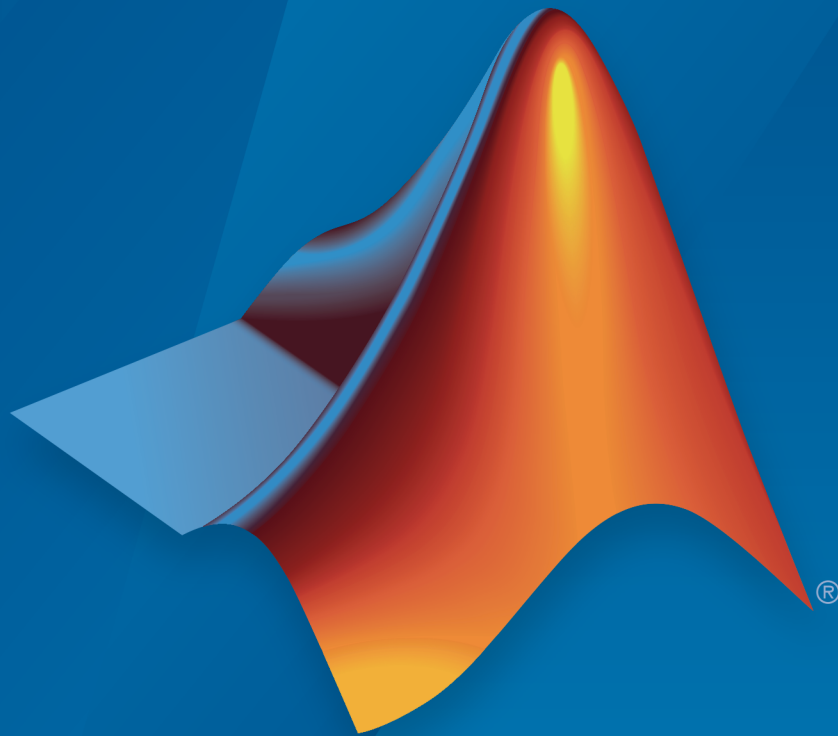


**Simscape™ Multibody™**

User's Guide



**MATLAB® & SIMULINK®**

R2016b



## How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



Phone: 508-647-7000



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

*Simscape™ Multibody™ User's Guide*

© COPYRIGHT 2002–2016 by The MathWorks, Inc.

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

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

### Trademarks

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

### Patents

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

### Revision History

March 2012	Online only	New for Version 4.0 (Release R2012a)
September 2012	Online only	Revised for Version 4.1 (Release R2012b)
March 2013	Online only	Revised for Version 4.2 (Release R2013a)
September 2013	Online only	Revised for Version 4.3 (Release R2013b)
March 2014	Online only	Revised for Version 4.4 (Release R2014a)
October 2014	Online only	Revised for Version 4.5 (Release R2014b)
March 2015	Online only	Revised for Version 4.6 (Release R2015a)
September 2015	Online only	Revised for Version 4.7 (Release R2015b)
March 2016	Online only	Revised for Version 4.8 (Release R2016a) (Renamed from <i>SimMechanics™ User's Guide</i> )
September 2016	Online only	Revised for Version 4.9 (Release R2016b)



## Multibody Modeling

### Spatial Relationships

1

<b>Frames and Frame Transforms</b> .....	<b>1-2</b>
What Are Frames? .....	1-2
Role of Frames .....	1-2
Working with Frames .....	1-3
What Are Frame Transforms? .....	1-4
Working with Frame Transforms .....	1-4
Visualizing Frame Transforms .....	1-5
Sensing Frame Transforms .....	1-6
Specify a Frame Transform .....	1-6
<b>Representing Frames</b> .....	<b>1-11</b>
Identity Relationships .....	1-12
Translation and Rotation .....	1-13
Interpreting a Frame Network .....	1-13
<b>World and Reference Frames</b> .....	<b>1-16</b>
World Frame .....	1-16
Reference Frame .....	1-19
<b>Common Issues with Frame Connections</b> .....	<b>1-22</b>
Rigidity Loops .....	1-22
Shorted Rigid Transforms .....	1-23

<b>Creating Body Subsystems</b> .....	2-2
Study the Bodies to Model .....	2-2
Model the Solids in Each Body .....	2-2
Connect the Solids Through Frames .....	2-3
Verify Body Subsystems .....	2-3
<b>Simscape Multibody Bodies</b> .....	2-4
Rigid Body Overview .....	2-4
Rigid Body Properties .....	2-5
Rigid Body Frames .....	2-6
Rigid Body Delimitation .....	2-7
Simple and Compound Rigid Bodies .....	2-9
<b>Solid Geometry</b> .....	2-10
Geometry Overview .....	2-10
Compound Shapes .....	2-10
Basic Shapes .....	2-11
General Extrusion and Revolution Shapes .....	2-12
Imported Shapes .....	2-13
<b>Solid and Multibody Visualization</b> .....	2-14
<b>Revolution and General Extrusion Shapes</b> .....	2-16
Shape Cross Sections .....	2-16
Coordinate Matrices .....	2-16
Hollow Cross Sections .....	2-17
3-D Shape Generation .....	2-18
<b>Solid Inertia</b> .....	2-20
Inertial Properties .....	2-20
Blocks with Inertia .....	2-20
Inertia in a Model .....	2-21
Inertia Parameterizations .....	2-22
Custom Inertia .....	2-22
Moments and Products of Inertia .....	2-23
Complex Inertias .....	2-25
<b>Specify Custom Inertia</b> .....	2-26
Custom Inertia .....	2-26

Model Overview . . . . .	2-26
Inertia Parameters . . . . .	2-27
Build Model . . . . .	2-29
Specify Inertia . . . . .	2-29
Add Motion Sensing . . . . .	2-31
Run Simulation . . . . .	2-32
<b>Add Frames to Solids . . . . .</b>	<b>2-34</b>
Solid Frames . . . . .	2-34
Frame-Creation Interface . . . . .	2-35
Model Solid Shape . . . . .	2-36
Create New Frame . . . . .	2-37
Specify Frame Origin . . . . .	2-38
Specify Primary Axis . . . . .	2-40
Specify Secondary Axis . . . . .	2-41
Save New Frame . . . . .	2-42
<b>Solid Visual Properties . . . . .</b>	<b>2-44</b>
Visual-Property Parameterizations . . . . .	2-44
RGB and RGBA Vectors . . . . .	2-45
Simple Visual Properties . . . . .	2-46
Advanced Visual Properties . . . . .	2-46
Adjust Solid Opacity . . . . .	2-47
Adjust Highlight Color . . . . .	2-48
Adjust Shadow Color . . . . .	2-48
Adjust Self-Illumination Color . . . . .	2-49
<b>Model a Solid of Revolution . . . . .</b>	<b>2-51</b>
Tutorial Overview . . . . .	2-51
Obtain the Cross-Section Coordinates . . . . .	2-51
Create the Dome Model . . . . .	2-52
Specify the Dome Properties . . . . .	2-53
Visualize the Dome Geometry . . . . .	2-54
<b>Model an Extruded Solid . . . . .</b>	<b>2-57</b>
Model Overview . . . . .	2-57
Modeling Approach . . . . .	2-57
Build Solid Model . . . . .	2-58
Define Solid Properties . . . . .	2-59
Visualize Solid Model . . . . .	2-60
<b>Model a Compound Body . . . . .</b>	<b>2-63</b>
Model Overview . . . . .	2-63

Build Model . . . . .	2-63
Generate Subsystem . . . . .	2-65
Visualize Model . . . . .	2-67
Open Reference Model . . . . .	2-67

