are automatically aligned at simulation start. Cannot be sensed or actuated. Must only be used in closed loops.

Connection parameters

Current base: <not connected>

Current follower: <not connected>

Parameters

Name	Primitive	Axis of Action [x y z]	Reference CS
Base	prismatic	[1 0 0]	World
Followe	prismatic	[0 1 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

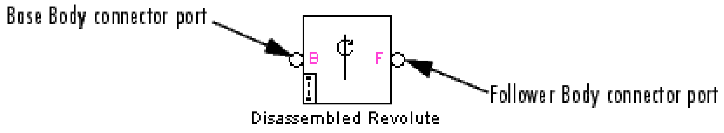
Connection Parameters

Current base

When you connect the base (B) connector port on the Disassembled Revolute block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Disassembled Revolute Base and Follower Body Connector Ports on page 2-86.

Current follower

When you connect the follower (F) connector port on the Disassembled Revolute block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Disassembled Revolute Base and Follower Body Connector Ports on page 2-86.

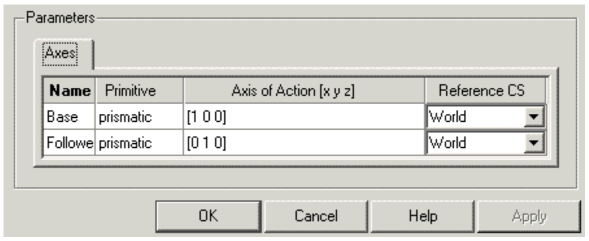


Disassembled Revolute Base and Follower Body Connector Ports

Parameters

There is one **Axes** tab.

The entries on the **Axes** tab are required. They specify the directions of the two misaligned axes of the rotational DoF that the Disassembled Revolute represents.



Name	Primitive	Axis of Action [x y z]	Reference CS
Base	prismatic	[1 0 0]	World
Followe	prismatic	[0 1 0]	World

Name
: This column automatically displays the names of the two misaligned rotation axes attached to base and follower bodies, respectively.

Axis of rotation [x y z]
: Enter here as two three-component vectors the two misaligned directional axes about which the base and follower bodies respectively can rotate. The default vectors are `[1 0 0]` and `[0 1 0]`, respectively. The axes are directed vectors whose overall signs matter.

Reference CS
: Using the pull-down menu, choose the coordinate systems (World, the base Body CS, or the follower Body CS) whose coordinate axes the two vector axes of rotation are oriented with respect to. The defaults are `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

See Also

Disassembled Cylindrical, Disassembled Prismatic, Disassembled Spherical, Revolute

See “Modeling Degrees of Freedom” for more on representing DoFs with Disassembled Joints.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on closing loops with disassembled joints.

Disassembled Spherical

Purpose

Primitive joint with misaligned base and follower axes containing three rotational degrees of freedom

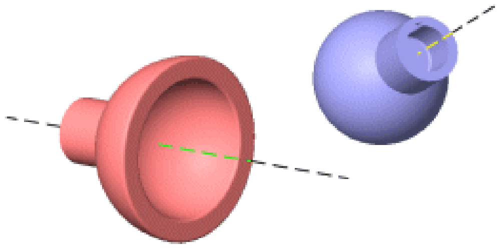
Library

Joints/Disassembled Joints

Description

The Disassembled Spherical block represents three rotational degrees of freedom (DoF) about a pair of specified dislocated pivots at the two bodies, separated “ball-in-socket” joints. SimMechanics simulation automatically assembles (collocates) the spherical pivots at simulation start when it defines a machine’s assembled configuration.

Two rotational DoFs specify a directional axis, and a third rotational DoF specifies rotation about that directional axis. (See the motion figure in the Spherical block reference page.) The sense of each rotational DoF is defined by the right-hand rule. Unlike the Gimbal block, the Disassembled Spherical block cannot become singular.



Disassembled Spherical Pivots of Follower (blue) and Base (red)

Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have a default of two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) point.

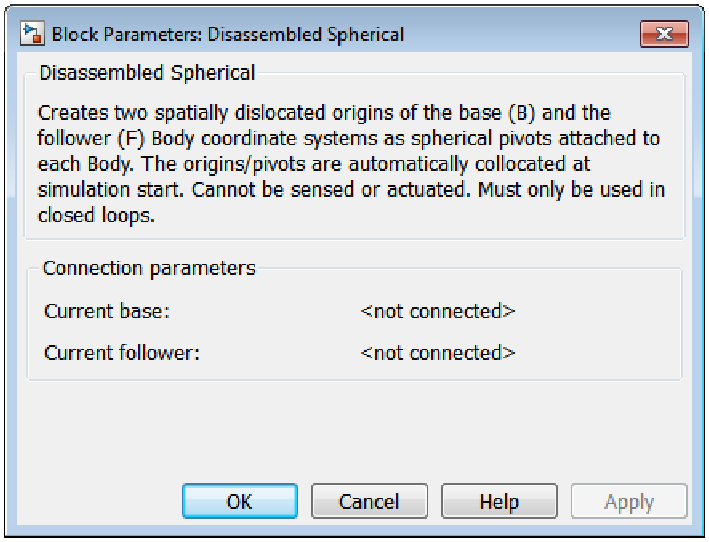
You specify the joint primitive axes, if any, in the Joint dialog. The two disassembled primitive joint axes are associated, in order, with the base and follower Bodies, respectively.

You cannot connect an Actuator or Sensor to a Disassembled Joint.

Assembly Restrictions on Disassembled Joints

This Joint block is disassembled: the connected Body CS origins do not need to be spatially collocated points or lie within the span of prismatic primitives, if any.

You can only use a Disassembled Joint block within a closed loop. One loop can have no more than one disassembled joint.



Block Parameters: Disassembled Spherical

Disassembled Spherical

Creates two spatially dislocated origins of the base (B) and the follower (F) Body coordinate systems as spherical pivots attached to each Body. The origins/pivots are automatically collocated at simulation start. Cannot be sensed or actuated. Must only be used in closed loops.

Connection parameters

Current base: <not connected>

Current follower: <not connected>

Dialog Box and Parameters

The dialog has one area, **Connection parameters**, which is inactive.

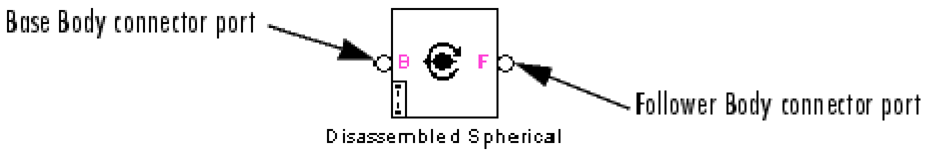
Connection Parameters

Current base

When you connect the base (B) connector port on the Disassembled Spherical block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Disassembled Spherical Base and Follower Body Connector Ports on page 2-90.

Current follower

When you connect the base (F) connector port on the Disassembled Spherical block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Disassembled Spherical Base and Follower Body Connector Ports on page 2-90.



Disassembled Spherical Base and Follower Body Connector Ports

See Also

Disassembled Cylindrical, Disassembled Prismatic, Disassembled Revolute, Gimbal, Spherical

See “Modeling Degrees of Freedom” for more on representing DoFs with Disassembled Joints.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on closing loops with disassembled joints.

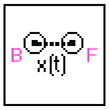
Purpose

Time-dependent distance between two body coordinate systems

Library

Constraints & Drivers

Description



The Distance Driver block drives the distance between the origins of two Body coordinate system (CS) as a function of time that you specify. This function must always remain nonnegative during the simulation.

Let $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ be the vector positions of the origins of CS1 on one Body and CS2 on the other Body, respectively. These vectors can be measured in any CS. The Distance Driver specifies the scalar distance $d = |\boldsymbol{r}_1 - \boldsymbol{r}_2|$ between these points as a function of time:

$$|\boldsymbol{r}_1 - \boldsymbol{r}_2| = d(t = 0) + f(t)$$

You connect the Distance Driver to a Driver Actuator block.

The Simulink input signal into the Driver Actuator specifies the time-dependent driving function $f(t)$ and its first two derivatives, as well as their units. If you do not actuate Distance Driver, this block acts as a time-independent constraint that freezes the distance between the two Body CSs at its initial value $d(t=0)$ during the simulation.

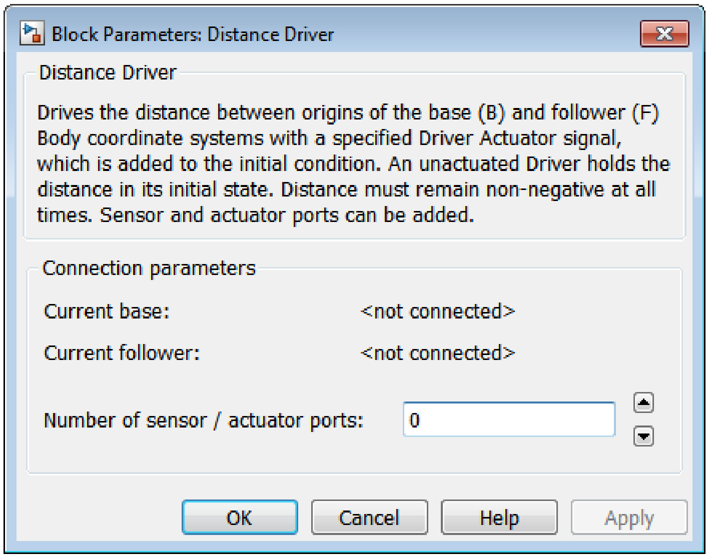
Caution

The block computes the initial distance $d(t=0)$ between the coordinate systems when the machine is in its home configuration, the original configuration of the machine's bodies before the simulation imposes initial conditions and implements joint assembly.

See "Kinematics and Machine Motion State" and "How SimMechanics Software Works".

Drivers restrict relative degrees of freedom (DoFs) between a pair of bodies as specified functions of time. Locally in a machine, they replace a Joint as the expression of the DoFs. Globally, Driver blocks must occur topologically in closed loops. Like Bodies connected to a Joint, the two Bodies connected to a Drivers are ordered as base and follower, fixing the direction of relative motion.

You can also connect a Constraint & Driver Sensor to any Driver and measure the reaction forces/torques between the driven bodies.



Drives the distance between origins of the base (B) and follower (F) Body coordinate systems with a specified Driver Actuator signal, which is added to the initial condition. An unactuated Driver holds the distance in its initial state. Distance must remain non-negative at all times. Sensor and actuator ports can be added.

Current base: <not connected>

Current follower: <not connected>

Number of sensor / actuator ports: 0

Dialog Box and Parameters

The dialog has one active area, **Connection parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive translation is the follower moving in the direction of the translation axis.

Current base

When you connect the base (B) connector port on the Distance Driver block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Distance Driver Base and Follower Body Connector Ports on page 2-93.

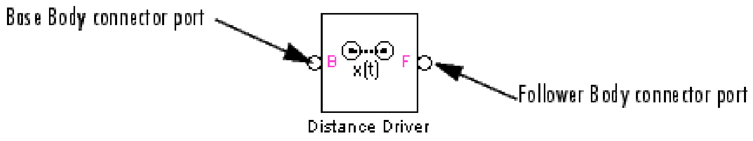
Current follower

When you connect the follower (F) connector port on the Distance Driver block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Distance Driver Base and Follower Body Connector Ports on page 2-93.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Driver Actuator and Constraint & Driver Sensor blocks to this Driver. The default is `0`.

To activate the Driver, connect a Driver Actuator.



Distance Driver Base and Follower Body Connector Ports

See Also

Constraint & Driver Sensor, Driver Actuator, Linear Driver, Weld

See “Constraining and Driving Degrees of Freedom” for more on restricting DoFs with Drivers.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on using drivers in closed loops.

Driver Actuator

Purpose Time-dependent motion input for driver blocks

Library Sensors & Actuators

Description The Driver Actuator block actuates a Driver block connected between a pair of bodies. You connect the block to the Driver block connected between the Bodies. The Driver block represents a time-dependent (rheonomic) constraint on a relative degrees of freedom (DoF) between the two bodies. A Driver requires a time-dependent function to specify the relative position, velocity, and acceleration of the connected Bodies. The input of the Driver Actuator is this time-dependent function *f(t)* and its first two derivatives.

The Driver connects a base (B) and a follower (F) Body. The base-follower sequence determines the sense of the actuation signal. The inport is the Simulink input signal. The output is the follower you connect to the Driver block you want to actuate.

Tip You do not have to connect a Driver block to a Driver Actuator. If you do not actuate a Driver, the Driver block acts as a time-independent constraint that freezes the driven relative DoF between the Bodies at its initial value during the simulation.

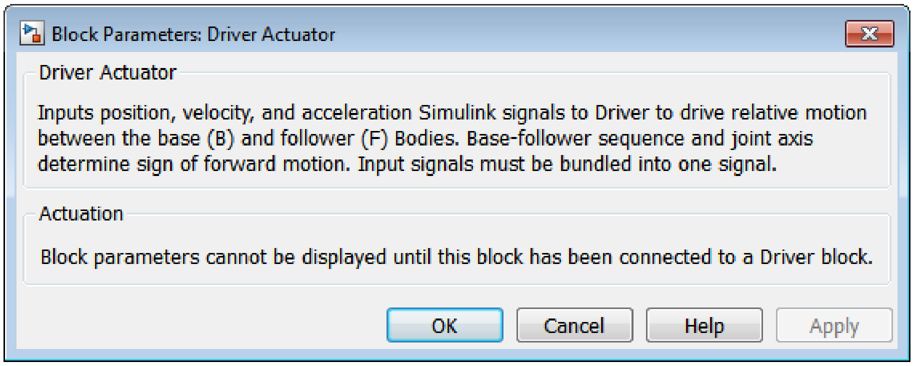
Driver Actuation Requires Two or Three Input Signals

You specify the three actuation functions as a bundled Simulink input signal (which can include a signal feedback from a Sensor block) satisfying these conditions:

- The signal must consist of three bundled components.
  - *Exception:* The Velocity Driver requires two bundled components.
- The three components must be ordered [$f(t)$, $df(t)/dt$, $d^2f(t)/dt^2$].
  - *Exception:* The Velocity Driver requires a function and its derivative [$f(t)$, $df(t)/dt$].

- Each successive signal component must be the time derivative of the previous component.
- The meaning of *f(t)* depends on the connected Driver block being actuated. Select the specific Driver block for details.

Linear Motions	Angular Motions
Distance Driver Linear Driver Velocity Driver	Angle Driver


Block Parameters: Driver Actuator

Driver Actuator

Inputs position, velocity, and acceleration Simulink signals to Driver to drive relative motion between the base (B) and follower (F) Bodies. Base-follower sequence and joint axis determine sign of forward motion. Input signals must be bundled into one signal.

Actuation

Block parameters cannot be displayed until this block has been connected to a Driver block.

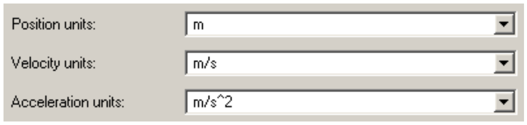
Dialog Box and Parameters

The dialog has one active area, **Actuation**. The block parameters are not displayed unless you connect it to a specific Driver block.

Actuation

The block dialog parameters depend on the specific Driver block to which you have connected it.

Driving Linear Motion



Position units:	m
Velocity units:	m/s
Acceleration units:	m/s^2

Position units

In the pull-down menu, choose the units of the actuating *f(t)* you apply to the relative motion of the bodies. The default is `m` (meters).

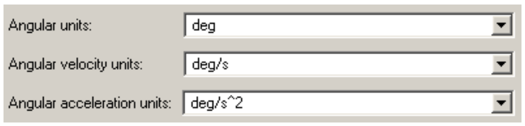
Velocity units

In the pull-down menu, choose the units of the actuating *df(t)/dt* you apply to the relative motion of the bodies. The default is `m/s` (meters/second).

Acceleration units

In the pull-down menu, choose the units of the actuating $d^2f(t)/dt^2$ you apply to the relative motion of the bodies. The default is `m/s`^{2} (meters/second^2).

Driving Angular Motion



Angular units: deg
Angular velocity units: deg/s
Angular acceleration units: deg/s^2

Angular units

In the pull-down menu, choose the units of the actuating *f(t)* you apply to the relative motion of the bodies. The default is `deg` (degrees).

Angular velocity units

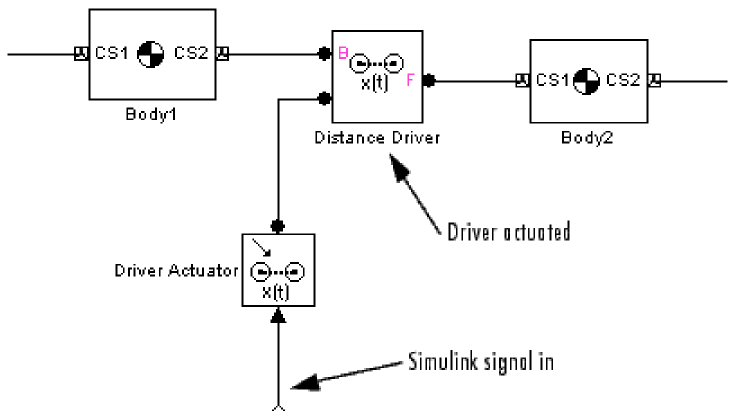
In the pull-down menu, choose the units of the actuating *df(t)/dt* you apply to the relative motion of the bodies. The default is `deg/s` (degrees/second).

Angular acceleration units

In the pull-down menu, choose the units of the actuating $d^2f(t)/dt^2$ you apply to the relative motion of the bodies. The default is `deg/s`^{2} (degrees/second^2).

Example

Here is a Driver Actuator connected to a Distance Driver, which connects two Bodies:



You must add an Actuator port (connector port) to the Driver block to connect the Driver Actuator to it. The base (B)-follower (F) Body sequence on the two sides of the Driver determines the sense of the Driver Actuator data.

The Driver Actuator drives the relative motion between the two Bodies connected to the Driver. The nature of the connected Driver block determines the exact meaning of the actuation data, including the choice of units.

See Also

Body Actuator, Constraint & Driver Sensor, Joint Actuator, Mechanical Branching Bar

Gear Constraint

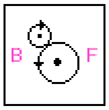
Purpose

Constraint that restricts body motion to rotation along tangent circles

Library

Constraints & Drivers

Description



The two Bodies connected by a Gear Constraint block are each restricted to turn relative to another along pitch circles centered at each body. The pitch circle centers are the origins of the Body coordinate systems (CSs) at which the Gear Constraint block is connected on either side. The pitch circles are tangent at one contact point.

Let $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ be the radius vectors of the two pitch circles and $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$ the angular velocity vectors of the two bodies. The Gear Constraint requires that:

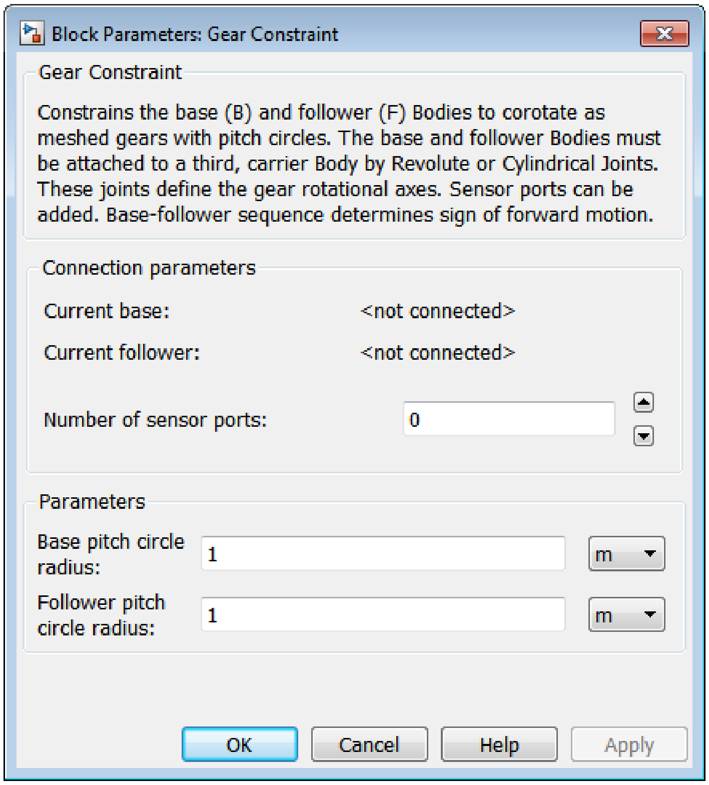
$$\boldsymbol{\omega}_1 \times \boldsymbol{r}_1 = \boldsymbol{\omega}_2 \times \boldsymbol{r}_2$$

You specify the scalar radii r_1, r_2 of the pitch circles.

You must also connect the two Bodies connected by a Gear Constraint to a third, carrier Body by Revolute or Cylindrical Joints. (The third carrier body can be ground, but you must use two Ground blocks in this case, because a Ground has only one Body CS port. Both Grounds represent the same immobile body.) The constrained pair of Bodies rotate relative to one another about distinct rotational axes defined by the angular velocity vectors ω_1, ω_2. These axes do not have to be parallel.

Constraints restrict relative degrees of freedom (DoFs) between a pair of bodies. Locally in a machine, they replace a Joint as the expression of the DoFs. Globally, Constraint blocks must occur topologically in closed loops. Like Bodies connected to a Joint, the two Bodies connected to a Constraint are ordered as base and follower, fixing the direction of relative motion.

You can connect a Constraint & Driver Sensor to a Constraint block, but not a Driver Actuator. The Constraint & Driver Sensor measures the reaction forces/torques between the constrained bodies.



Block Parameters: Gear Constraint

Gear Constraint

Constrains the base (B) and follower (F) Bodies to corotate as meshed gears with pitch circles. The base and follower Bodies must be attached to a third, carrier Body by Revolute or Cylindrical Joints. These joints define the gear rotational axes. Sensor ports can be added. Base-follower sequence determines sign of forward motion.

Connection parameters

Current base: <not connected>

Current follower: <not connected>

Number of sensor ports: 0

Parameters

Base pitch circle radius: 1 m

Follower pitch circle radius: 1 m

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive rotation is the follower rotating in the right-handed sense about the rotation axis.

Current base

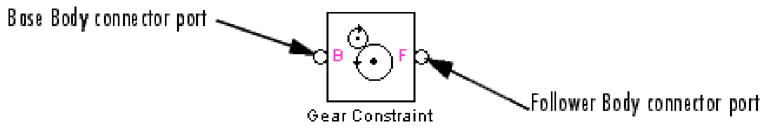
When you connect the base (B) connector port on the Gear Constraint block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Gear Constraint Base and Follower Body Connector Ports on page 2-100.

Current follower

When you connect the follower (F) connector port on the Gear Constraint block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Gear Constraint Base and Follower Body Connector Ports on page 2-100.

Number of sensor ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Constraint & Driver Sensor blocks to this Constraint. The default is `0`.



Gear Constraint Base and Follower Body Connector Ports

Parameters

Base pitch circle radius

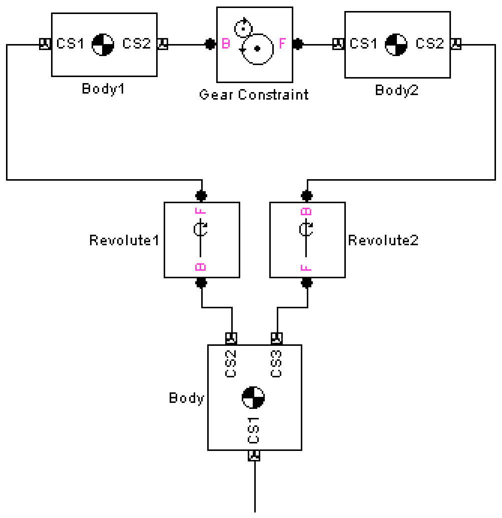
Enter a radius for the pitch circle centered at base Body CS. In the pull-down menu to the right, select units. The defaults are `1` and `m` (meters), respectively.

Follower pitch circle radius

Enter a radius for the pitch circle centered at follower Body CS. In the pull-down menu to the right, select units. The defaults are `1` and `m` (meters), respectively.

Example

A simple example of a valid part of a model with a Gear Constraint:



The Body CS origins CS2@Body1 and CS1@Body2 must be separated and oriented in such a way that the gear pitch circles are in contact and tangent at one point.

See Also

Body, Constraint & Driver Sensor, Cylindrical, Ground, Revolute

See “Constraining and Driving Degrees of Freedom” for more on restricting DoFs with Constraints.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on using constraints in closed loops.

Gimbal

Purpose

Joint with three revolute joint primitives

Library

Joints

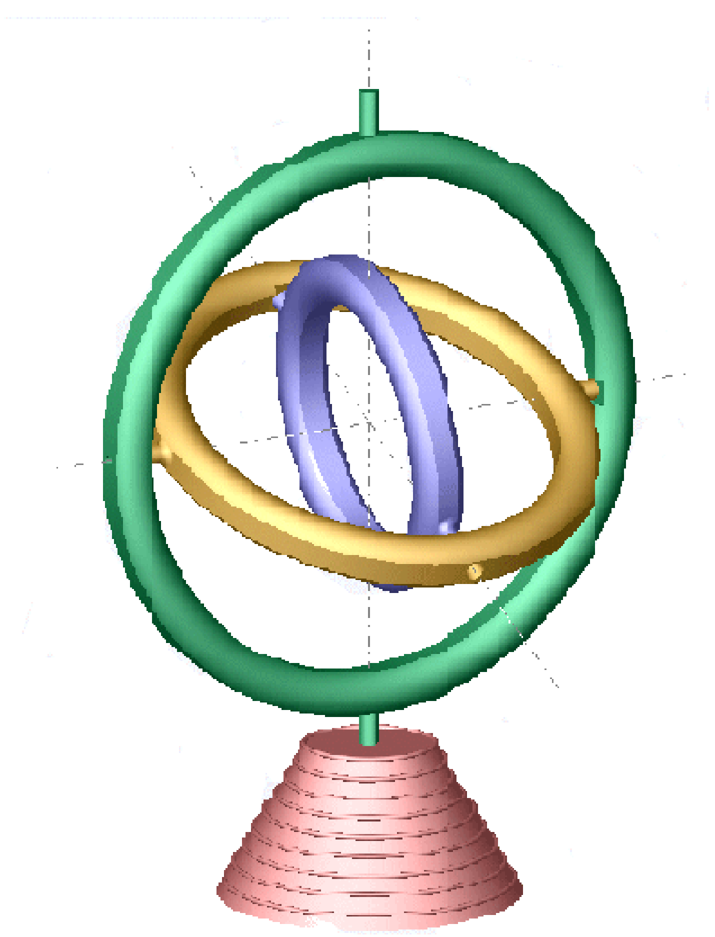
Description

The Gimbal block represents a composite joint with three rotational degrees of freedom (DoFs) as three revolute primitives. There are no constraints among the primitives.

Warning

A joint with three revolute primitives becomes singular if two or three of the rotation axes become parallel ("gimbal lock"). The simulation stops with an error in this case.

A joint with three revolute primitives must be configured in the initial state with the three revolute primitive axes mutually orthogonal. There are no restrictions on the primitive axes once the simulation starts, except to prevent any two of the primitive axes from becoming parallel.



Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower).

You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

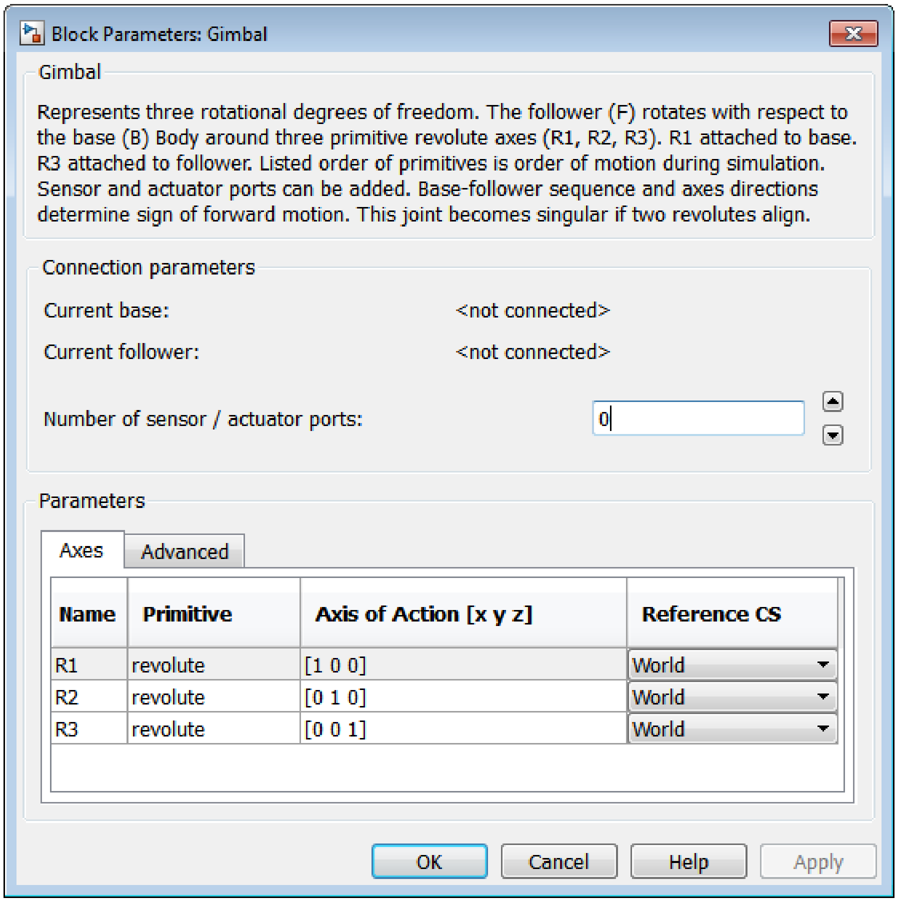
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space



Gimbal

Represents three rotational degrees of freedom. The follower (F) rotates with respect to the base (B) Body around three primitive revolute axes (R1, R2, R3). R1 attached to base. R3 attached to follower. Listed order of primitives is order of motion during simulation. Sensor and actuator ports can be added. Base-follower sequence and axes directions determine sign of forward motion. This joint becomes singular if two revolutes align.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>
Number of sensor / actuator ports:	0

Parameters

Name	Primitive	Axis of Action [x y z]	Reference CS
R1	revolute	[1 0 0]	World
R2	revolute	[0 1 0]	World
R3	revolute	[0 0 1]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive rotation is the follower moving around the rotational axis following the right-hand rule.

Current base

When you connect the base (B) connector port on the Gimbal block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Gimbal Base and Follower Body Connector Ports on page 2-106.

The base Body is automatically connected to the first joint primitive `R1` in the primitive list in **Parameters**.

Current follower

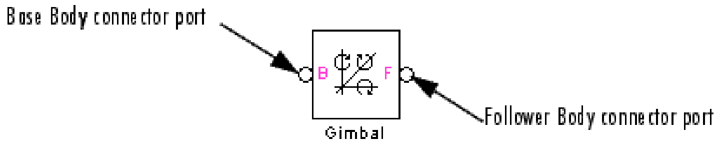
When you connect the follower (F) connector port on the Gimbal block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Gimbal Base and Follower Body Connector Ports on page 2-106.

The follower Body is automatically connected to the last joint primitive `R3` in the primitive list in **Parameters**.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`.

The motion of revolute primitives is specified in angular units.



Gimbal Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. Each DoF primitive in Gimbal has an entry line. These lines specify the direction of the axes of action of the DoFs that the Gimbal represents.

Name - Primitive

The primitive list states the names and types of joint primitives that make up the Gimbal block: revolute primitives `R1`, `R2`, `R3`.

Axis of Action [x y z]

Enter here as a three-component vector the directional axes defining the allowed motions of these primitives and their corresponding DoFs:

- Revolute: axis of rotation

The default vectors are shown in the dialog above. The axis is a directed vector whose overall sign matters.

To prevent singularities and simulation errors, no two of the revolute axes can be parallel.

Reference CS

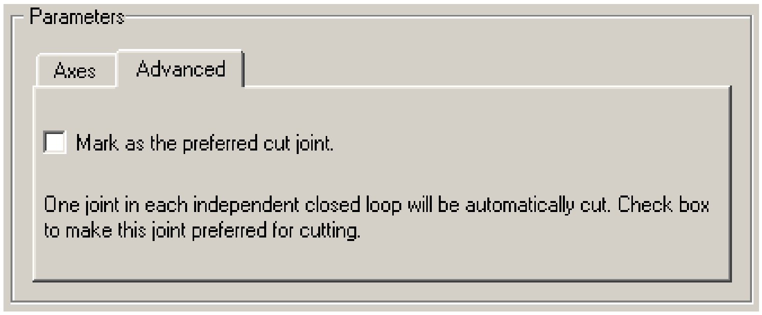
Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of action is oriented with respect to. This CS also determines the absolute meaning of forces/torques and motion along/about the joint axis. The default is `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Revolute, Spherical

See "Modeling Degrees of Freedom" for more on representing DoFs with Joints.

See "Checking Model Topology" and "How SimMechanics Software Works" for more on closed loops and cutting.

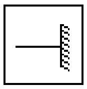
Purpose

Fixed point attached to world

Library

Bodies

Description



A Ground block represents an immobile ground point at rest in the absolute inertial World reference frame. Connecting it to a Joint prevents one side of that Joint from moving. You can also connect a Ground block to a machine Environment block.

Caution

Every valid SimMechanics machine *must* have at least one Ground block.

You *must* connect exactly one Ground block in each machine of your model to a Machine Environment block.

Ground is a type of Body, but you can connect only one side of a Ground to a Joint block. A Ground block automatically carries a grounded coordinate system (CS). This Grounded CS is inertial, at rest in the World reference frame, with coordinate axes parallel to the World axes:

+*x* points right

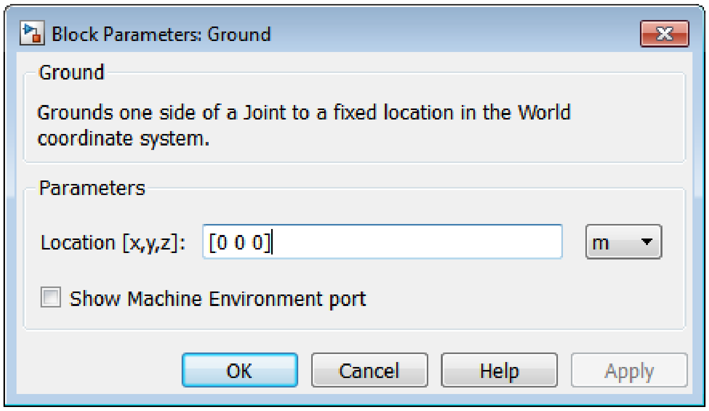
+*y* points up (gravity in -*y* direction)

+*z* points out of the screen, in three dimensions

But a Ground's origin is the ground point, which in general is shifted with respect to the World origin.

Multiple Ground blocks represent different fixed points in the global inertial World. In the topology of a model, multiple Ground blocks function as a single body.

You cannot connect a Sensor or Actuator to a Ground block, because the ground point cannot be moved.



Block Parameters: Ground

Ground

Grounds one side of a Joint to a fixed location in the World coordinate system.

Parameters

Location [x,y,z]: [0 0 0] m

☐ Show Machine Environment port

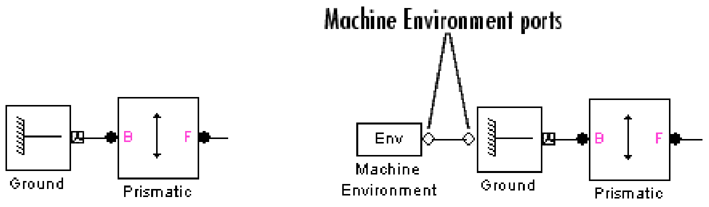
Dialog Box and Parameters

Location [x,y,z]

Enter the position of the ground point translated from the origin of the World CS. The position is specified as a translation vector *(x,y,z)*, with components projected onto the fixed World CS axes. Set the Ground position units using the pull-down menu to the right. The defaults are `[0 0 0]` and `m` (meters).

Show Machine Environment port

Select to enable the Machine Environment port on the Ground block. This port allows you to connect a Machine Environment block to the Ground and the machine that the Ground is a part of. The default is not selected.



A Ground Without and With a Connected Machine Environment Block

Configuring the Mechanical Environment

If you connect a Machine Environment block to a Ground, you can adjust the mechanical environment for the machine of which that Ground is a part. Consult the Machine Environment block reference.

See Also

Body, Machine Environment, Shared Environment

See “Representing Machines with Models”, “Modeling Grounds and Bodies”, and “Modeling Degrees of Freedom” for more on creating valid SimMechanics models and setting up Grounds.

See the relevant entries in the Glossary: ground, grounded CS, machine, and World.

In-Plane

Purpose

Joint with two coplanar prismatic joint primitives

Library

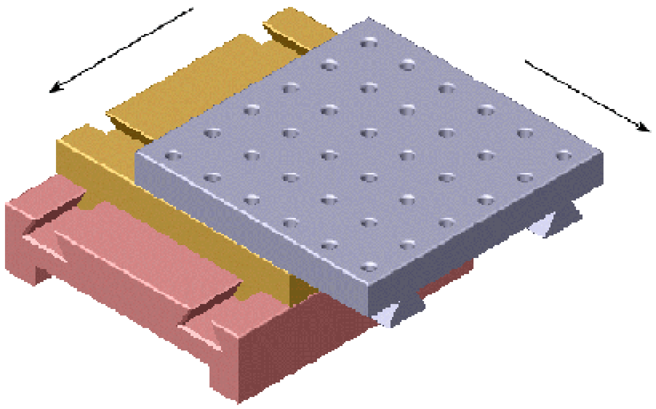
Joints

Description

The In-Plane block represents a composite joint with two translational degrees of freedom (DoFs) as two prismatic primitives. There are no constraints among the primitives.

Warning

A joint with two prismatic primitives becomes singular if the two translation axes become parallel. The simulation stops with an error in this case.



Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

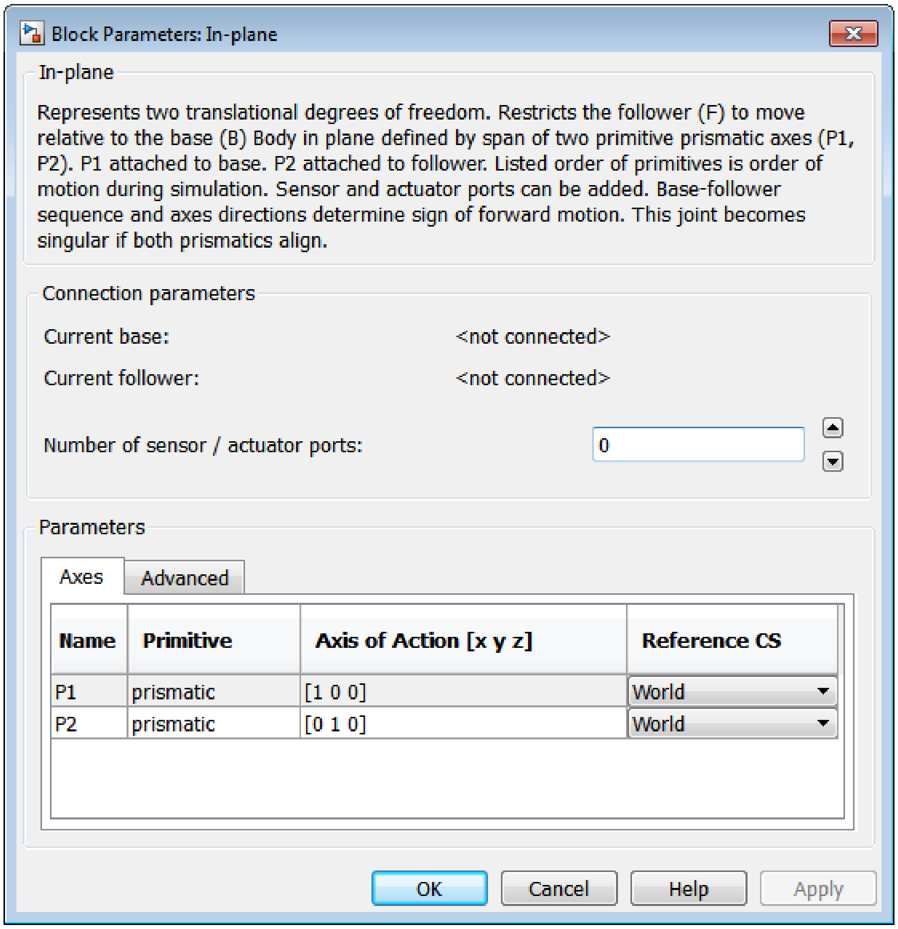
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space



In-plane

Represents two translational degrees of freedom. Restricts the follower (F) to move relative to the base (B) Body in plane defined by span of two primitive prismatic axes (P1, P2). P1 attached to base. P2 attached to follower. Listed order of primitives is order of motion during simulation. Sensor and actuator ports can be added. Base-follower sequence and axes directions determine sign of forward motion. This joint becomes singular if both prismatics align.

Connection parameters

Current base: <not connected>

Current follower: <not connected>

Number of sensor / actuator ports: 0

Parameters

Axes | Advanced

Name	Primitive	Axis of Action [x y z]	Reference CS
P1	prismatic	[1 0 0]	World
P2	prismatic	[0 1 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive translation is the follower moving in the direction of the translation axis.

Current base
When you connect the base (B) connector port on the In-Plane block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, In-Plane Base and Follower Body Connector Ports on page 2-115.

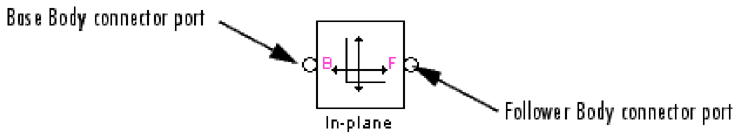
The base Body is automatically connected to the first joint primitive `P1` in the primitive list in **Parameters**.

Current follower
When you connect the follower (F) connector port on the In-Plane block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, In-Plane Base and Follower Body Connector Ports on page 2-115.

The follower Body is automatically connected to the last joint primitive `P2` in the primitive list in **Parameters**.

Number of sensor/actuator ports
Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`.

The motion of prismatic primitives is specified in linear units.



In-Plane Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. Each DoF primitive in In-Plane has an entry line. These lines specify the direction of the axes of action of the DoFs that the In-Plane represents.

Name - Primitive

The primitive list states the names and types of joint primitives that make up the In-Plane block: prismatic primitives P1, P2.

Axis of Action [x y z]

Enter here as a three-component vector the directional axes defining the allowed motions of these primitives and their corresponding DoFs:

- Prismatic: axis of translation

The default vectors are shown in the dialog above. The axis is a directed vector whose overall sign matters.

To prevent singularities and simulation errors, the two prismatic axes cannot be parallel.

Reference CS

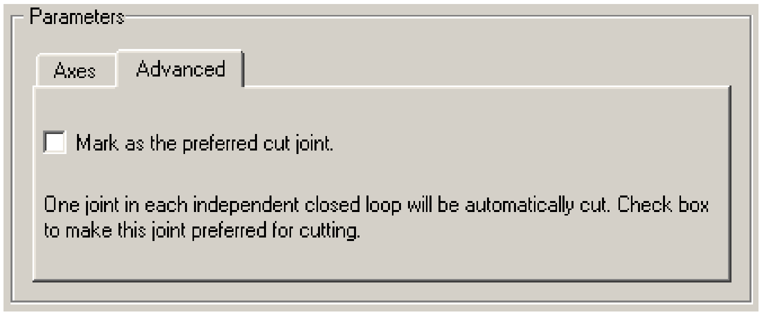
Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of action is oriented with respect to. This CS also determines the absolute meaning of forces/torques and motion along/about the joint axis. The default is World.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Planar, Prismatic

See “Modeling Degrees of Freedom” for more on representing DoFs with Joints.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on closed loops and cutting.

Purpose

Time-dependent force, torque, or motion input to a joint

Library

Sensors & Actuators

Description

A joint between two bodies represents relative degrees of freedom (DoFs) between the bodies. The Joint Actuator block actuates a Joint block connected between two Bodies with one of these signals:

- A generalized force:
  - Force for translational motion along a prismatic joint primitive
  - Torque for rotational motion about a revolute joint primitive
- A motion:
  - Translational motion for a prismatic joint primitive, in terms of linear position, velocity, and acceleration.
  - Rotational motion for a revolute joint primitive, in terms of angular position, velocity, and acceleration.

The generalized force or the motion is a function of time specified by a Simulink input signal, which can include a signal feedback from a Sensor block.

The Joint Actuator applies the actuation signal along/about the joint axis in the reference coordinate system (CS) specified for that joint primitive in the Joint's dialog. The Joint connects a base and a follower Body. The base-follower sequence determines the sense of the actuation signal.

The inport is the Simulink input signal. The output is the connector port you connect to the Joint block you want to actuate. A Joint Actuator block actuates one joint primitive at a time:

- A primitive Joint (Prismatic or Revolute) has only one primitive within the Joint to actuate.
- A composite Joint has multiple joint primitives within, and you must choose which of those primitives to actuate with the Joint Actuator.

You cannot connect a Joint Actuator to a Spherical or spherical primitive.

Caution You cannot simultaneously actuate a joint primitive with Joint Actuator motion actuation and with a Joint Initial Condition Actuator.

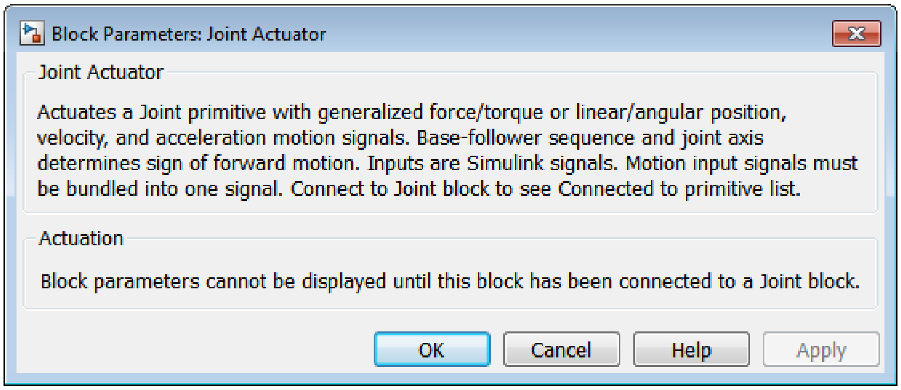
Joint Motion Actuation Requires Acceleration, Velocity, and Position Input Signals

- If the motion is translational, the block requires three input signals, corresponding to position, velocity, and acceleration.

  The velocity signal must be the derivative of the position signal, and the acceleration the derivative of the velocity.

- If the motion is rotational, the block requires three input signals, corresponding to angle, angular velocity, and angular acceleration.

  The angular velocity signal must be the derivative of the angle signal, and the angular acceleration the derivative of the angular velocity.



Actuates a Joint primitive with generalized force/torque or linear/angular position, velocity, and acceleration motion signals. Base-follower sequence and joint axis determines sign of forward motion. Inputs are Simulink signals. Motion input signals must be bundled into one signal. Connect to Joint block to see Connected to primitive list.

Block parameters cannot be displayed until this block has been connected to a Joint block.

Dialog Box and Parameters

The dialog has one active area, **Actuation**. The full block parameters are not displayed unless you connect it to a specific Joint block.

Actuation

Connected to primitive

In the pull-down menu, choose the joint primitive within the Joint that you want to actuate with the Joint Actuator. A primitive Joint block has only one joint primitive.

You cannot connect a Joint Actuator to a spherical primitive.

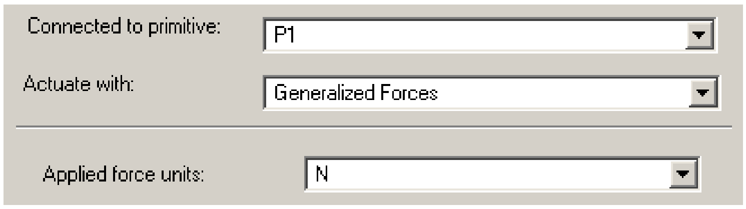
If the Joint Actuator is not connected to a Joint block, this menu is blank.

Actuate with

In the pull-down menu, choose one of two types of actuation, `Generalized Forces` or `Motion`.

Generalized Forces

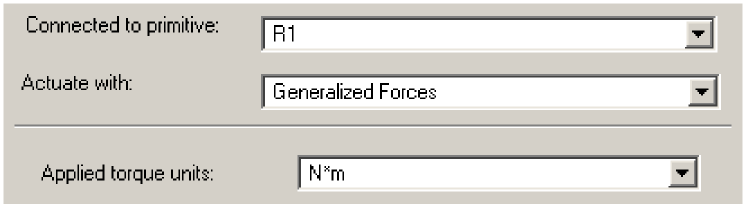
This option interprets the actuation signal as a force or a torque between the Bodies connected by the Joint block you are actuating. Choose units depending on whether you are actuating a prismatic or revolute primitive.


Connected to primitive: P1
Actuate with: Generalized Forces
Applied force units: N

Applied force units

In the pull-down menu, choose units for the actuation force. The default is `N` (newtons).

The Simulink input is a 1-component signal.


Connected to primitive: R1
Actuate with: Generalized Forces
Applied torque units: N*m

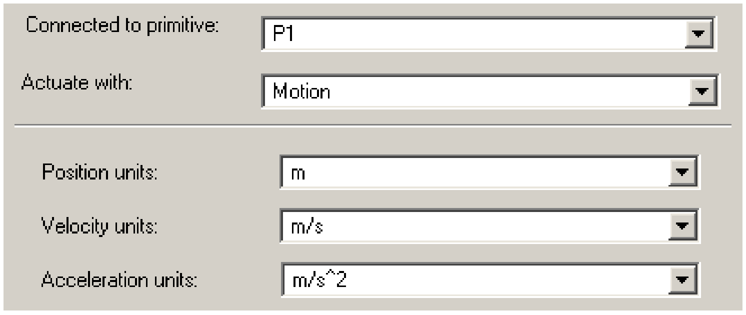
Applied torque units

In the pull-down menu, choose units for the actuation torque. The default is `N*m` (newton-meters).

The Simulink input is a 1-component signal.

Motion

This option interprets the actuation signal as a motion of the joint primitive to which you connect the Joint Actuator. Choose units depending on whether you are actuating a prismatic or revolute primitive. The Simulink input is a bundled 3-component signal with components in the order shown in the dialog.

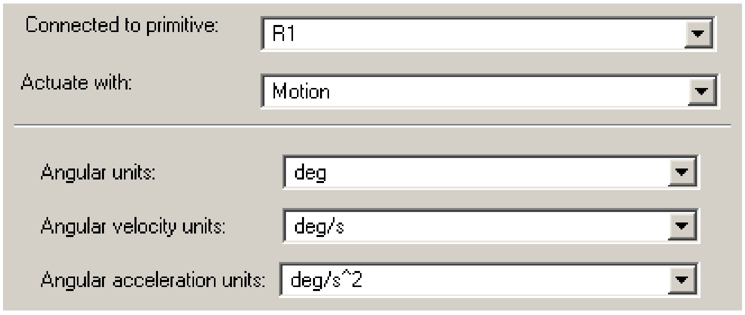

Connected to primitive: P1
Actuate with: Motion
Position units: m
Velocity units: m/s
Acceleration units: m/s^2

Position units

In the pull-down menu, choose units for the linear position motion actuation. The default is `m` (meters).

Velocity units
: In the pull-down menu, choose units for the linear velocity motion actuation. The default is `m/s` (meters/second).

Acceleration units
: In the pull-down menu, choose units for the linear acceleration motion actuation. The default is `m/s`^{2} (meters/second^2).



Connected to primitive:	R1
Actuate with:	Motion
Angular units:	deg
Angular velocity units:	deg/s
Angular acceleration units:	deg/s^2

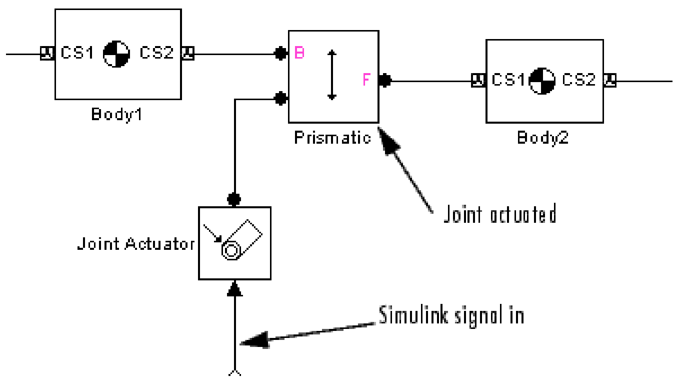
Angular units
: In the pull-down menu, choose units for the angular motion actuation. The default is `deg` (degrees).

Angular velocity units
: In the pull-down menu, choose units for the angular velocity motion actuation . The default is `deg/s` (degrees/second).

Angular acceleration units
: In the pull-down menu, choose units for the angular acceleration motion actuation. The default is `deg/s`^{2} (degrees/second^2).

Example

Here is a Joint Actuator connected to a Prismatic that connects two Bodies:



You must add an Actuator port (connector port) to the Joint block to connect the Joint Actuator to it. The base (B)-follower (F) Body sequence on the two sides of the Joint determines the sense of the Joint Actuator data.

See Also

Joint Initial Condition Actuator, Joint Sensor, Joint Stiction Actuator, Mechanical Branching Bar, Prismatic, Revolute

Joint Initial Condition Actuator

Purpose

Initial joint position and velocity

Library

Sensors & Actuators

Description

IC

The Joint Initial Condition Actuator block supplies the prismatic and revolute joint primitives of a Joint block with initial value data. The initial values are the positions and velocities of the joint primitives and fully specify the initial state of motion (initial kinematic state) of those primitives.

You can set initial positions and velocities for two primitive types:

- Translational initial conditions for a prismatic primitive, in terms of linear position and velocity
- Rotational initial conditions for a revolute primitive, in terms of angular position and velocity

This block can actuate one, some, or all of the prismatic and revolute primitives of a Joint.

The Joint Initial Condition Actuator applies the initial state along/about the joint axis in the reference coordinate system (CS) specified for that joint primitive in the Joint's dialog. The Joint connects a base and a follower Body. The base-follower sequence determines the sense of the actuation signal.

The output is the connector port you connect to the Joint block whose initial conditions you want to set. You set the initial linear and/or angular positions and velocities in the block's dialog, so there is no input signal.

You cannot actuate a Spherical or spherical primitive with a Joint Initial Condition Actuator.

Caution You cannot simultaneously actuate a joint primitive with a Joint Initial Condition Actuator and with Joint Actuator motion actuation.

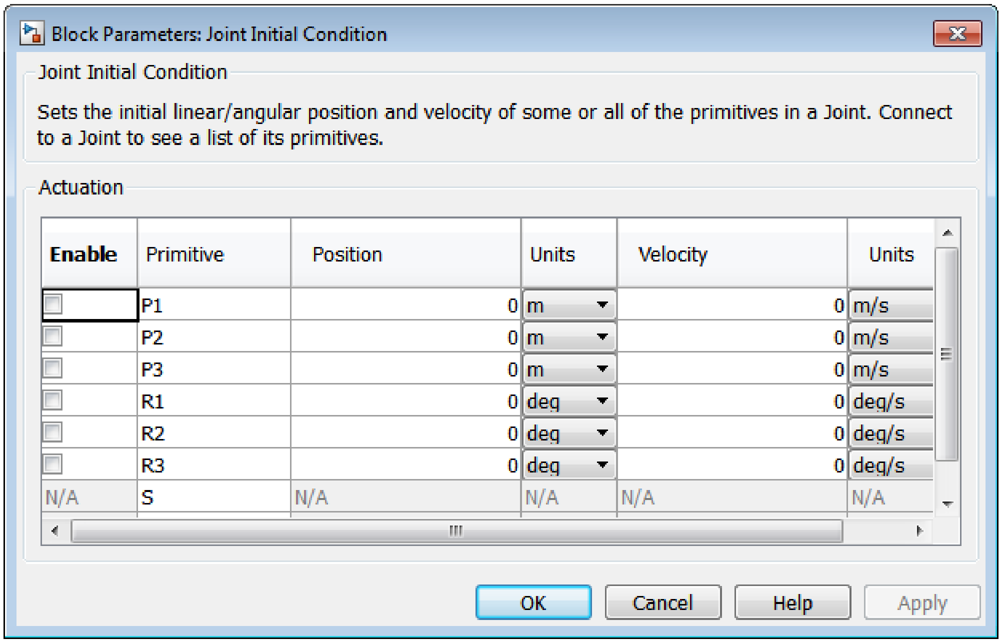
Initial Geometric Versus Kinematic Configuration

When you build your model, the geometric configuration of the Bodies (the home configuration) implicitly specifies the initial positions/angles of bodies relative to one another and to World. The Ground, Body, and Joint layout specifies only initial coordinates (degrees of freedom or DoFs), not their corresponding velocities. Starting a simulation in this state sets all initial velocities to zero. You can set the full initial kinematic state (the initial configuration), both positions and velocities, of joint primitives by using Joint Initial Condition Actuator blocks.

Initial Condition Actuation in Open and Closed Topologies

In a SimMechanics model, the DoFs represented by Joints are relative. Suppose you actuate a Joint with initial conditions, and that Joint has other Joints in a sequence connected to it through intermediate Bodies. Then the initial conditions applied to the first Joint change the absolute positions and velocities of the other Joints (as measured in World) because the initial conditions of the other Joints are defined relative to the first.

The one exception to this rule occurs if the actuated Joint is part of a closed loop. SimMechanics simulation cuts one Joint in each closed loop. The initial conditions applied to a Joint indirectly affect the initial conditions of the other connected Joints only up to (but not including) the cut Joint.



Block Parameters: Joint Initial Condition

Joint Initial Condition

Sets the initial linear/angular position and velocity of some or all of the primitives in a Joint. Connect to a Joint to see a list of its primitives.

Actuation

Enable	Primitive	Position	Units	Velocity	Units
☐	P1	0	m	0	m/s
☐	P2	0	m	0	m/s
☐	P3	0	m	0	m/s
☐	R1	0	deg	0	deg/s
☐	R2	0	deg	0	deg/s
☐	R3	0	deg	0	deg/s
N/A	S	N/A	N/A	N/A	N/A

Dialog Box and Parameters

The dialog has one active area, **Actuation**.

Actuation

The menu choices are available for every primitive in the Joint to which the Joint Initial Condition Actuator is connected. If you connect the Actuator with its dialog open, the primitive list is automatically updated to reflect the connected Joint's primitives. If the Actuator is unconnected, all primitive types are shown, including two that cannot be actuated, spherical (S) and weld (W).

Enable
Select this check box if you want to actuate the primitive with initial conditions. The default is not selected.

Primitive
Displays the name of the primitive within the Joint. Not an active field.

Position
: Enter a value for the initial position of the primitive, either prismatic or revolute. The default is `0`.

Units
: In the pull-down menu, select units for the initial position. The defaults are `m` (meters) for prismatic primitives and `deg` (degrees) for revolute primitives.

Velocity
: Enter a value for the initial velocity of the primitive, either prismatic or revolute. The default is `0`.

Units
: In the pull-down menu, select units for the initial velocity. The defaults are `m/s` (meters/second) for prismatic primitives and `deg/s` (degrees/second) for revolute primitives.

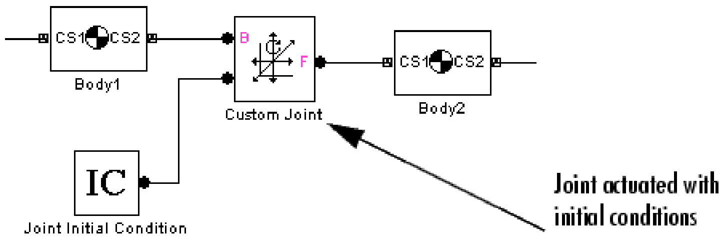
Restricted Parameters

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

- The **Actuation** (joint primitives) parameter table

Example

Here is a Joint Initial Condition Actuator connected to a Custom Joint, which connects two Bodies:



You must add an Actuator port (connector port) to the Joint block to connect the Joint Initial Condition Actuator to it. The base (B)-follower (F) Body sequence on the two sides of the Joint determines the sense of the Joint Initial Condition Actuator data.

See Also

Joint Actuator, Joint Sensor, Joint Stiction Actuator, Mechanical Branching Bar, Prismatic, Revolute

See “Using JICA Blocks” for setting general initial conditions (positions and velocities) of Joint DoFs. “Cutting Machine Diagram Loops” and “Checking Model Topology” discuss how SimMechanics simulation cuts Joints in closed loops.

Purpose

Joint force, torque, and motion sensor

Library

Sensors & Actuators

Description

The Joint Sensor block measures the position, velocity, and/or acceleration of a joint primitive in a Joint block. It also measures the reaction force and torque across the Joint.

The Joint Sensor measures the motion along/about the joint axis (or about the pivot point for a spherical primitive) in the reference coordinate system (CS) specified for that joint primitive in the Joint's dialog. The Joint connects a base and a follower Body at one body coordinate system on each body. The base-follower sequence determines the sense of the motion, which is defined as follower relative to base.

Depending on the joint primitive being sensed, you measure a combination of outputs from one of these motion types:

Motion Type	Joint Primitive Type	Measurements	Relative Motions of Connected Follower and Base Coordinate Systems
Translational	Prismatic	Linear position Linear velocity Linear acceleration	Distance moved along prismatic primitive axis of follower body CS origin relative to base body CS origin
Rotational	Revolute	Angle Angular velocity Acceleration	Rotation angle about revolute primitive axis of follower body CS origin relative to base body CS origin
Spherical	Spherical	Quaternion Quaternion first derivative Quaternion	Rotational orientation of follower body CS axes relative to base body CS axes

Motion Type	Joint Primitive Type	Measurements	Relative Motions of Connected Follower and Base Coordinate Systems
		second derivative	

The input is the connector port connected to the Joint being sensed. The outport is a set of Simulink signals or one bundled Simulink signal of the position, velocity, acceleration, computed force, and/or reaction force of the joint primitive.

A Joint Sensor block measures one joint primitive at a time:

- A primitive Joint (Prismatic or Revolute) has only one primitive within the Joint to sense.
- A composite Joint has multiple joint primitives within, and you must choose which primitive to sense with the Joint Sensor.

Relative Orientation of Follower and Base Bodies in Spherical Motion

In the spherical primitive case, the joint motion is a three-dimensional rotation, which is the orientation of the base body CS axes relative to the follower body CS axes. The sensor returns a quaternion q corresponding to a R that represents this orientation.

The components of the same vector $\boldsymbol{v}$ as measured in the follower and base body CS axes are related by $\boldsymbol{v}_F = R^T \cdot \boldsymbol{v}_B$. The column vector $\boldsymbol{v}_B$ lists the vector $\boldsymbol{v}$'s three components measured in the base body CS axes. The column vector $\boldsymbol{v}_F$ lists the vector $\boldsymbol{v}$'s three components measured in the follower body CS axes. The columns of the rotation matrix R are the components of the follower body CS unit basis vectors measured with respect to the base body CS axes.

See "Representations of Body Motion" and "Representations of Body Orientation" for more details on representing body position and orientation, rotation matrices, and angular velocity, as well as the relationship of R and q.

Joint Motion and the Home Configuration

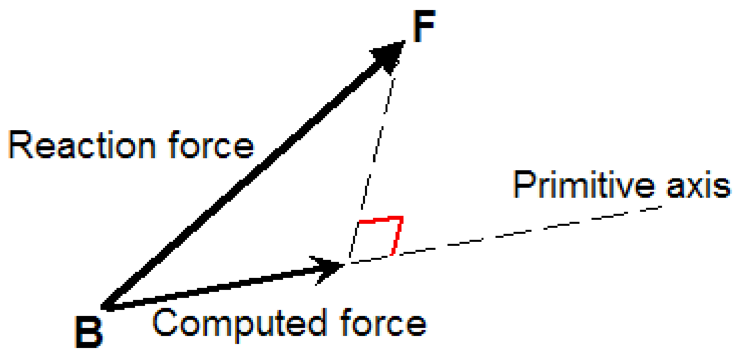
The Joint Sensor block measures the state of a degree of freedom, translational or rotational. It measures this state relative to the home configuration, the machine state *before* the application of initial condition actuators and assembly of disassembled joints. Thus the Joint Sensor measures the effect of the latter, which act before the simulation starts.

Reaction and Computed Force on a Joint

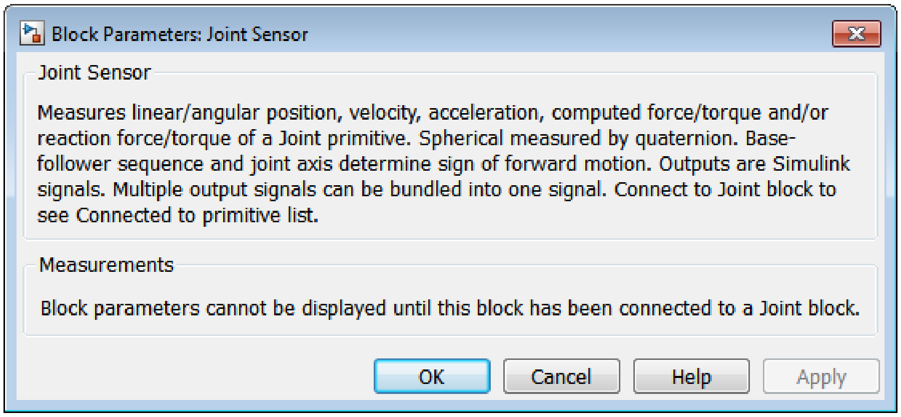
The *reaction force* and *torque* are three-component vectors of the force and torque that the joint transfers from the base Body to the follower Body.

The *computed force* is the component of the reaction force projected along the prismatic primitive axis. It is also the force along the prismatic axis that reproduces the follower motion with respect to the base.

The *computed torque* is the component of the reaction torque projected along the revolute primitive axis. It is also the torque about the revolute axis that reproduces the follower motion with respect to the base.



Joint Reaction and Computed Force (Prismatic Case)



Block Parameters: Joint Sensor

Joint Sensor

Measures linear/angular position, velocity, acceleration, computed force/torque and/or reaction force/torque of a Joint primitive. Spherical measured by quaternion. Base-follower sequence and joint axis determine sign of forward motion. Outputs are Simulink signals. Multiple output signals can be bundled into one signal. Connect to Joint block to see Connected to primitive list.

Measurements

Block parameters cannot be displayed until this block has been connected to a Joint block.

Dialog Box and Parameters

The dialog has one active area, **Measurements**. The block parameters are not displayed unless you connect it to a specific Joint block.

Measurements

Connected to primitive

In the pull-down menu, choose the joint primitive within the Joint that you want to measure with the Joint Sensor. A primitive Joint block has only one joint primitive.

If the Joint Sensor is not connected to a Joint block, this menu is not shown.

Output selected parameters as one signal

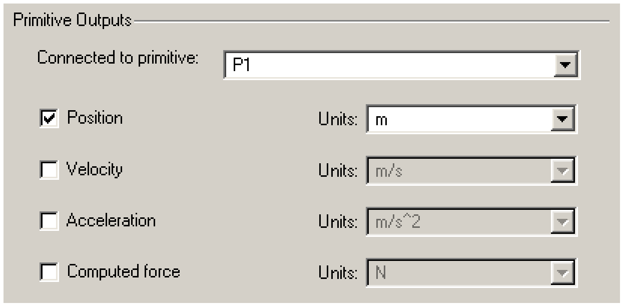
☑ Output selected parameters as one signal.

Select this check box to convert all the output signals into a single bundled signal. The default is selected. If you clear it, the Joint Sensor block will grow as many Simulink outports as there are active signals selected, in the same order top to bottom, in the dialog.

If the check box is selected, the Simulink signal out has all the active signals ordered into a single row vector. The order and type of the signal components depend on the joint primitive, as listed in the Simulink signal tables following.

The **Measurements** tab you see in the Joint Sensor dialog depends on the type of joint primitive to which you connect the Joint Sensor.

Measuring Prismatic Motion



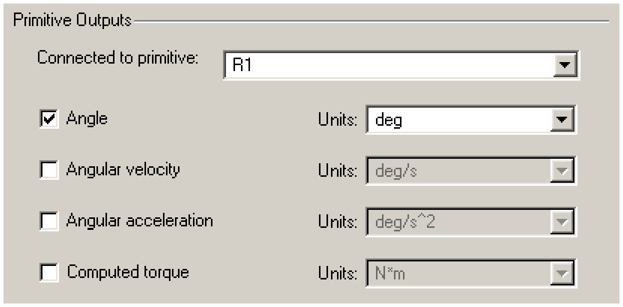
In the **Primitive Outputs** area, select the check box(es) for each of the possible measurements you want to make: **Position**, **Velocity**, **Acceleration**, and **Computed force**.

In the **Units** pull-down menus, choose the units for each of the measurements you want. The defaults are `m` (meters), `m/s` (meters/second), `m/s`^{2} (meters/second^2), `N` (newtons), respectively, for **Position**, **Velocity**, **Acceleration**, and **Computed force**.

The bundled Simulink output signal for a prismatic primitive has these measurements ordered in a row vector. Nonselected components are removed from the vector signal:

Position	Velocity	Acceleration	Computed Force	Reaction Torque (3-vector)	Reaction Force (3-vector)

Measuring Revolute Motion



Primitive Outputs

Connected to primitive: R1

[✓] Angle — Units: deg

[ ] Angular velocity — Units: deg/s

[ ] Angular acceleration — Units: deg/s^2

[ ] Computed torque — Units: N*m

In the **Primitive Outputs** area, select the check box(es) for each of the possible measurements you want to make: **Angle**, **Angular velocity**, **Angular acceleration**, and **Computed torque**.

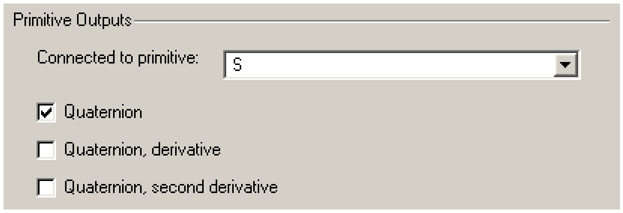
In the **Units** pull-down menus, choose the units for each of the measurements you want. The defaults are `deg` (degrees), `deg/s` (degrees/second), `deg/s`^{2} (degrees/second^2), `N*m` (newton-meters), and `N` (newtons), respectively, for **Angle**, **Angular Velocity**, **Angular Acceleration**, and **Computed torque**.

The bundled Simulink output signal for a revolute primitive has these measurements ordered in a row vector. Nonselected components are removed from the vector signal:

Angle	Angular Velocity	Angular Acceleration	Computed Torque	Reaction Torque (3-vector)	Reaction Force (3-vector)

Tip The absolute angle of revolute motion is mapped on to the interval (-180^o, +180^o] degrees or (-π,+π] radians.

Measuring Spherical Motion



Primitive Outputs

Connected to primitive: S

☑ Quaternion

☐ Quaternion, derivative

☐ Quaternion, second derivative

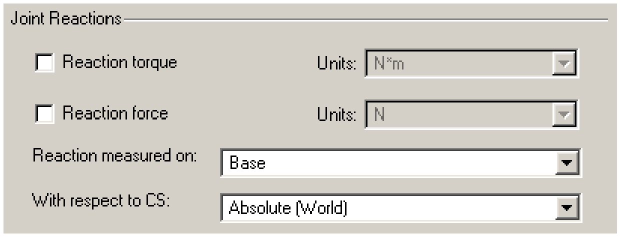
In the **Primitive Outputs** area, select the check box(es) for each of the possible measurements you want to make: **Quaternion**, **Quaternion, derivative**, and **Quaternion, second derivative**.

Quaternions are dimensionless, 4-component row vectors. The time unit for the derivatives is seconds.

The bundled Simulink output signal for a spherical primitive has these quaternion measurements ordered into a larger row vector. Nonselected components are removed from the vector signal:

Quaternion (4-vector)	Quaternion, derivative (4-vector)	Quaternion, second derivative (4-vector)	Reaction Torque (3-vector)	Reaction Force (3-vector)

Reaction Force and Torque



Joint Reactions

☐ Reaction torque Units: N*m

☐ Reaction force Units: N

Reaction measured on: Base

With respect to CS: Absolute (World)

In the **Joint Reactions** area, select the check box(es) for each of the possible measurements you want to make.

Reaction torque
: Select the check box to output the reaction torque.

Reaction force
: Select the check box to output the reaction force.

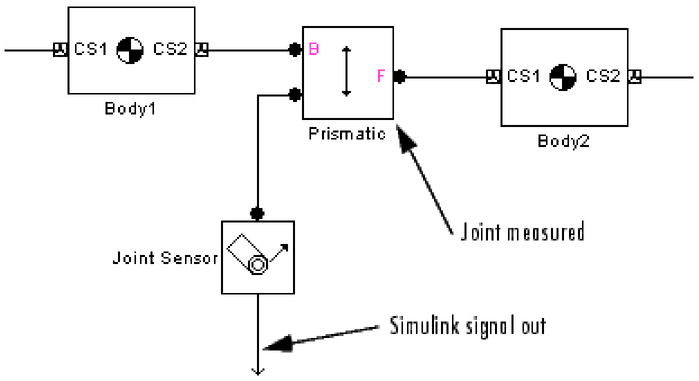
Reaction measured on
: Choose the Body on which the reaction force and torque vectors are measured, `Base` or `Follower`. The default is `Base`.

With respect to CS
: In the pull-down menu, choose the coordinate system in which the reaction torque and force vectors are measured: either the `Local (Body CS)` to which the Sensor is connected or the default `Absolute (World)`.

  In the `Absolute` case, the force and torque vectors have components measured relative to the inertial World CS axes. In the `Local` case, the same force and torque signals are premultiplied by the inverse orientation rotation matrix $R^{-1} = R^{T}$ for the Body selected in **Reactions measured on**.

Example

Here is a Joint Sensor connected to a Prismatic that connects two Bodies:



You must add an Sensor port (connector port) to the Joint block to connect the Joint Sensor to it. The base (B)-follower (F) Body sequence on the two sides of the Joint determines the sense of the Joint Sensor data.

See Also

Body Sensor, Constraint & Driver Sensor, Joint Actuator, Joint Initial Condition Actuator, Joint Stiction Actuator, Mechanical Branching Bar, Prismatic, Revolute, Spherical

See “Kinematics and Machine Motion State”, “Representations of Body Motion”, and “Sensing Motions and Forces”.

Joint Spring & Damper

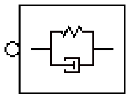
Purpose

Damped linear oscillator force or torque acting on a joint

Library

Force Elements

Description



The Joint Spring & Damper block models a damped linear oscillator force acting along a prismatic primitive or a damped linear oscillator torque acting about a revolute primitive. The joint primitives are connected between two bodies, and the force or torque acts between these bodies. The sign of the force or torque is set by the base (B)-to-follower (F) sequence of the bodies. These models represent damped linear translational and torsional springs in the prismatic and revolute cases, respectively.

You connect this block to a Joint at one of the Joint's sensor/actuator ports. (If the Joint lacks a sensor/actuator port, open its dialog and create one.) The Joint represents any mixture of translational and rotational degrees of freedom (DoFs). With the Joint Spring & Damper block, you can then apply any combination of damped linear oscillator forces on any prismatics and damped linear torsion torques on any revolutes.

Note Each Joint Spring & Damper block connected to a revolute primitive adds a normal Simulink state to your model.

This feature does not change the mechanical states of your model.

Joint Spring and Damper Theory

Connect two Bodies with a Joint having some combination of prismatic and revolute primitives.

Caution The Joint Spring & Damper uses a Joint Sensor to measure the degree of freedom in the Joint. These values are measured relative to the home configuration of the DoF, its state *before* the application of initial condition actuators and assembly of disassembled joints.

Translational Case

If x represents the displacement along a prismatic axis, and $v = dx/dt$ is the prismatic DoF's linear speed, then the damped spring force acting along this prismatic and between the Bodies connected by this Joint is

$$F = -k(x - x_0) - bv$$

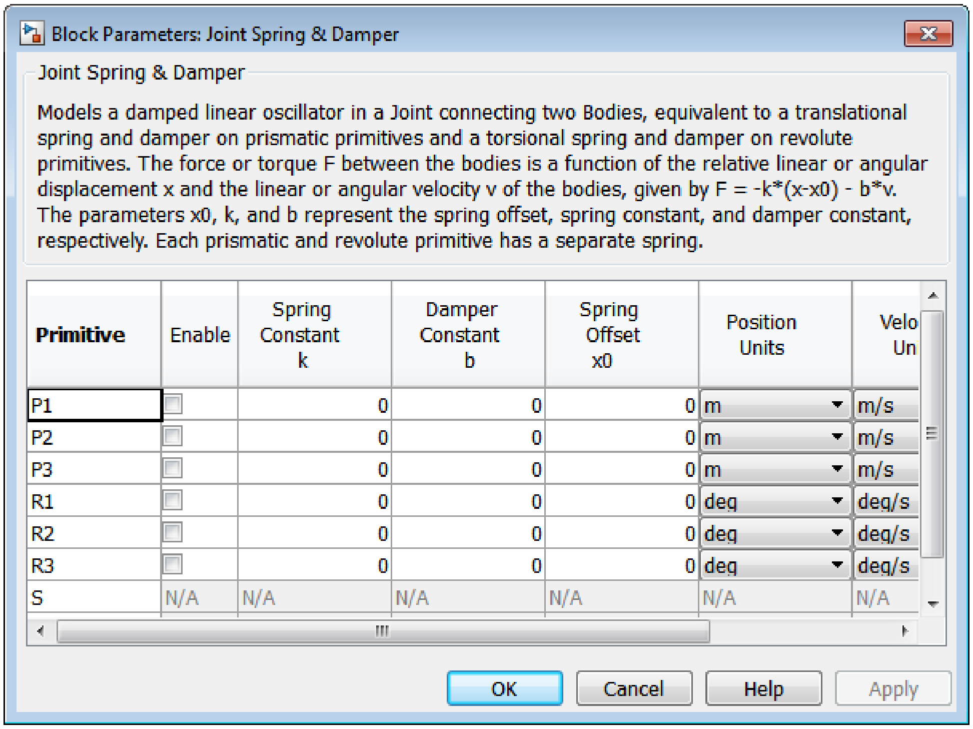
The model parameters are the spring constant k, the natural spring length (offset) x_0, and the damping constant b. The natural length is the spring's length with no forces acting on it and should be nonnegative: $x_0 \geq 0$. A stable spring requires $k > 0$. A damping representing dissipation and respecting the second law of thermodynamics requires $b \geq 0$. You can use a negative b to represent energy pumping.