## Multibody Systems

### 3

<b>Assembling Multibody Systems . . . . .</b>	<b>3-2</b>
Step 1: Study the Joints and Constraints to Model . . . . .	3-2
Step 2: Assemble Bodies Using Joints and Constraints . . . . .	3-2
Step 3: Guide Model Assembly . . . . .	3-3
Step 4: Verify Model Assembly . . . . .	3-3
<b>Joint Connections . . . . .</b>	<b>3-4</b>
Role of Joints in a Model . . . . .	3-4
Joint Degrees of Freedom . . . . .	3-4
Joint Primitives . . . . .	3-6
Joint Inertia . . . . .	3-8
<b>Multibody Assembly . . . . .</b>	<b>3-9</b>
Model Assembly . . . . .	3-9
Connecting Joints . . . . .	3-9
Orienting Joints . . . . .	3-10
Guiding Assembly . . . . .	3-11
Verifying Model Assembly . . . . .	3-11
<b>Mechanism Degrees of Freedom . . . . .</b>	<b>3-15</b>
<b>Model an Open-Loop Kinematic Chain . . . . .</b>	<b>3-16</b>
Model Overview . . . . .	3-16
Build Model . . . . .	3-17
Guide Model Assembly . . . . .	3-18
Visualize Model and Check Assembly Status . . . . .	3-18
Simulate Model . . . . .	3-20
Open Reference Model . . . . .	3-20
<b>Model a Closed-Loop Kinematic Chain . . . . .</b>	<b>3-21</b>
Model Overview . . . . .	3-21
Build Model . . . . .	3-22



Specify Block Parameters . . . . .	3-24
Guide Assembly and Visualize Model . . . . .	3-24
Verify Model Assembly . . . . .	3-25
Simulate Model . . . . .	3-26
Open Reference Model . . . . .	3-26
<b>Troubleshoot Assembly Errors in Aiming Mechanism</b>	
<b>Model</b> . . . . .	3-27
Model Overview . . . . .	3-27
Explore Model . . . . .	3-28
Update Model . . . . .	3-31
Troubleshoot Assembly Error . . . . .	3-31
Correct Assembly Error . . . . .	3-33
Simulate Model . . . . .	3-34
<b>Gear Constraints</b> . . . . .	3-36
Gear Types . . . . .	3-36
Featured Examples . . . . .	3-37
Inertia, Geometry, and Efficiency . . . . .	3-37
Using Gear Blocks . . . . .	3-37
Assembling Rigid Bodies with Gear Constraints . . . . .	3-39
Common Gear Assembly and Simulation . . . . .	3-40
Rack and Pinion Assembly and Simulation . . . . .	3-42
<b>Model a Rack-and-Pinion Constraint</b> . . . . .	3-45
Model Overview . . . . .	3-45
Model Pinion . . . . .	3-47
Model Rack . . . . .	3-49
Add Rack and Pinion Constraint . . . . .	3-52
Actuate Model . . . . .	3-53
Simulate Model . . . . .	3-55
Open Complete Model . . . . .	3-56
<b>Model a Common Gear Constraint</b> . . . . .	3-57
Model Overview . . . . .	3-57
Model Sun-Planet Gear Set . . . . .	3-58
Constrain Sun-Planet Gear Motion . . . . .	3-61
Add Ring Gear . . . . .	3-63
Add Gear Carrier . . . . .	3-66
Add More Planet Gears . . . . .	3-71
<b>Model a Cam Constraint</b> . . . . .	3-73
Model Overview . . . . .	3-73

Geometry Ports .....	3-74
Spline Curves .....	3-76
Point On Curve Constraints .....	3-77
Model Eccentric Cam .....	3-77
Model Cam Follower .....	3-83
Interactively Create Frame at Follower Tip .....	3-85
Constrain Cam and Follower .....	3-89

## Internal Mechanics, Actuation and Sensing

# 4

<b>Modeling and Sensing System Dynamics</b> .....	4-2
Provide Joint Actuation Inputs .....	4-2
Specify Joint Internal Mechanics .....	4-2
Model Body Interactions and External Loads .....	4-3
Sense Dynamical Variables .....	4-3
 <b>Modeling Gravity</b> .....	4-4
Gravity Models .....	4-4
Gravitational Force Magnitude .....	4-5
Force Position and Direction .....	4-6
Gravitational Torques .....	4-7
 <b>Model Gravity in a Planetary System</b> .....	4-9
Model Overview .....	4-9
Step 1: Start a New Model .....	4-10
Step 2: Add the Solar System Bodies .....	4-11
Step 3: Add the Degrees of Freedom .....	4-15
Step 4: Add the Initial State Targets .....	4-17
Step 5: Add the Gravitational Fields .....	4-21
Step 6: Configure and Run the Simulation .....	4-23
Open an Example Model .....	4-24
 <b>Joint Actuation</b> .....	4-25
Actuation Modes .....	4-25
Motion Input .....	4-28
Input Handling .....	4-30
Assembly and Simulation .....	4-31
Specifying Motion Input Derivatives .....	4-32

<b>Joint Actuation Limitations</b> .....	<b>4-34</b>
Closed Loop Restriction .....	4-34
Motion Actuation Not Available in Spherical Primitives .....	4-34
Redundant Actuation Mode Not Supported .....	4-34
Model Report and Mechanics Explorer Restrictions .....	4-35
Motion-Controlled DOF Restriction .....	4-35
<b>Actuating and Sensing with Physical Signals</b> .....	<b>4-36</b>
Exposing Physical Signal Ports .....	4-36
Converting Actuation Inputs .....	4-36
Obtaining Sensing Signals .....	4-37
<b>Sensing</b> .....	<b>4-39</b>
Sensing Overview .....	4-39
Variables You Can Sense .....	4-40
Blocks with Sensing Capability .....	4-40
Sensing Output Format .....	4-40
<b>Force and Torque Sensing</b> .....	<b>4-42</b>
Blocks with Force and Torque Sensing .....	4-42
Joint Forces and Torques You can Sense .....	4-43
Force and Torque Measurement Direction .....	4-45
<b>Motion Sensing</b> .....	<b>4-46</b>
Sensing Spatial Relationships Between Joint Frames .....	4-46
Sensing Spatial Relationships Between Arbitrary Frames .....	4-48
<b>Rotational Measurements</b> .....	<b>4-51</b>
Rotation Sensing Overview .....	4-51
Measuring Rotation .....	4-51
Axis-Angle Measurements .....	4-51
Quaternion Measurements .....	4-52
Transform Measurements .....	4-54
<b>Translational Measurements</b> .....	<b>4-56</b>
Translation Sensing Overview .....	4-56
Measuring Translation .....	4-56
Cartesian Measurements .....	4-57
Cylindrical Measurements .....	4-59
Spherical Measurements .....	4-61

<b>Measurement Frames</b> .....	<b>4-64</b>
Measurement Frame Purpose .....	4-64
Measurement Frame Types .....	4-65
<b>Sense Motion</b> .....	<b>4-67</b>
Model Overview .....	4-67
Modeling Approach .....	4-68
Build Model .....	4-68
Guide Model Assembly .....	4-70
Simulate Model .....	4-70
Save Model .....	4-72
<b>Specify Joint Actuation Torque</b> .....	<b>4-73</b>
Model Overview .....	4-73
Four-Bar Linkages .....	4-74
Modeling Approach .....	4-76
Build Model .....	4-77
Simulate Model .....	4-80
<b>Analyze Motion at Various Parameter Values</b> .....	<b>4-85</b>
Model Overview .....	4-85
Build Model .....	4-85
Specify Block Parameters .....	4-87
Create Simulation Script .....	4-88
Run Simulation Script .....	4-89
<b>Sense Forces and Torques Acting at Joints</b> .....	<b>4-91</b>
Overview .....	4-91
Open Model .....	4-92
Sense Actuation Torque .....	4-92
Sense Constraint Forces .....	4-95
Sense Total Forces .....	4-97
<b>Sense Constraint Forces</b> .....	<b>4-99</b>
Model Overview .....	4-99
Add Weld Joint Block to Model .....	4-100
Add Constraint Force Sensing .....	4-101
Add Damping to Joints .....	4-102
Simulate Model .....	4-102
Plot Constraint Forces .....	4-103
<b>Specify Joint Motion Profile</b> .....	<b>4-105</b>
Model Overview .....	4-105