Rotational Case

If θ represents the displacement about a revolute axis, and $\omega = d\theta/dt$ is the revolute DoF's angular speed, then the damped torsion torque acting about this revolute and between the Bodies connected by this Joint is

$$\tau = -k(\theta - \theta_0) - b\omega$$

The model parameters are the torsion constant k, the natural torsion angle (offset) θ_0, and the damping constant b. The natural angle is the torsion balance's direction with no torques acting on it and can have any sign. A stable torsion requires $k > 0$. A damping representing dissipation and respecting the second law of thermodynamics requires $b \geq 0$. You can use a negative b to represent energy pumping.


Block Parameters: Joint Spring & Damper

Joint Spring & Damper

Models a damped linear oscillator in a Joint connecting two Bodies, equivalent to a translational spring and damper on prismatic primitives and a torsional spring and damper on revolute primitives. The force or torque F between the bodies is a function of the relative linear or angular displacement x and the linear or angular velocity v of the bodies, given by F = -k*(x-x0) - b*v. The parameters x0, k, and b represent the spring offset, spring constant, and damper constant, respectively. Each prismatic and revolute primitive has a separate spring.

Primitive	Enable	Spring Constant k	Damper Constant b	Spring Offset x0	Position Units	Velo Un
P1	☐	0	0	0	m	m/s
P2	☐	0	0	0	m	m/s
P3	☐	0	0	0	m	m/s
R1	☐	0	0	0	deg	deg/s
R2	☐	0	0	0	deg	deg/s
R3	☐	0	0	0	deg	deg/s
S	N/A	N/A	N/A	N/A	N/A	N/A

Dialog Box and Parameters

Actuation

The menu lists all the active primitives in the Joint to which the Joint Spring & Damper block is connected. If you connect the Joint Spring & Damper with its dialog open, the primitive list is automatically updated to reflect the connected Joint's primitives.

Primitive

Lists the active primitives in the Joint to which the block is connected. `P` represents a prismatic primitive, `R` a revolute primitive, `S` a spherical primitive, and `W` a weld primitive.

Enable

To enable force or torque actuation on any particular primitive in the Joint, select the **Enable** check box next to that primitive's name in the Primitive column. You cannot actuate spherical or weld primitives.

Spring Constant k

Enter the spring or torsion constant *k*, for a prismatic or revolute primitive, respectively. The default is `0`.

The units for *k* are derived implicitly from your choice of position and force/torque units.

Damper Constant b

Enter the spring or torsion damping constant *b*, for a prismatic or revolute primitive, respectively. The default is `0`.

The units for *b* are derived implicitly from your choice of velocity and force/torque units.

Spring Offset x0

Enter the natural spring length x_0 or the natural torsion angle θ_0, for a prismatic or revolute primitive, respectively. The default is `0`.

Position Units

In the pull-down menu, select linear or angular units for prismatic or revolute primitives, respectively. The default is `m` (meters) or `deg` (degrees).

Velocity Units

In the pull-down menu, select linear or angular velocity units for prismatic or revolute primitives, respectively. The default is `m/s` (meters/second) or `deg/s` (degrees/second).

Force/Torque Units

In the pull-down menu, select force or torque units for prismatic or revolute primitives, respectively. The default is `N` (newtons) or `N*m` (newton-meters).

Restricted Parameters

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

- The **Actuation** (joint primitives) parameter table

See Also

Body, Body Spring & Damper, Custom Joint, Joint Actuator, Joint Sensor, Prismatic, Revolute

See “Adding Internal Forces”.

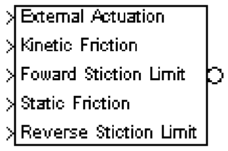
Purpose

Joint static and kinetic friction

Library

Sensors & Actuators

Description



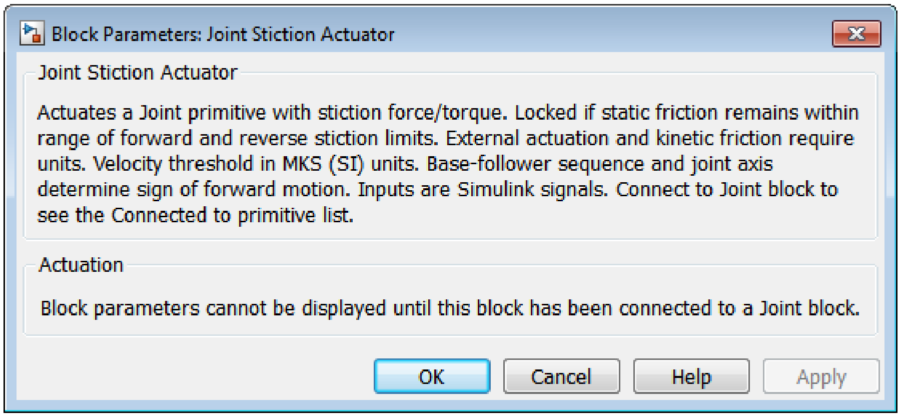
The Joint Stiction Actuator block applies Coulomb kinetic and static friction to a prismatic or revolute joint primitive. The stiction is regulated by a friction model whose parameters you specify. (See "Stiction Theory and Implementation" on page 2-147.) The Joint Stiction Actuator applies stiction to the joint primitive as a relative force/torque between the joint's connected Bodies. The bodies can experience additional forces independent of the applied stiction.

The inports are Simulink signals. The output is a connector port. You cannot connect a Joint Stiction Actuator to a Spherical block or spherical primitive. Restrictions on simultaneous actuators and sensors include:

- You cannot actuate a joint primitive simultaneously with a Joint Stiction Actuator and a Joint Actuator. But with the Joint Stiction Actuator inport `External Actuation`, you can apply to the joint primitive an external (nonfrictional) force/torque actuation signal equivalent to applying a Joint Actuator.
- You can simultaneously actuate a joint primitive with a Joint Stiction Actuator and a Joint Initial Condition Actuator.
- You can also simultaneously actuate a joint primitive with a Joint Stiction Actuator and measure the force/torque along/around the joint primitive with a Joint Sensor.

Caution You cannot trim or linearize a SimMechanics model that contains a Joint Stiction Actuator block.

The Joint Stiction Actuator block implements locking friction through non-time-increment simulation steps (algebraic loops) that display warnings at the MATLAB command line.



Joint Stiction Actuator

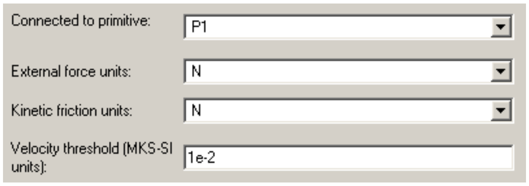
Actuates a Joint primitive with stiction force/torque. Locked if static friction remains within range of forward and reverse stiction limits. External actuation and kinetic friction require units. Velocity threshold in MKS (SI) units. Base-follower sequence and joint axis determine sign of forward motion. Inputs are Simulink signals. Connect to Joint block to see the Connected to primitive list.

Actuation

Block parameters cannot be displayed until this block has been connected to a Joint block.

Dialog Box and Parameters

The dialog has one active area, **Actuation**. The block parameters are not displayed unless you connect it to a specific Joint block.



Connected to primitive:	P1
External force units:	N
Kinetic friction units:	N
Velocity threshold (MKS-SI units):	1e-2

Connected to primitive

In the pull-down menu, choose the joint primitive within the Joint that you want to actuate with the Joint Stiction Actuator. A primitive Joint block has only one joint primitive.

You cannot connect a Joint Stiction Actuator to a spherical primitive.

If the Joint Stiction Actuator is not connected to a Joint block, this menu displays `Unknown`.

External force units

In the pull-down menu, choose units for the external nonfrictional force/torque F_{ext}. The default is `N` (newtons) if connected to a prismatic primitive and `N*m` (newton-meters) if connected to a revolute primitive.

Kinetic friction units

In the pull-down menu, choose units for the kinetic friction force/torque F_K. The default is `N` (newtons) if connected to a prismatic primitive and `N*m` (newton-meters) if connected to a revolute primitive.

Velocity threshold (MKS-SI units)

Enter the positive relative speed v_{th} of the joint primitive below which the joint locks by static friction. Above that speed, the joint is unlocked.

The units must be MKS or SI: for a prismatic primitive, meters/second; for a revolute primitive, radians/second.

Warning

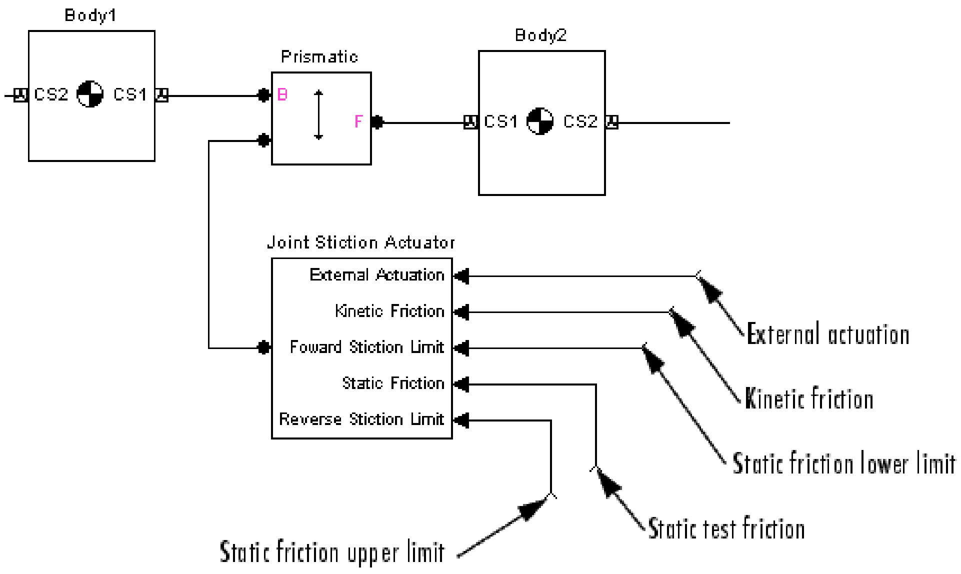
The velocity threshold v_{th} must be set greater than the Absolute tolerance in the Solver node of your model's Configuration Parameters dialog to avoid a meaningless threshold value.

Never set Absolute tolerance to `auto` if stiction actuators are present in a model. A recommended setting is to make v_{th} at least 10 times the Absolute tolerance value.

See "Configuring Methods of Solution" for more about setting simulation parameters.

Summary of Joint Stiction Actuator Inport Signals

All the Simulink inports are one-component signals. Here is an example of a prismatic joint connected between two bodies and actuated with stiction:



Joint Stiction Actuator Simulink® Inport Signals

Simulink Inport	Friction Model	Description
`External Actuation`	F_{ext}	External nonfrictional force/torque
`Kinetic Friction`	F_{K}	Kinetic friction
`Forward Stiction Limit`	$F_{\text{S}}^{\text{f}} < 0$	Static friction lower limit
`Static Test Friction`	F_{test}	Static test friction
`Reverse Stiction Limit`	$F_{\text{S}}^{\text{r}} > 0$	Static friction upper limit

Units

You specify units in the dialog only for the external nonfrictional and kinetic friction forces/torques, F_{ext} and F_K. These two friction signals are used to integrate the motion of the joint and have physical significance in the model. Thus units are necessary for F_{ext} and F_K.

The other three signals are compared only to one another in the locking condition $F_S^f < F_{test} < F_S^r$. These friction signals are not used to integrate motion and thus do not have units set in the dialog. But they must have the same implicit units for a valid comparison.

Example

The `mech_dpen_sticky` model in the Demos library has two revolute joints actuated with stiction. See "Joint Stiction Actuator Example: Mixed Static and Kinetic Friction".

Stiction Theory and Implementation

Kinematics

v and a are the velocity and acceleration along or around a joint primitive axis. These quantities are relative between the two bodies at the joint ends and signed ± to indicate forward or reverse. The joint directionality is set by the base (B)-to-follower (F) Body sequence of Bodies attached to the joint primitive being actuated.

Continuous Motion

A joint subject to stiction, if unlocked, moves in continuous motion. During this motion, you can apply two forces/torques at the joint primitive:

- A kinetic friction force/torque F_K:
  - $F_K < 0$ retards forward motion
  - $F_K > 0$ retards reverse motion
- An external, nonfrictional force/torque F_{ext}

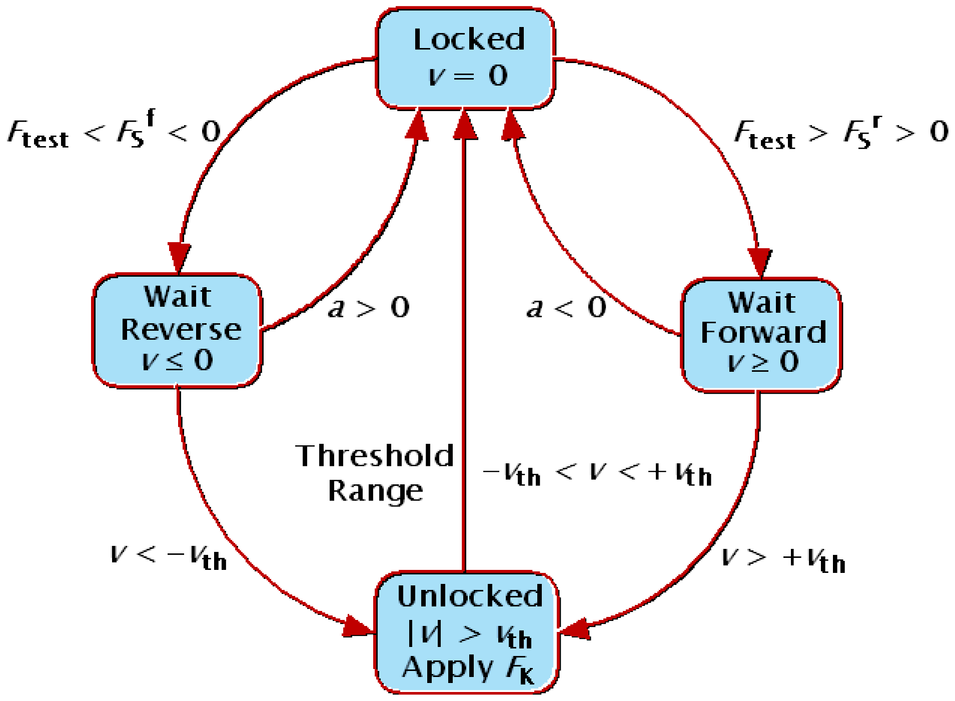
Discrete Joint Modes: Unlocked, Locked, Wait

A joint primitive actuated by stiction has potentially three discrete motion modes. The Joint Stiction Actuator switches the primitive among the three modes.

- In the *unlocked* mode, the joint primitive moves with the kinetic friction and external nonfrictional forces/torques applied.
- In the *locked* mode, the joint primitive locks rigidly and cannot move.
- In the *wait* mode, between locked and unlocked, the joint primitive is in a virtual motion state, awaiting determination of unlocking. The wait mode prevents infinite cycling between locked and unlocked modes, although it can noticeably slow down the simulation. The mode search uses a nonphysical algebraic loop.

Caution

The Joint Stiction Actuator uses a specialized type of *zero-crossing detection* (ZCD) to solve the joint locking and unlocking conditions. To avoid infinite loops and zero-crossing conflicts, disable any other ZCD conditions applied to a stiction-actuated joint by normal Simulink blocks connected directly or indirectly to it.



Joint Stiction Modes and Transition Conditions

Unlocking

You define the unlocking criteria by a two-condition threshold, constructed from four inputs that you specify.

- Joint unlocking threshold velocity $v_{th} > 0$ via the block dialog.
- *Static* friction limits $F_S{}^f < 0$ and $F_S{}^r > 0$ for forward and reverse motion, and a *static test friction* F_{test}, all three specified via Simulink signals. The static test friction F_{test} and forward/reverse limits $F_S{}^f$ and $F_S{}^r$ can be functions of the machine state and/or time.

The static test and kinetic frictions F_{test} and F_K can be discontinuous, but should be physically sensible.

Locking

You specify the locking criterion with the velocity threshold alone.

- Joint locking threshold velocity $v_{th} > 0$ via the block dialog.

Locked Mode

In this mode, v and a of the joint are zero. The static computed force/torque F_S at the joint is internally computed to maintain this mode: $F_{ext} + F_S + F_F - F_B = 0$. The forces/torques F_B, F_F are the forces/torques on the base and follower Bodies apart from those forces/torques acting at the joint.

The joint remains locked as long as $F_S^f < F_{test} < F_S^r$.

In most realistic friction models, you would set F_{test} equal to the computed F_S.

Wait Mode

If the static test friction F_{test} leaves the static friction range $[F_S^f, F_S^r]$, the joint has passed the first condition for unlocking, and the simulation enters wait mode, suspending the mechanical motion.

A search begins for a consistent state of all stiction-actuated joints in your model.

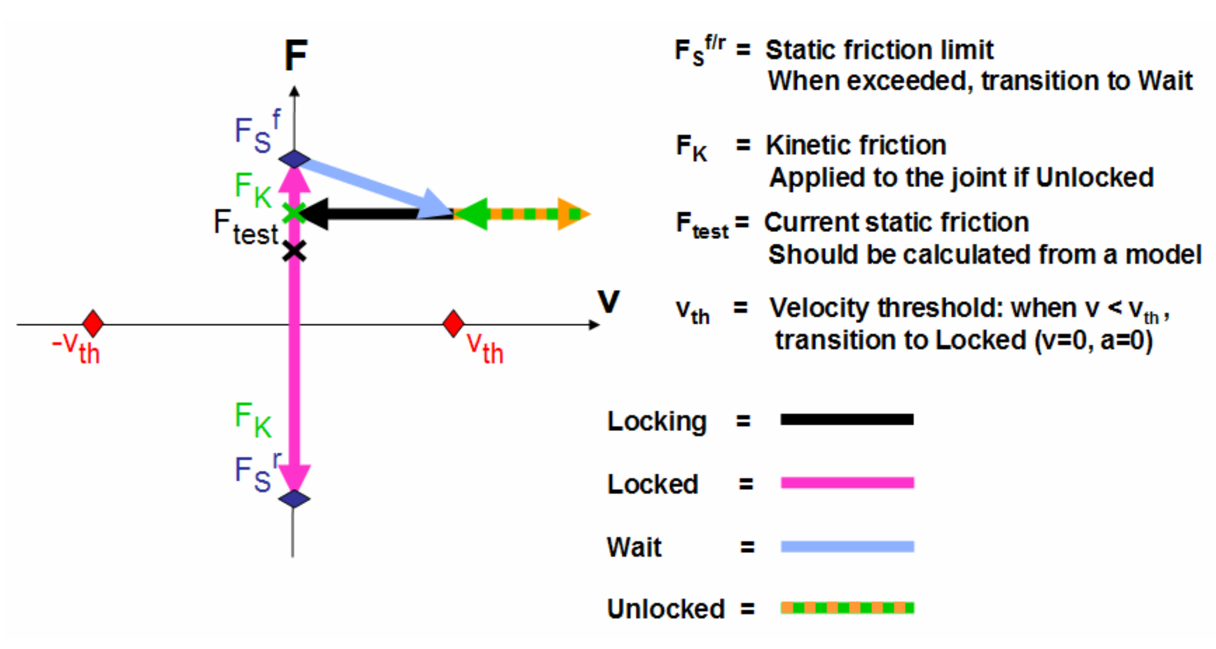
- The potential direction of motion after unlocking is determined by all the nonfrictional forces on the bodies.
- During the search, the net force/torque $F = F_{ext} + F_K$ at the joint primitive is computed, where F_K is the kinetic friction, and a is determined.
- For potential motion in the forward (reverse) direction, if $a < 0$ ($a > 0$), the search returns to the locked mode.

Once a consistent state for all stiction-actuated joints are found, mechanical motion restarts. The simulation integrates a to obtain v. When $|v|$ exceeds v_{th}, the second condition, the joint unlocks.

Unlocked Mode

In the unlocked mode, the joint primitive moves, actuated by the sum of the external, nonfrictional force/torque F_{ext} and the kinetic friction F_K.

The joint returns to the locked mode if v falls into the range $-v_{th} < v < +v_{th}$. If the simulation steps in time over this velocity range, it instead catches the zero of velocity with Simulink zero-crossing detection.


F
F_S^f
F_K
F_{test}
v
$-v_{th}$
v_{th}
F_K
F_S^r
$F_S^{f/r}$ = Static friction limit
When exceeded, transition to Wait
F_K = Kinetic friction
Applied to the joint if Unlocked
F_{test} = Current static friction
Should be calculated from a model
v_{th} = Velocity threshold: when $v < v_{th}$, transition to Locked (v=0, a=0)
Locking =
Locked =
Wait =
Unlocked =

Static and Kinetic Friction and Relative Velocity

Reference

[1] Moler, C. B., *Numerical Computing with MATLAB,* Philadelphia, Society for Industrial and Applied Mathematics, 2004, Chapter 7.

See Also

Joint Actuator, Joint Initial Condition Actuator, Joint Sensor, Mechanical Branching Bar, Prismatic, Revolute

See “Actuating a Joint”.

Linear Driver

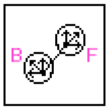
Purpose

Time-dependent signal of a vector position component between two body coordinate systems

Library

Constraints & Drivers

Description



The Linear Driver block specifies a component of the vector difference of Body coordinate system (CS) origins as a function of time.

Let $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ be the vector positions of the origins of CS1 on one Body, CS2 on the other Body, and $\boldsymbol{R} = \boldsymbol{r}_1 - \boldsymbol{r}_2$. The Linear Driver block specifies one of the vector components of R = *(X,Y,Z)*, projected on to the World CS axes, as a function of time:

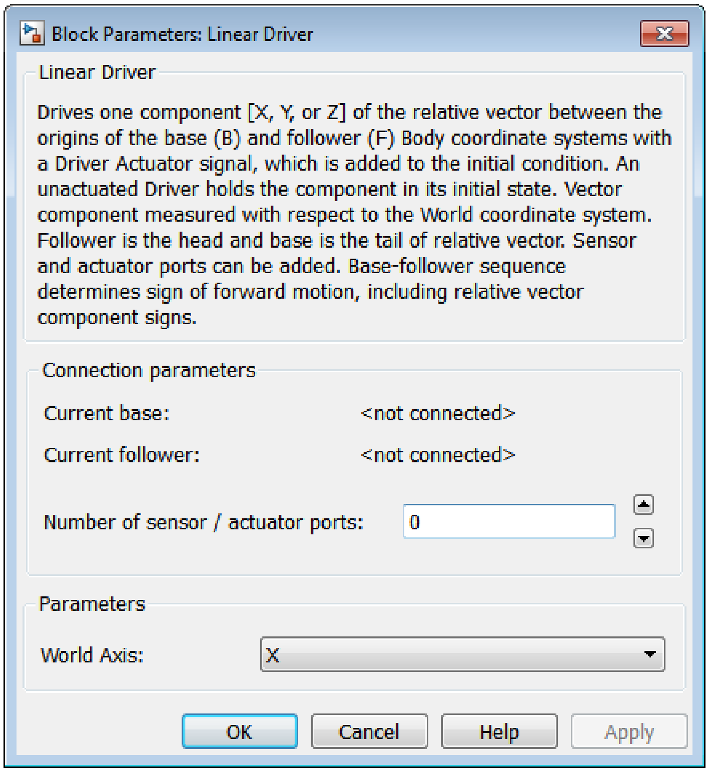
$$X,\ Y,\ \text{or}\ Z = X(t=0),\ Y(t=0),\ \text{or}\ Z(t=0) + f(t)$$

You connect a Driver Actuator block to the Linear Driver.

The Simulink input signal into the Driver Actuator specifies the time-dependent driving function *f(t)* and its first two derivatives, as well as their units. If you do not actuate Linear Driver, this block acts as a time-independent constraint that freezes the vector component between the two Body CS origins at its initial value *X(t=0)*, *Y(t=0)*, or *Z(t=0)* during the simulation.

Drivers restrict relative degrees of freedom (DoFs) between a pair of bodies as specified functions of time. Locally in a machine, they replace a Joint as the expression of the DoFs. Globally, Driver blocks must occur topologically in closed loops. Like Bodies connected to a Joint, the two Bodies connected to a Drivers are ordered as base and follower, fixing the direction of relative motion.

You can also connect a Constraint & Driver Sensor to any Driver and measure the reaction forces/torques between the driven bodies.


Block Parameters: Linear Driver

Linear Driver

Drives one component [X, Y, or Z] of the relative vector between the origins of the base (B) and follower (F) Body coordinate systems with a Driver Actuator signal, which is added to the initial condition. An unactuated Driver holds the component in its initial state. Vector component measured with respect to the World coordinate system. Follower is the head and base is the tail of relative vector. Sensor and actuator ports can be added. Base-follower sequence determines sign of forward motion, including relative vector component signs.

Connection parameters

Current base: <not connected>

Current follower: <not connected>

Number of sensor / actuator ports: 0

Parameters

World Axis: X

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive translation is the follower moving in the direction of the translation axis.

Current base

When you connect the base (B) connector port on the Linear Driver block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Linear Driver Base and Follower Body Connector Ports on page 2-154.

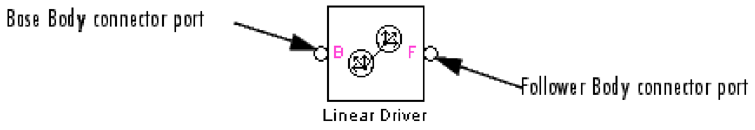
Current follower

When you connect the follower (F) connector port on the Linear Driver block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Linear Driver Base and Follower Body Connector Ports on page 2-154.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Driver Actuator and Constraint & Driver Sensor blocks to this Driver. The default is `0`.

To activate the Driver, connect a Driver Actuator.



Linear Driver Base and Follower Body Connector Ports

Parameters

World Axis

In the pull-down menu, choose the component of the vector difference ***R*** between the Body CS origins that you want to drive as a function of time. The components are measured with respect to the World CS axes. The choices are `X`, `Y`, or `Z`. The default is `X`.

See Also

Constraint & Driver Sensor, Distance Driver, Driver Actuator

See “Constraining and Driving Degrees of Freedom” for more on restricting DoFs with Drivers.

See “Checking Model Topology” and “How SimMechanics Software Works”.

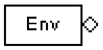
Purpose

Mechanical simulation parameters of a machine

Library

Bodies

Description



The Machine Environment block allows you to view and change the mechanical environment settings for one machine in your model.

Caution

A SimMechanics model consists of one or more machines. A machine is a complete, connected diagram of SimMechanics blocks topologically distinct from other complete SimMechanics block diagrams. Each machine must have one or more Ground blocks.

A machine can be a composite of submachines connected by Shared Environment blocks. Each submachine must have one or more Ground blocks.

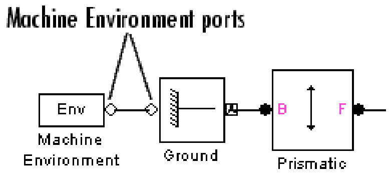
Exactly one Ground per machine, simple or composite, must be connected to a Machine Environment block for your SimMechanics model to be valid.

This block determines the following settings for the machine:

- How to simulate the machine
- How to interpret mechanical constraints
- How to linearize the simulation
- Whether and how to display the machine in visualization

The Machine Environment Port and Connecting the Block

You connect this block to a Ground by enabling that Ground's Machine Environment port from the Ground dialog.



Gravity as a Simulink Signal

This block also allows you to input gravity as a variable Simulink signal. If you choose to do this, a Simulink inport > also appears on the block for connection to a three-component Simulink signal line.

Opening the Simulink Configuration Parameters Dialog

You can open the Simulink Configuration Parameters dialog for viewing and editing by clicking the **Configuration Parameters** button on the lower left of the block dialog.

Dialog Box and Parameters

In the lower half of its dialog, the Machine Environment block has four active tabs that you can view and modify after selecting the corresponding tabs. You can apply your settings at any time by clicking **Apply** or **OK**.

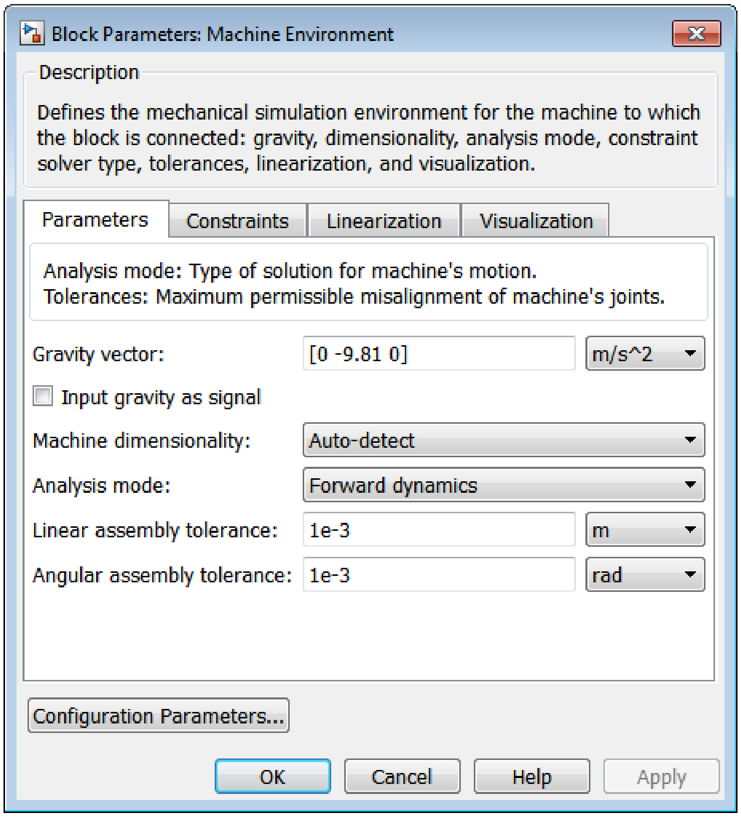
- **Parameters**
- **Constraints**
- **Linearization**
- **Visualization**

Restricted Parameters

When your model is in Restricted editing mode, you can modify all the fields in the dialog, with the exception of the following parameters:

- The **Analysis mode** pull-down menu
- The **Input gravity as signal** check box

These options are exceptions to the Simscape editing mode rules.



Block Parameters: Machine Environment

Description

Defines the mechanical simulation environment for the machine to which the block is connected: gravity, dimensionality, analysis mode, constraint solver type, tolerances, linearization, and visualization.

Analysis mode: Type of solution for machine's motion.
Tolerances: Maximum permissible misalignment of machine's joints.

Gravity vector: [0 -9.81 0] m/s^2

☐ Input gravity as signal

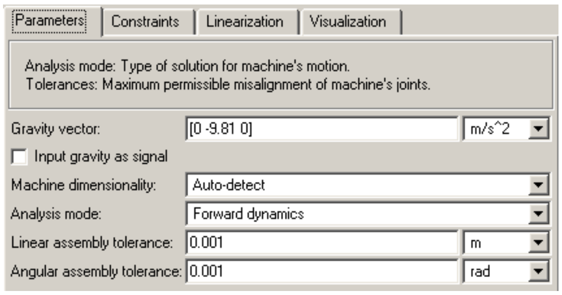
Machine dimensionality: Auto-detect

Analysis mode: Forward dynamics

Linear assembly tolerance: 1e-3 m

Angular assembly tolerance: 1e-3 rad

Configuring the Dynamics


Parameters | Constraints | Linearization | Visualization

Analysis mode: Type of solution for machine's motion.
Tolerances: Maximum permissible misalignment of machine's joints.

Gravity vector:	[0 -9.81 0]	m/s^2
☐ Input gravity as signal		
Machine dimensionality:	Auto-detect	
Analysis mode:	Forward dynamics	
Linear assembly tolerance:	0.001	m
Angular assembly tolerance:	0.001	rad

In this tab, you configure settings that control the mechanical dynamics.

Gravity vector

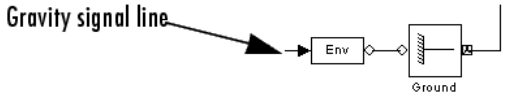
The value of this parameter is a MATLAB vector that specifies the magnitude and direction of gravitational acceleration in the model's world coordinate system. It must be a three-component vector. The default vector is `[0 -9.81 0]`. This field is disabled if you choose to input gravity as a signal.

The default units are `m/s`^{2} (meters per square second). Use the pull-down menu to the right if you want to reset the units.

Input gravity as signal

Select this check box if you want to disable the **Gravity vector** field and instead input gravity as a variable Simulink signal. The default is not selected.

If you select this check box, a Simulink inport appears on the block in addition to the existing Machine Environment port. You input the gravity vector as a three-component Simulink signal to this port. The components are, respectively, *x*, *y*, and *z*.



Machine dimensionality

In the pull-down menu, select in how many dimensions you want to simulate your machine: in `3D Only` or `2D Only`, or let the SimMechanics simulation choose for you with `Auto`. The default is `3D Only`.

You must take care, if you choose `2D Only`, that the machine actually moves in only two dimensions. If it does not, the simulation stops with an error.

Analysis mode

Specifies the type of analysis to be performed during the simulation. Choose one from the pull-down menu.

Analysis Mode	Description
`Forward dynamics`	Computes the positions and velocities of the system's bodies, given forces, torques, and initial conditions. This is the default mode.
`Inverse dynamics`	Computes the forces and torques required to produce the specified motions of an open machine.
`Kinematics`	Computes the forces and torques required to produce the specified motions of a closed-loop machine.
`Trimming`	Variant of Forward Dynamics mode to be used with the Simulink `trim` command. Determines steady-state or other points in system state space.

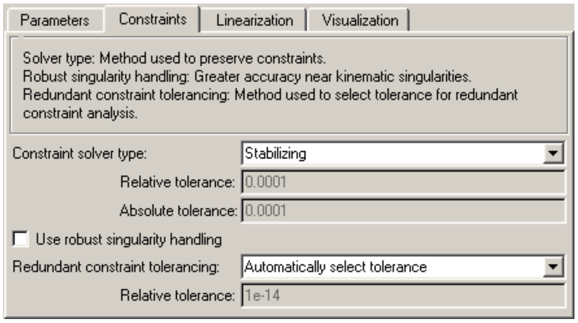
Linear assembly tolerance
Maximum position error allowed between bodies connected by prismatic joints. The default is `1e-3 m`. Use the menu on the right to set the units.

Angular assembly tolerance
Maximum angular error allowed between bodies connected by revolute joints. Default is `1e-3 rad`. Use the menu on the right to set the units.

Tunable Parameters

The **Gravity vector** field is tunable during simulation.

Implementing Constraints



Solver type: Method used to preserve constraints.
Robust singularity handling: Greater accuracy near kinematic singularities.
Redundant constraint tolerancing: Method used to select tolerance for redundant constraint analysis.

Constraint solver type: Stabilizing
Relative tolerance: 0.0001
Absolute tolerance: 0.0001
☐ Use robust singularity handling
Redundant constraint tolerancing: Automatically select tolerance
Relative tolerance: 1e-14

In this tab, you tell the SimMechanics simulation how to interpret mechanical constraints in machines that contain blocks from the Constraints & Drivers library; cut Joint, Constraint, and Driver blocks in closed loops; or both.

Constraint solver type
Type of solver used to solve constraints on the mechanical system's states. Choose one from the pull-down menu.

Solver Type	Description
`Stabilizing`	Adds a self-correcting term to the dynamics that stabilizes the numerical solution so that it drifts toward the constraint manifold. This is the default.
`Tolerancing`	Solves the constraints on the system's states to a specified degree of accuracy.
`Machine precision`	Solves the constraints to the numerical precision of the computer on which the simulation is running.

Relative tolerance

The relative tolerance used by the tolerancing constraint solver to determine when to stop refining a solution. Default is `1e-4`.

Enabled only if **Constraint solver type** is set to `Tolerancing`.

Absolute tolerance

The absolute tolerance used by the tolerancing constraint solver to determine when to stop refining the solution of a machine state. Default is `1e-4`.

Enabled only if **Constraint solver type** is set to `Tolerancing`.

Use robust singularity handling

Select this check box if you want Simulink to take extra steps to handle singularities in a system's equations of motion. The default is not selected.

This option increases the computational cost of solving a system's equations of motion, regardless of whether they have singularities. Select this option only as a last resort, i.e., only if the Simulink solvers cannot otherwise solve the system's equations of motion or require an excessively long time to do so.

Redundant constraint tolerancing

Select `Specify tolerance` in this pull-down menu if you want to control how precisely the SimMechanics simulation distinguishes constraints. The default is `Automatically select tolerance`.

This option is important if you have two or more constraints that impose almost identical restrictions on the motion of your machine. More constraints means fewer degrees of freedom.

- If two or more constraints are almost the same, the simulation eliminates one or more of them as redundant.
- If the constraints are dissimilar enough, the simulation treats them as independent constraints.

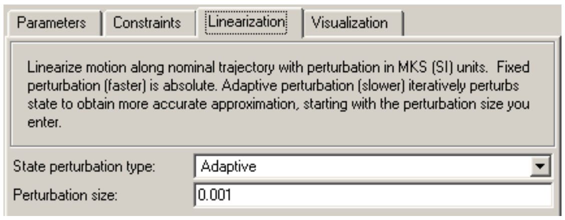
Selecting `Specify tolerance` enables the **Relative tolerance** field.

Relative tolerance

The relative tolerance of redundant constraint analysis that the simulation implements. This field is enabled only if you select `Specify tolerance` in the **Redundant constraint tolerancing** pull-down menu. The default is `1e-14`.

- Making this tolerance larger means the simulation treats similar constraints as the same, i.e., redundant.
- Making this tolerance smaller means the simulation treats similar constraints as distinct, i.e., not redundant.

Configuring Linearization



Linearize motion along nominal trajectory with perturbation in MKS (SI) units. Fixed perturbation (faster) is absolute. Adaptive perturbation (slower) iteratively perturbs state to obtain more accurate approximation, starting with the perturbation size you enter.

State perturbation type: Adaptive

Perturbation size: 0.001

In this tab, you tell the simulation how to linearize your machine.

State perturbation type

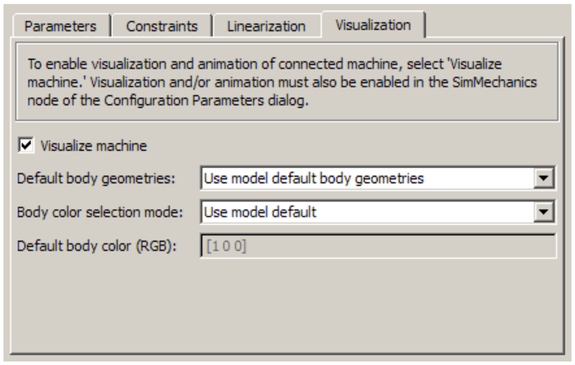
Specifies the type of state perturbation used by `linmod` to linearize a machine. The default is `Fixed`.

- `Adaptive` recomputes the size of the perturbation used at each step in the linearization process to ensure accurate computation of the linearization coefficients. It starts with the entry in the **Perturbation size** field as an initial guess.
- `Fixed` uses the perturbation size specified in the **Perturbation size** field for every step.

Perturbation size

Specifies the relative size of the perturbation used by the `Fixed perturbation` option. Specifies the relative size of the initial guess perturbation used by the `Adaptive` perturbation type. The perturbation size is relative to the size of the state being perturbed. The default is `1e-5`.

Configuring Machine Visualization



To enable visualization and animation of connected machine, select 'Visualize machine.' Visualization and/or animation must also be enabled in the SimMechanics node of the Configuration Parameters dialog.