Build Model . . . . .	4-105
Simulate Model . . . . .	4-108
<b>Prescribe Joint Motion in Planar Manipulator Model</b>	<b>4-110</b>
Model Overview . . . . .	4-110
Add Virtual Joint . . . . .	4-111
Prescribe Motion Inputs . . . . .	4-112
Sense Joint Actuation Torques . . . . .	4-116
Simulate Model . . . . .	4-117

## Simulation and Analysis

### Simulation

## 5

<b>Update and Simulate a Model</b> . . . . .	<b>5-2</b>
Create or Open a Model . . . . .	5-2
Update the Block Diagram . . . . .	5-2
Examine the Model Assembly . . . . .	5-3
Configure the Solver Settings . . . . .	5-3
Run Simulation and Analyze Results . . . . .	5-4
<b>Troubleshoot Simulation Issues</b> . . . . .	<b>5-5</b>
Models with For Each Subsystem Blocks Have Limited Visualization . . . . .	5-5
Models with Model Blocks Have No Visualization . . . . .	5-5
Simscape Local Solvers Do Not Work with Simscape Multibody . . . . .	5-5

### Visualization and Animation

## 6

<b>Enable Mechanics Explorer</b> . . . . .	<b>6-2</b>
--	------------

<b>Model Animation</b> .....	<b>6-3</b>
Animation Playback .....	6-3
Looping Playback .....	6-3
Changing Playback Speed .....	6-3
Jumping to Playback Time .....	6-4
<b>Manipulate the Visualization Viewpoint</b> .....	<b>6-5</b>
Model Visualization .....	6-5
Select a Standard View .....	6-5
Set View Convention .....	6-6
Rotate, Roll, Pan, and Zoom .....	6-7
Split Model View .....	6-8
<b>Visualization Cameras</b> .....	<b>6-10</b>
Camera Types .....	6-10
Global Camera .....	6-11
Dynamic Cameras .....	6-11
Camera Trajectory Modes .....	6-12
Dynamic Camera Selection .....	6-13
Dynamic Camera Reuse .....	6-14
<b>Create a Dynamic Camera</b> .....	<b>6-15</b>
Start a New Camera Definition .....	6-15
Define a Keyframes Camera .....	6-16
Define a Tracking Camera .....	6-17
Select a Dynamic Camera .....	6-17
<b>Selective Model Visualization</b> .....	<b>6-19</b>
What Is Visualization Filtering? .....	6-19
Changing Component Visibility .....	6-20
Visualization Filtering Options .....	6-21
Components You Can Filter .....	6-21
Model Hierarchy and Tree Nodes .....	6-22
Filtering Hierarchical Subsystems .....	6-22
Updating Models with Hidden Nodes .....	6-23
Alternative Ways to Enhance Visibility .....	6-24
<b>Selectively Show and Hide Model Components</b> .....	<b>6-25</b>
Visualization Filtering .....	6-25
Open Example Model .....	6-26
Update Example Model .....	6-26
Hide Half-Cylinder Subsystem .....	6-27
Show Solid in Hidden Subsystem .....	6-28

Show Only Piston Subsystem .....	6-29
Show Everything .....	6-30
<b>Visualize Simscape Multibody Frames .....</b>	<b>6-32</b>
What Are Frames? .....	6-32
Show All Frames .....	6-32
Highlight Specific Frames .....	6-33
Visualize Frames via Graphical Markers .....	6-35
<b>Go to a Block from Mechanics Explorer .....</b>	<b>6-37</b>
<b>Create a Model Animation Video .....</b>	<b>6-39</b>
UI and Command-Line Tools .....	6-39
Before Creating a Video .....	6-39
Create a Video Using Video Creator .....	6-39
Create a Video Using smwritevideo .....	6-40
<b>Multibody Visualization Issues .....</b>	<b>6-41</b>
Mechanics Explorer Not Opening .....	6-41
Model Oriented Sideways .....	6-41
Imported CAD Parts Missing .....	6-42
Imported Part Colors Missing .....	6-43

## CAD Import

### About CAD Import

# 7

<b>CAD Translation .....</b>	<b>7-2</b>
<b>Translated Model .....</b>	<b>7-3</b>
<b>Model Data File .....</b>	<b>7-7</b>
<b>CAD Model Export .....</b>	<b>7-8</b>
<b>CAD Export Errors .....</b>	<b>7-9</b>

<b>XML File Import</b> .....	<b>7-10</b>
<b>CAD Import Errors</b> .....	<b>7-11</b>
<b>Data File Update</b> .....	<b>7-12</b>
<b>Install Simscape Multibody Link Plug-In</b> .....	<b>7-13</b>
Before You Begin .....	<b>7-13</b>
Step 1: Get Installation Files .....	<b>7-13</b>
Step 2: Run Installation Function .....	<b>7-13</b>
Step 3: Register MATLAB as Automation Server .....	<b>7-14</b>
Step 4: Enable Simscape Multibody Link Plug-In .....	<b>7-14</b>
<b>Import a CAD Assembly Model</b> .....	<b>7-15</b>
Before You Begin .....	<b>7-15</b>
Example Files .....	<b>7-15</b>
Import Model .....	<b>7-15</b>
After Import .....	<b>7-16</b>
<b>Import a Robotic Arm Model</b> .....	<b>7-17</b>
Example Overview .....	<b>7-17</b>
Example Files .....	<b>7-17</b>
Import the Model .....	<b>7-17</b>
Visualize the Model .....	<b>7-18</b>
Build on the Model .....	<b>7-20</b>

## Deployment

## Code Generation

# 8

<b>Code Generation Applications</b> .....	<b>8-2</b>
Code Generation Overview .....	<b>8-2</b>
Simulation Acceleration .....	<b>8-2</b>
Model Deployment .....	<b>8-3</b>
<b>Code Generation Setup</b> .....	<b>8-4</b>
Before You Begin .....	<b>8-4</b>



Solver Selection .....	8-4
Target Selection .....	8-4
<b>Generate Code for a Multibody Model .....</b>	<b>8-5</b>
<b>Configure Model for Rapid Accelerator Mode .....</b>	<b>8-7</b>
Model Overview .....	8-7
Configure Model .....	8-8



# **Multibody Modeling**



# Spatial Relationships

---

- “Frames and Frame Transforms” on page 1-2
- “Representing Frames” on page 1-11
- “World and Reference Frames” on page 1-16
- “Common Issues with Frame Connections” on page 1-22

## Frames and Frame Transforms

### In this section...

“What Are Frames?” on page 1-2

“Role of Frames” on page 1-2

“Working with Frames” on page 1-3

“What Are Frame Transforms?” on page 1-4

“Working with Frame Transforms” on page 1-4

“Visualizing Frame Transforms” on page 1-5

“Sensing Frame Transforms” on page 1-6

“Specify a Frame Transform” on page 1-6

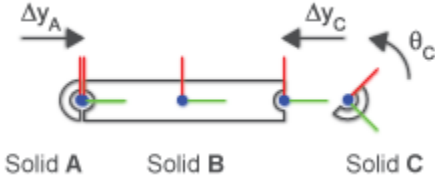
### What Are Frames?

Frames are axis triads that encode position and orientation data in a 3-D multibody model. Each triad consists of three perpendicular axes that intersect at an origin. The origin determines the frame position and the axes determine the frame orientation. The axes are color-coded, with the  $x$ -axis in red, the  $y$ -axis in green, and the  $z$ -axis in blue.



### Role of Frames

Every solid component has one or more local frames to which it is rigidly attached. By positioning and orienting the component frames, you position and orient the components themselves. This is the role of frames in a model—to enable you to specify the spatial relationships between components.

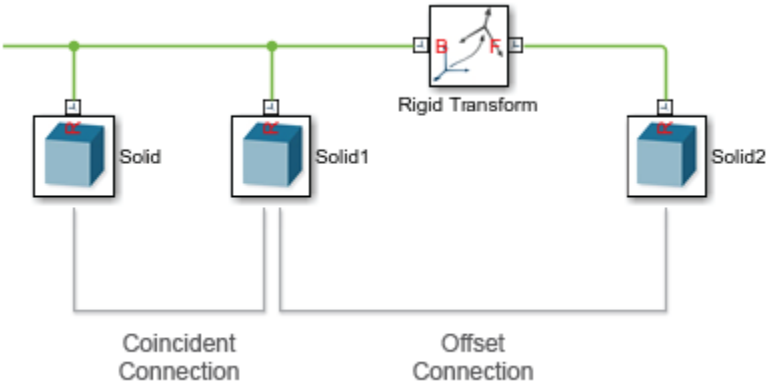


### Working with Frames

A frame port identifies a local frame on a component. For example, the R frame port of a Solid block identifies the local reference frame of a solid. Every block has one or more frame ports that you connect in order to locate the associated components in space. The figure shows the reference frame ports on Body Elements blocks.

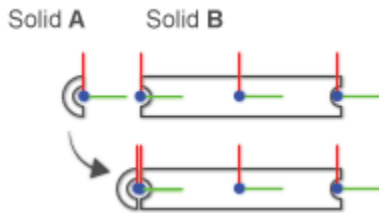


The connections between frame ports determine the spatial relationships between their frames. A direct frame connection line makes the connected frames coincident in space. A Rigid Transform block sets the rotational and translational offsets between the frames. The figure shows coincident and offset frame connections.



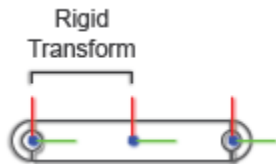
A coincident relationship between solid frames does not, by itself, constitute a coincident relationship between solid geometries. The spatial arrangement of two solid geometries depends not only on the spatial arrangement of the respective reference frames, but also on how the geometries are defined relative to those frames.

If two geometries differ from each other, or if their positions and orientations relative to their reference frames differ from each other, then making the reference frames coincident will cause the solid geometries to be offset. In the figure, connecting the frame of Solid A to the left frame of Solid B joins the solids such that their geometries are offset from each other.



## What Are Frame Transforms?

The rotational and translational offsets between frames are called transforms. If the transforms are constant through time, they are called rigid. Rigid transforms enable you to fix the relative positions and orientations of components in space, e.g., to assemble solids into rigid bodies.

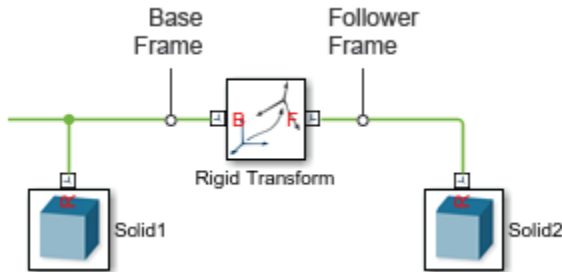


## Working with Frame Transforms

You use the Rigid Transform block to specify a rotational, translational, or mixed rigid transform between frames. The transforms are directional. They set the rotation and translation of a frame known as follower relative to a frame known as base.



The frame port labels on the Rigid Transform block identify the base and follower frames. The frame connected to port B serves as base. The frame connected to port F serves as follower. Reversing the port connections reverses the direction in which the frame transform is applied.



You can specify a transform using different methods. For rotational transforms, these include axis-angle pairs, rotation matrices, and rotation sequences. For translational transforms, they include translational offset vectors defined in Cartesian or cylindrical coordinate systems.

If the rotational and translational transforms are both zero, the connected frames are coincident in space. This relationship is known as *identity* and it is equivalent to a direct frame connection line between frame ports—i.e., one without a Rigid Transform block.

## Visualizing Frame Transforms

You can visualize frames and examine the transforms between frames using the Solid block visualization pane or Mechanics Explorer. Use the Solid block visualization pane to examine the frames of a single solid element. Click the **Toggle frame visibility** button in the visualization toolstrip to show all the solid frames.

Use Mechanics Explorer to visualize the frames of more than a single solid element—e.g., in compound bodies, multibody subsystems, or complete multibody models. Click the **Toggle frame visibility** button in the visualization toolstrip to show all frames. Select a node from the tree view pane to show only those frames belonging to the selected component.

## Sensing Frame Transforms

You can sense rotational and translational transforms between frames. To sense a transform between any two frames, you use the **Transform Sensor** block. To sense a transform between frames connected by a **Joint** block, you use the **Joint** block itself.

The **Transform Sensor** block provides the broadest selection of transform measurements. You can sense rotation transforms as angles about axes, rotation matrices, and quaternions. You can sense translation transforms as offsets in Cartesian, cylindrical, and spherical coordinate systems.

**Joint** blocks provide a limited selection of transform measurements. The transforms that you can sense depend closely on the degrees of freedom that the joint provides. For example, a **Prismatic Joint** block provides a single translational degree of freedom and therefore senses translational variables only.

Transforms sensed through **Transform Sensor** blocks are given for the follower frame relative to the base frame. The transform measurements are resolved in a choice of five measurement frames, including the World frame, rotating and non-rotating versions of the base frame, and rotating and non-rotating versions of the follower frame.

Transforms sensed through joint blocks are given for the follower frame relative to the base frame. However, unless the joint has a spherical primitive, the transform measurements are resolved in the base frame only. Joint blocks with spherical primitives add the option to resolve measurements in the follower frame.

## Specify a Frame Transform

This example shows how to move two solids relative to each other by specifying a frame transform between the solid reference frames. The transform consists of a **-45 deg** rotation about the  $z$  axis followed by a **1 m** translation along the  $x$ -axis and a **1 m** translation along the  $y$ -axis.

### Add the solids to the model

- 1 Drag two **Solid** blocks from the **Body Elements** library and place them in a new model.

Each **Solid** block specifies the default geometry of a cube **1 m** in width.

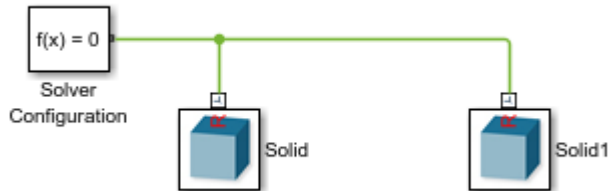
- 2 Connect the **Solid** block **R** frame ports.



The frame connection line makes the reference frames—and cubes—coincident in space.

### Visualize the solid frames

- 1 Drag a Solver Configuration block from the Simscape™ Foundation Utilities library and connect it anywhere on the model.



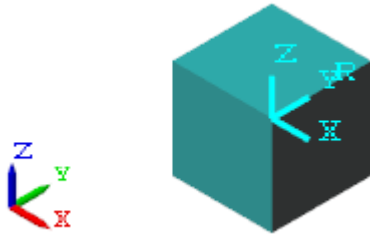
This block is required for model update and simulation.

- 2 Update the diagram, e.g., by selecting **Simulation > Update Diagram**.

Mechanics Explorer opens with a model visualization.



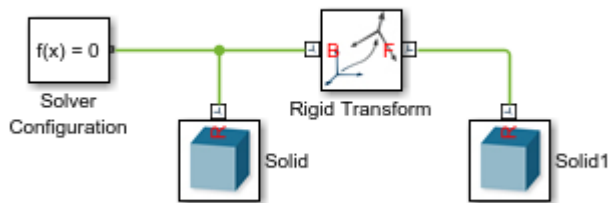
- 3 In the tree view pane, alternately click the **Solid** and **Solid1** nodes.