[✓] Visualize machine

Default body geometries: Use model default body geometries

Body color selection mode: Use model default

Default body color (RGB): [1 0 0]

In this tab, you determine whether SimMechanics visualization displays this machine and choose the default body geometry (surface shape) and color for all the Bodies within the connected machine.

The machine inherits the model-wide defaults for body geometry and color. But you can change these machine-wide defaults to differ from the model-wide defaults.

Visualize machine

Select this check box if you want the machine to which this block is connected to appear in the visualization window. The default is selected.

Default body geometries

From the pull-down menu, select a machine-wide default body geometry.

- `Use model default body geometries` (the default)
- `Convex hull from Body CS locations` for convex hulls
- `Equivalent ellipsoid from mass properties` for equivalent ellipsoids

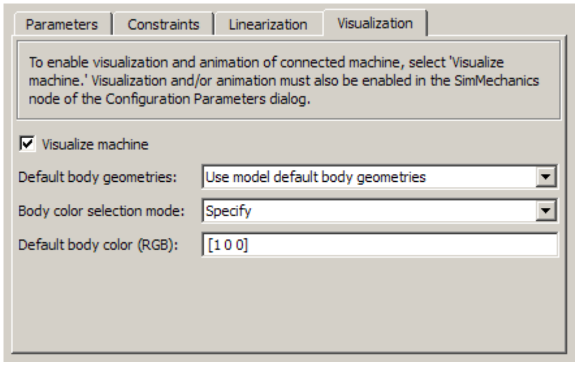
Body color selection mode

From the pull-down menu, choose whether to use the model-wide default for the color of all Bodies in this machine (the default), or to specify a machine-wide default different from the model-wide default.

Default body color (RGB)

If you select `Specify` in the **Body color selection mode** pull-down menu, you can specify the machine-wide default body color in the field. You specify RGB values according to the MATLAB Graphics `ColorSpec`.

The default is `[1 0 0]`.



To enable visualization and animation of connected machine, select 'Visualize machine.' Visualization and/or animation must also be enabled in the SimMechanics node of the Configuration Parameters dialog.

☑ Visualize machine

Default body geometries:	Use model default body geometries
Body color selection mode:	Specify
Default body color (RGB):	[1 0 0]

See Also

Body, Ground, Shared Environment

See the relevant entries in the Glossary: constraint, dynamics, ground, kinematics, machine, machine precision constraint, stabilizing constraint, and tolerancing constraint.

Setting Up, Configuring, and Running Machines and Models

For more about SimMechanics models and machines, see "Representing Machines with Models". For more about using Grounds and creating valid SimMechanics models, see "Modeling Grounds and Bodies" and "Validating Mechanical Models". For more about modeling constraints, see "Constraining and Driving Degrees of Freedom".

Visualizing Machines and Models

For more about configuring visualization for simulation, see "Starting Visualization and Simulation". For complete information about machine and body visualization, including default and custom body geometries and colors, see the SimMechanics Visualization and Import Guide.

Working Together with Simulink

For more about running SimMechanics software with Simulink, see "Configuring SimMechanics Models in Simulink", "Machine Settings via the Machine Environment Block", and "Configuring Methods of Solution".

For more about configuring simulations in Simulink, consult the section on the Configuration Parameters dialog in the Simulink documentation.

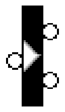
Purpose

Utility that maps multiple sensor and actuation signals into a single connection line

Library

Utilities

Description



The Mechanical Branching Bar bundles multiple actuator and sensor connection lines into one line, allowing you to connect multiple actuators and/or sensors to a single connector port on a Joint, Constraint, or Driver, or to a single Body coordinate system (CS) port on a Body. You can choose any number of sensor/actuator ports on the Mechanical Branching Bar.

- In the case of a Body, a single Body CS port represents a single Body CS. If the needed Body CS port does not exist, open the Body dialog and create one. You can connect the selected Body CS to multiple Body Actuators and Sensors through the Mechanical Branching Bar.
- In the case of a Joint, you need a single sensor/actuator port on the Joint. If the needed port does not exist, open the Joint's dialog and create one. You can connect this sensor/actuator port to multiple Actuators and Sensors through the Mechanical Branching Bar.

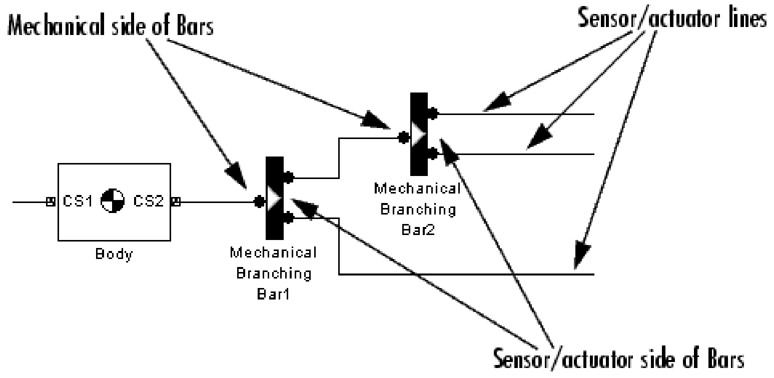
  Using the Mechanical Branching Bar, you can connect a Joint block to any combination of Joint Sensors, Joint Actuators, Joint Initial Condition Actuators, and Joint Stiction Actuators. The Actuator and Sensor dialogs display the Joint's primitives as if they were directly connected to the Joint.
- The procedure for Constraints and Drivers is the same as it is for Joints, except that you need to choose to measure reaction forces/torques or to actuate motions.

Cascading Mechanical Branching Bars and Avoiding Closed Loops

You can connect multiple Mechanical Branching Bar blocks in series, creating a cascade. Connect the mechanical side of the first Branching Bar to a Joint, Constraint, Driver, or Body. Then connect its sensor/actuator side to the mechanical side of the second Branching Bar, and so on.

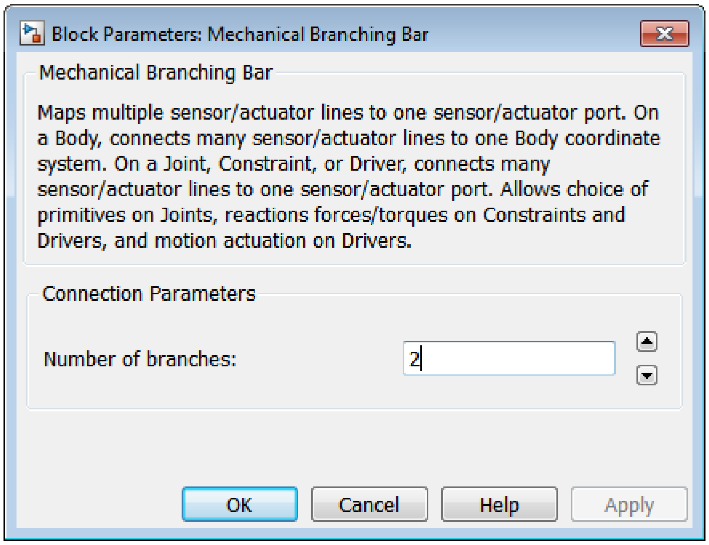
The only restriction on cascading Mechanical Branching Bars is that you must avoid connecting them into closed loops.

The following diagram shows a cascade, starting at a Body.


Mechanical side of Bars
Sensor/actuator lines
CS1
CS2
Body
Mechanical Branching Bar1
Mechanical Branching Bar2
Sensor/actuator side of Bars

Caution To avoid simulation errors, you should not create a cascade of Mechanical Branching Bars that closes on itself in a loop.

You should not connect the mechanical side of one Mechanical Branching Bar to the mechanical side of another Mechanical Branching Bar. You should also not connect the sensor/actuator side of one Mechanical Branching Bar to the sensor/actuator side of another Mechanical Branching Bar.


Block Parameters: Mechanical Branching Bar

Mechanical Branching Bar

Maps multiple sensor/actuator lines to one sensor/actuator port. On a Body, connects many sensor/actuator lines to one Body coordinate system. On a Joint, Constraint, or Driver, connects many sensor/actuator lines to one sensor/actuator port. Allows choice of primitives on Joints, reactions forces/torques on Constraints and Drivers, and motion actuation on Drivers.

Connection Parameters

Number of branches: 2

Dialog Box and Parameters

The dialog has one active area, **Connection parameters**.

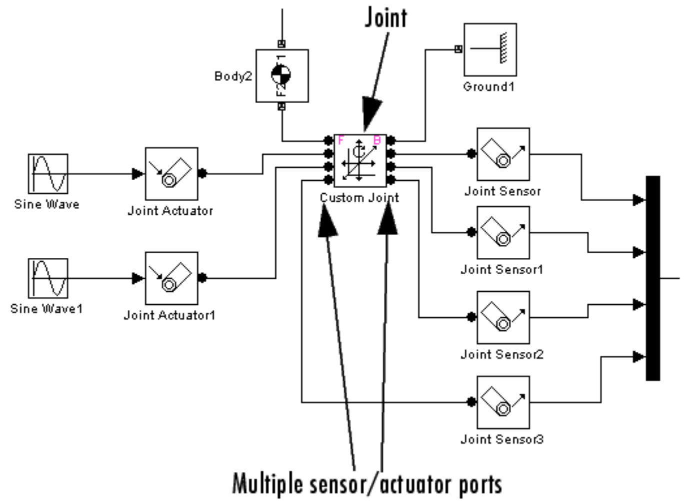
Connection Parameters

Number of branches

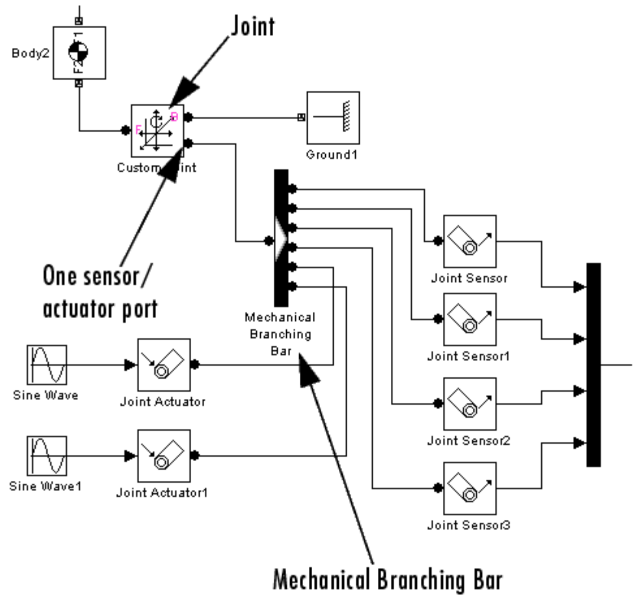
Using this spinner menu, you can set the number of extra connector ports needed for connecting Actuator and Sensor blocks to the Mechanical Branching Bar. The default is 2.

Example

Without the Mechanical Branching Bar, you must connect multiple Sensors and Actuators to a Joint by creating a sensor/actuator port on the Joint for each Sensor and each Actuator:



With the Mechanical Branching Bar block, you can combine all the sensor and actuator ports for a single Joint into one sensor/actuator port:



See Also

Body, Body Actuator, Body Sensor, Constraint & Driver Sensor, Driver Actuator, Joint Actuator, Joint Initial Condition Actuator, Joint Sensor, Joint Stiction Actuator

Parallel Constraint

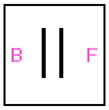
Purpose

Constant parallel relationship between two body axis vectors

Library

Constraints & Drivers

Description



The two Bodies connected by a Parallel Constraint are restricted in their relative rotational motion. The Parallel Constraint is connected on either side to a Body CS, one on each Body. A vector $\boldsymbol{a}_B$ defined in one Body CS on the base body remains parallel to a second vector $\boldsymbol{a}_F$ defined in another Body CS on the follower body.

The Parallel Constraint block requires that:

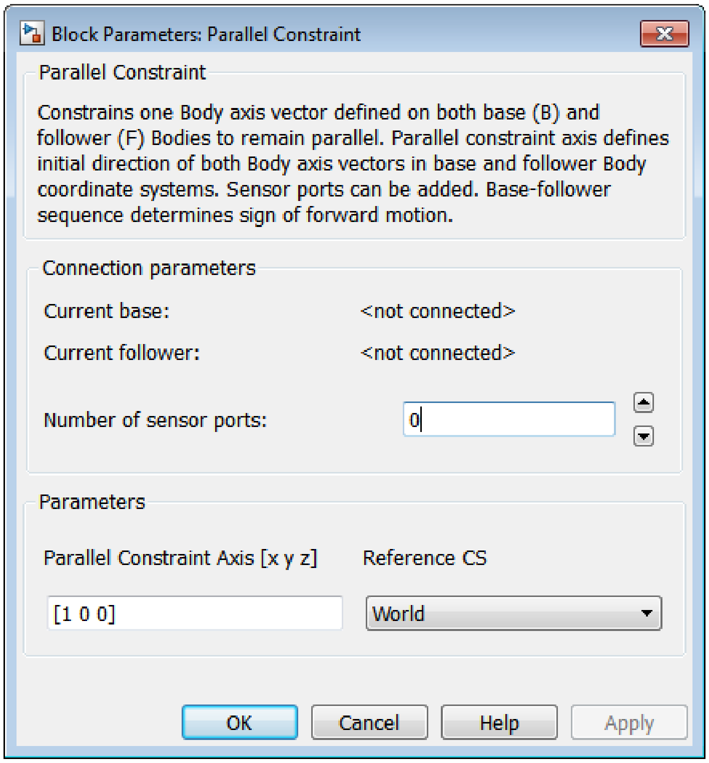
$$|\boldsymbol{a}_B \cdot \boldsymbol{a}_F| / (|\boldsymbol{a}_B||\boldsymbol{a}_F|) = 1$$

You specify the initial direction to which both vectors must remain parallel.

Constraints restrict relative degrees of freedom (DoFs) between a pair of bodies. Locally in a machine, they replace a Joint as the expression of the DoFs. Globally, Constraint blocks must occur topologically in closed loops. Like Bodies connected to a Joint, the two Bodies connected to a Constraint are ordered as base and follower, fixing the direction of relative motion.

Parallel Constraint is assembled: the Body CS origin on the base body must be initially collocated with the Body CS origin on the follower body, to within assembly tolerance.

You can connect a Constraint & Driver Sensor to any Constraint block, but not a Driver Actuator. The Constraint & Driver Sensor measures the reaction forces/torques between the constrained bodies.



Block Parameters: Parallel Constraint

Parallel Constraint

Constrains one Body axis vector defined on both base (B) and follower (F) Bodies to remain parallel. Parallel constraint axis defines initial direction of both Body axis vectors in base and follower Body coordinate systems. Sensor ports can be added. Base-follower sequence determines sign of forward motion.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>
Number of sensor ports:	0

Parameters

Parallel Constraint Axis [x y z]	Reference CS
[1 0 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive rotation is the follower rotating in the right-handed sense about the rotation axis.

Current base

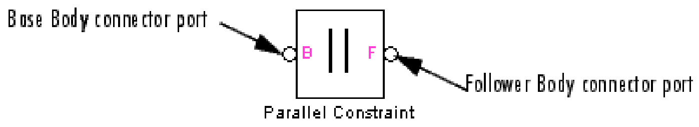
When you connect the base (B) connector port on the Parallel Constraint block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Parallel Constraint Base and Follower Body Connector Ports on page 2-174.

Current follower

When you connect the follower (F) connector port on the Parallel Constraint block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Parallel Constraint Base and Follower Body Connector Ports on page 2-174.

Number of sensor ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Constraint & Driver Sensor blocks to this Constraint. The default is `0`.



Parallel Constraint Base and Follower Body Connector Ports

Parameters

Parallel Constraint Axis [x y z]

Enter the axis vector defining the initial direction of the two body axis vectors $\boldsymbol{a}_b$, $\boldsymbol{a}_f$. These body axis vectors are restricted to always remain parallel to this initial axis. The default is `[1 0 0]`.

Reference CS

Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the initial **Parallel constraint axis** is oriented with respect to. This CS also determines the absolute meaning of reaction forces/torques at this Constraint. The default is `World`.

See Also

Angle Driver, Constraint & Driver Sensor, Velocity Driver

See “Constraining and Driving Degrees of Freedom” for more on restricting DoFs with Constraints.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on using constraints in closed loops.

Planar

Purpose

Joint with one revolute and two prismatic joint primitives

Library

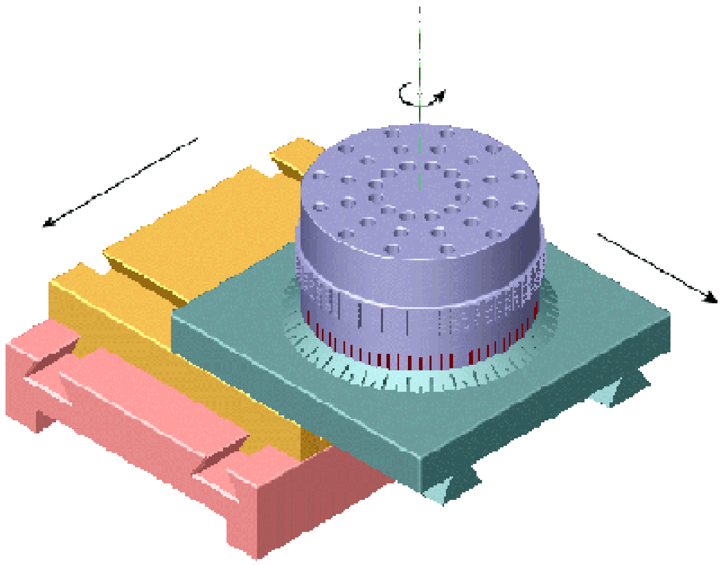
Joints

Description

The Planar block represents a composite joint with two translational degrees of freedom (DoFs) as two prismatic primitives and one rotational DoFs as one revolute primitives. The rotation axis must be orthogonal to the plane defined by the two translation axes.

Warning

A joint with two prismatic primitives becomes singular if the two translation axes become parallel. The simulation stops with an error in this case.



Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

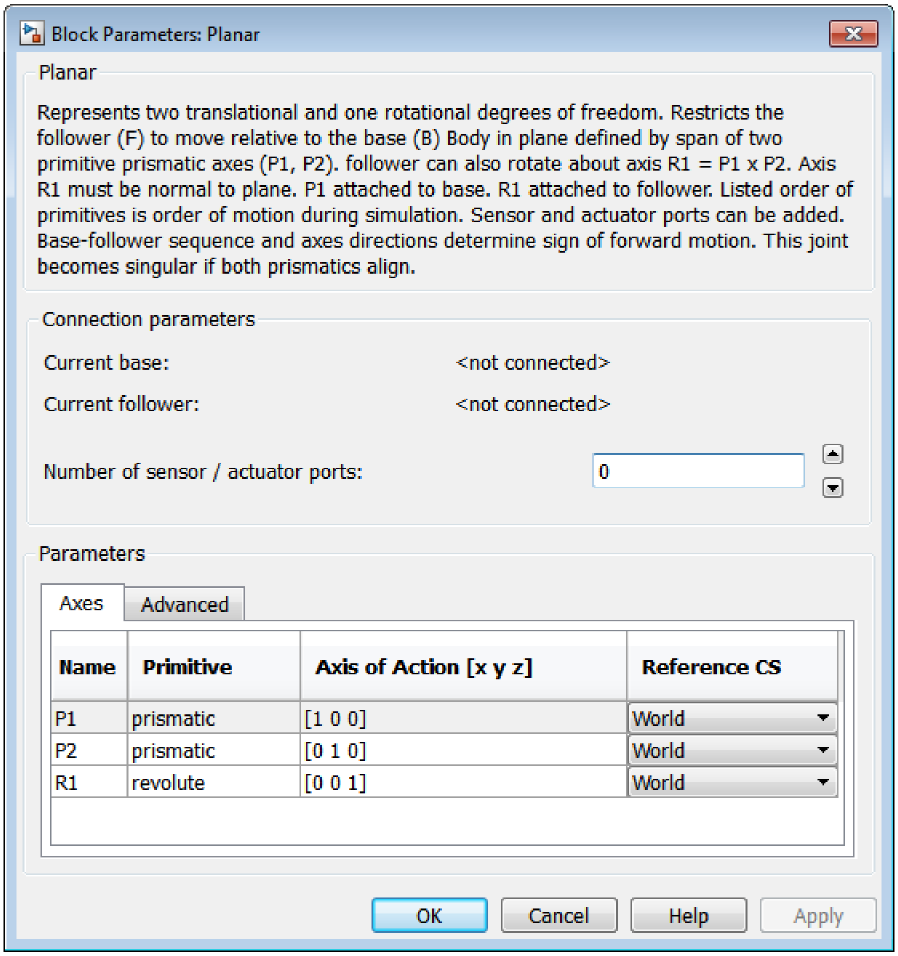
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space



Planar

Represents two translational and one rotational degrees of freedom. Restricts the follower (F) to move relative to the base (B) Body in plane defined by span of two primitive prismatic axes (P1, P2). follower can also rotate about axis R1 = P1 x P2. Axis R1 must be normal to plane. P1 attached to base. R1 attached to follower. Listed order of primitives is order of motion during simulation. Sensor and actuator ports can be added. Base-follower sequence and axes directions determine sign of forward motion. This joint becomes singular if both prismatics align.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>
Number of sensor / actuator ports:	0

Parameters

Axes | Advanced

Name	Primitive	Axis of Action [x y z]	Reference CS
P1	prismatic	[1 0 0]	World
P2	prismatic	[0 1 0]	World
R1	revolute	[0 0 1]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive translation is the follower moving in the direction of the translation axis. Positive rotation is the follower moving around the rotational axis following the right-hand rule.

Current base

When you connect the base (B) connector port on the Planar block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Planar Base and Follower Body Connector Ports on page 2-179.

The base Body is automatically connected to the first joint primitive `P1` in the primitive list in **Parameters**.

Current follower

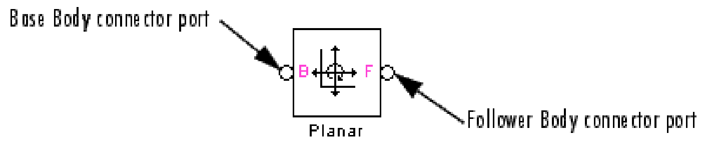
When you connect the follower (F) connector port on the Planar block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Planar Base and Follower Body Connector Ports on page 2-179.

The follower Body is automatically connected to the last joint primitive `R1` in the primitive list in **Parameters**.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`.

The motions of prismatic and revolute primitives are specified in linear and angular units, respectively.



Planar Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. Each DoF primitive in Planar has an entry line. These lines specify the direction of the axes of action of the DoFs that the Planar represents.

Name - Primitive
The primitive list states the names and types of joint primitives that make up the Planar block: prismatic primitives `P1`, `P2` and revolute primitives `R1`.

Axis of Action [x y z]
Enter here as a three-component vector the directional axes defining the allowed motions of these primitives and their corresponding DoFs:

- Prismatic: axis of translation
- Revolute: axis of rotation

The default vectors are shown in the dialog above. The axis is a directed vector whose overall sign matters.

To prevent singularities and simulation errors, the two prismatic axes cannot be parallel.

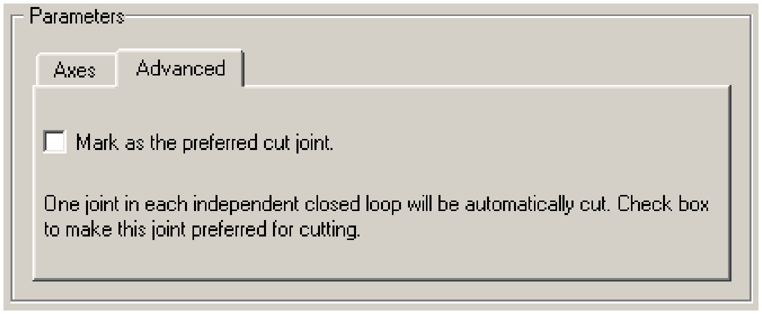
Reference CS
Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of action is oriented with respect to. This CS also determines the absolute meaning of forces/torques and motion along/about the joint axis. The default is `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

In-Plane, Prismatic, Revolute

See "Modeling Degrees of Freedom" for more on representing DoFs with Joints.

See "Checking Model Topology" and "How SimMechanics Software Works" for more on closed loops and cutting.

Point-Curve Constraint

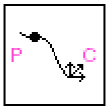
Purpose

Constraint that restricts body motion to a specified path

Library

Constraints & Drivers

Description



The two Bodies connected by a Point-Curve Constraint can only move relative to one another if a point on one body moves along a curve on the other body. The point on one body is the origin of the Body coordinate system (CS) to which one side of the Point-Curve Constraint is connected. The corresponding curve starting point on the other body is the origin of the Body CS to which the other side of the Point-Curve Constraint is connected. The point is constrained to move along the curve and cannot move perpendicularly to the curve.

Specifying the Curve You specify the curve function on the second body as a *spline* with break points and end conditions. The spline is a piecewise cubic polynomial, with the pieces joined at *breakpoints* that you specify:

$$(x_1,y_1,z_1)\ ,\ (x_2,y_2,z_2)\ ,\ \ldots\ ,\ (x_N,y_N,z_N)$$

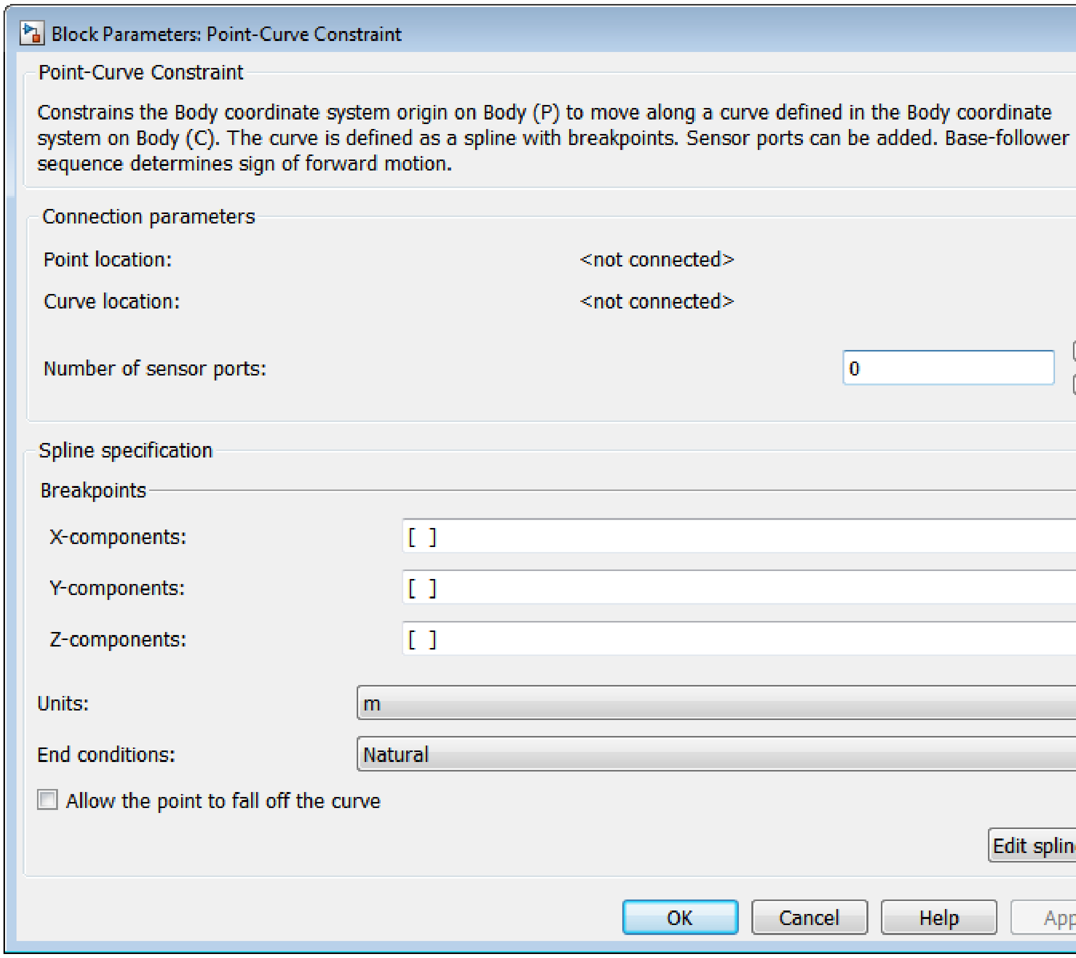
and boundary conditions applied at the spline's *endpoints*, (x_0,y_0,z_0) and $(x_{N+1},y_{N+1},z_{N+1})$. The spline curve and its first two derivatives are continuous at each breakpoint.

Constraints restrict relative degrees of freedom (DoFs) between a pair of bodies. Locally in a machine, they replace a Joint as the expression of the DoFs. Globally, Constraint blocks must occur topologically in closed loops. Like Bodies connected to a Joint, the two Bodies connected to a Constraint are ordered as base and follower, fixing the direction of relative motion.

For the Point-Curve Constraint, the base (P) is the Body carrying the point, and the follower (C) is the Body carrying the curve. The Point-Curve Constraint is assembled: the Body CS origin on the base

(Point) body must be initially collocated with the Body CS origin on the follower (Curve) body, to within assembly tolerance.

You can connect a Constraint & Driver Sensor to any Constraint block, but not a Driver Actuator. The Constraint & Driver Sensor measures the reaction forces/torques between the constrained bodies.


Block Parameters: Point-Curve Constraint

Point-Curve Constraint

Constrains the Body coordinate system origin on Body (P) to move along a curve defined in the Body coordinate system on Body (C). The curve is defined as a spline with breakpoints. Sensor ports can be added. Base-follower sequence determines sign of forward motion.

Connection parameters

Point location:	<not connected>
Curve location:	<not connected>
Number of sensor ports:	0

Spline specification

Breakpoints

X-components:	[ ]
Y-components:	[ ]
Z-components:	[ ]
Units:	m
End conditions:	Natural

☐ Allow the point to fall off the curve

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Spline specification**. It stores the defining information of a single spline for the constraint.

Connection Parameters

The base (P)-follower (C) Body sequence determines the sense of positive motion. Positive translation is the follower moving in the direction of the translation axis.

Point location

When you connect the base (P) connector port on the Point-Curve Constraint block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Point-Curve Constraint Base and Follower Body Connector Ports on page 2-185.

This Body CS origin is the point of the Point-Curve Constraint.

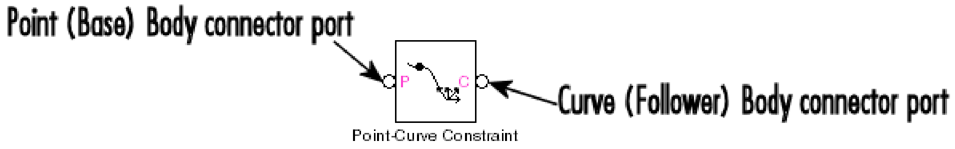
Curve location

When you connect the follower (C) connector port on the Point-Curve Constraint block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Point-Curve Constraint Base and Follower Body Connector Ports on page 2-185.

This Body CS origin is the starting point of the curve of the Point-Curve Constraint.

Number of sensor ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Constraint & Driver Sensor blocks to this Constraint. The default is `0`.


Point (Base) Body connector port
P
C
Curve (Follower) Body connector port
Point-Curve Constraint

Point-Curve Constraint Base and Follower Body Connector Ports

Specifying the Spline

The Point-Curve Constraint dialog gives you two ways to specify the spline curve. The first way is entering in this dialog the coordinates of breakpoints and endpoints on the follower and is valid for defining curves in up to three dimensions.

The second way is graphically displaying and editing the spline in the spline editor (see following), valid only for two-dimensional curves on the follower.

Breakpoints

List here the *x*-components, *y*-components, and *z*-components, respectively, of the breakpoints and endpoints that define the spline:

X-components: enter $(x_0, x_1, ..., x_{N+1})$ as a vector.

Y-components: enter $(y_0, y_1, ..., y_{N+1})$ as a vector.

Z-components: enter $(z_0, z_1, ..., z_{N+1})$ as a vector.

All three fields require nonnull entries. The number of components in each vector should be the same. *Exception and shortcut:* if all the ***Z*** components are the same, just enter one number in the ***Z*** vector. The **Breakpoints** list replicates this number to expand out a full vector.

If there are no ***X*** and/or ***Y*** components, you must still enter `[0 ... 0]` in that/those field(s). If there are no ***Z*** components, you must still enter at least `[0]` in the ***Z*** field (using the replication/expansion shortcut).

The pull-down menu for each spatial dimension lists the history of those previous breakpoints created by the graphical spline editor (see following) within a single dialog session. Closing the dialog destroys this history, and only the current breakpoint list is retained.

Units

In the pull-down menu, choose the linear units for distances on the constrained bodies. The default is `m` (meters).

End conditions

In the pull-down menu, choose the type of end (boundary) condition on the spline curve. The possible conditions are:

End Condition	Definition	Minimum Number of Points	Notes
`Natural`	Match each endslope to the slope of the cubic that fits the first four points at that end	Two points	Default
`Not-a-knot`	Only the curve and its first derivative are continuous at first and last interior points	Four points	
`Periodic`	Match the first and second derivatives of the two endpoints	Two points (three recommended)	This choice closes the spline by connecting the endpoints

Allow the point to fall off the curve

If the check box is selected, the base point continues with unconstrained motion if it reaches an endpoint and leaves the spline on the follower. The direction of motion at the instant the base point leaves the constraint is tangent to the spline.

If the check box is not selected, and the base point attempts to leave the spline on the follower, the simulation stops with an error. The default is not selected.

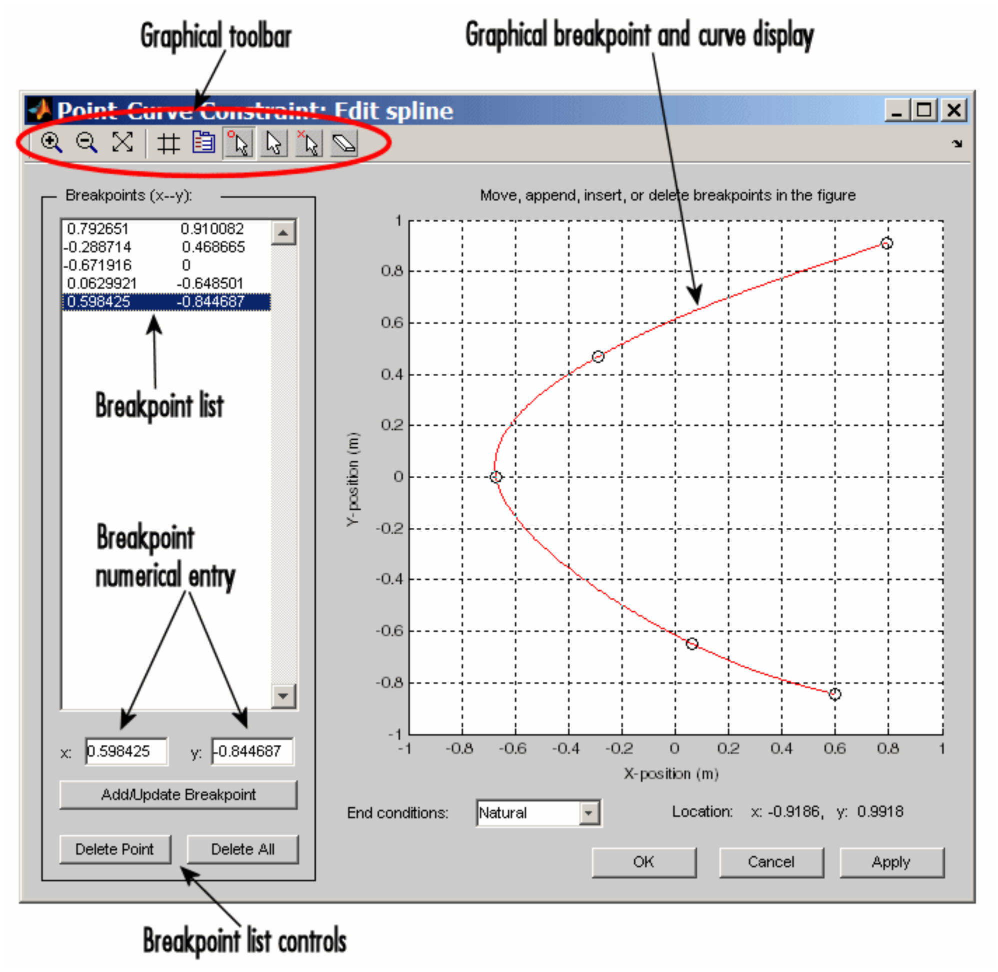
Edit spline

Click here to open the optional Edit spline dialog.

The Edit spline dialog provides alternative numerical entry and graphical editing methods for defining the constraint spline. But it can define only two-dimensional curves in the *x*-*y* coordinate directions on the follower Body. The spline editor ignores any *z*-components in existing breakpoints.

Edit Spline

The numerical entry area lies on the left side of the Edit spline dialog, the graphical editing area on the right side.



0.792651	0.910082
-0.288714	0.468665
-0.671916	0
0.0629921	-0.648501
0.598425	-0.844687

x: 0.598425 y: -0.844687

Location: x: -0.9186, y: 0.9918

Point-Curve Constraint Spline Editor

Graphical Editing of Spline Points

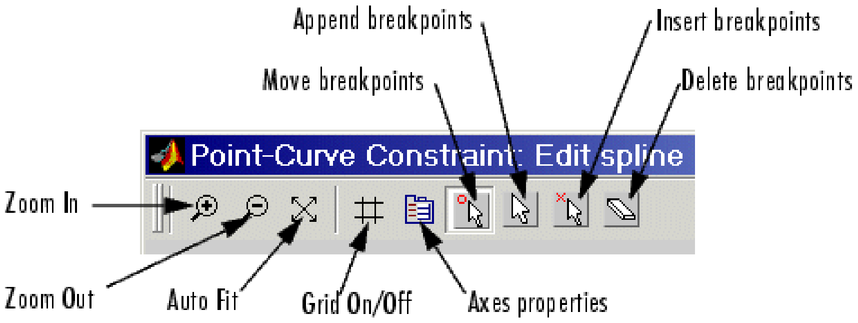
1 To place a breakpoint in the graphical display, place your cursor at the position where you want the breakpoint. The **Location** display

in the lower right indicates your current cursor coordinates in the curve display.

2 Then click at the desired point. A circle appears where you clicked, and simultaneously, the breakpoint is listed in the **Breakpoints (x-y)** list.

Continuing to add breakpoints generates the spline (red curve).

3 Use the Graphical toolbar controls to edit the spline graphically in the display:



- Remove points by clicking on the **Delete breakpoints** icon. Your cursor turns into an eraser symbol. With it, select and click the breakpoints you want to delete.
- Insert new (interior) breakpoints by clicking on the **Insert breakpoints** icon. Your cursor acquires a small circle. Click on the positions, near the existing curve, where you want the new breakpoints. The editor modifies the spline to fit the new breakpoints.
- Add new endpoints and extend the curve by clicking on the **Append breakpoints** icon. Your cursor acquires a small circle. Click on the positions, near the existing endpoints, to where you want to extend the curve. The editor modifies the spline to fit the new endpoints.

- Move existing endpoints by clicking the **Move breakpoints** icon. Click and drag the breakpoints you want to move, then drop them where you want them.

The editor modifies both the spline red curve in the graphical display and the **Breakpoints (x-y)** list as you make these changes.