The visualization pane shows the solid reference frames. The frames are coincident in space.

### Apply the rotation transform

- 1 Drag a Rigid Transform block from the Frames and Transforms library and connect it between the two Solid blocks.

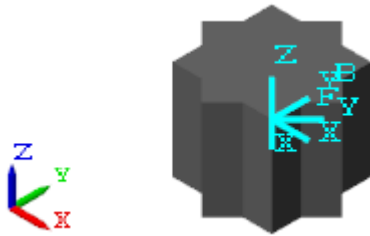


- 2 In the Rigid Transform block dialog box, set:
  - **Rotation** > **Method** to Standard Axis.
  - **Rotation** > **Axis** to -Z.
  - **Rotation** > **Angle** to 45.
- 3 Click **OK** and update the block diagram.



The model visualization updates to show the rotated—and still overlapping—solids.

- 4 In the tree view pane, click the **Rigid Transform** node.



The visualization pane shows the rotated frames.

### Apply the translation transform

- 1 In the Rigid Transform block dialog box, set:
  - **Translation > Method** to Cartesian.
  - **Translation > Offset** to [1 1 0].

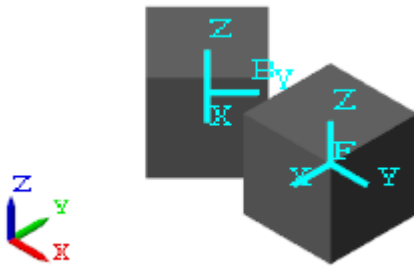
The array elements are the translation offsets along the base frame  $x$ ,  $y$ , and  $z$  axes.

- 2 Click **OK** and update the block diagram.



The model visualization updates to show the translated solids.

- 3 In the tree view pane, click the **Rigid Transform** node.



The visualization pane shows the translated frames.

The Rigid Transform block always applies the rotation transform first. The translation transform is relative to the rotated frame resulting from the rotation transform. To apply the translation transform first, use separate Rigid Transform blocks for each transform and connect them in the desired order between the Solid blocks.

## Representing Frames

### In this section...

“Identity Relationships” on page 1-12

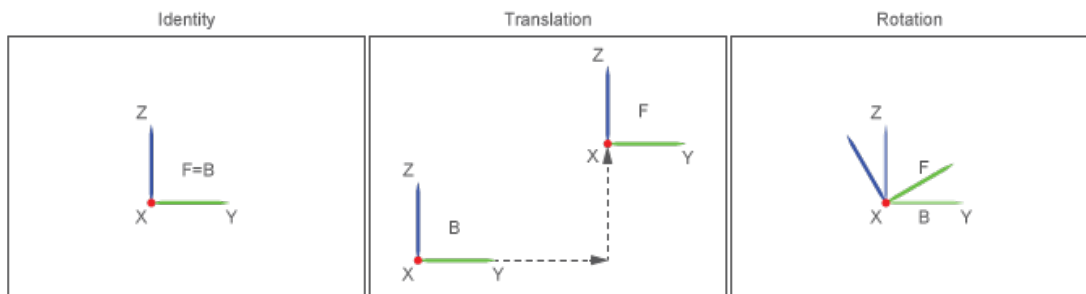
“Translation and Rotation” on page 1-13


“Interpreting a Frame Network” on page 1-13

You represent frames with frame ports, lines, and nodes. Each of these frame entities represents one frame. You connect one frame entity to any other using a connection line. When you do so, you apply a spatial relationship between the two frames. Spatial relationships that you can specify include:

- Identity — Make two frames coincident with each other.
- Translation — Maintain an offset distance between two frame origins.
- Rotation — Maintain an angle between two frames.

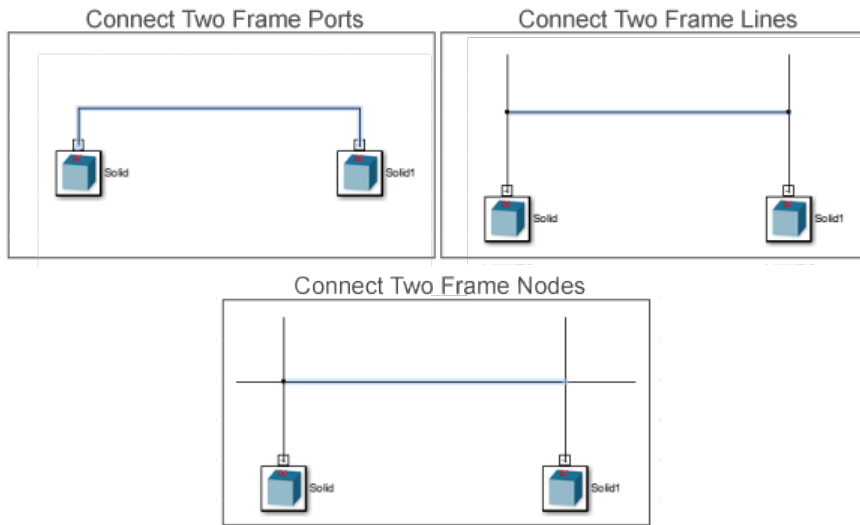
The figure illustrates these spatial relationships. Letters B and F represent the two frames between which you apply a spatial relationship.



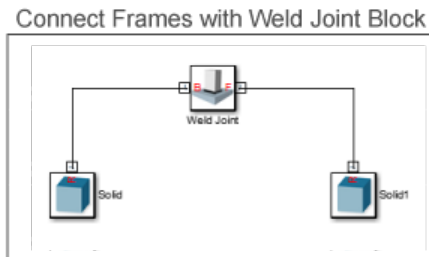
A frame port is any port with the frame icon . A frame line is any connection line that joins two frame ports. A frame node is the junction point between two or more frame lines. You can connect one frame entity only to another frame entity. Connecting frame ports, lines, or nodes to other types of ports, lines, or nodes is invalid. For example, you cannot connect a frame port to a physical signal port.

## Identity Relationships

To make two frames coincident in space, connect the corresponding frame entities with a frame line. The frame line applies a rigid identity relationship between the two frames. During simulation, the two frames can move only as a single unit. They cannot move with respect to each other. The figure shows three ways to make two frames coincident.



Alternatively, use the **Weld Joint** block to make two frames coincident for all time. The Weld Joint block fixes the relative positions and orientations of frames belonging to different rigid bodies.





---

**Note:** Ensure each joint frame port connects rigidly to a Solid or Inertia block. The connection can be direct, through a connection line, or indirect, through one or more Rigid Transform blocks. Joint frame ports not rigidly connected to components with inertia (those containing at least one Solid or Inertia block) can cause a degenerate-mass error during simulation.

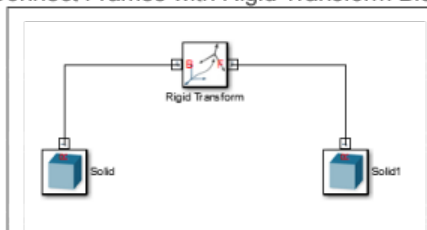
---

## Translation and Rotation

To separate two frames in space, you use the **Rigid Transform** block. By connecting two frame entities to the base and follower frame ports of this block, you apply the rigid transformation that the block specifies. Rigid transformations include translation and rotation.

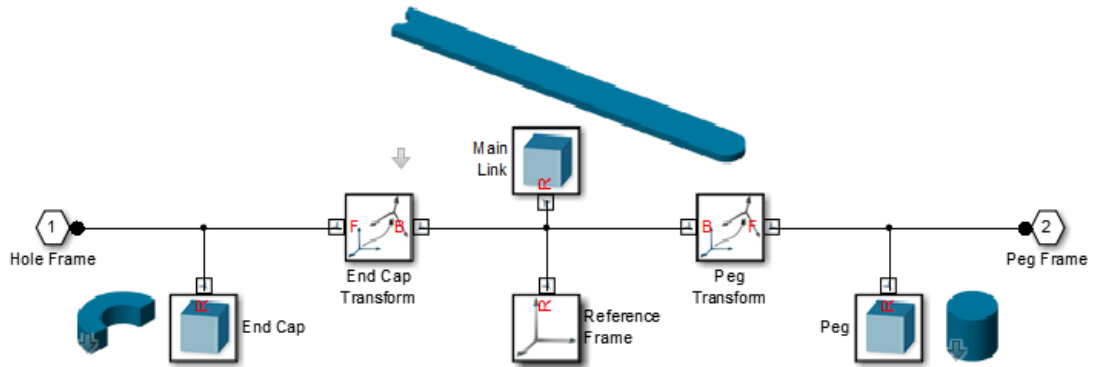
You can apply an offset distance between two frame origins, a rotation angle between the frame axes, or both. Two frames that you connect using a Rigid Transform block behave as a single entity. If you specify neither translation or rotation, the Rigid Transform block represents the identity relationship. The two frames become coincident in space. In the figure, a Rigid Transform block applies a rigid transformation between two solid reference frames.

Connect Frames with Rigid Transform Block

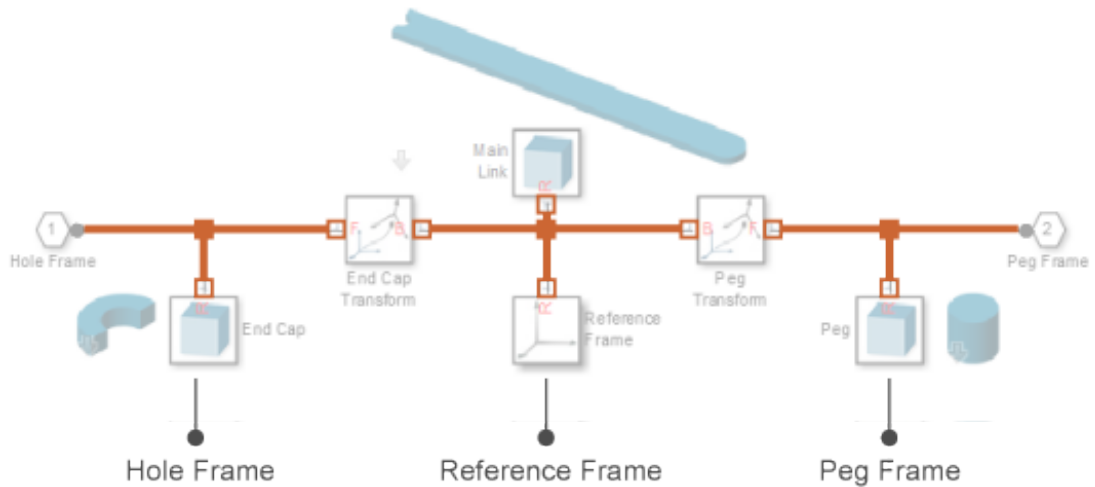


## Interpreting a Frame Network

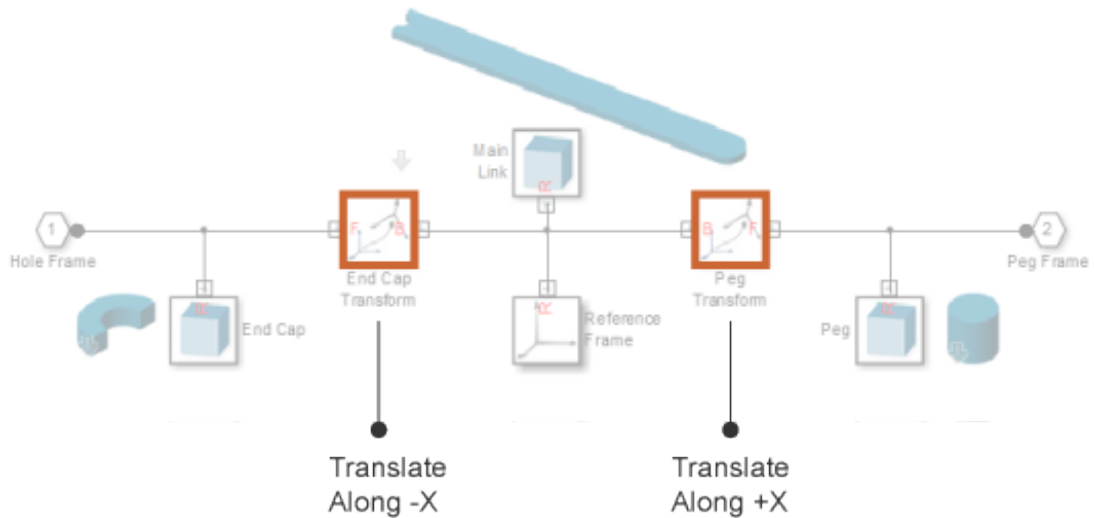
As an example, consider the frame network of a binary link. Simscape Multibody™ provides a model of this rigid body. To open it, at the MATLAB® command prompt, enter `sm_compound_body`. Double-click the Compound Body subsystem block to view the underlying block diagram. The figure shows this diagram.



To represent the binary link, the Compound Body subsystem contains three Solid blocks. The blocks represent the main, peg, and hole sections. Three frames provide the position and orientation of the three solids according to the guidelines that section “Identity Relationships” on page 1-12 introduces. Each group of frame ports, lines, and nodes that directly connect to each other represents one frame. The figure shows the three frames in the block diagram.



Two Rigid Transform blocks represent the spatial relationships between the three frames. One block translates the hole frame with respect to the reference frame along the common  $-X$  axis. The other block translates the peg frame with respect to the reference frame along the common  $+X$  axis. The figure shows these two blocks.



## More About

- “World and Reference Frames” on page 1-16
- “Common Issues with Frame Connections” on page 1-22

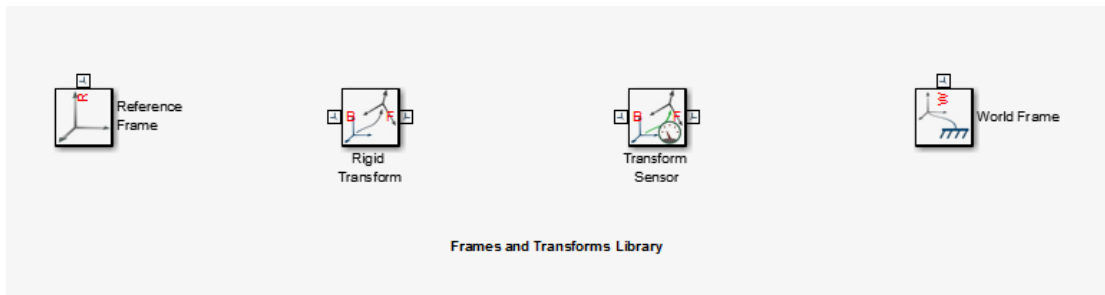
## World and Reference Frames

**In this section...**

“World Frame” on page 1-16

“Reference Frame” on page 1-19

Two preset frames are available in Simscape Multibody: World and Reference. These are standalone frames with respect to which you can define other frames in a model. New frames can in turn serve as the basis to define yet other frames. However, directly or indirectly, all frames depend on either World or Reference frames. Both frames are available as blocks in the Frames and Transforms library.