Additional graphical toolbar controls:

- **Zoom In/Zoom Out** and **Auto Fit**: Standard MATLAB Graphics zooming and auto resizing of graphics display.
- **Axes properties**: Edit properties of graphical display.
- **Grid On/Off**: Turn the graphical display *x*-*y* grid on or off.

Numerical Editing of Spline Points

Use the numerical entry controls, instead of the graphical editing tools, to edit breakpoints by text entry.

Breakpoints (x-y)

You can also add, delete, and edit the breakpoints via this breakpoints list:

- Select an existing breakpoint by highlighting it with your cursor.
- Add a breakpoint by moving the highlighted selection to the empty line below the last breakpoint with your cursor control.
- In the **x:** and **y:** fields, enter the *x*- and *y*-coordinates of the currently selected breakpoint.

Add/Update Breakpoint

After editing an existing breakpoint or entering a new one in the **x:–y:** fields, update the breakpoint list by clicking here.

The new or changed breakpoint appears in the graphical display as a circle.

Delete Point

Click here to delete the currently selected breakpoint.

Delete All

Click here to delete all the breakpoints in the breakpoint list.

End conditions

In the pull-down menu, choose the type of end (boundary) condition on the spline curve. The possible conditions are `natural`, `not-a-knot`, and `periodic`. The default is `natural`.

Closing the Edit Spline Dialog

Clicking **Apply** or **OK** updates the breakpoints stored in the main Point-Curve Constraint dialog.

Previous breakpoint lists are stored in the history pull-down menus of the main Point-Curve Constraint dialog's **Breakpoints** list. This history is destroyed if you close the main dialog, and only the current breakpoint list is retained.

See Also

Constraint & Driver Sensor

See "Constraining and Driving Degrees of Freedom" for more on restricting DoFs with Constraints.

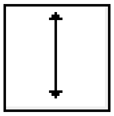
See "Checking Model Topology" and "How SimMechanics Software Works" for more on using constraints in closed loops.

For more about representing curves as splines, see the *Curve Fitting Toolbox documentation*.

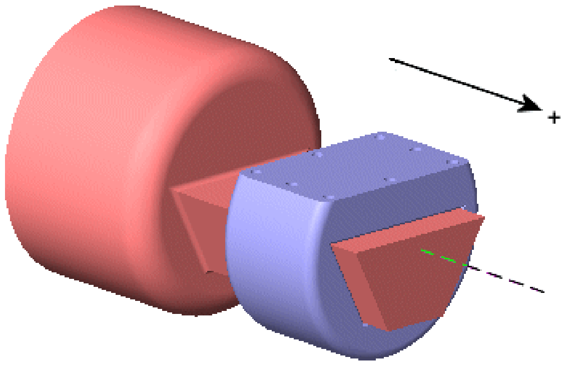
Purpose Primitive joint with one translational degree of freedom

Library Joints

Description



The Prismatic block represents a single translational degrees of freedom (DoF) along a specified axis between two bodies. A prismatic joint is one of SimMechanics primitive joints, along with revolute and spherical.



Prismatic Motion of Follower (blue) Relative to Base (red)

Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

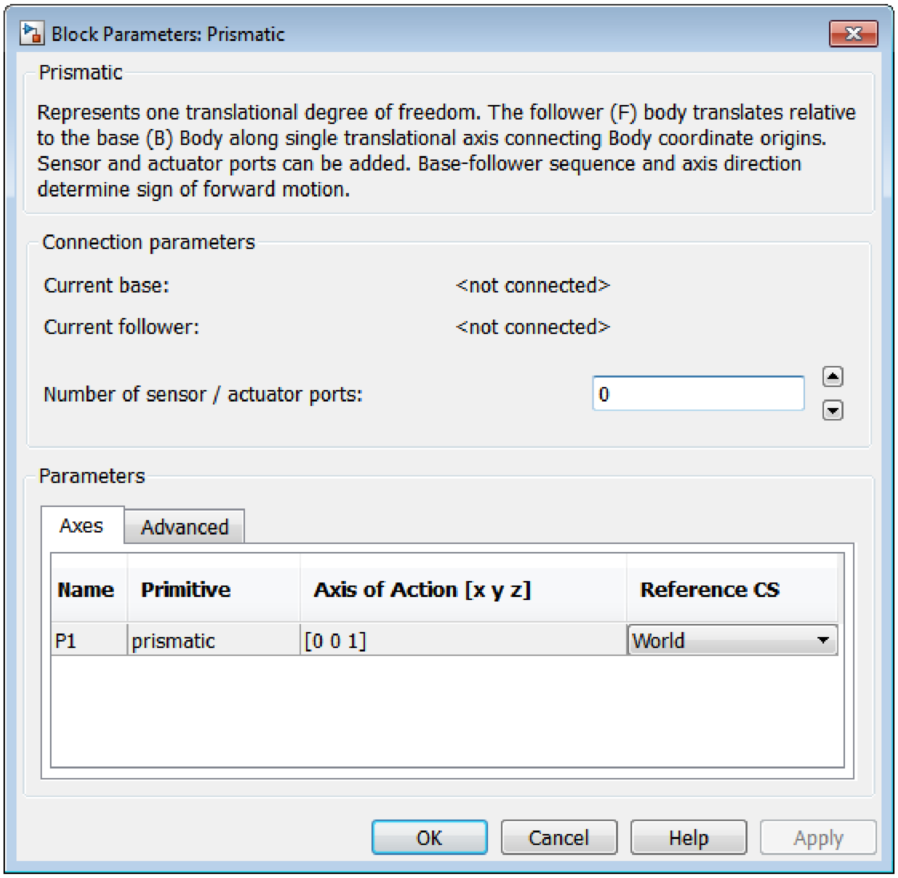
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space



Block Parameters: Prismatic

Prismatic

Represents one translational degree of freedom. The follower (F) body translates relative to the base (B) Body along single translational axis connecting Body coordinate origins. Sensor and actuator ports can be added. Base-follower sequence and axis direction determine sign of forward motion.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>
Number of sensor / actuator ports:	0

Parameters

Axes | Advanced

Name	Primitive	Axis of Action [x y z]	Reference CS
P1	prismatic	[0 0 1]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive translation is the follower moving in the direction of the translation axis.

Current base

When you connect the base (B) connector port on the Prismatic block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Prismatic Base and Follower Body Connector Ports on page 2-196.

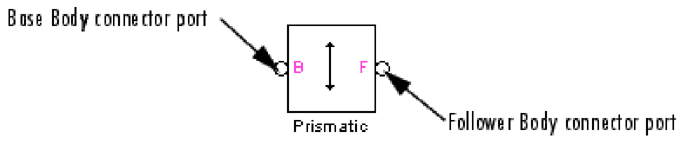
Current follower

When you connect the follower (F) connector port on the Prismatic block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Prismatic Base and Follower Body Connector Ports on page 2-196.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`.

The motion of a Prismatic is specified in linear units.



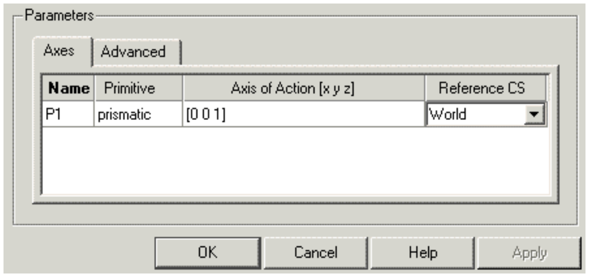
Prismatic Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. They specify the direction of the translational DoF that the Prismatic represents.



Name	Primitive	Axis of Action [x y z]	Reference CS
P1	prismatic	[0 0 1]	World

Name
: This column automatically displays the name of each primitive joint contained in the Joint block. For Prismatic, there is only one primitive joint, a prismatic, labeled `P1`.

Primitive
: This column automatically displays the type of each primitive joint contained in the Joint block. For Prismatic, there is only one primitive type, labeled `Prismatic`.

Axis of translation [x y z]
: Enter here as a three-component vector the directional axis along which this translational DoF can move. The default vector is `[0 0 1]`. The axis is a directed vector whose overall sign matters.

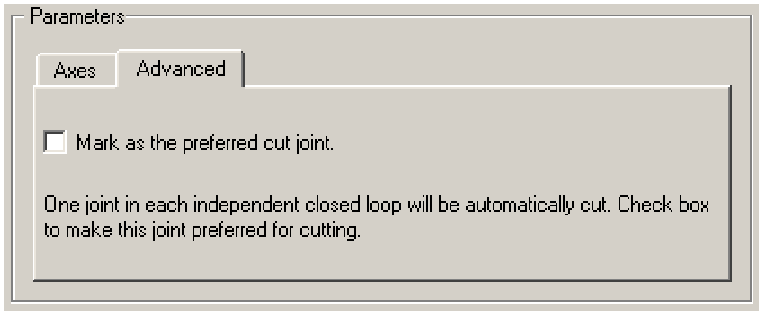
Reference CS
: Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of translation is oriented with respect to. This CS also determines the absolute meaning of force and motion along the joint axis. The default is `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Disassembled Prismatic, Joint Actuator, Joint Initial Condition Actuator, Joint Sensor, Joint Stiction Actuator, Revolute, Spherical

See "Modeling Degrees of Freedom" for more on representing DoFs with Joints.

See "Checking Model Topology" and "How SimMechanics Software Works" for more on closed loops and cutting.

Prismatic-Translational Interface

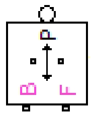
Purpose

Connection interface between prismatic primitive and Simscape mechanical translational elements

Library

Interface Elements

Description



This block connects a SimMechanics prismatic joint primitive with mechanical translational elements from the Simscape Foundation library. Dynamically, such a connection is equivalent to connecting the mechanical translational elements between the two bodies that are connected to the prismatic primitive. The validity of the connection is guaranteed only if the SimMechanics and Simscape mechanical modeling rules are both respected.

Simscape software supports a domain representing one-dimensional translational motion. In a Simscape mechanical translational circuit, force flows through a connection line, while velocity is defined across nodes connected by connection lines. The Prismatic-Translational Interface block rigidly couples a SimMechanics prismatic primitive to a Simscape mechanical translational degree of freedom. By itself, this interface adds or removes no degrees of freedom (DoFs) to or from the combined machine-circuit. Prismatic-Translational Interface preserves force through and motion across the block, conserving mechanical power.

- The mechanical connector port ○ acts, on the SimMechanics side, like a force-actuated Joint Actuator, feeding force from the circuit through the connected Joint to its connected Bodies, while maintaining the velocity across the block. The force exerted by the Simscape mechanical translational elements is applied between the two SimMechanics bodies along the Joint prismatic primitive axis.
- The physical conserving ports ◘ act, on the Simscape side, like a motion actuator, feeding a prescribed motion from the machine into the circuit, while preserving the flow of force through the block. The motion of the SimMechanics bodies along the Joint prismatic primitive axis is applied as relative motion between the Simscape mechanical translational ports of the block.

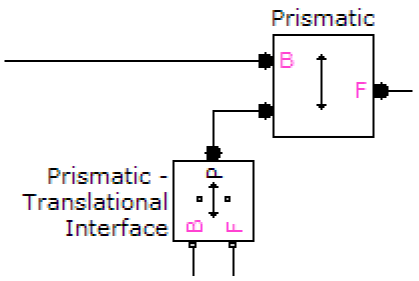
- The directionality of motion (base-to-follower, or B-to-F) is preserved across the interface, from the directionality of the connected Joint to the directionality of force and motion in the Simscape circuit.

You interface the machine and the mechanical circuit through the block ports.

- Connect the SimMechanics port P to a Sensor/Actuator port on a Joint. In the Interface block dialog, you then choose which prismatic primitive in that Joint should be connected to the Simscape circuit through this interface.
- Connect the Simscape mechanical translational ports (B and F) along the circuit connection line, which in turn connects to Simscape mechanical translational elements.

Joint Block Required for Complete Interface

The complete interface between a SimMechanics machine and a Simscape mechanical circuit includes both an Interface block and its connected Joint block containing the joint primitive chosen to couple to the one-dimensional circuit.



Avoiding Masses in the Interfaced Simscape Circuit

Warning

Simulating with mass elements in a Simscape circuit interfaced to a SimMechanics machine leads to unphysical results.

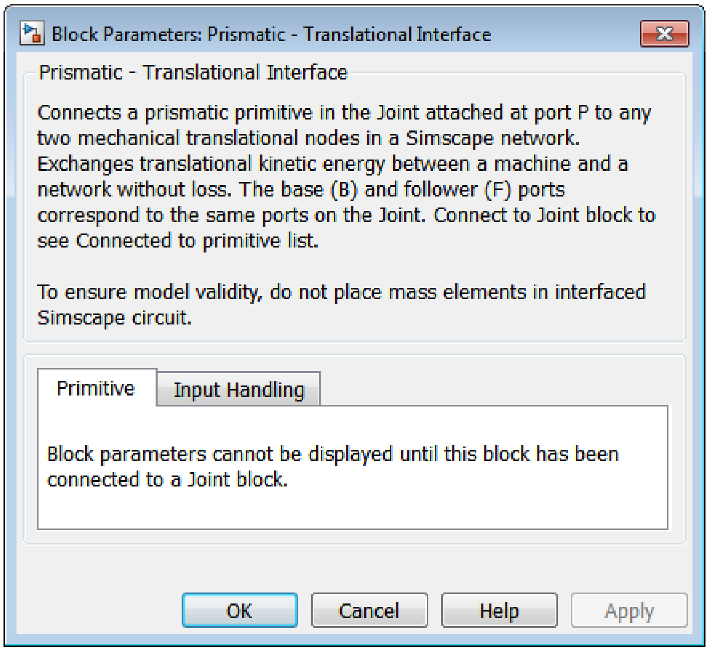
To avoid such unphysical situations, model all masses in your machine using SimMechanics Body blocks, and do not include mass elements in the Simscape circuit interfaced to a SimMechanics machine.

The dynamics of a Simscape mass element is simulated only:

- Along one dimension (the chosen prismatic primitive axis)
- With forces on the mass element explicitly modeled by blocks and connections in the circuit

If a Simscape mechanical circuit is interfaced to a SimMechanics machine and includes mass elements:

- The *acceleration* of the circuit's frame within the larger machine and thus noninertial *pseudoforces* are neglected in the dynamics of the circuit's mass elements.
- The interfaced Joint, from the point of view of the machine, effectively carries any masses modeled in the interfaced circuit. SimMechanics Joints are assumed massless.


Block Parameters: Prismatic - Translational Interface

Prismatic - Translational Interface

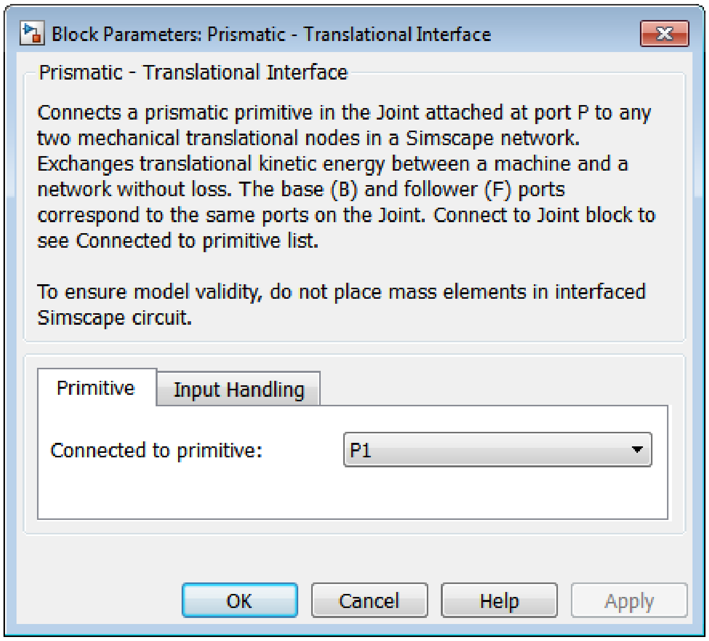
Connects a prismatic primitive in the Joint attached at port P to any two mechanical translational nodes in a Simscape network. Exchanges translational kinetic energy between a machine and a network without loss. The base (B) and follower (F) ports correspond to the same ports on the Joint. Connect to Joint block to see Connected to primitive list.

To ensure model validity, do not place mass elements in interfaced Simscape circuit.

Block parameters cannot be displayed until this block has been connected to a Joint block.

Dialog Box and Parameters

The dialog has one active area, **Primitive**. The block parameters are not displayed unless you connect it to a specific Joint block.


Block Parameters: Prismatic - Translational Interface

Prismatic - Translational Interface

Connects a prismatic primitive in the Joint attached at port P to any two mechanical translational nodes in a Simscape network. Exchanges translational kinetic energy between a machine and a network without loss. The base (B) and follower (F) ports correspond to the same ports on the Joint. Connect to Joint block to see Connected to primitive list.

To ensure model validity, do not place mass elements in interfaced Simscape circuit.

Connected to primitive: P1

Primitive

When connected to a Joint block, the dialog lists all the prismatic primitives present in the Joint.

Connected to primitive
From the pull-down menu, select the prismatic primitive in the Joint that you want to connect to the mechanical translational elements.

Examples

These demos illustrate how to interface between a three-dimensional machine and a one-dimensional mechanical translational circuit.

- mech_interface_crate_transfer
- mech_interface_dspring_damper
- mech_interface_hyd_cylinder
- mech_interface_trans_spr_damper

See Also

For more about using Interface Element blocks, see “Combining One- and Three-Dimensional Mechanical Elements”.

For more about dynamics, see “Mechanical Dynamics”.

Consult the Simscape documentation for more about one-dimensional domains.

Body, Joint Actuator, Prismatic, Revolute-Rotational Interface

Purpose

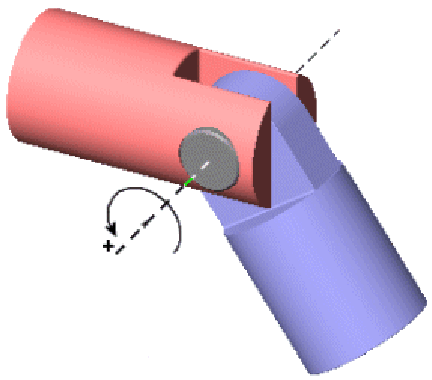
Primitive joint with one rotational degree of freedom

Library

Joints

Description

The Revolute block represents a single rotational degrees of freedom (DoF) about a specified axis between two bodies. The rotational sense is defined by the right-hand rule. A revolute joint is one of SimMechanics primitive joints, along with prismatic and spherical.



Revolute Motion of Follower (blue) Relative to Base (red)

Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

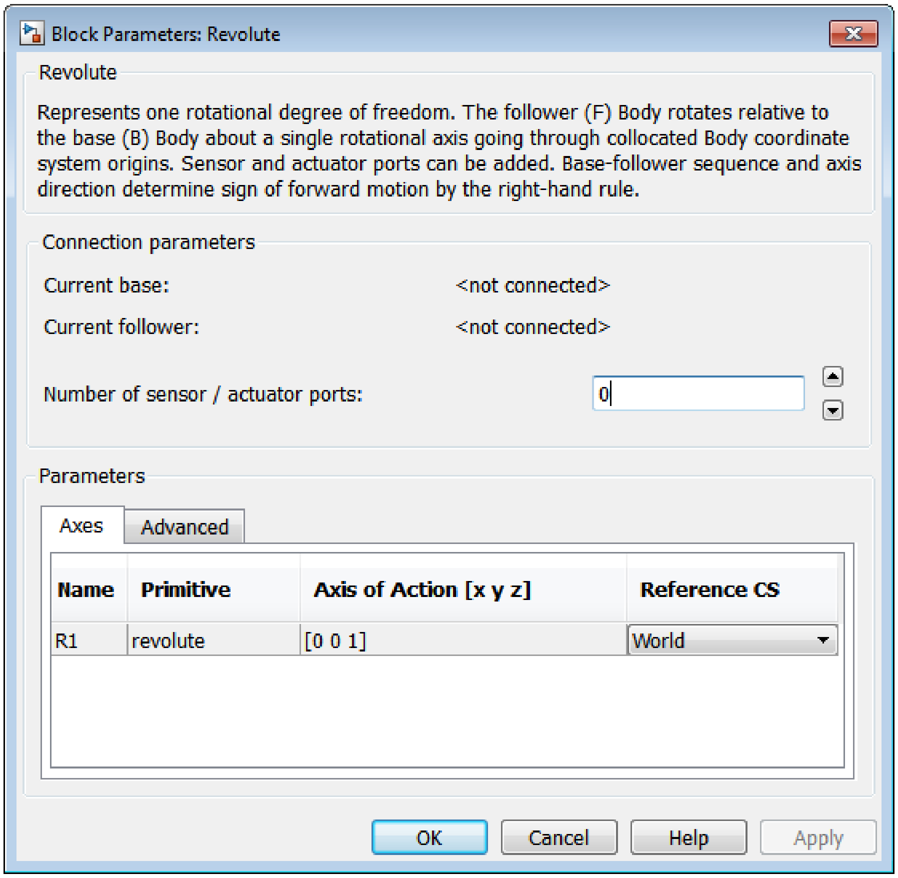
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space



Block Parameters: Revolute

Revolute

Represents one rotational degree of freedom. The follower (F) Body rotates relative to the base (B) Body about a single rotational axis going through collocated Body coordinate system origins. Sensor and actuator ports can be added. Base-follower sequence and axis direction determine sign of forward motion by the right-hand rule.

Connection parameters

Current base: <not connected>

Current follower: <not connected>

Number of sensor / actuator ports: 0

Parameters

Axes | Advanced

Name	Primitive	Axis of Action [x y z]	Reference CS
R1	revolute	[0 0 1]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive rotation is the follower rotating in the right-hand rule about the rotation axis.

Current base

When you connect the base (B) connector port on the Revolute block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Revolute Base and Follower Body Connector Ports on page 2-208.

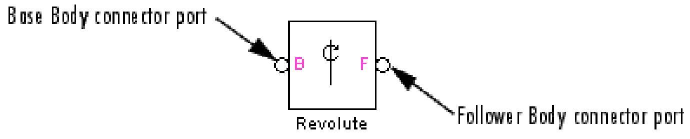
Current follower

When you connect the follower (F) connector port on the Revolute block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Revolute Base and Follower Body Connector Ports on page 2-208.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`.

The motion of a Revolute is specified in angular units.



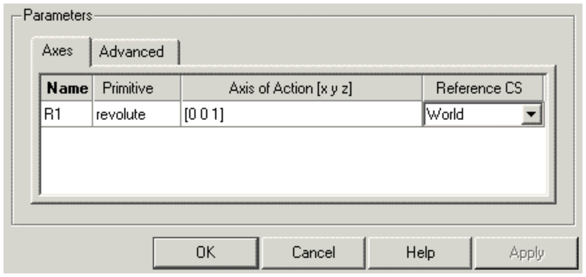
Revolute Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. They specify the direction of the rotation axis of the DoF that the Revolute represents.



Name	Primitive	Axis of Action [x y z]	Reference CS
R1	revolute	[0 0 1]	World

Name
: This column automatically displays the name of each primitive joint contained in the Joint block. For Revolute, there is only one primitive joint, a revolute, labeled `R1`.

Primitive
: This column automatically displays the type of each primitive joint contained in the Joint block. For Revolute, there is only one primitive type, labeled `Revolute`.

Axis of rotation [x y z]
: Enter here as a three-component vector the directional axis about which this rotational DoF can move. The default vector is `[0 0 1]`. The axis is a directed vector whose overall sign matters.

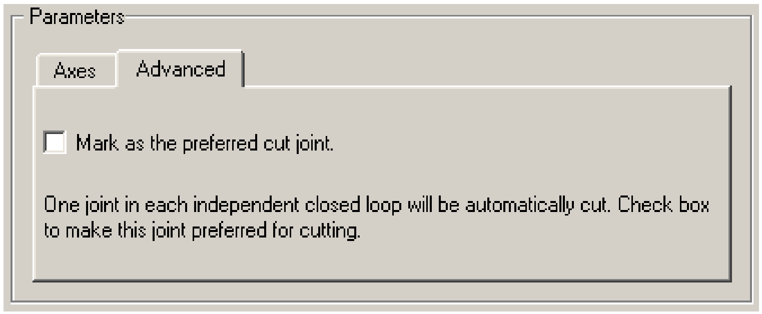
Reference CS
: Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of rotation is oriented with respect to. This CS also determines the absolute meaning of torque and motion about the joint axis. The default is `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Disassembled Revolute, Joint Actuator, Joint Initial Condition Actuator, Joint Sensor, Joint Stiction Actuator, Prismatic, Spherical

See "Modeling Degrees of Freedom" for more on representing DoFs with Joints.

See "Checking Model Topology" and "How SimMechanics Software Works" for more on closed loops and cutting.

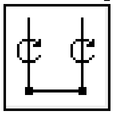
Purpose

Constant-length joint connector with two spatially separated revolute axes

Library

Joints/Massless Connectors

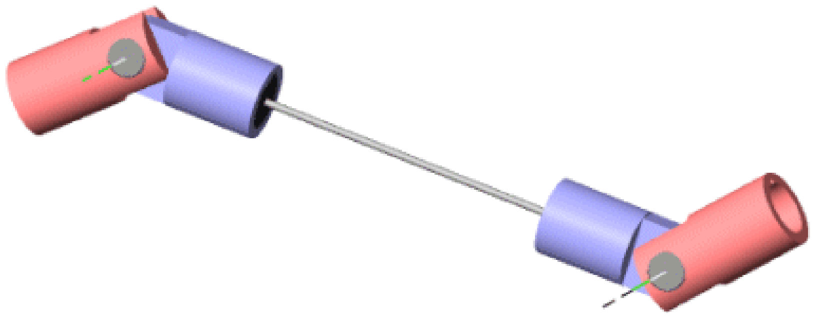
Description



The Revolute-Revolute block represents a composite joint composed of two revolute joint primitives. The Body coordinate systems (CSs) on either side of the Joint are each connected to a revolute primitive. The primitives are separated spatially by a vector of constant length but variable direction connecting the two Body CS origins.

Warning

This joint becomes singular if the two revolute primitive axes align with the vector separating the primitives. The simulation stops with an error in this case.



Massless Connector Between Revolute and Revolute Joints

Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have a default of two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) point.

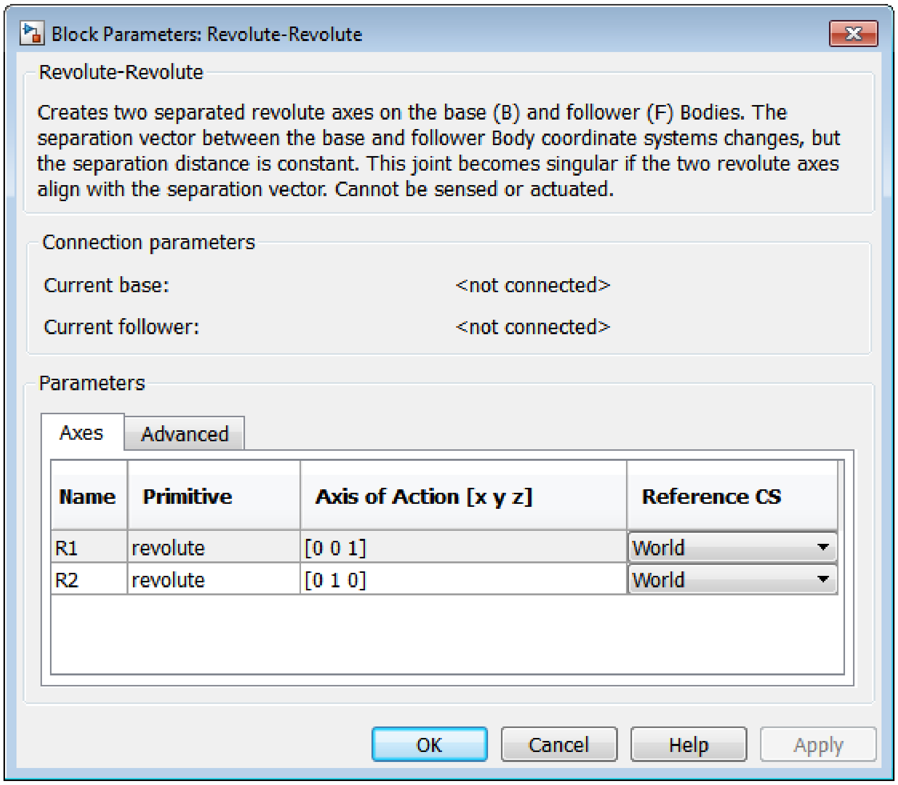
You specify the joint primitive axes, if any, in the Joint dialog.

You cannot connect an Actuator or Sensor to a Massless Connector.

Assembly Restrictions on Massless Connectors

Both joint primitives are assembled but spatially separated. That is, the connected Body CS origins must not be spatially collocated points.

The distance separating the two ends of the connector is computed automatically from the Body CS origins to which the Joint is connected. This distance (the magnitude of the vector between the Body CS origins) remains fixed at its initial value during the simulation. This initial value must be nonzero.



Revolute-Revolute

Creates two separated revolute axes on the base (B) and follower (F) Bodies. The separation vector between the base and follower Body coordinate systems changes, but the separation distance is constant. This joint becomes singular if the two revolute axes align with the separation vector. Cannot be sensed or actuated.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>

Parameters

Name	Primitive	Axis of Action [x y z]	Reference CS
R1	revolute	[0 0 1]	World
R2	revolute	[0 1 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

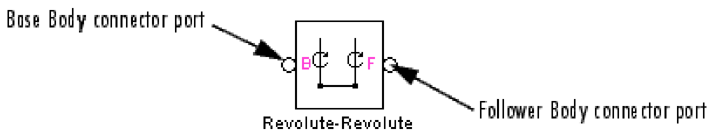
The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive rotation is the base or follower rotating in the right-handed sense about its respective rotation axis.

Current base
When you connect the base (B) connector port on the Revolute-Revolute block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See

the following figure, Revolute-Revolute Base and Follower Body Connector Ports on page 2-214.

Current follower

When you connect the follower (F) connector port on the Revolute-Revolute block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Revolute-Revolute Base and Follower Body Connector Ports on page 2-214.



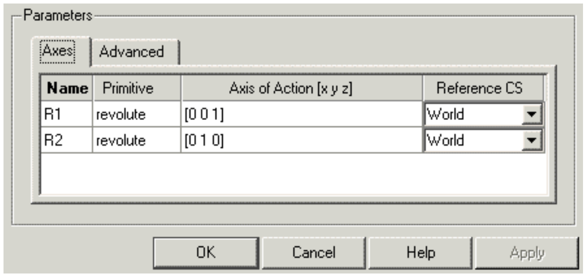
Revolute-Revolute Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. They specify the direction of the rotation axes of these DoFs that the Revolute-Revolute represents.



Name	Primitive	Axis of Action [x y z]	Reference CS
R1	revolute	[0 0 1]	World
R2	revolute	[0 1 0]	World

Name

This column automatically displays the name of each primitive joint contained in the Joint block. For Revolute-Revolute, there are two revolute primitives, labeled `R1` and `R2`, connecting to base and follower, respectively.

Primitive
This column automatically displays the type of each primitive joint contained in the Joint block. For Revolute-Revolute, there is only one primitive type, labeled `Revolute`.

Axis of Action [x y z]
Enter here as a three-component vector the directional axis about which these rotational DoFs can move. The default vectors are `[0 0 1]` and `[0 1 0]`. The axes are directed vectors whose overall signs matter.

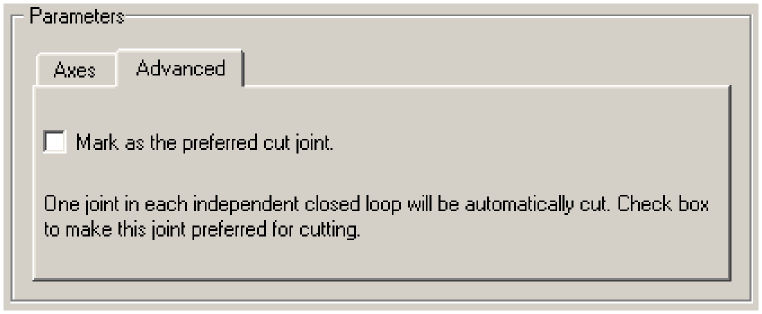
Reference CS
Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axes of rotation are oriented with respect to. These CSs also determine the absolute meaning of torque and motion about the primitive axes. The defaults are `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Revolute

See "Modeling Degrees of Freedom" for more on representing DoFs with Massless Connectors.

See "Checking Model Topology" and "How SimMechanics Software Works" for more on closed loops and cutting.

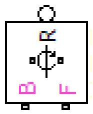
Purpose

Connection interface between revolute primitive and Simscape mechanical rotational elements

Library

Interface Elements

Description



This block connects a SimMechanics revolute joint primitive with mechanical rotational elements from the Simscape Foundation library. Dynamically, such a connection is equivalent to connecting the mechanical rotational elements between the two bodies that are connected to the revolute primitive. The validity of the connection is guaranteed only if the SimMechanics and Simscape mechanical modeling rules are both respected.

Simscape software supports a domain representing one-dimensional rotational motion. In a Simscape mechanical rotational circuit, torque flows through a connection line, while angular velocity is defined across nodes connected by connection lines. The Revolute-Rotational Interface block rigidly couples a SimMechanics revolute primitive to a Simscape mechanical rotational degree of freedom. By itself, this interface adds or removes no degrees of freedom (DoFs) to or from the combined machine-circuit. Revolute-Rotational Interface preserves torque through and angular motion across the block, conserving mechanical power.

- The mechanical connector port ○ acts, on the SimMechanics side, like a torque-actuated Joint Actuator, feeding torque from the circuit through the connected Joint to its connected Bodies, while maintaining the angular velocity across the block. The torque exerted by the Simscape mechanical rotational elements is applied between the two SimMechanics bodies about the Joint revolute primitive axis.
- The physical conserving ports ▣ act, on the Simscape side, like a motion actuator, feeding a prescribed motion from the machine into the circuit, while preserving the flow of torque through the block. The motion of the SimMechanics bodies about the Joint revolute primitive axis is applied as relative motion between the Simscape mechanical rotational ports of the block.

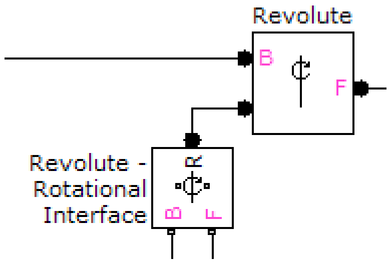
- The directionality of motion (base-to-follower, or B-to-F) is preserved across the interface, from the directionality of the connected Joint to the directionality of torque and angular motion in the Simscape circuit.

You interface the machine and the mechanical circuit through the block ports.

- Connect the SimMechanics port P to a Sensor/Actuator port on a Joint. In the Interface block dialog, you then choose which revolute primitive in that Joint should be connected to the Simscape circuit through this interface.
- Connect the Simscape mechanical rotational ports (B and F) along the circuit connection line, which in turn connects to Simscape mechanical rotational elements.

Joint Block Required for Complete Interface

The complete interface between a SimMechanics machine and a Simscape mechanical circuit includes both an Interface block and its connected Joint block containing the joint primitive chosen to couple to the one-dimensional circuit.



Avoiding Inertias in the Interfaced Simscape Circuit

Warning

Simulating with inertia elements in a Simscape circuit interfaced to a SimMechanics machine leads to unphysical results.

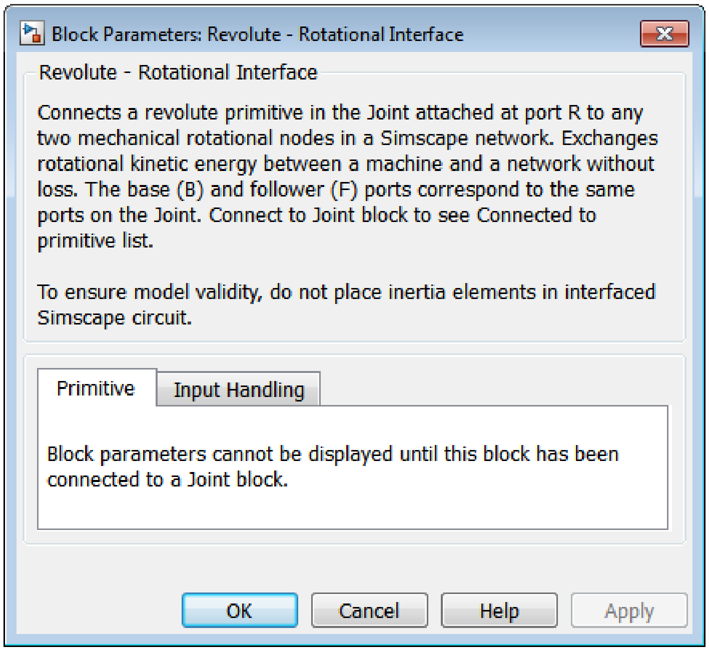
To avoid such unphysical situations, model all inertias in your machine using SimMechanics Body blocks, and do not include inertia elements in the Simscape circuit interfaced to a SimMechanics machine.

The dynamics of a Simscape inertia element is simulated only:

- About one dimension (the chosen revolute primitive axis)
- With torques on the inertia element explicitly modeled by blocks and connections in the circuit

If a Simscape mechanical circuit is interfaced to a SimMechanics machine and includes inertia elements:

- The *acceleration* of the circuit's frame within the larger machine and thus noninertial *pseudoforces* are neglected in the dynamics of the circuit's inertia elements.
- The interfaced Joint, from the point of view of the machine, effectively carries any inertias modeled in the interfaced circuit. SimMechanics Joints are assumed massless.



Block Parameters: Revolute - Rotational Interface

Revolute - Rotational Interface

Connects a revolute primitive in the Joint attached at port R to any two mechanical rotational nodes in a Simscape network. Exchanges rotational kinetic energy between a machine and a network without loss. The base (B) and follower (F) ports correspond to the same ports on the Joint. Connect to Joint block to see Connected to primitive list.

To ensure model validity, do not place inertia elements in interfaced Simscape circuit.

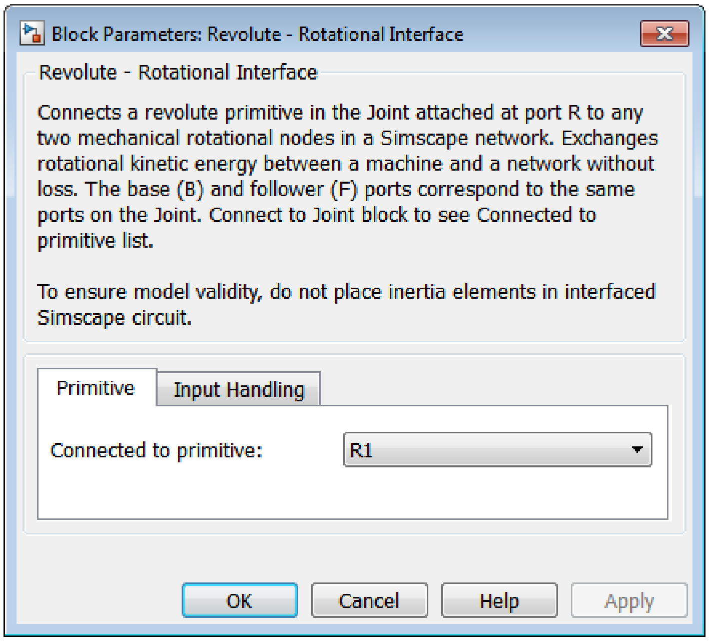
Primitive | Input Handling

Block parameters cannot be displayed until this block has been connected to a Joint block.