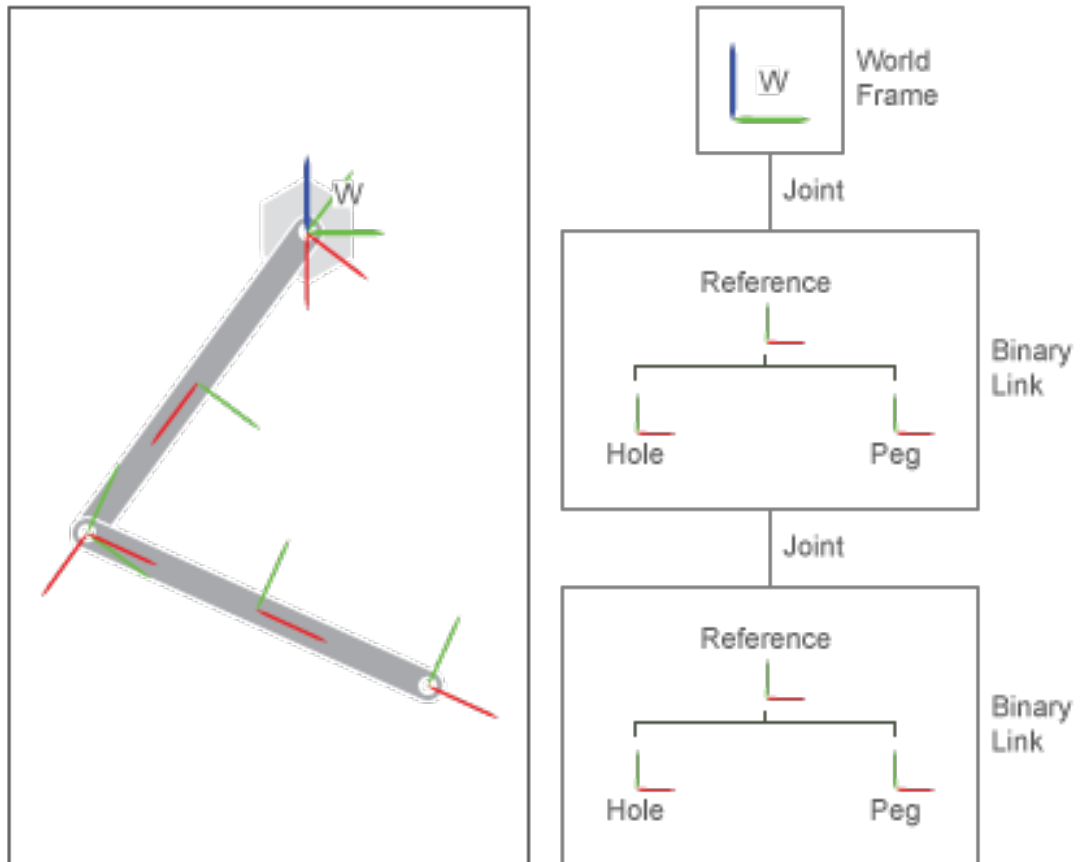
### World Frame

The World frame represents the external environment of a mechanical system. It is always at absolute rest, and therefore experiences zero acceleration. As a consequence, centripetal and other pseudo-forces are not present in the world frame, and it is said to be inertial. Rigidly connecting any frame to the World frame makes that frame also inertial. To add the World frame to a model, use the World Frame block.

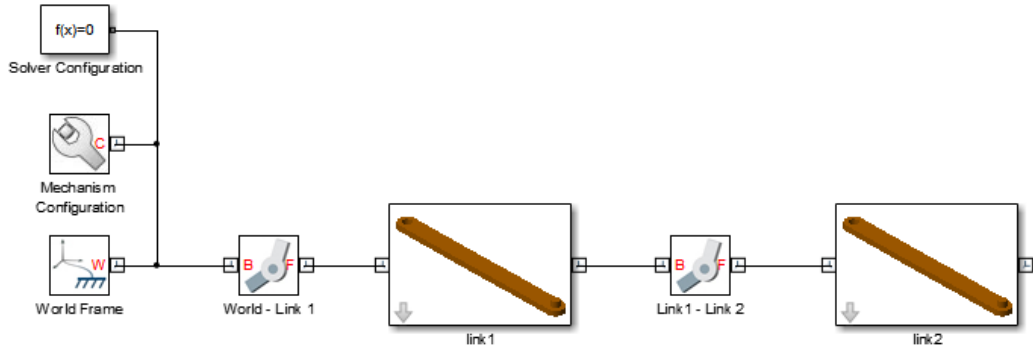


The World frame is the ultimate reference frame. Its position and orientation are predefined and do not depend on any other frame. You can always apply a transform to

the World frame and obtain a new frame. Applying a transform to the resulting frame in turn yields more new frames, all indirectly related to the World frame. The result is a frame tree with the World frame at the root. The figure shows such a frame tree for a double-pendulum system.

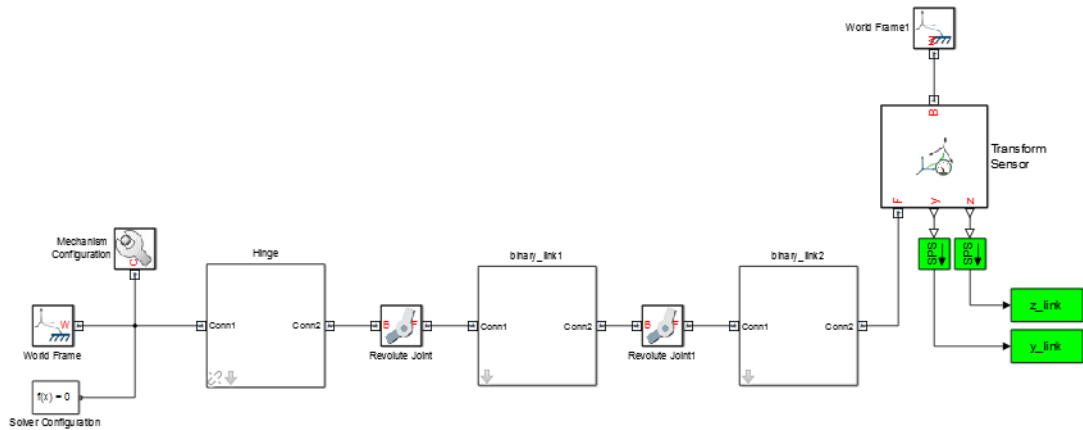


The double-pendulum block diagram is based on this frame tree. The World Frame block identifies the root of the frame tree. A Revolute Joint block applies the variable transform that relates the World frame to the binary link peg frame. A second Revolute Joint block applies a similar variable transform between the hole and peg frames of adjoining binary links. The figure shows this block diagram.



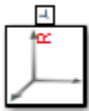
The World frame is present in every model. However, the World Frame block is strictly optional. If you do not add this block to a model, Simscape Multibody assigns one of the existing frames as the World frame. This implicit World frame connects to the rest of the model via an implicit 6-DOF joint, which in the absence of counteracting forces allows a machine to fall under gravity.

You can connect multiple World Frame blocks to a model. However, all World Frame blocks represent the same frame. In this sense, the World frame is unique. You can add multiple World Frame blocks to simplify modeling tasks, e.g., sensing motion with respect to the World frame. The figure shows the model of a double-pendulum with two World Frame blocks. Both World Frame blocks represent the same frame.