Dialog Box and Parameters

The dialog has one active area, **Primitive**. The block parameters are not displayed unless you connect it to a specific Joint block.

Primitive


Block Parameters: Revolute - Rotational Interface

Revolute - Rotational Interface

Connects a revolute primitive in the Joint attached at port R to any two mechanical rotational nodes in a Simscape network. Exchanges rotational kinetic energy between a machine and a network without loss. The base (B) and follower (F) ports correspond to the same ports on the Joint. Connect to Joint block to see Connected to primitive list.

To ensure model validity, do not place inertia elements in interfaced Simscape circuit.

Connected to primitive: R1

When connected to a Joint block, the dialog lists all the revolute primitives present in the Joint.

Connected to primitive
From the pull-down menu, select the revolute primitive in the Joint that you want to connect to the mechanical rotational elements.

Examples

These demos illustrate how to interface between a three-dimensional machine and a one-dimensional mechanical rotational circuit.

- mech_interface_hyd_slidercrank
- mech_interface_rot_spr_damper

See Also

For more about using Interface Element blocks, see "Combining One- and Three-Dimensional Mechanical Elements".

For more about dynamics, see “Mechanical Dynamics”.

Consult the Simscape documentation for more about one-dimensional domains.

Body, Joint Actuator, Prismatic-Translational Interface, Revolute

Purpose

Constant-length joint connector with spatially separated revolute axis and spherical pivot point

Library

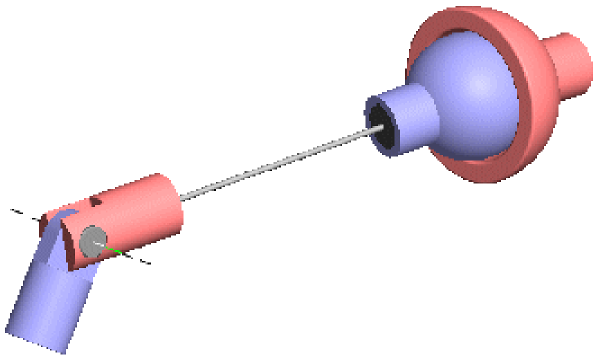
Joints/Massless Connectors

Description

The Revolute-Spherical block represents a composite joint composed of a revolute and a spherical joint primitive. The base Body coordinate system (CS) on one side of the Joint is connected to the revolute primitive, and the follower Body CS is connected to the spherical primitive. The primitives are separated spatially by a vector of constant length but variable direction connecting the two Body CS origins.

Warning

This joint becomes singular if the revolute primitive axis aligns with the vector separating the primitives. The simulation stops with an error in this case.



Massless Connector Between Revolute and Spherical Joints

Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have a default of two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) point.

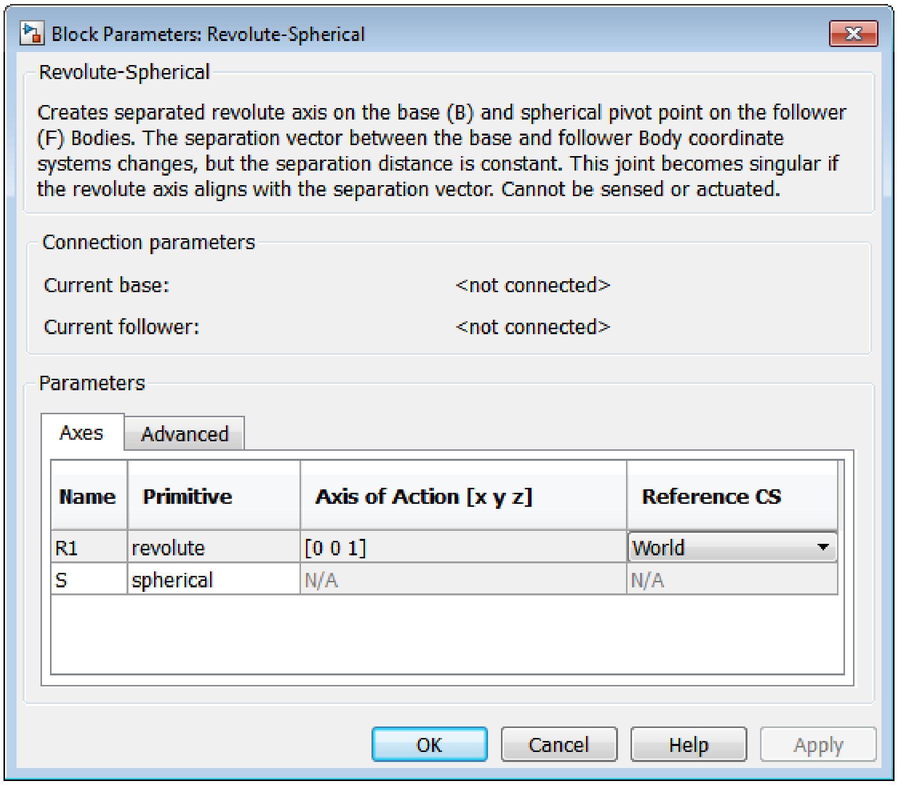
You specify the joint primitive axes, if any, in the Joint dialog.

You cannot connect an Actuator or Sensor to a Massless Connector.

Assembly Restrictions on Massless Connectors

Both joint primitives are assembled but spatially separated. That is, the connected Body CS origins must not be spatially collocated points.

The distance separating the two ends of the connector is computed automatically from the Body CS origins to which the Joint is connected. This distance (the magnitude of the vector between the Body CS origins) remains fixed at its initial value during the simulation. This initial value must be nonzero.



Revolute-Spherical

Creates separated revolute axis on the base (B) and spherical pivot point on the follower (F) Bodies. The separation vector between the base and follower Body coordinate systems changes, but the separation distance is constant. This joint becomes singular if the revolute axis aligns with the separation vector. Cannot be sensed or actuated.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>

Parameters

Name	Primitive	Axis of Action [x y z]	Reference CS
R1	revolute	[0 0 1]	World
S	spherical	N/A	N/A

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

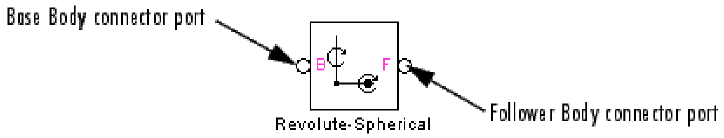
The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive rotation is the base rotating in the right-handed sense about its rotation axis or the follower pivoting as shown for the Spherical Joint.

Current base

When you connect the base (B) connector port on the Revolute-Spherical block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Revolute-Spherical Base and Follower Body Connector Ports on page 2-226.

Current follower

When you connect the follower (F) connector port on the Revolute-Spherical block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Revolute-Spherical Base and Follower Body Connector Ports on page 2-226.



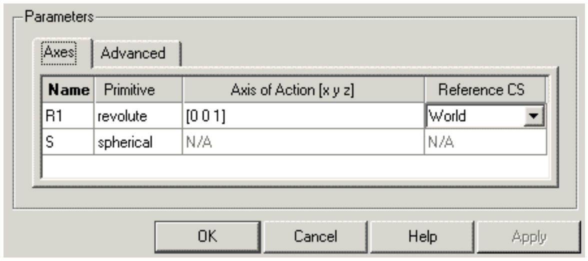
Revolute-Spherical Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. They specify the direction of the rotation axis of one of the DoFs that Revolute-Spherical represents.



Name	Primitive	Axis of Action [x y z]	Reference CS
R1	revolute	[0 0 1]	World
S	spherical	N/A	N/A

Name

This column automatically displays the name of each primitive joint contained in the Joint block. For Revolute-Spherical, there are one revolute and one spherical primitive, labeled `R1` and `S`, connecting to base and follower, respectively.

Primitive

This column automatically displays the type of each primitive joint contained in the Joint block. For Revolute-Spherical, there are two primitive types, labeled `Revolute` and `Spherical`.

Axis of Action [x y z]

Enter here as a three-component vector the directional axis about which the rotational DoF can move. The default vector is `[0 0 1]`. The axis is a directed vector whose overall sign matters.

This field is not active for the Spherical primitive.

Reference CS

Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of rotation is oriented with respect to. This CS also determines the absolute meaning of torque and motion about the primitive axis. The default is `World`.

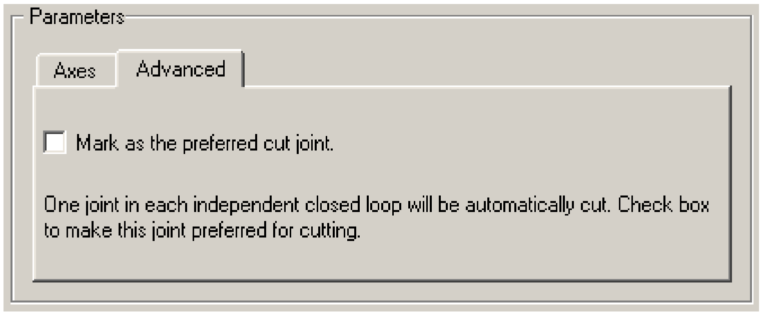
This field is not active for the Spherical primitive.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Revolute, Spherical

See “Modeling Degrees of Freedom” for more on representing DoFs with Massless Connectors.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on closed loops and cutting.

Purpose

Utility that transforms 3x3 rotation matrix into rotation axis-angle 4-vector

Library

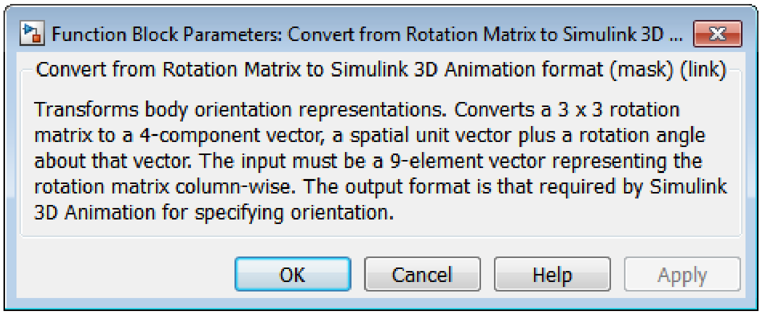
Utilities

Description

A rotation with respect to an initial orientation has many equivalent representations. A common and important one is the 3-by-3 orthogonal rotation matrix R, where $R^{-1} = R^{T}$ and $R^{T}R = RR^{T} = I$, the 3-by-3 identity matrix. Another important representation is the combination of rotation axis (a unit vector **n**) and angle of rotation θ about that axis. The sign of rotation follows the right-hand-rule.

The RotationMatrix2VR block converts the 3-by-3 rotation matrix representation of orientation to its equivalent representation as a rotation axis and angle about that axis, the form used in Virtual Reality Modeling Language (VRML) for orienting bodies. The input and output signals are bundled Simulink signals.

The most common use of rotations is to represent the orientation of a body with respect to some coordinate system (CS) axes.


Function Block Parameters: Convert from Rotation Matrix to Simulink 3D ...

Convert from Rotation Matrix to Simulink 3D Animation format (mask) (link)

Transforms body orientation representations. Converts a 3 x 3 rotation matrix to a 4-component vector, a spatial unit vector plus a rotation angle about that vector. The input must be a 9-element vector representing the rotation matrix column-wise. The output format is that required by Simulink 3D Animation for specifying orientation.

Dialog Box and Parameters

The dialog has no active areas.

Representations of Rotation Signals

The rotation matrix R has the form:

$$\begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}$$

The input signal to the RotationMatrix2VR block is the R matrix components passed column-wise and bundled into a single 9-component Simulink signal: `[R11 R21 R31 R12 ...]`.

The output signal is the equivalent rotation represented as the axis of rotation, a unit vector $\boldsymbol{n} = (n_x,n_y,n_z)$, with

$$\boldsymbol{n}\cdot\boldsymbol{n} = n_x^2 + n_y^2 + n_z^2 = 1,$$

and the angle of rotation θ about that axis. The sign of the rotation follows the right-hand rule. The output signal is bundled into a single 4-component Simulink signal:

$[n_x \; n_y \; n_z \; \theta]$.

See Also

Body

See “Representations of Body Motion” and “Representations of Body Orientation” for more details on representing body rotations.

See entries on axis-angle rotation, Euler angles, quaternion, and rotation matrix in the Glossary for summaries of body orientation representations.

For more on virtual reality and VRML, see the *Simulink 3D Animation™ User’s Guide*.

Screw

Purpose

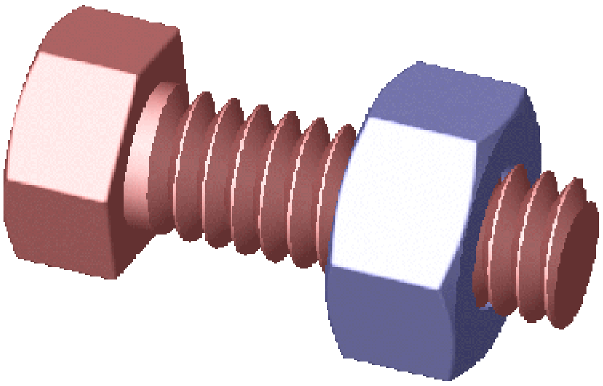
Joint with coupled rotational and translational degrees of freedom

Library

Joints

Description

The Screw block represents a composite joint with one translational degrees of freedom (DoF) as one prismatic primitive and one rotational DoF as one revolute primitive. The translation and rotation axes are parallel. The translational and rotational DoFs are constrained by a pitch constraint to have proportional motion.



Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

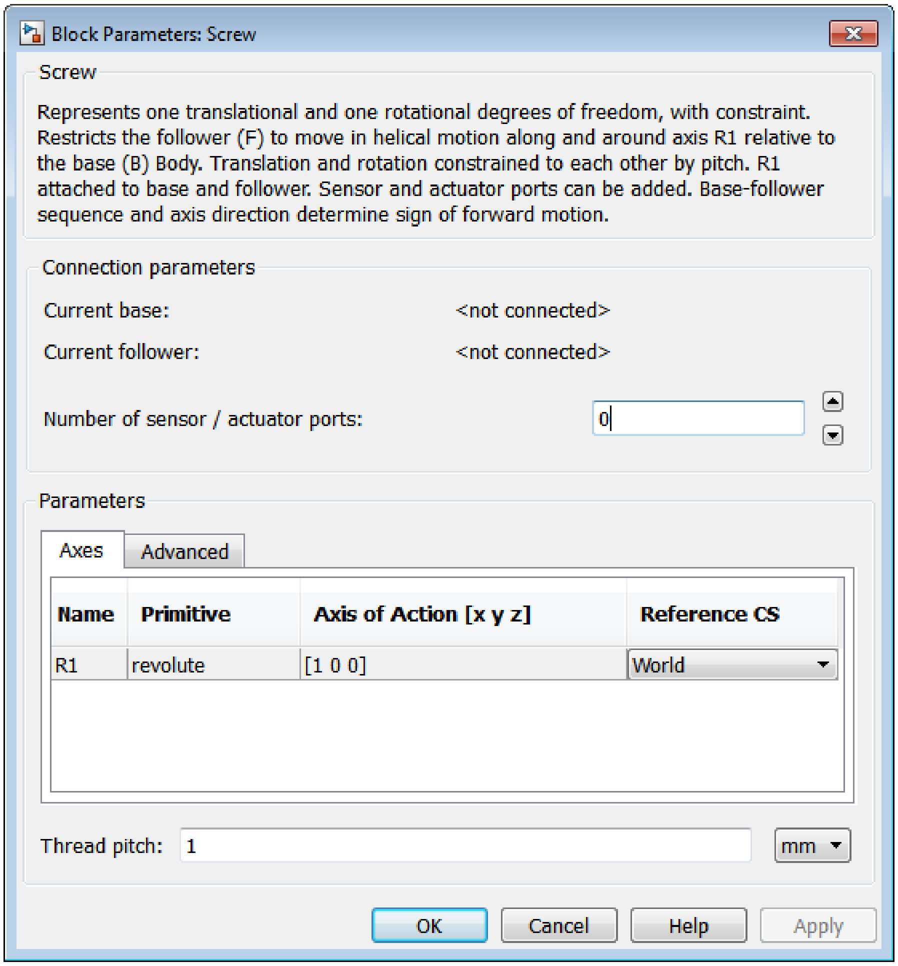
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space


Block Parameters: Screw

Screw

Represents one translational and one rotational degrees of freedom, with constraint. Restricts the follower (F) to move in helical motion along and around axis R1 relative to the base (B) Body. Translation and rotation constrained to each other by pitch. R1 attached to base and follower. Sensor and actuator ports can be added. Base-follower sequence and axis direction determine sign of forward motion.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>
Number of sensor / actuator ports:	0

Parameters

Axes | Advanced

Name	Primitive	Axis of Action [x y z]	Reference CS
R1	revolute	[1 0 0]	World

Thread pitch: 1 mm

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive rotation is the follower moving around the rotational axis following the right-hand rule.

Current base

When you connect the base (B) connector port on the Screw block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Screw Base and Follower Body Connector Ports on page 2-235.

The base Body is automatically connected to the joint primitive `R1` in the primitive list in **Parameters**.

Current follower

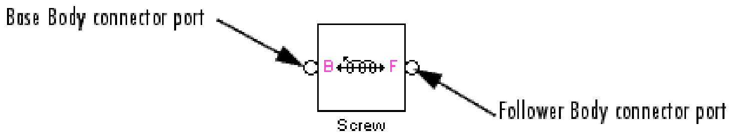
When you connect the follower (F) connector port on the Screw block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Screw Base and Follower Body Connector Ports on page 2-235.

The follower Body is automatically connected to the joint primitive `R1` in the primitive list in **Parameters**.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`.

The motion of revolute primitives is specified in angular units.



Screw Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. Each DoF primitive in Screw has an entry line. These lines specify the direction of the axes of action of the DoFs that the Screw represents.

Name - Primitive

The primitive list states the name and type of the joint primitive that makes up the Screw block: revolute primitive `R1`.

Axis of Action [x y z]

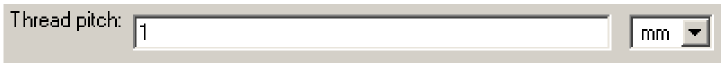
Enter here as a three-component vector the directional axes defining the allowed motions of this primitive and its corresponding DoF:

- Revolute: axis of rotation

The default vectors are shown in the dialog above. The axis is a directed vector whose overall sign matters.

Reference CS

Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of action is oriented with respect to. This CS also determines the absolute meaning of forces/torques and motion along/about the joint axis. The default is `World`.


Thread pitch: 1 mm

The thread pitch controls the amount of translation for each turn of the screw.

Thread pitch

Linear distance the screw travels along the screw axis for each complete revolution of 2π radians (360^o). The default is `1`.

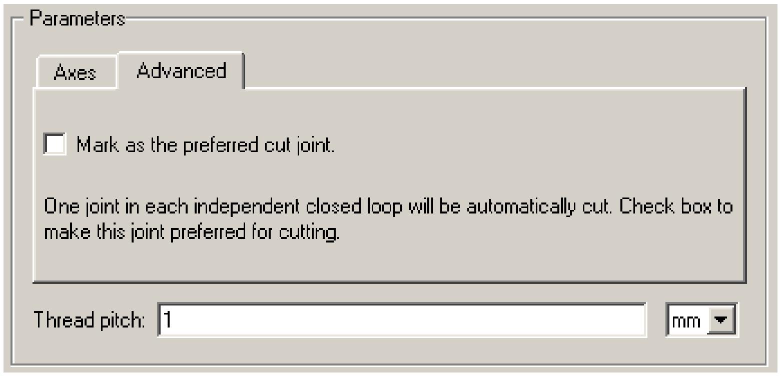
In pull-down menu, select units. The default is `mm` (millimeters).

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab


Parameters

Axes | Advanced

☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

Thread pitch: 1 mm

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Cylindrical, Prismatic, Revolute

See "Modeling Degrees of Freedom" for more on representing DoFs with Joints.

See "Checking Model Topology" and "How SimMechanics Software Works" for more on closed loops and cutting.

Shared Environment

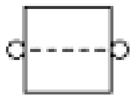
Purpose

Utility that connects two independent machines in a single mechanical environment

Library

Bodies

Description

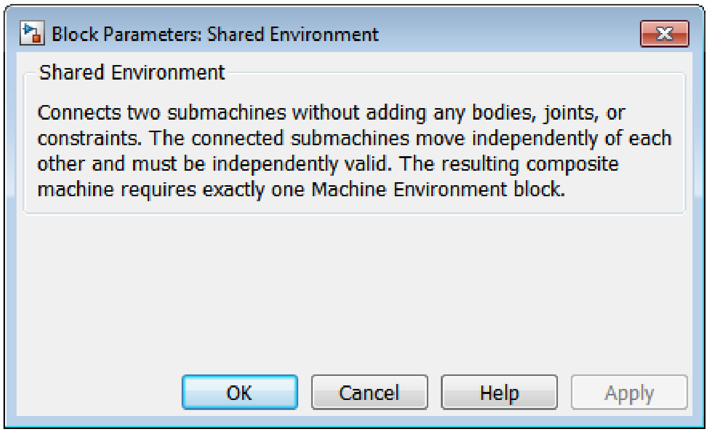


The Shared Environment block provides a nonphysical connection between two independent mechanical block diagrams, or submachines. The block carries no inertia, adds no joints, imposes no constraints, and transfers no motion, force, or torque between the SimMechanics blocks to which it is connected.

You can use this block to connect two independent machines into one machine, so that the two submachines then share the same machine environment. Making this connection does not change the structure or dynamics of either submachine.

Caution The two connected submachines have to be independently valid, and *each* submachine requires at least one Ground block.

The resulting composite machine needs exactly one Machine Environment block, not two.



Connects two submachines without adding any bodies, joints, or constraints. The connected submachines move independently of each other and must be independently valid. The resulting composite machine requires exactly one Machine Environment block.

Dialog Box and Parameters

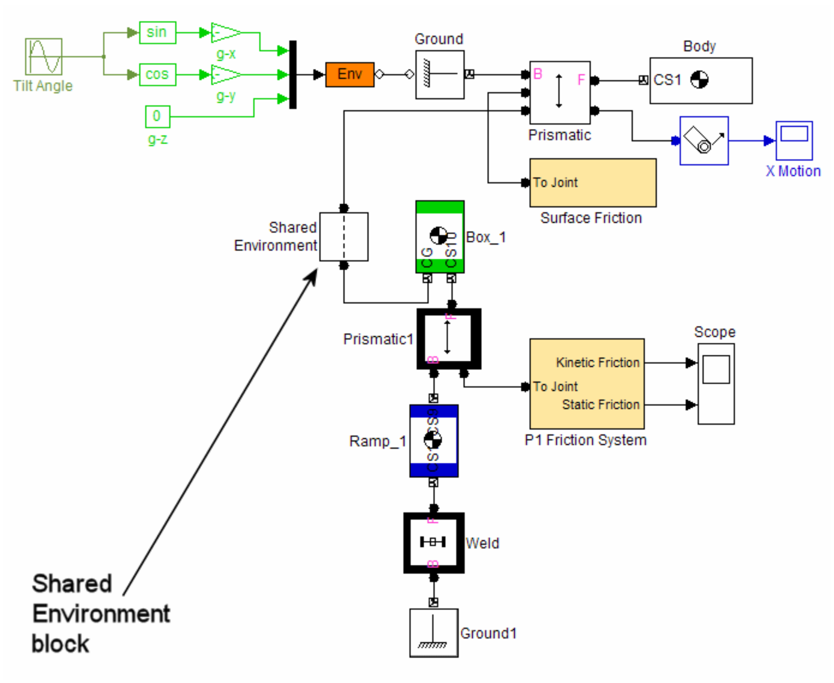
This block has no parameters.

Using the Shared Environment Block

The Shared Environment block features two general-purpose SimMechanics connector ports ○. To connect two machines to each other so that they share the same machine environment, connect each of these ports to either:

- Another general-purpose connector port ○
- A Body coordinate system (CS) port ⊞

This block diagram illustrates a valid composite machine formed by connecting two separate machines with a Shared Environment block. The composite machine requires and contains one and only one Machine Environment block.



See Also

See "Connecting SimMechanics Blocks".

Ground, Machine Environment

Six-DoF

Purpose

Joint with three revolute and three prismatic joint primitives

Library

Joints

Description

The Six-DoF block represents a composite joint with three translational degrees of freedom (DoFs) as three prismatic primitives and three rotational DoFs as one spherical primitives. There are no constraints among the primitives. Unlike Bushing, Six-DoF represents the rotational DoFs as one spherical, rather than as three revolutes.

Warning

A joint with three prismatic primitives becomes singular if two or three of the translation axes become parallel. The simulation stops with an error in this case.

Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

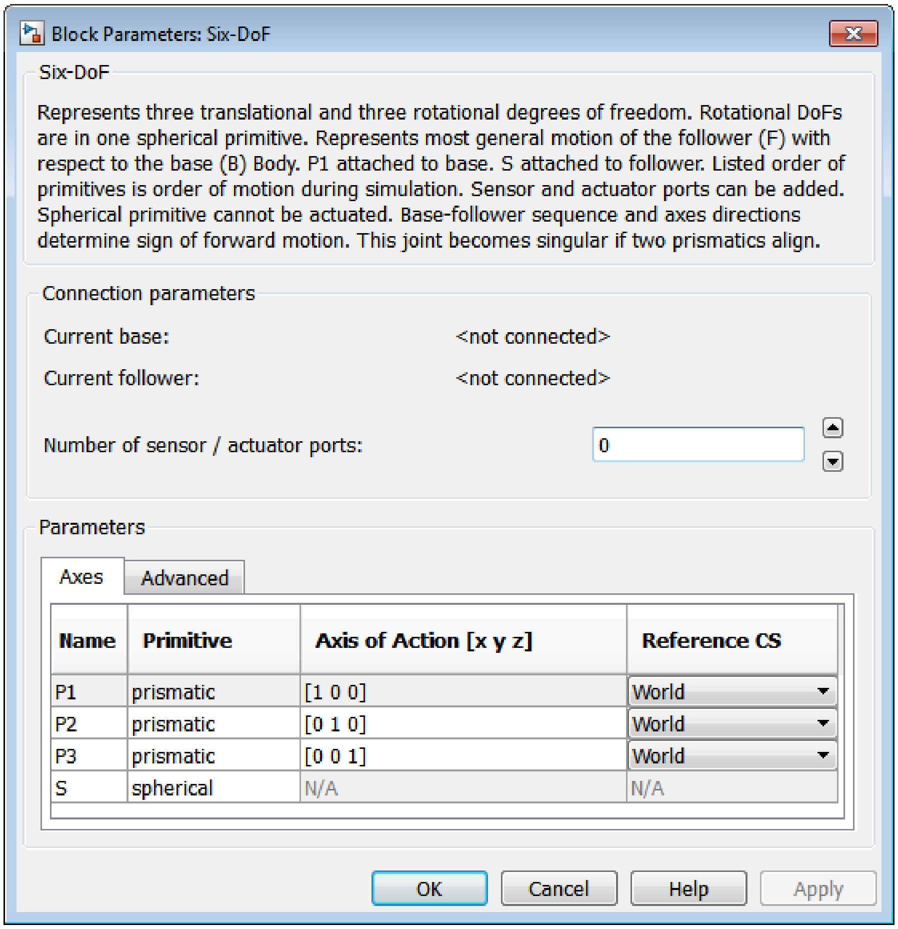
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space



Represents three translational and three rotational degrees of freedom. Rotational DoFs are in one spherical primitive. Represents most general motion of the follower (F) with respect to the base (B) Body. P1 attached to base. S attached to follower. Listed order of primitives is order of motion during simulation. Sensor and actuator ports can be added. Spherical primitive cannot be actuated. Base-follower sequence and axes directions determine sign of forward motion. This joint becomes singular if two prismatics align.

Connection parameters

Current base: <not connected>

Current follower: <not connected>

Number of sensor / actuator ports: 0

Parameters

Name	Primitive	Axis of Action [x y z]	Reference CS
P1	prismatic	[1 0 0]	World
P2	prismatic	[0 1 0]	World
P3	prismatic	[0 0 1]	World
S	spherical	N/A	N/A

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive translation is the follower moving in the direction of the translation axis. Positive spherical motion is the follower rotating in the right-handed sense as shown in the Spherical block figure.

Current base

When you connect the base (B) connector port on the Six-DoF block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Six-DoF Base and Follower Body Connector Ports on page 2-244.

The base Body is automatically connected to the first joint primitive `P1` in the primitive list in **Parameters**.

Current follower

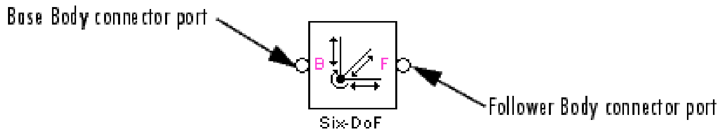
When you connect the follower (F) connector port on the Six-DoF block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Six-DoF Base and Follower Body Connector Ports on page 2-244.

The follower Body is automatically connected to the last joint primitive `S` in the primitive list in **Parameters**.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`.

The motion of prismatic primitives is specified in linear units. The motion of spherical primitives is specified by a dimensionless quaternion.



Six-DoF Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. Each DoF primitive in Six-DoF has an entry line. These lines specify the direction of the axes of action of the DoFs that the Six-DoF represents.

Name - Primitive
The primitive list states the names and types of joint primitives that make up the Six-DoF block: prismatic primitives `P1`, `P2`, `P3`, and spherical primitive `S`.

Axis of Action [x y z]
Enter here as a three-component vector the directional axes defining the allowed motions of these primitives and their corresponding DoFs:

- Prismatic: axis of translation
- Spherical: field is not active

The default vectors are shown in the dialog above. The axis is a directed vector whose overall sign matters.

To prevent singularities and simulation errors, no two of the prismatic axes can be parallel.

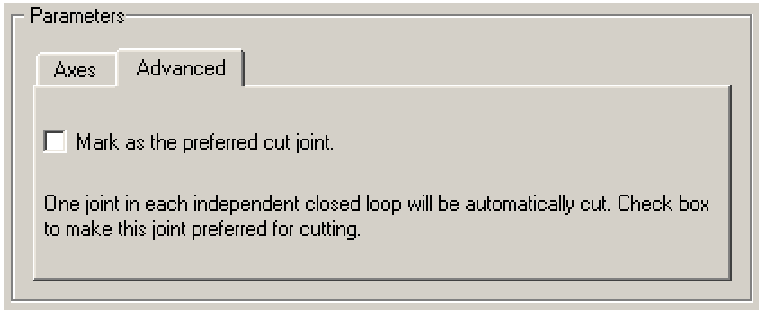
Reference CS
Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of action is oriented with respect to. This CS also determines the absolute meaning of forces/torques and motion along/about the joint axis. The default is `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Bushing, Gimbal, Prismatic, Spherical

See “Modeling Degrees of Freedom” for more on representing DoFs with Joints.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on closed loops and cutting.

Purpose

Primitive joint with three rotational degrees of freedom

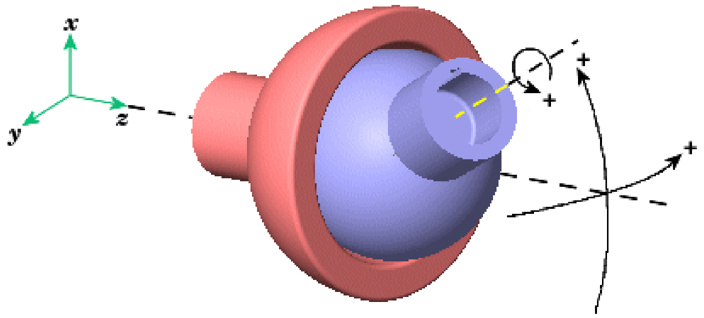
Library

Joints

Description

The Spherical block represents three rotational degrees of freedom (DoFs) at a single pivot point, a "ball-in-socket" joint. Two rotational DoFs specify a directional axis, and a third rotational DoF specifies rotation about that directional axis. The sense of each rotational DoF is defined by the right-hand rule, and the three rotations together form a right-handed system. A spherical joint is one of the SimMechanics primitive joints, along with prismatic and revolute.

You cannot connect an Actuator to a Spherical. Unlike the Gimbal block, the Spherical block cannot become singular.



Spherical Motion of Follower (blue) Relative to Base (red)

Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower).

You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

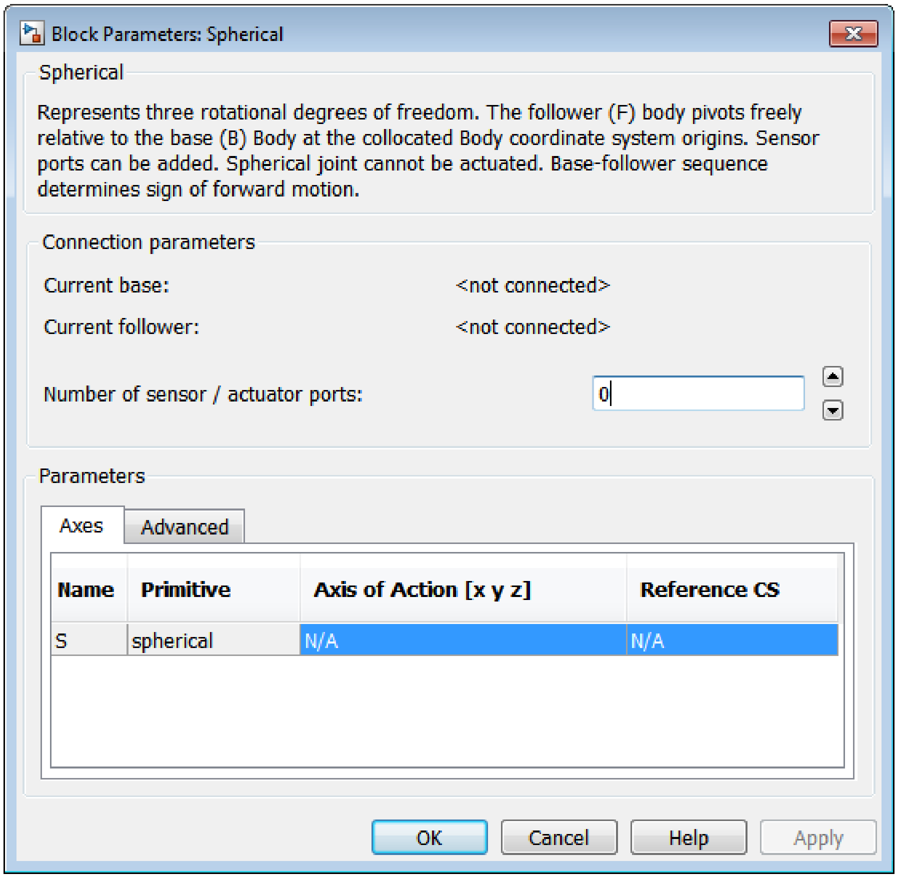
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space



Block Parameters: Spherical

Spherical

Represents three rotational degrees of freedom. The follower (F) body pivots freely relative to the base (B) Body at the collocated Body coordinate system origins. Sensor ports can be added. Spherical joint cannot be actuated. Base-follower sequence determines sign of forward motion.

Connection parameters

Current base: <not connected>

Current follower: <not connected>

Number of sensor / actuator ports: 0

Parameters

Axes | Advanced

Name	Primitive	Axis of Action [x y z]	Reference CS
S	spherical	N/A	N/A

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive rotation is the follower rotating in the right-hand sense as shown in the figure above.

Current base

When you connect the base (B) connector port on the Spherical block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Spherical Base and Follower Body Connector Ports on page 2-250.

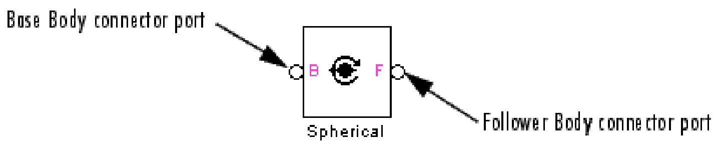
Current follower

When you connect the follower (F) connector port on the Spherical block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Spherical Base and Follower Body Connector Ports on page 2-250.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Sensor blocks to this Joint. The default is `0`. A Spherical cannot be connected to a Joint Actuator.

The motion of a Spherical is three DoFs specified in quaternion form.



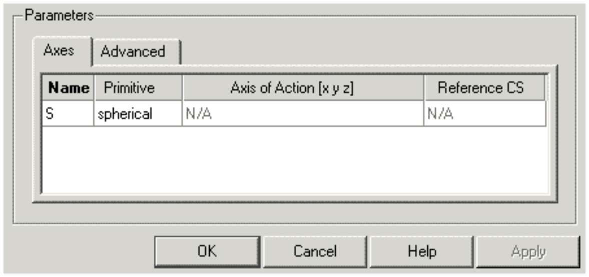
Spherical Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are automatic. They specify the orientation of the spherical DoF that the Spherical represents.



Name	Primitive	Axis of Action [x y z]	Reference CS
S	spherical	N/A	N/A

Name

This column automatically displays the name of each primitive joint contained in the Joint block. For Spherical, there is only one primitive joint, a spherical, labeled `S`.

Primitive

This column automatically displays the type of each primitive joint contained in the Joint block. For Spherical, there is only one primitive type, labeled `Spherical`.

Reference orientation [x y z]

This field is not active.

Reference CS

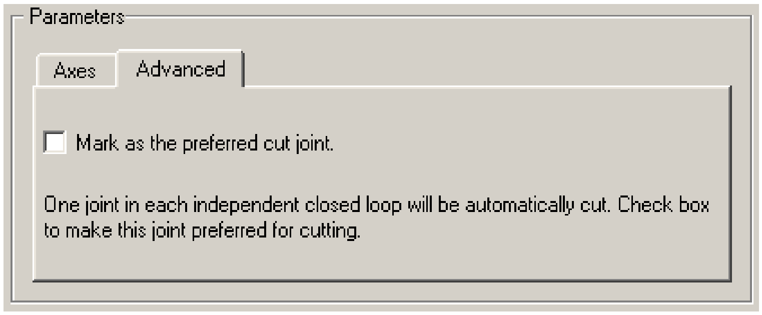
This field is not active.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Disassembled Spherical, Gimbal, Joint Sensor, Prismatic, Revolute

See "Modeling Degrees of Freedom" for more on representing DoFs with Joints.

See "Checking Model Topology" and "How SimMechanics Software Works" for more on closed loops and cutting.

Purpose

Constant-length joint connector with two spatially separated spherical pivot points

Library

Joints/Massless Connectors

Description

The Spherical-Spherical block represents a composite joint composed of two spherical joint primitives. The Body coordinate system (CSs) on either side of the Joint are connected to the spherical primitives. The primitives are separated spatially by a vector of constant length but variable direction connecting the two Body CS origins.

Massless Connector Between Spherical and Spherical Joints

Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have a default of two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) point.

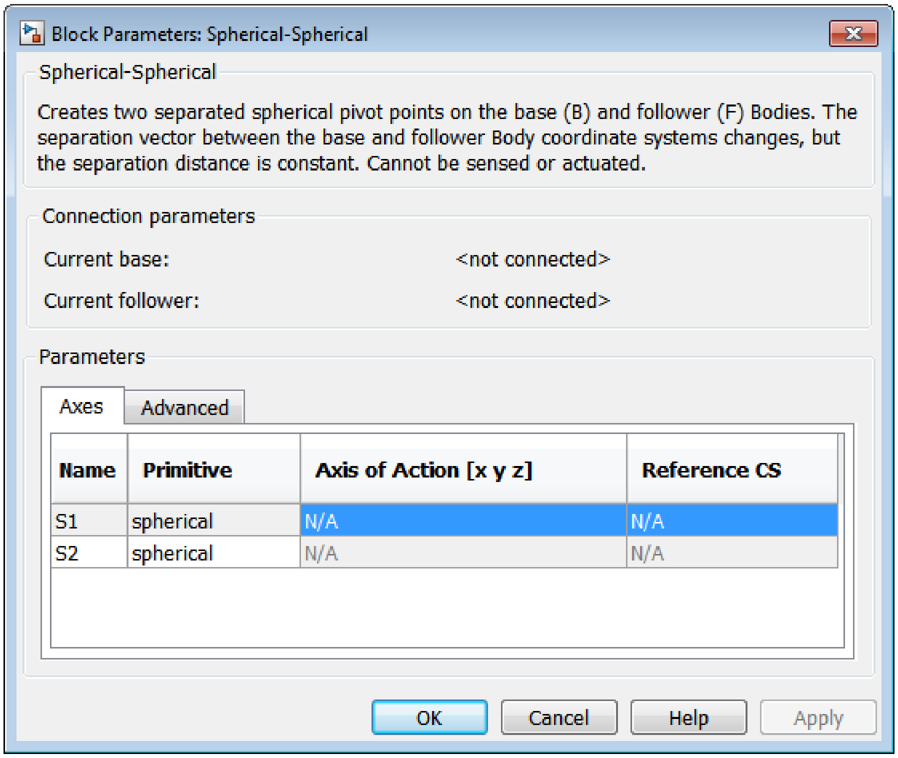
You specify the joint primitive axes, if any, in the Joint dialog.

You cannot connect an Actuator or Sensor to a Massless Connector.

Assembly Restrictions on Massless Connectors

Both joint primitives are assembled but spatially separated. That is, the connected Body CS origins must not be spatially collocated points.

The distance separating the two ends of the connector is computed automatically from the Body CS origins to which the Joint is connected. This distance (the magnitude of the vector between the Body CS origins) remains fixed at its initial value during the simulation. This initial value must be nonzero.



Spherical-Spherical

Creates two separated spherical pivot points on the base (B) and follower (F) Bodies. The separation vector between the base and follower Body coordinate systems changes, but the separation distance is constant. Cannot be sensed or actuated.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>

Parameters

Axes | Advanced

Name	Primitive	Axis of Action [x y z]	Reference CS
S1	spherical	N/A	N/A
S2	spherical	N/A	N/A

Dialog Box and Parameters

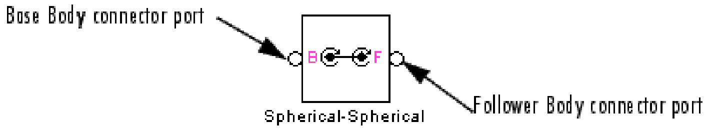
The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive rotation is the base or follower pivoting as shown by the motion figure in the Spherical block reference page.

Current base
When you connect the base (B) connector port on the Spherical-Spherical block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Spherical-Spherical Base and Follower Body Connector Ports on page 2-255.

Current follower
When you connect the follower (F) connector port on the Spherical-Spherical block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Spherical-Spherical Base and Follower Body Connector Ports on page 2-255.



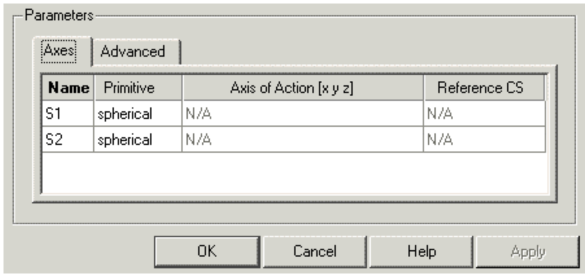
Spherical-Spherical Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are automatic. They specify the orientation of the spherical DoFs that the Spherical-Spherical represents.



Name	Primitive	Axis of Action [x y z]	Reference CS
S1	spherical	N/A	N/A
S2	spherical	N/A	N/A

Name
: This column automatically displays the name of each primitive joint contained in the Joint block. For Spherical-Spherical, there are two spherical primitives, labeled `S1` and `S2`, connecting to base and follower, respectively.

Primitive
: This column automatically displays the type of each primitive joint contained in the Joint block. For Spherical-Spherical, there is only one primitive type, labeled `Spherical`.

Axis of Action [x y z]
: These fields are not active.

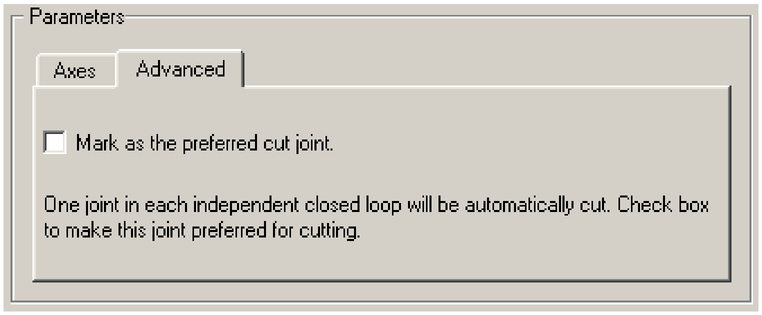
Reference CS
: These fields are not active.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Spherical

See "Modeling Degrees of Freedom" for more on representing DoFs with Massless Connectors.

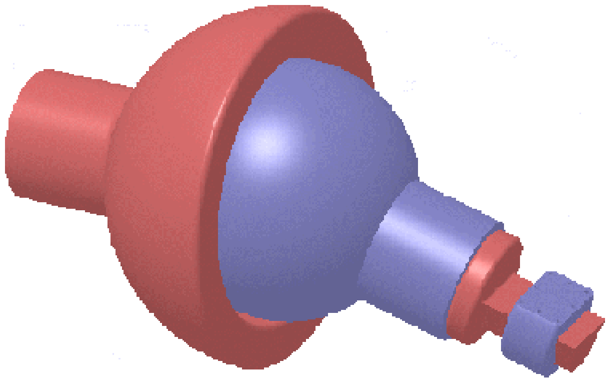
See "Checking Model Topology" and "How SimMechanics Software Works" for more on closed loops and cutting.

Telescoping

Purpose Joint with three revolute and one prismatic joint primitives

Library Joints

Description The Telescoping block represents a composite joint with one translational degrees of freedom (DoF) as one prismatic primitive and three rotational DoFs as one spherical primitive. There are no constraints among the primitives. Unlike Bearing, Telescoping represents the rotational DoFs as one spherical, rather than as three revolutes.



Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

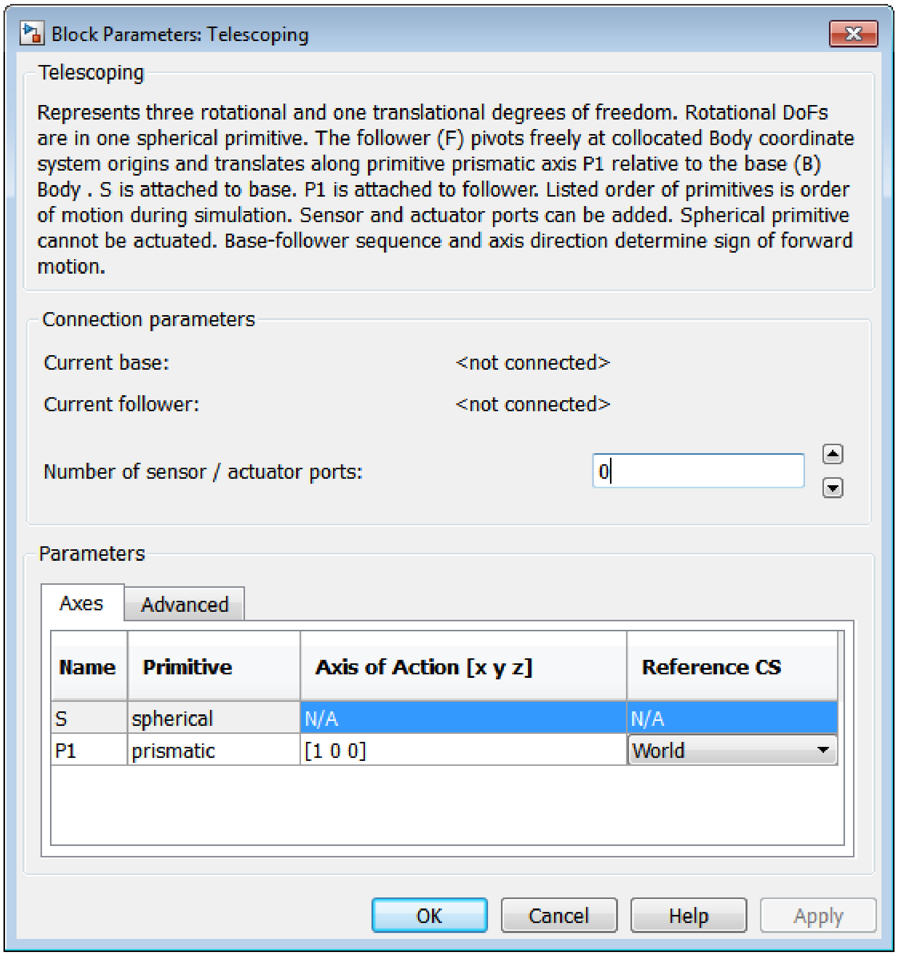
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space



Block Parameters: Telescoping

Telescoping

Represents three rotational and one translational degrees of freedom. Rotational DoFs are in one spherical primitive. The follower (F) pivots freely at collocated Body coordinate system origins and translates along primitive prismatic axis P1 relative to the base (B) Body . S is attached to base. P1 is attached to follower. Listed order of primitives is order of motion during simulation. Sensor and actuator ports can be added. Spherical primitive cannot be actuated. Base-follower sequence and axis direction determine sign of forward motion.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>
Number of sensor / actuator ports:	0

Parameters

Axes | Advanced

Name	Primitive	Axis of Action [x y z]	Reference CS
S	spherical	N/A	N/A
P1	prismatic	[1 0 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive translation is the follower moving in the direction of the translation axis. Positive spherical motion is the follower rotating in the right-handed sense as shown in the Spherical block figure.

Current base

When you connect the base (B) connector port on the Telescoping block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Telescoping Base and Follower Body Connector Ports on page 2-262.

The base Body is automatically connected to the first joint primitive `S` in the primitive list in **Parameters**.

Current follower

When you connect the follower (F) connector port on the Telescoping block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Telescoping Base and Follower Body Connector Ports on page 2-262.

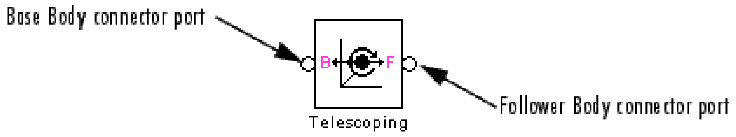
The follower Body is automatically connected to the last joint primitive `P1` in the primitive list in **Parameters**.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`.

The motion of prismatic primitives is specified in linear units. The motion of spherical primitives is specified by a dimensionless quaternion.

Telescoping



Telescoping Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. Each DoF primitive in Telescoping has an entry line. These lines specify the direction of the axes of action of the DoFs that Telescoping represents.

Name - Primitive
The primitive list states the names and types of joint primitives that make up the Telescoping block: spherical primitive `S` and prismatic primitives `P1`.

Axis of Action [x y z]
Enter here as a three-component vector the directional axes defining the allowed motions of these primitives and their corresponding DoFs:

- Prismatic: axis of translation
- Spherical: field is not active

The default vectors are shown in the dialog above. The axis is a directed vector whose overall sign matters.

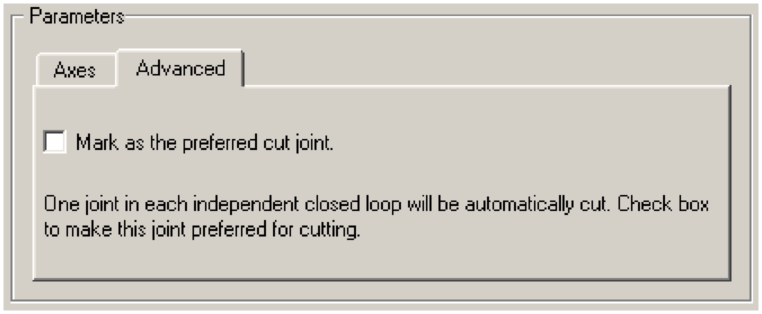
Reference CS
Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of action is oriented with respect to. This CS also determines the absolute meaning of forces/torques and motion along/about the joint axis. The default is `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Bearing, Prismatic, Six-DoF, Spherical

See “Modeling Degrees of Freedom” for more on representing DoFs with Joints.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on closed loops and cutting.

Universal

Purpose

Joint with two revolute joint primitives

Library

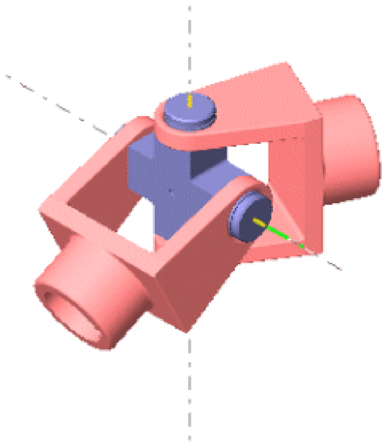
Joints

Description

The Universal block represents a composite joint with two rotational degrees of freedom (DoFs) as two revolute primitives. There are no constraints among the primitives.

Warning

A joint with two revolute primitives becomes singular if the two rotation axes become parallel ("gimbal lock"). The simulation stops with an error in this case.



Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower).

You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

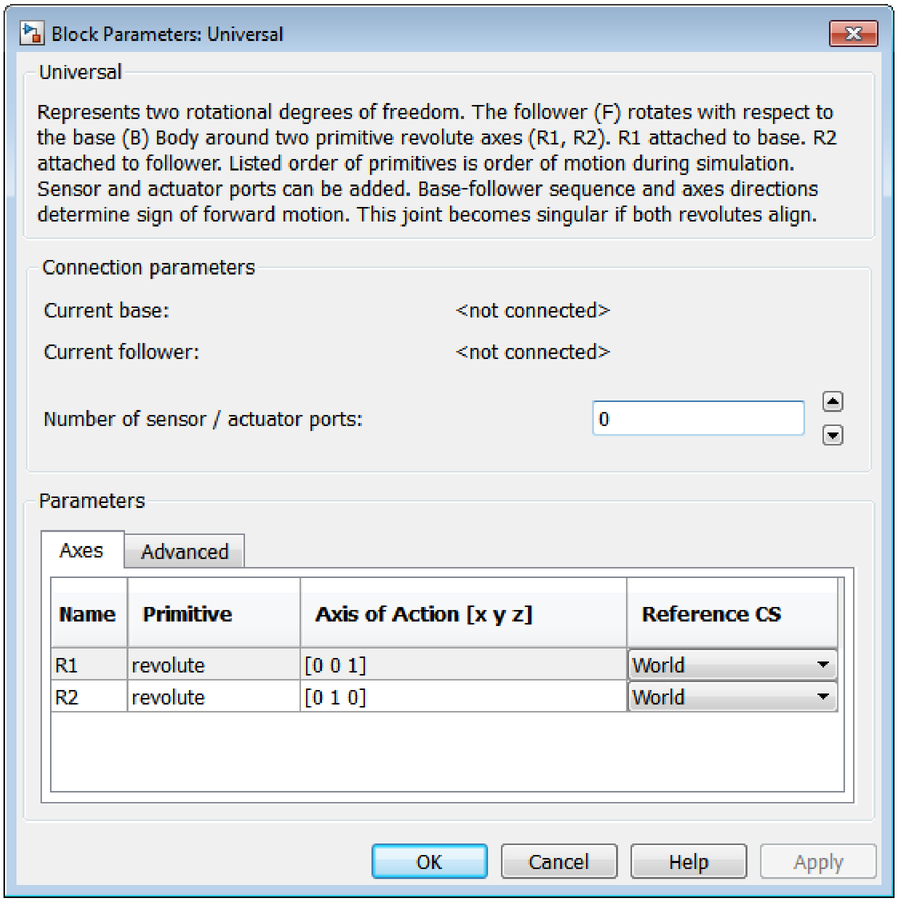
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space



Block Parameters: Universal

Universal

Represents two rotational degrees of freedom. The follower (F) rotates with respect to the base (B) Body around two primitive revolute axes (R1, R2). R1 attached to base. R2 attached to follower. Listed order of primitives is order of motion during simulation. Sensor and actuator ports can be added. Base-follower sequence and axes directions determine sign of forward motion. This joint becomes singular if both revolutes align.

Connection parameters

Current base: <not connected>

Current follower: <not connected>

Number of sensor / actuator ports: 0

Parameters

Axes | Advanced

Name	Primitive	Axis of Action [x y z]	Reference CS
R1	revolute	[0 0 1]	World
R2	revolute	[0 1 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive rotation is the follower moving around the rotational axis following the right-hand rule.

Current base

When you connect the base (B) connector port on the Universal block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Universal Base and Follower Body Connector Ports on page 2-267.

The base Body is automatically connected to the first joint primitive `R1` in the primitive list in **Parameters**.

Current follower

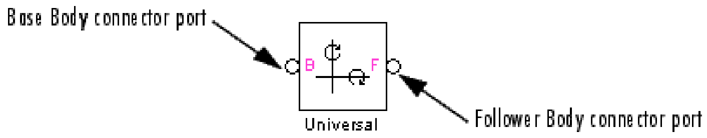
When you connect the follower (F) connector port on the Universal block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Universal Base and Follower Body Connector Ports on page 2-267.

The follower Body is automatically connected to the last joint primitive `R2` in the primitive list in **Parameters**.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Actuator and Joint Sensor blocks to this Joint. The default is `0`.

The motion of revolute primitives is specified in angular units.



Universal Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are required. Each DoF primitive in Universal has an entry line. These lines specify the direction of the axes of action of the DoFs that the Universal represents.

Name - Primitive

The primitive list states the names and types of joint primitives that make up the Universal block: revolute primitives `R1`, `R2`.

Axis of Action [x y z]

Enter here as a three-component vector the directional axes defining the allowed motions of these primitives and their corresponding DoFs:

- Revolute: axis of rotation

The default vectors are shown in the dialog above. The axis is a directed vector whose overall sign matters.

To prevent singularities and simulation errors, the two revolute axes cannot be parallel.

Reference CS

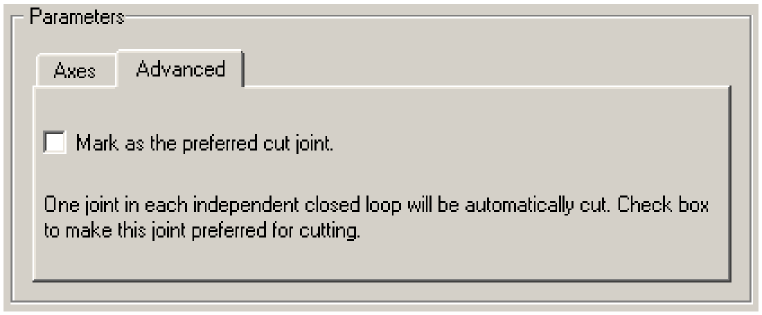
Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of action is oriented with respect to. This CS also determines the absolute meaning of forces/torques and motion along/about the joint axis. The default is `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint

In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Gimbal, Revolute

See “Modeling Degrees of Freedom” for more on representing DoFs with Joints.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on closed loops and cutting.

Variable Mass & Inertia Actuator

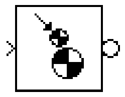
Purpose

Time-dependent mass and inertia parameters

Library

Sensors & Actuators

Description



The Variable Mass & Inertia Actuator block allows you to vary the mass m and/or inertia tensor $\boldsymbol{I}$ of the Body to which it is connected. The general form of Newton's second law for linear or angular motion is

(*mass* or *inertia*) * *acceleration* = external *force* or *torque*

This block externally varies the leftmost parameter in this law of motion with a Simulink signal.

Caution The Variable Mass & Inertia Actuator does *not* apply any thrust forces or torques associated with the Body's mass loss or gain. Such thrust effects would occur on the left-hand side of the force or torque law as terms proportional to the time derivatives of the mass or inertia tensor, dm/dt or $d\boldsymbol{I}/dt$, multiplied by the related thrust velocities. You must separately apply such thrust forces or torques to the Body with Body Actuators.

How the Actuator Varies a Body's Mass and Inertia Tensor

You connect the Variable Mass & Inertia Actuator block to the original Body at a Body coordinate system (CS). You can connect multiple Variable Mass & Inertia Actuators to a single Body, each Actuator at a separate Body CS port. If Body CS ports are lacking, open the Body dialog and create them as needed.

At each Body CS so connected, the Variable Mass & Inertia Actuator creates an invisible body. The attachment is equivalent to connecting another Body with a Weld, except that the other body's mass properties vary with time. This invisible body has a time-varying mass and/or symmetric inertia tensor supplied by the external Simulink signal. The center of gravity coordinate system (CG CS) of the invisible body is identical to the attached Body CS. The inertia tensor of the invisible body is evaluated at this CS, in this coordinate system's axes.

The Composite Body

Once started, a SimMechanics simulation creates a combined or *composite* body, made of the invisible, time-varying body created by the Actuator and the original Body. The total mass of the composite body is the sum of the visible Body and the invisible body's masses. The CG of this composite body is recomputed at each time step. The inertia tensor of the composite body is formed at each time step by combining the inertia tensors of the visible Body and the invisible body. The combined inertia tensor is then evaluated at the composite body's new CG.

What The Invisible Body Requires

The time-varying mass and inertia tensor of the invisible body must satisfy these requirements:

- The mass and principal inertial moments can be positive, negative, or zero.

  The only restriction is that the total mass and the principal inertial moments of the composite body be nonnegative.
- The time-varying inertia tensor of the invisible body must be symmetric.

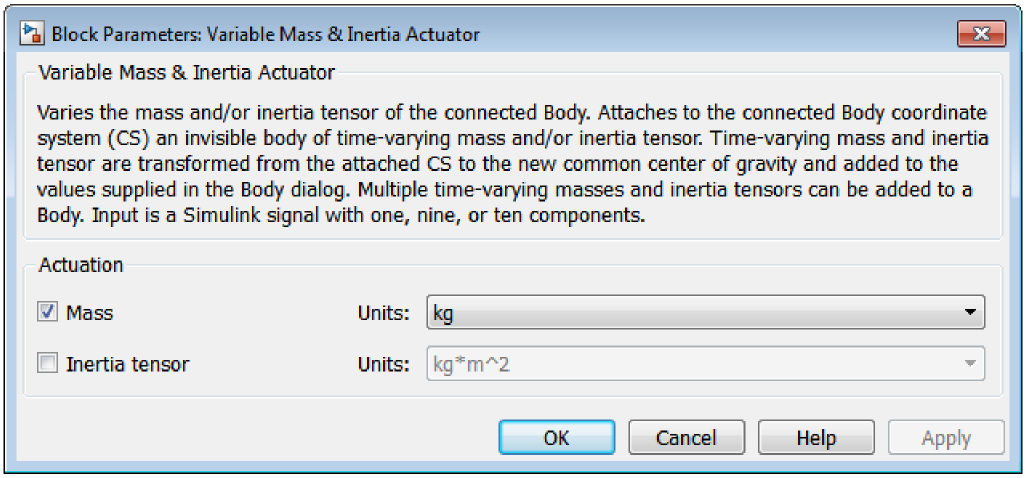
You can mix variable mass and/or variable inertia tensor actuation.

Actuation	Effect on Connected Body
Variable mass alone	Adds a time-varying point mass at the attached Body CS
Variable inertia tensor alone	Adds time-varying inertia tensor at the attached Body CS without changing the composite body's total mass
Variable mass and inertia tensor combined	Adds invisible body with time-varying mass and inertia tensor at the attached Body CS

What Does Not Vary in the Original Body

While the invisible, attached body and the invisible composite body have time-varying mass properties, you do not see any visible changes in the original Body that you are actuating. The mass properties in its dialog do not change.

If you are visualizing the varying-mass/inertia actuated Body as an equivalent ellipsoid, the ellipsoid is displayed using the static data in the Body dialog itself. The displayed ellipsoid ignores the effect of any Variable Mass & Inertia Actuators attached to the Body. See "About Body Color and Geometry".


Block Parameters: Variable Mass & Inertia Actuator

Variable Mass & Inertia Actuator

Varies the mass and/or inertia tensor of the connected Body. Attaches to the connected Body coordinate system (CS) an invisible body of time-varying mass and/or inertia tensor. Time-varying mass and inertia tensor are transformed from the attached CS to the new common center of gravity and added to the values supplied in the Body dialog. Multiple time-varying masses and inertia tensors can be added to a Body. Input is a Simulink signal with one, nine, or ten components.

Actuation

[✓] Mass Units: kg

[ ] Inertia tensor Units: kg*m^2

Dialog Box and Parameters

The dialog has one active area, **Actuation**.

Actuation

You can apply a variable mass, a variable inertia tensor, or both, to a body.

If you apply both, you need to bundle the variable mass and inertia tensor into a 10-component signal, in the order shown in the dialog.

Mass

Select the check box to apply an external time-varying mass from the input Simulink signal. In the pull-down menu to the right, select units for this time-varying mass. The default is `kg` (kilograms).

Inertia tensor

Select the check box to apply an external time-varying inertia tensor from the input Simulink signal. In the pull-down menu to the right, select units for this time-varying inertia tensor. The default is `kg-m`^{2} (kilogram-meters^2).

The Simulink input signal has the following components. For variable mass or inertia tensor actuation alone, omit the missing components.

Time-varying mass (scalar)	Time-varying inertia tensor (9-vector): $(I_{11}, I_{21}, I_{31}, I_{12}, \ldots)$

References

[1] Corbin, H. C., and P. Stehle, *Classical Mechanics*, Second Edition, New York, Dover Publications, 1994 (original edition, 1960), chapters 2, 5, and 9.

[2] Goldstein, H., *Classical Mechanics*, Second Edition, Reading, Massachusetts, Addison-Wesley, 1980, chapters 4 and 5.

[3] Piscane, V. L., and R. C. Moore, eds., *Fundamentals of Space Systems*, Johns Hopkins University/Applied Physics Laboratory Series, New York, Oxford University Press, 1994, chapters 3, 4, and 5.

See Also

Body, Body Actuator, Weld

See "Varying a Body's Mass and Inertia Tensor" for more on varying the mass and inertia tensor of a body.

Velocity Driver

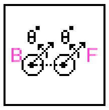
Purpose

Linear and angular velocity components of base and follower body coordinate systems

Library

Constraints & Drivers

Description



The Velocity Driver block drives a linear combination of the projected translational and angular velocities of two Bodies. The velocities are projected by inner products on to constant vectors you specify.

The subscripts B and F refer to base and follower bodies, respectively. Let

- $\boldsymbol{v}_B$, $\boldsymbol{v}_F$ be the two body velocity vectors, measured in World.
- ω_B, ω_F be the two body angular velocity vectors, measured in World.
- $\boldsymbol{c}_B$, $\boldsymbol{c}_F$, $\boldsymbol{d}_B$, $\boldsymbol{d}_F$ be constant vectors.

The Velocity Driver block specifies the linear combination Ω:

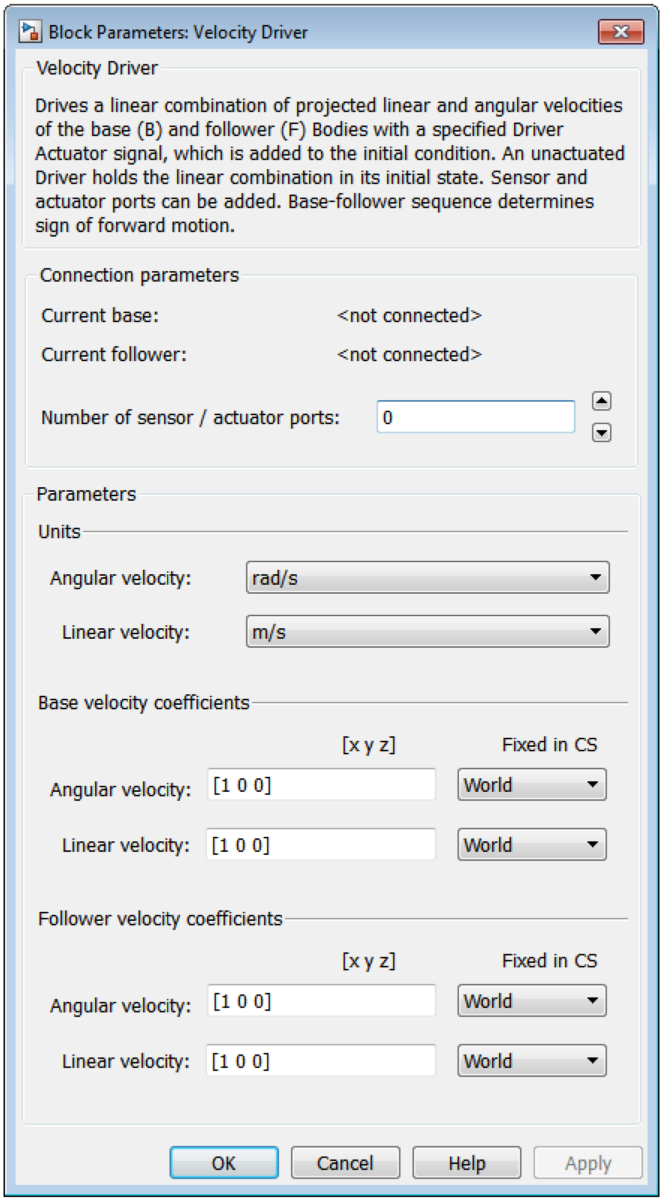
$$\Omega = \boldsymbol{c}_B \cdot \boldsymbol{v}_B + \boldsymbol{d}_B \cdot \boldsymbol{\omega}_B - \boldsymbol{c}_F \cdot \boldsymbol{v}_F - \boldsymbol{d}_F \cdot \boldsymbol{\omega}_F = \Omega(t=0) + f(t)$$

as a function of time *f(t)*. You specify the vectors $\boldsymbol{c}_B$, $\boldsymbol{c}_F$, $\boldsymbol{d}_B$, $\boldsymbol{d}_F$. You also connect a Driver Actuator block to the Velocity Driver.

The Simulink input signal into the Driver Actuator specifies the time-dependent driving function *f(t)* and its first two derivatives, as well as their units. If you do not actuate Velocity Driver, this block acts as a time-independent constraint that freezes the constraint linear combination at its initial value Ω(*t*=0) during the simulation.

Drivers restrict relative degrees of freedom (DoFs) between a pair of bodies as specified functions of time. Locally in a machine, they replace a Joint as the expression of the DoFs. Globally, Driver blocks must occur topologically in closed loop. Like Bodies connected to a Joint, the two Bodies connected to a Drivers are ordered as base and follower, fixing the direction of relative motion.

You can also connect a Constraint & Driver Sensor to any Driver block and measure the reaction forces/torques between the driven bodies.


Block Parameters: Velocity Driver

Velocity Driver

Drives a linear combination of projected linear and angular velocities of the base (B) and follower (F) Bodies with a specified Driver Actuator signal, which is added to the initial condition. An unactuated Driver holds the linear combination in its initial state. Sensor and actuator ports can be added. Base-follower sequence determines sign of forward motion.

Connection parameters

Current base:	<not connected>
Current follower:	<not connected>
Number of sensor / actuator ports:	0

Parameters

Units

Angular velocity:	rad/s
Linear velocity:	m/s

Base velocity coefficients

	[x y z]	Fixed in CS
Angular velocity:	[1 0 0]	World
Linear velocity:	[1 0 0]	World

Follower velocity coefficients

	[x y z]	Fixed in CS
Angular velocity:	[1 0 0]	World
Linear velocity:	[1 0 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

The base (B)-follower (F) Body sequence determines the sense of positive motion. Positive translation is the follower moving in the direction of the translation axis. Positive rotation is the follower rotating in the right-handed sense about the rotation axis.

Current base

When you connect the base (B) connector port on the Velocity Driver block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Velocity Driver Base and Follower Body Connector Ports on page 2-276.

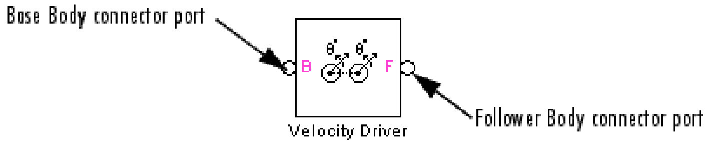
Current follower

When you connect the follower (F) connector port on the Velocity Driver block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Velocity Driver Base and Follower Body Connector Ports on page 2-276.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Driver Actuator and Constraint & Driver Sensor blocks to this Driver. The default is `0`.

To activate the Driver, connect a Driver Actuator.



Velocity Driver Base and Follower Body Connector Ports

Parameters

The **Parameters** fields are grouped into three sets, **Units**, **Base velocity coefficients**, and **Follower velocity coefficients**.

Units

The vectors $\boldsymbol{c}_B$, $\boldsymbol{c}_F$, $\boldsymbol{d}_B$, $\boldsymbol{d}_F$ carry the implicit units conversion to convert all velocities to the common linear velocity units of *f(t)* that you set in the Driver Actuator connected to the Velocity Driver block.

Angular velocity

From the pull-down menu, choose the common units for all angular velocities. The default is `rad/s` (radians/second).

The vectors $\boldsymbol{d}_B$ and $\boldsymbol{d}_F$ implicitly carry the units conversion of length/angle. The driving function *f(t)* has the linear velocity units that you set in the Driver Actuator block that you connect to Velocity Driver. If the *f(t)* units differ from the units set in **Linear velocity units** in this dialog, the vectors $\boldsymbol{d}_B$ and $\boldsymbol{d}_F$ implicitly carry the additional units conversion.

Linear velocity

From the pull-down menu, choose the common units for all linear velocities. The default is `m/s` (meters/second).

The driving function *f(t)* has the linear velocity units that you set in the Driver Actuator block that you connect to the Velocity Driver. If the *f(t)* units differ from the units set here, the vectors $\boldsymbol{c}_B$ and $\boldsymbol{c}_F$ implicitly carry the units conversion.

Base Velocity Coefficients

Angular velocity

Under **[x y z]**, enter the **Angular velocity** coefficient vectors for the base Body. These are the components of $\boldsymbol{d}_B$. The default is `[1 0 0]`.

In the **Fixed in CS** pull-down menu, choose which set of coordinates axes, `World` or `Base`, define these vector coefficients. The default is `World`.

Linear Velocity

Under **[x y z]**, enter the **Linear velocity** coefficient vectors for the base Body. These are the components of $\boldsymbol{c}_{\mathrm{B}}$. The default is `[1 0 0]`.

In the **Fixed in CS** pull-down menu, choose which set of coordinates axes, `World` or `Base`, define these vector coefficients. The default is `World`.

Follower Velocity Coefficients

Angular velocity

Under **[x y z]**, enter the **Angular velocity** coefficient vector for the follower Body. These are the components of $\boldsymbol{d}_{\mathrm{F}}$. The default is `[1 0 0]`.

In the **Fixed in CS** pull-down menu, choose which set of coordinates axes, `World` or `Follower`, define these vector coefficients. The default is `World`.

Linear Velocity

Under **[x y z]**, enter the **Linear velocity** coefficient vector for the base Body. These are the components of $\boldsymbol{c}_{\mathrm{F}}$. The default is `[1 0 0]`.

In the **Fixed in CS** pull-down menu, choose which set of coordinates axes, `World` or `Follower`, define these vector coefficients. The default is `World`.

See Also

Angle Driver, Constraint & Driver Sensor, Driver Actuator, Parallel Constraint

See “Constraining and Driving Degrees of Freedom” for more on restricting DoFs with Drivers.

See “Checking Model Topology” and “How SimMechanics Software Works” for more on using drivers in closed loops.

Purpose

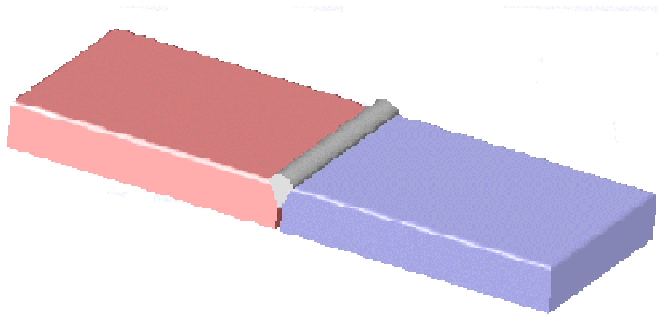
Joint with zero degrees of freedom

Library

Joints

Description

The Weld block represents a joint with no degrees of freedom (DoFs). The two Bodies connected to either side of the Weld block are locked rigidly to one another, with no possible relative motion.



Satisfying Joint Requirements

A Joint block represents the relative degrees of freedom between two bodies, not the bodies themselves.

You must connect any Joint block to two and only two Body blocks, the base and the follower. All Joints have two connector ports for these connections, defining the direction of joint motion (base to follower). You connect each side of the Joint block to these Body blocks at a Body coordinate system (CS) port.

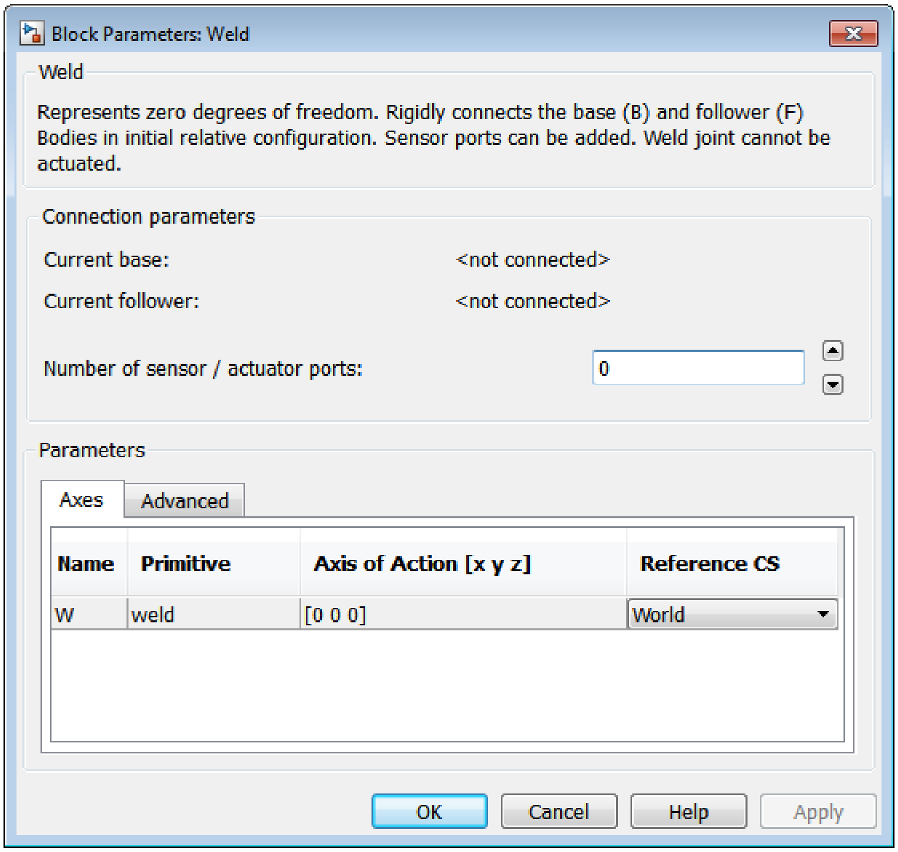
You specify the joint primitive axes, if any, in the Joint dialog.

Assembly Restrictions on Assembled Joints

This Joint block is assembled and places restrictions on the connected Body CSs.

- If the Joint has no prismatic primitives, the origins of the connected Body CSs on either side of the Joint must be spatially collocated points, to within assembly tolerances.
- If the Joint has one or more prismatic primitives, the origins of the connected Body CSs must lie in the span of the prismatic axes:

Number of Prismatic Primitives	Span of Primitive Axes
One	Along the primitive axis
Two	In the plane of the primitive axes
Three	Anywhere in three-dimensional space


Block Parameters: Weld

Weld

Represents zero degrees of freedom. Rigidly connects the base (B) and follower (F) Bodies in initial relative configuration. Sensor ports can be added. Weld joint cannot be actuated.

Connection parameters

Current base: <not connected>

Current follower: <not connected>

Number of sensor / actuator ports: 0

Parameters

Name	Primitive	Axis of Action [x y z]	Reference CS
W	weld	[0 0 0]	World

Dialog Box and Parameters

The dialog has two active areas, **Connection parameters** and **Parameters**.

Connection Parameters

Current base

When you connect the base (B) connector port on the Weld block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Weld Base and Follower Body Connector Ports on page 2-282.

The base Body is automatically connected to the joint primitive `W` in the primitive list in **Parameters**.

Current follower

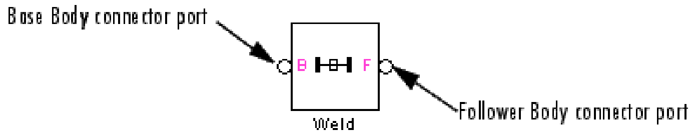
When you connect the follower (F) connector port on the Bushing block to a Body CS Port on a Body, this parameter is automatically reset to the name of this Body CS. See the following figure, Weld Base and Follower Body Connector Ports on page 2-282.

The follower Body is automatically connected to the joint primitive `W` in the primitive list in **Parameters**.

Number of sensor/actuator ports

Using this spinner menu, you can set the number of extra connector ports needed for connecting Joint Sensor blocks to this Joint. The default is `0`.

You cannot actuate a Weld joint, and a Weld joint undergoes no motion. A Joint Sensor measures zero motion, but in general nonzero reaction forces, at this joint.



Weld Base and Follower Body Connector Ports

Parameters

Switch between the **Axes** and **Advanced** tabs.

Axes Tab

The entries on the **Axes** tab are inactive for Weld. This block has no DoF primitives.

Name - Primitive

The primitive list states the names and types of joint primitives that make up the Weld block: a rigid primitive `W` representing no motion.

Axis of Action [x y z]
: This field is inactive.

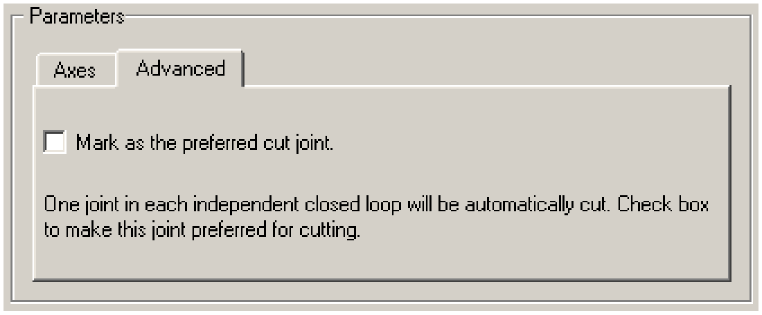
Reference CS
: Using the pull-down menu, choose the coordinate system (World, the base Body CS, or the follower Body CS) whose coordinate axes the vector axis of action is oriented with respect to. This CS also determines the absolute meaning of forces/torques and motion along/about the joint axis. The default is `World`.

Restricted Parameters

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

- The **Axes** (joint primitives) parameters table

Advanced Tab



☐ Mark as the preferred cut joint.

One joint in each independent closed loop will be automatically cut. Check box to make this joint preferred for cutting.

The **Advanced** tab is optional. You use it to control the way SimMechanics simulation interprets the topology of your schematic diagram.

Mark as the preferred cut joint
: In a closed loop, the simulation internally and automatically cuts one and only one joint.

If you want this particular joint to be weighted preferentially for cutting during the simulation, select the check box. The default is not selected.

See Also

Distance Driver

See "Modeling Degrees of Freedom" for more on representing DoFs with Joints.

See "Checking Model Topology" and "How SimMechanics Software Works" for more on closed loops and cutting.

Function Reference

import_physmod

Purpose

Generate model from Physical Modeling XML file

Note `import_physmod` will be removed in a future release. Use `mech_import` instead.

Syntax

```
import_physmod
import_physmod('filename.xml')
import_physmod('filename.xml', option1, value1, option2,
   value2, ...)
```

Description

`import_physmod` with no inputs opens the Import Physical Modeling XML dialog box. In the dialog box, you select the XML file to import and the function options.

`import_physmod('filename.xml')` generates a SimMechanics model from a Physical Modeling XML file called `filename.xml`. The `.xml` extension for the `filename` input is optional. For a computer-aided design (CAD)-generated XML file, the name of the generated model is the same as the original CAD assembly, regardless of the name of the XML file.

`import_physmod('filename.xml', option1, value1, option2, value2, ...)` generates the SimMechanics model `filename` from `filename.xml` using the specified option-value pairs when importing. The `.xml` extension for the `filename` input is optional.

See Also

`mech_import`

Purpose

SimMechanics states from Simulink state vector

Syntax

```
[vector_mgr, mech_states] = mech_get_states(X, block)
[vector_mgr, mech_states] = mech_get_states(X, vectorMgr)
```

Description

`[vector_mgr, mech_states] = mech_get_states(X, block)` returns a state vector manager object. The object's values reflect the SimMechanics states in the Simulink state vector `X` for the SimMechanics machine containing `block`. The function also returns the mechanical states `vector_mgr.X` in `mech_states`.

`[vector_mgr, mech_states] = mech_get_states(X, vectorMgr)` extracts the mechanical states from the Simulink state vector `X` and returns them as an array of states assigned to `vector_mgr.X`.

The returned vector manager is an instance of the `MECH.StateVectorMgr` class, the same class as returned by the `mech_stateVectorMgr` function.

Input Arguments

`mech_get_states` accepts two possible combinations of two inputs.

X

The Simulink state vector for your model. `X` must be compatible with your Simulink model.

block

A string or block handle specifying the absolute path of a block in the machine that you want to query.

vectorMgr

A mechanical state vector manager object of the `MECH.StateVectorMgr` class.

Output Arguments

`mech_get_states` produces one or two outputs.

vector_mgr

An instance of the MECH.StateVectorMgr object class whose values reflect the mechanical state of the model.

mech_states

A vector of the mechanical state values. Identical to vector_mgr.X, if you use the first form of the function.

Examples

Assigning a Final Model State

Simulate a Stewart platform model for 10 seconds, capturing the model state over the entire simulation:

```
simOut = sim('mech_stewart_trajectory','StopTime','10', ...
   'SaveState','on','StateSaveName','xout');
xout = simOut.get('xout');
```

Populate the state vector manager with this model's final state:

```
stewartStateVectorMgr = mech_get_states(xout(end,:), ...
   'mech_stewart_trajectory/Plant/Machine Environment');
```

See Also

mech_runtime_states | mech_set_states | mech_stateVectorMgr | mech_transfer_states | sim | states

Purpose

Generate model from Physical Modeling XML file

Syntax

```
mech_import
mech_import('filename.xml')
mech_import('filename.xml', option1, value1, option2, value2,
   ...)
```

Description

`mech_import` opens the Import Physical Modeling XML dialog box. See "Alternatives" on page 3-10.

`mech_import('filename.xml')` generates a SimMechanics model from a Physical Modeling XML file called `filename.xml`. For an XML file generated from computer-aided design (CAD), the name of the generated model is the same as the original CAD assembly, regardless of the name of the XML file.

`mech_import('filename.xml', option1, value1, option2, value2, ...)` generates the SimMechanics model from `filename.xml`, using the specified option-value pairs when importing.

The `mech_import` function either generates a new SimMechanics model or updates a previously generated SimMechanics model that you specify. It also generates or updates the associated body geometry files.

Input Arguments

`mech_import` requires one input and accepts additional optional inputs.

'filename.xml'

Input XML file name. Replace *`filename`* with the name of the XML file that you want to import. The `.xml` extension in this input is optional.

This input is required.

Default: `''`

'ImportMode'

`0` – Import XML into a new model.

`1` – Use XML to update contents of a model or subsystem.

`2` – Add contents of new XML to a model or subsystem.

`3` – Import contents of new XML into a model or subsystem and delete the existing contents.

This input is optional.

> **Default:** `0`

'ModelToImportInto'

String specifying the target model for importing or updating existing blocks. This input is optional.

> **Default:** Original CAD assembly name

'SubsystemToImportInto'

String specifying the target subsystem for importing or updating existing blocks. This input is optional.

> **Default:** Original CAD subassembly name

'CreateTopSS'

`true` or `false` – Whether to create a new subsystem if the specified subsystem (`'SubsystemToImportInto'`) does not exist in the target model. Applies when `'ImportMode'` = 2. This input is optional.

> **Default:** `false`

'ModelSimplificationOpt'

`0` – Follow the imported hierarchy and make no simplifications to the generated subsystem hierarchy.

`1` – Bring all joints to the top model or subsystem. Group and place welded bodies into newly created rigid subsystems at each level of model hierarchy.

`2` – Group and place welded bodies at each level of model hierarchy into newly created rigid subsystems at the same level.

This input is optional.

Default: 0

'UseDefaultJointNames'

`'on'` – Use default joint names such as Revolute1, Revolute2, etc., and ignore joint names specified in the XML file.

`'off'` – Use the joint names from the XML file.

This input is optional.

Default: `'on'`

'UseBlockNamesForSpacing'

`'on'` – Space blocks so that their names do not overlap.

`'off'` – Use only block size to space blocks.

This input is optional.

Default: `'off'`

'LayoutDirection'

`'LR'` – Diagram grows from left to right.

`'TB'` – Diagram grows from top to bottom.

This input is optional.

Default: `'LR'`

'LayoutWithUpdate'

`true` or `false` – Whether to arrange all imported blocks after updating their parameters. Applies only when updating. If false, arrange newly imported blocks separately from the existing blocks. This input is optional.

Default: `false`

'EnableIndvlBlkUpdCtrl'

`true` or `false` – Whether to respect the update settings of individual blocks. If false, update all blocks regardless. This input is optional.

Default: `true`

'NewJointsAtTopLevel'

`true` or `false` – Whether or not to add new joints in the updated model at the highest level of the model hierarchy specified by either the `ModelToImportInto` or the `SubsystemToImportInto` option. Applies when `'ImportMode'` = 1. This input is optional.

Default: `false`

'BackupMode'

`0` – Do not create a backup version of the model while updating an existing model.

`1` – Create a backup version of the model in the same folder as the updated model.

`2` – Specify a name for the backup version of the model and a location.

This input is optional.

Default: 1

'BackupLocation'

Specify the complete path where importer will create the backup version of the model. Applies when `'BackupMode'` = 2. This input is optional.

Default: `''`

Output Arguments

`mech_import` produces or updates one model file.

The default output model name is the same as the assembly name specified in the Physical Modeling XML file. The output model name is independent of the name of the XML file used as the input.

Examples

Importing a New Model

Create the Physical Modeling XML file for a hypothetical CAD assembly called `assembly`:

1. From your CAD platform, export the assembly. Call the file `assembly.xml`.
2. Place or copy the XML file into your current MATLAB working folder.
3. At the command line, enter
4.

   ```
   mech_import('assembly.xml')
   ```

   SimMechanics software generates the block diagram in a model file called `assembly.mdl`. You can save, rename, modify, and run this model.

Importing a New Model with a Nondefault Name

Change a model's name from the default:

- When you invoke the function to import the XML file, use the `'ModelToImportInto'` option.
- When you use the dialog box to import the XML file, use the **Specify model to import into** field.
- After you import the XML file and generate the model, when you save the model.

Apply a nondefault name to a final generated model in this way:

```
mech_import('robot.xml','ImportMode',0, ...
          'ModelToImportInto','myrobot')
```

This form of the function imports the example file, `robot.xml`, into a new SimMechanics model called myrobot.

Importing a New Model While Simplifying Model Hierarchy

At the command line, enter

```
showdemo mech_modelsimplification
```

to display the example page and learn about to simplify the joints and welded bodies in a generated model.

Updating an Existing Model with Several Options

Update an existing model (existingModel) with the XML file `existingModelUpdate.xml`:

```
mech_import('existingModelUpdate.xml','ImportMode',1, ...
           'ModelToImportInto','existingModel', ...
           'LayoutWithUpdate',true,'EnableIndvlBlkUpdCtrl',true, ...
           'BackupMode',2,'BackupLocation','C:\backupdir\')
```

The importer arranges the updated blocks, while respecting the individual block update settings. The function also creates a backup version of the model and specifies a nondefault backup location (`C:\backupdir\`).

Alternatives

If you enter `mech_import` at the command line with no inputs, the Import Physical Modeling XML dialog box opens.

Use the context-sensitive help in this dialog box for information about its parameters. These parameters provide the same control over generating SimMechanics models from Physical Modeling XML files as do the function inputs.

See Also

Body | Custom Joint | Ground | Machine Environment | Weld

How To

- “Mechanical Import”

mech_runtime_states

Purpose

SimMechanics states from running simulation

Syntax

```
X = mech_runtime_states(vectorMgr)
[X, RTO] = mech_runtime_states(vectorMgr)
[X, RTO] = mech_runtime_states(vectorMgr, RTO)
```

Description

`X = mech_runtime_states(vectorMgr)` returns the vector of instantaneous SimMechanics states in the machine associated with the state vector manager `vectorMgr` in an executing model. The contents of `vectorMgr` are not altered.

`[X, RTO] = mech_runtime_states(vectorMgr)` also returns the Simulink runtime object (RTO) that contains the SimMechanics states. You can use this form to speed up future calls to this function.

`[X, RTO] = mech_runtime_states(vectorMgr, RTO)` uses the Simulink runtime object `RTO` assumed to correspond to the machine associated with `vectorMgr`. This form of the function is faster than the others.

The returned value `X` has a format that you can assign to `vectorMgr.X`, which you can then use to interpret the states.

Input Arguments

`mech_runtime_states` accepts one or two inputs.

vectorMgr

A mechanical state vector manager as returned by the `mech_stateVectorMgr` function. This input is required.

RTO

A Simulink runtime object that corresponds to the machine referenced by `vectorMgr`. This input is optional.

Output Arguments

`mech_runtime_states` produces one or two outputs.

X

The vector of mechanical state values at the instant you query the running model.

RTO

A Simulink runtime object.

Examples

Displaying Mechanical State Values During Simulation

Enter the following script at the command line.

```
modelName = 'mech_stewart_trajectory';
open(modelName);
groundBlock = find_system(modelName, 'Name', 'Ground1');

vm = mech_stateVectorMgr(groundBlock{1});
set_param(modelName, 'SimulationCommand', 'Start')
[vm.X, rto] = mech_runtime_states(vm)

try
   while (true)
      vm.X = mech_runtime_states(vm, rto);
      clc;
      vm.BlockStates(1) % display
      pause(0.5);
   end
catch myException
   set_param(modelName, 'SimulationCommand', 'Stop')
   throw(myException), throwAsCaller(myException);
end

set_param(modelName, 'SimulationCommand', 'Stop')
```

This script displays and updates the state of the first Joint in the Stewart platform model, mech_stewart_trajectory. The results appear in the command window. Press **Ctrl+C** at the keyboard to halt the simulation. (Ignore the error message that results.)

See Also

mech_get_states | mech_set_states | mech_stateVectorMgr | mech_transfer_states | sim | states

How To

- “Access Block Data During Simulation”

Purpose

Populate SimMechanics states in Simulink state vector

Syntax

```
X = mech_set_states(vector_mgr)
X = mech_set_states(vector_mgr, X)
X = mech_set_states(vector_mgr, X, mech_states)
```

Description

`X = mech_set_states(vector_mgr)` returns the Simulink state vector `X` for the model associated with `vector_mgr`. The state vector entries corresponding to the SimMechanics states are filled with the values specified in `vector_mgr`. Entries in `X` that do not correspond to SimMechanics states are set to their initial values, as reported by Simulink.

`X = mech_set_states(vector_mgr, X)` overwrites the entries that correspond to SimMechanics states in the input state vector `X` with the values specified in `vector_mgr`. Entries of `X` that do not correspond to SimMechanics states are left unchanged. Specify `[]` in place of `X` to obtain the initial values for the nonmechanical states, as reported by Simulink.

`X = mech_set_states(vector_mgr, X, mech_states)` overwrites the entries that correspond to SimMechanics states in the input state vector `X` with the values specified in `mech_states` (for example, as reported by `vector_mgr.X`). Entries of `X` that do not correspond to SimMechanics states are left unchanged.

Input Arguments

`mech_set_states` accepts one, two, or three inputs.

vector_mgr

An instance of the object class `MECH.StateVectorMgr` corresponding to the machine. This input is required.

X

A vector of Simulink state values. This input is optional.

mech_states

A vector of the mechanical state values, assigned to `vector_mgr.X`. This input is optional and requires the second optional input, `X`.

Output Arguments

X

The Simulink state vector for the model you are querying, with the mechanical state values set to the values specified by the function.

Examples

Setting Mechanical State Values

Set state values in a simple pendulum model:

```
load_system('mech_spen');
set_param('mech_spen','InitInArrayFormatMsg','None');
vm = mech_stateVectorMgr('mech_spen/Machine Environment');
vm.x = [pi/4 0];
simulinkState = mech_set_states(vm);
```

Simulate, capturing the model state over the entire simulation:

```
simOutIS = sim('mech_spen','InitialState','simulinkState','LoadInitialSta
   'StopTime','5','SaveState','on','StateSaveName','xoutIS');
xoutIS = simOutIS.get('xoutIS');
```

Compare the initial condition `xoutIS(1,:)` with the initial condition obtained without setting the initial state in `xout0(1,:)`:

```
simOut0 = sim('mech_spen','StopTime','5','SaveState','on',...
   'StateSaveName','xout0');
xout0 = simOut0.get('xout0');
```

See Also

`mech_get_states` | `mech_runtime_states` | `mech_stateVectorMgr` | `mech_transfer_states` | `sim` | `states`

Purpose

Create machine state vector manager object

Syntax

```
MachineState = mech_stateVectorMgr('pathname')
MachineState = mech_stateVectorMgr('handle')
MachineState = mech_stateVectorMgr(gcb)
MachineState = mech_stateVectorMgr(gcbh)
```

Description

`MachineState = mech_stateVectorMgr('pathname')` returns a data structure of `MECH.StateVectorMgr` class, starting from the path name of any SimMechanics block in the machine whose state you want.

`MachineState = mech_stateVectorMgr('handle')` returns a data structure of `MECH.StateVectorMgr` class, the handle of any SimMechanics block in the machine whose state you want.

`MachineState = mech_stateVectorMgr(gcb)` returns a data structure of `MECH.StateVectorMgr` class, starting with an indirect call to the block path name of a selected SimMechanics block in your machine.

`MachineState = mech_stateVectorMgr(gcbh)` returns a data structure of `MECH.StateVectorMgr` class, starting with an indirect call to the block handle of a selected SimMechanics block in your machine.

The data structure returned by `mech_stateVectorMgr` exposes the structure of the machine's mechanical state vector.

Input Arguments

`mech_stateVectorMgr` accepts one input.

pathname

A SimMechanics block's full path name or handle, or an indirect call to the path name or handle.

- Specify the full path name starting with the model name and continuing through any subsystem hierarchy: `pathname = modelname/subsystem1/etc.../blockname`.

  You can obtain the path name or handle of any block by selecting that block in a window and entering `gcb` or `gcbh` at the command line.

- Combine the previous steps into one with an indirect path name or handle call. Select a SimMechanics block in the model window and enter the function passing `gcb` or `gcbh` as the input.

Output Arguments

The output of `mech_stateVectorMgr` is an object of the `MECH.StateVectorMgr` class associated with a particular machine.

The returned object has four properties, fixed by the machine structure and naming.

MachineState.MachineName

A string specifying the path to the root Ground block: `'modelname/subsystem1/etc.../rootgroundblock'`.

MachineState.X

A 1-by-*N* real array with value `[ 0 0 ...  0 ]`. The length of the vector indicates the number of state components.

You can change the values of this array's components, but these values do not propagate to the model.

MachineState.BlockStates

An array of *N* block state managers, containing the Joint primitive and Point-Curve Constraint states. These managers are structures, arranged in the array by class: `MECH.RPJointStateMgr`, `MECH.SJointStateMgr`, and `MECH.PointCurveStateMgr`.

The components of each manager contain:

- The joint block and prismatic, revolute, and spherical primitive names
- The position and velocity values and units
- The `FixedAtT_0` flag indicating if the primitive is actuated with initial conditions

MachineState.StateNames

A cell array of *N* strings, containing the names of joint primitives and Point-Curve Constraints. This structure's strings are grouped and ordered in the same way that the block state managers of `BlockStates` are.

Querying the Mechanical State Vector

Entering the `mech_stateVectorMgr` function or querying the entire object returns a summary of the object by property. Once you create a state vector manager object, you can query these properties individually by entering the full property name:

```
MachineState.MachineName
MachineState.X
```

The state vector manager object is a singleton. If you create the object in `A`, then reassigning `B = A` makes `A` and `B` point to each other, not independent copies. Changing `B` automatically changes `A` and vice versa.

Changing Values of Mechanical States

This data structure does not contain actual state values. You can subsequently assign values to the states in this object, but these values do not propagate to your model. That requires changing the mechanical part of the model's Simulink state vector. Consult `mech_get_states`, `mech_runtime_states`, `mech_set_states`, and `mech_transfer_states` about changing a model's mechanical state vector values.

Definitions

Machines

A machine is a connected set of SimMechanics blocks.

Mechanical State Vector and Simulink State Vector

The data structure returned by `mech_stateVectorMgr` includes only the states of a machine made of SimMechanics blocks.

SimMechanics names the mechanical states in a model. Simulink recognizes and uses these SimMechanics state names.

Examples

Mechanical State Vector with One Primitive

Open the example model `mech_spen`, select a SimMechanics block, and enter:

```
machinestate = mech_stateVectorMgr(gcb)
```

at the command line. The function returns

```
machinestate =
    MECH.StateVectorMgr
    MachineName: 'mech_spen/Ground'
              X: [0 0]
    BlockStates: [1x1 MECH.RPJointStateMgr]
     StateNames: {2x1 cell}
```

The first line in the object is the class and the last four are the properties. The model `mech_spen` contains one Joint block (a Revolute), with two states (angle and angular velocity).

Query individual properties. For example, enter `machinestate.machinename`. The structure returns

```
mech_spen/Ground
```

referring to the one Ground block in the model.

Enter `machinestate.X`. The structure returns a two-component state vector ($N = 2$, position and velocity):

```
0      0
```

Enter `machinestate.blockstates`. The structure returns

```
MECH.RPJointStateMgr
        BlockName: 'Revolute'
        Primitive: 'R1'
         Position: 0
    PositionUnits: 'rad'
         Velocity: 0
    VelocityUnits: 'rad/s'
       FixedAtT_0: 'off'
```

There is one Joint (a Revolute) and no Point-Curve Constraints, so there is only one state manager of class `MECH.RPJointStateMgr`. This property gives detailed Joint information: block name, primitive name, position and velocity values and units, and the absence of initial condition actuators.

Enter `machinestate.statenames`. The structure returns the names of the Joint block, the joint primitive, and the states:

```
'Revolute:R1:Position'
'Revolute:R1:Velocity'
```

Mechanical State Vector with Multiple Primitives

Construct an unnamed model with Ground and Body blocks connected by a Telescoping Joint. Then select one of the blocks and enter `machinestate = mech_stateVectorMgr(gcb)` at the command line. Simulink returns

```
machinestate =
    MECH.StateVectorMgr
    MachineName: 'untitled/Ground'
              X: [0 0 0 0 0 0 0 0 0 0]
    BlockStates: [2x1 MECH.BlockStateMgr]
     StateNames: {10x1 cell}
```

The unnamed model is associated with its Ground block and has a spherical and a prismatic primitive, hence 10 components in the state vector. To see those primitive names, enter `machinestate.statenames` to obtain

```
'Telescoping:S:Quaternion:1'
'Telescoping:S:Quaternion:2'
'Telescoping:S:Quaternion:3'
'Telescoping:S:Quaternion:4'
'Telescoping:P1:Position'
```

```
'Telescoping:S:Quaternion_dot:1'
'Telescoping:S:Quaternion_dot:2'
'Telescoping:S:Quaternion_dot:3'
'Telescoping:S:Quaternion_dot:4'
'Telescoping:P1:Velocity'
```

The quaternion and the prismatic position make the first five components, while the quaternion derivative and prismatic velocity make the last five.

Algorithms

Counting Machine State Vector Components

Excluding any motion-actuated joint primitives, the total number of mechanical state components is

N = 2*(# of uncut prismatics + # of uncut revolutes) + 8*(# of uncut sphericals)
+ (# of Point-Curve Constraints)

The machine state consists of translational and angular positions and velocities of these degrees of freedom (DoFs) in the machine:

- If uncut, prismatic and revolute joint primitives each contribute two state components, a position and a velocity.
- If uncut, spherical joint primitives each contribute eight state components, a quaternion and a quaternion derivative.
- A joint primitive actuated by a Joint Initial Condition Actuator (JICA) is counted like other joint primitives. Because the JICA externally specifies such a primitive's initial position and velocity, the primitive has an active `FixedAtT_0` flag.
- Point-Curve Constraints each contribute one predictor state component, the arc parameter velocity of the point along the curve.

Some Joint Primitives Not Counted for Mechanical States

- The joint primitives of a cut Joint in a closed loop are not counted in the machine state.

- If the joint primitive is motion actuated with a Joint Actuator block, that joint primitive is not counted in machine state.
- The weld primitive contributes no machine state components.

Distinguishing Independent States and Degrees of Freedom

Not all the machine states have to represent independent DoFs.

Closed Loops and Dependent States

The number of independent DoFs is reduced by:

- Constraint and Driver blocks, which must occur in closed loops
- Additional constraints arising from cut joints in closed loops

In such cases, some of the machine states are dependent.

Open Topology, Constraint-Free Machines

If the machine contains no closed loops, and thus no cuts, Constraints, or Drivers, the machine states and independent DoFs are identical.

See Also

`gcb` | `gcbh` | `gcs` | `mech_get_states` | `mech_runtime_states` | `mech_set_states` | `mech_transfer_states` | Point-Curve Constraint | Prismatic | Revolute | Spherical

Tutorials

- “Trimming Mechanical Models”
- “Linearizing Mechanical Models”
- “About the Stewart Platform”
- “Modeling the Stewart Platform”

How To

- “Representing Motion”
- “Cutting Machine Diagram Loops”
- “Counting Model Degrees of Freedom”

mech_transfer_states

Purpose

Map states from one machine's state vector manager to another

Syntax

```
dststates = mech_transfer_states(srcMgr, dstMgr,
   transferstruct)
dststates = mech_transfer_states(srcMgr, dstMgr,
   transferstruct, srcstates)
dststates = mech_transfer_states(srcMgr, dstMgr,
   transferstruct, srcstates, dststates)
```

Description

`dststates = mech_transfer_states(srcMgr, dstMgr, transferstruct)` copies selected state entries from the source state vector manager `srcMgr` to the state vector `dststates`, which can be assigned to `dstMgr.X`, where `dstMgr` is the state vector manager for the destination machine. The states to be copied are specified in `transferstruct`. Any destination states not specified in `transferstruct` retain their values as specified initially in `dstMgr`.

`dststates = mech_transfer_states(srcMgr, dstMgr, transferstruct, srcstates)` performs the same transfer, but takes the "source" states in `srcstates` rather than the values in `srcMgr.X`.

`dststates = mech_transfer_states(srcMgr, dstMgr, transferstruct, srcstates, dststates)` performs the same transfer, but also takes the "destination" states in `dststates` rather than the values in `dstMgr.X` for those destination states that are not the target of an assignment.

Input Arguments

`mech_transfer_states` accepts three, four, or five inputs.

srcMgr

A state vector manager for the source system. It is an object of the `MECH.StateVectorMgr` class as returned by the `mech_stateVectorMgr` function. This input is required.

dstMgr

A state vector manager for the destination system. It is an object of the `MECH.StateVectorMgr` class as returned by the `mech_stateVectorMgr` function. This input is required.

transferstruct

A structured array whose `src` and `dst` fields contain either the indices or the names of the source and destination states as known to their respective state managers. This input is required.

Either or both of the `dst` and `src` fields in `transferstruct` can also be indices for their respective vector manager's `X` vectors.

srcstates

A source mechanical state vector. This input is optional.

dststates

A destination mechanical state vector. This input is optional and requires the optional fourth input.

Output Arguments

dststates

The destination mechanical state vector.

Examples

Copying Mechanical States

Copy the first state in `srcMgr` to the third state in `dstMgr`:

```
transferstruct(1).src=srcMgr.StateNames{1};
transferstruct(1).dst=dstMgr.StateNames{3};
dstMgr.X = mech_transfer_states(srcMgr, dstMgr, transferstruct);
```

Using `transferstruct` Fields as Mechanical State Vector Manager Indices

Define the same `transferstruct` structure as in the preceding example.

If both of the `dst` and `src` fields in `transferstruct` are indices for their respective vector manager's `X` vectors, this function is equivalent to:

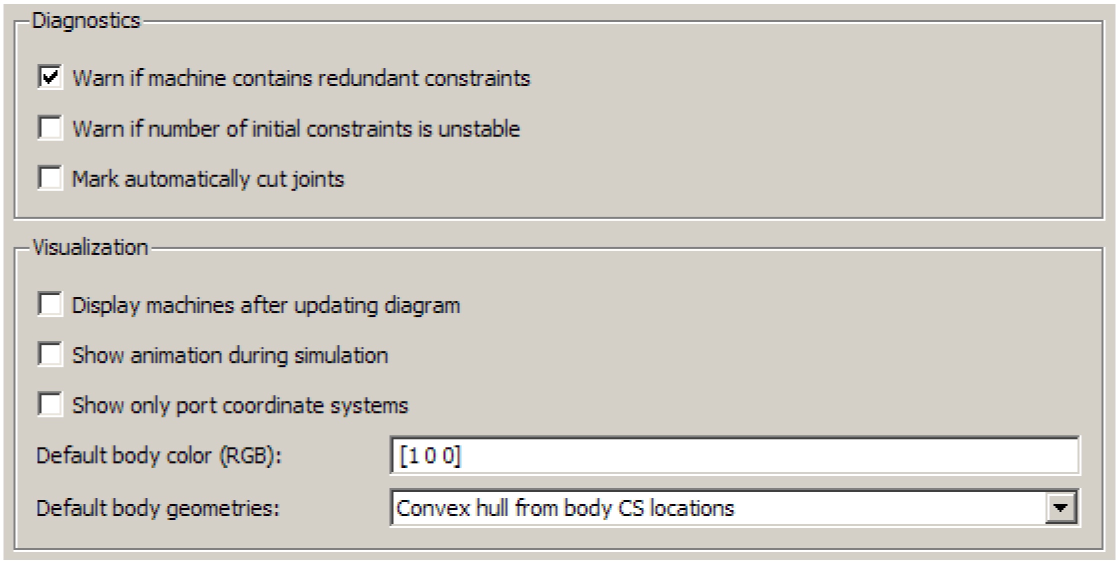
```
dststates(transferstruct.dst) = srcstates(transferstruct.src)
```

See Also

`mech_get_states` | `mech_runtime_states` | `mech_set_states` | `mech_stateVectorMgr` | `sim` | `states`

Configuration Parameters

SimMechanics Pane: General


Diagnostics

[✓] Warn if machine contains redundant constraints

[ ] Warn if number of initial constraints is unstable

[ ] Mark automatically cut joints

Visualization

[ ] Display machines after updating diagram

[ ] Show animation during simulation

[ ] Show only port coordinate systems

Default body color (RGB): [1 0 0]

Default body geometries: Convex hull from body CS locations

In this section...

SimMechanics Pane Overview

Configure the mechanical settings for an entire SimMechanics 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 SimMechanics product.
- The settings in this pane are saved only if your model contains at least one SimMechanics block.

See Also

- Configuring SimMechanics Models in Simulink
- Configuring Methods of Solution
- Starting Visualization and Simulation

Warn if machine contains redundant constraints

Require SimMechanics software to warn you if your model has more constraints than necessary.

Settings

Default: on

☑ On

SimMechanics software warns you if your model has too many constraints.

☐ Off

SimMechanics software does not warn you if your model has too many constraints.

Tip

Use this option to check if your model has too many and possibly conflicting constraints. Inconsistent constraints cause the model simulation to stop with an error.

Command-Line Information

Parameter: `WarnOnRedundantConstraints`
Type: string
Value: `'on'` | `'off'`
Default: `'on'`

Recommended Settings

Application	Setting
Debugging	On
Traceability	On
Efficiency	No impact
Safety precaution	On

See Also

Configuring Methods of Solution: Avoiding Simulation Failures

Warn if number of initial constraints is unstable

Require SimMechanics software to warn you if small changes to your model's initial state lead to changes in the number of constraints.

Settings

Default: off

☑ On

SimMechanics software warns you if your model's constraints are unstable against small changes to its initial state.

☐ Off

SimMechanics software does not warn you if your model's constraints are unstable against small changes to its initial state.

Tip

Use this option to check if your model has:

- Too many and possibly conflicting constraints
- Too few constraints, leaving the model underdetermined

Either condition prevents SimMechanics software from successfully simulating your model.

Command-Line Information

Parameter: `WarnOnSingularInitialAssembly`
Type: string
Value: `'on' | 'off'`
Default: `'off'`

Recommended Settings

Application	Setting
Debugging	On
Traceability	On

Application	Setting
Efficiency	No impact
Safety precaution	On

See Also

Configuring Methods of Solution: Avoiding Simulation Failures

Mark automatically cut joints

Require Simulink to mark any Joint blocks in closed loops that it cuts during simulation.

Settings

Default: off

☑ On
: Simulink marks the icons of any Joint blocks that it cuts during simulation.

☐ Off
: Simulink does not mark the icons of any Joint blocks, whether the blocks are cut during simulation or not.

Tip

SimMechanics software chooses the Joint to cut in each closed loop unless you enter your preference in one of the Joint dialogs. To do so, go to the **Axes** tab of that Joint dialog's **Parameters** area and select the **Mark as the preferred cut joint** check box.

Dependency

This parameter applies only if your SimMechanics model contains at least one closed loop.

Command-Line Information

Parameter: `ShowCutJoints`
Type: string
Value: `'on' | 'off'`
Default: `'off'`

Recommended Settings

Application	Setting
Debugging	On
Traceability	On
Efficiency	No impact
Safety precaution	No impact

See Also

- Configuring Methods of Solution: Maintaining Constraints
- Machine Environment Block: Implementing Constraints

Display machines after updating diagram

Enable a model-wide static view in the SimMechanics visualization window.

Settings

Default: off

☑ On

Enable a static display of all machines in your model that also have machine-specific visualization enabled. The view refreshes when you select **Update Diagram** in the model's **Edit** menu.

☐ Off

Disable static display for all machines in your model.

Tips

- Enable or disable, and configure, visualization for each machine in your model in the **Visualization** pane of the machine's Machine Environment block.
- All other visualization controls appear in the individual Body block dialogs or on the visualization window itself.

Dependency

Visualization is enabled for each machine in your model by the **Visualize machine** check box of the machine's Machine Environment block.

Command-Line Information

Parameter: `VisOnUpdateDiagram`
Type: string
Value: `'on' | 'off'`
Default: `'off'`

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	Off

See Also

Starting Visualization and Simulation: Setting Up Visualization

Show animation during simulation

Enable model-wide animation in the SimMechanics visualization window.

Settings

Default: off

☑ On

Enables model-wide animation in the visualization window for all machines in your model with machine-specific visualization also enabled. The animation starts when you start model simulation.

☐ Off

Disables model-wide animation in the visualization window for all machines in your model.

Tips

- Enable or disable, and configure, visualization for each machine in your model in the **Visualization** pane of the machine's Machine Environment block.
- All other visualization controls appear in the individual Body block dialogs or on the SimMechanics visualization window itself.

Dependency

Visualization is enabled for each machine in your model by the **Visualize machine** check box of the machine's Machine Environment block.

Command-Line Information

Parameter: `VisDuringSimulation`
Type: string
Value: `'on' | 'off'`
Default: `'off'`

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	Off
Safety precaution	Off

See Also

Starting Visualization and Simulation: Setting Up Visualization

Show only port coordinate systems

Suppress display of Body coordinate systems on bodies in the SimMechanics visualization window, unless the CS ports associated with the Body coordinate systems are also visible on the Body blocks in the model.

Settings

Default: off

☑ On

Suppresses display of Body coordinate systems on bodies in the visualization window, if the associated CS ports are not visible on the Body blocks in the model.

☐ Off

Enables display of all Body coordinate systems on bodies in the visualization window, whether or not the associated CS ports are visible on the Body blocks in the model.

Tips

- Enable or disable, and configure, visualization for each machine in your model in the **Visualization** pane of the machine's Machine Environment block.
- All other visualization controls appear in the individual Body block dialogs or on the visualization window itself.

Dependency

Visualization is enabled for each machine in your model by the **Visualize machine** check box of the machine's Machine Environment block.

Command-Line Information

Parameter: `ShowOnlyPortCS`
Type: string
Value: `'on' | 'off'`
Default: `'off'`

Recommended Settings

Application	Setting
Debugging	Off
Traceability	Off
Efficiency	On
Safety precaution	No impact

See Also

Starting Visualization and Simulation: Setting Up Visualization

Default body color (RGB)

Set default color of all bodies in a model visualized in the SimMechanics visualization window.

Settings

Specify the color value with the MATLAB Graphics `ColorSpec [r g b]`, where `r`, `g`, and `b` specify red, green, and blue color components.

Default: `[1 0 0]` (red)

Tips

- Configure visualization for each machine in your model in the **Visualization** pane of the machine's Machine Environment block.
- All other visualization controls appear in the individual Body block dialogs or on the visualization window itself.

Dependencies

Visualization is enabled for each machine in your model by the **Visualize machine** check box of the machine's Machine Environment block.

Command-Line Information

Parameter: `DefaultBodyColor`
Type: string
Value: `'[r g b]'`, where `r`, `g`, and `b` are real numbers between 0 and 1, inclusive
Default: `'[1 0 0]'`

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	No impact

See Also

Starting Visualization and Simulation: Setting Up Visualization

Default body geometries

Set default body geometry of all bodies visualized in the SimMechanics visualization window.

Settings

Specify the geometry by one of these choices:

- Convex hull from body coordinate system (CS) locations
- Equivalent ellipsoid from body mass properties

Default: `Convex hull from body CS locations`

Tips

- Configure visualization for each machine in your model in the **Visualization** pane of the machine's Machine Environment block.
- All other visualization controls appear in the individual Body block dialogs or on the visualization window itself.

Dependency

Visualization is enabled for each machine in your model by the **Visualize machine** check box of the machine's Machine Environment block.

Command-Line Information

Parameter: `MDLBodyVisualizationType`
Type: string
Value: `'Convex hull from body CS locations' | 'Equivalent ellipsoid from mass properties'`
Default: `'Convex hull from body CS locations'`

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

Starting Visualization and Simulation: Setting Up Visualization

Index