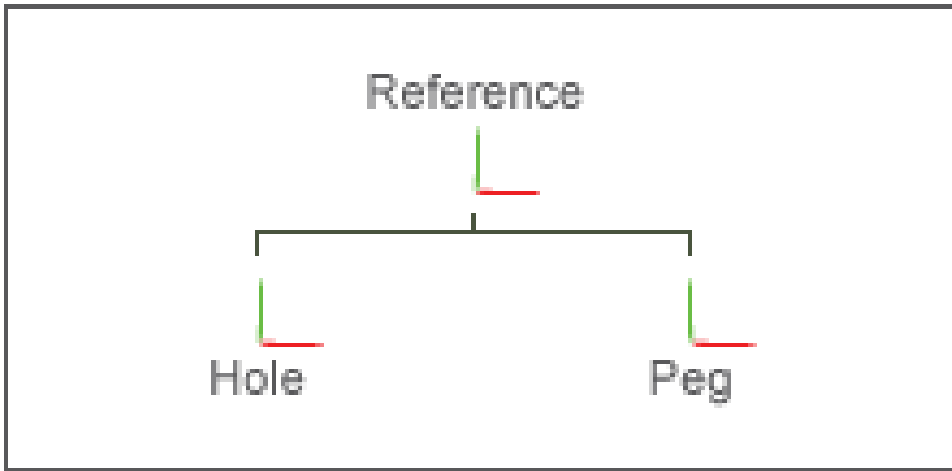
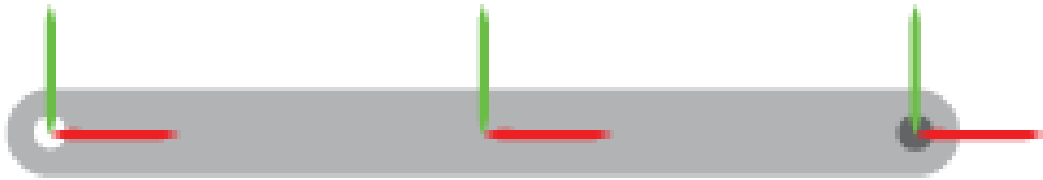


## Reference Frame

The Reference frame represents the root of a rigid body or multibody subsystem. Within a subsystem, it denotes the frame against which all remaining frames are defined. To add a Reference frame, use the Reference Frame block. Use this block to mark the top level of a subsystem frame tree.

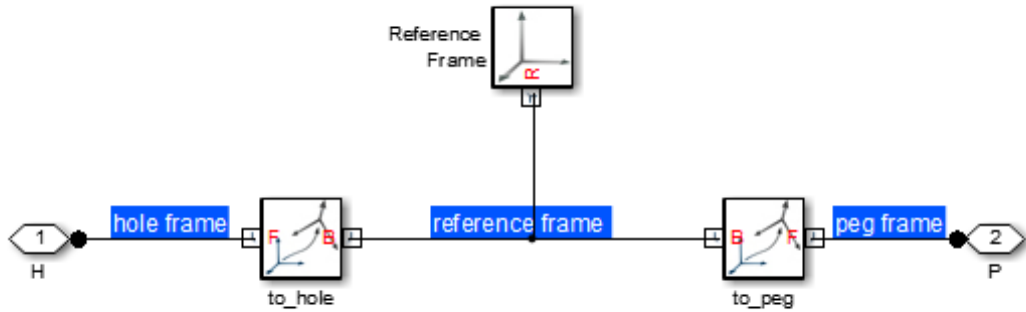


Applying a transform to the Reference frame yields other frames. Applying transforms to these other frames yields still more frames. The overall set of frames forms a frame tree with the Reference frame at the root. The figure shows such a frame tree for one of the binary links used in the double-pendulum system.



The block diagram of the binary link subsystem is based on this frame tree. The following figure shows the binary link block diagram. The Reference Frame block identifies the root of the frame tree. Rigid Transform block to\_hole adds the hole frame. Rigid Transform block to\_peg adds the peg frame. It is a simple task to add the main, peg, and hole solids once these frames are defined.





The distinguishing feature of the Reference frame is that it can move with respect to other frames. Depending on the dynamics of a model, a Reference frame can accelerate, giving rise to pseudo-forces that render this frame non-inertial. Rigidly connecting any frame to a non-inertial Reference frame makes that frame also non-inertial.

The Reference frame is present in every subsystem. However, the Reference Frame block is strictly optional. If you do not add this block to a subsystem, Simscape Multibody assigns one of the existing frames as the Reference frame.

## More About

- “Representing Frames” on page 1-11

## Common Issues with Frame Connections

**In this section...**

“Rigidity Loops” on page 1-22

“Shorted Rigid Transforms” on page 1-23

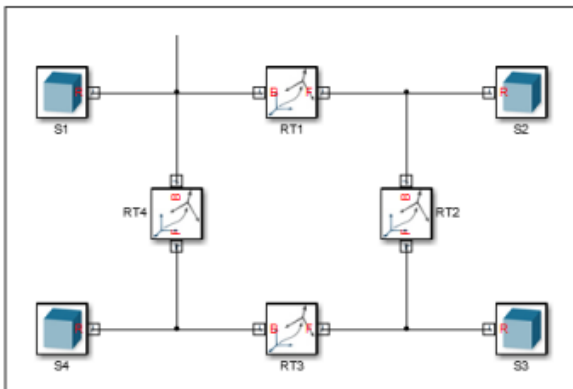
If your model contains an invalid frame connection, Simscape Multibody issues an error and the model does not simulate. Possible error sources include:

- Rigidity loops — Rigidly connecting multiple frames in a closed loop
- Shorted Rigid Transform Blocks — Rigidly connecting base and follower frame ports of a Rigid Transform block

### Rigidity Loops

A rigidity loop is a closed loop of Rigid Transform blocks. The loop contains one redundant Rigid Transform block that over-constrains the subsystem. If a rigidity loop is present, Simscape Multibody issues an error and the model does not simulate.

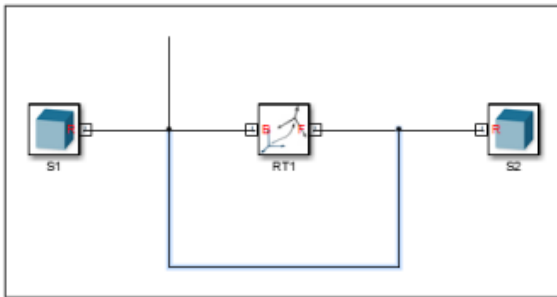
To remove the simulation error, disconnect one Rigid Transform block. This step removes the redundant constraint, and allows the model to simulate. The following figure shows a rigidity loop. The loop contains four Rigid Transform blocks directly connected to each other.



## Shorted Rigid Transforms

A shorted Rigid Transform block contains a direct connection line between base (B) and follower frames (F). The connection line makes the two port frames coincident in space. However, the Rigid Transform block enforces a spatial transformation that translates or rotates one port frame relative to the other. The result is a conflict in the frame definition.

If a shorted Rigid Transform block is present, Simscape Multibody issues an error and the model does not simulate. The error remains even if the Rigid Transform block specifies no rotation and no translation. To remove the simulation error, delete the direct connection line between base and follower frame ports of the Rigid Transform block. The following figure shows a shorted Rigid Transform block.



## More About

- “Representing Frames” on page 1-11



# Rigid Bodies

---

- “Creating Body Subsystems” on page 2-2
- “Simscape Multibody Bodies” on page 2-4
- “Solid Geometry” on page 2-10
- “Solid and Multibody Visualization” on page 2-14
- “Revolution and General Extrusion Shapes” on page 2-16
- “Solid Inertia” on page 2-20
- “Specify Custom Inertia” on page 2-26
- “Add Frames to Solids” on page 2-34
- “Solid Visual Properties” on page 2-44
- “Model a Solid of Revolution” on page 2-51
- “Model an Extruded Solid” on page 2-57
- “Model a Compound Body” on page 2-63

## Creating Body Subsystems

### In this section...

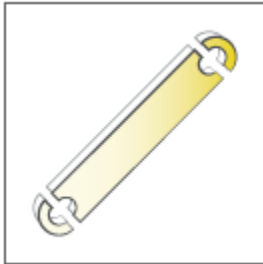
“Study the Bodies to Model” on page 2-2

“Model the Solids in Each Body” on page 2-2

“Connect the Solids Through Frames” on page 2-3

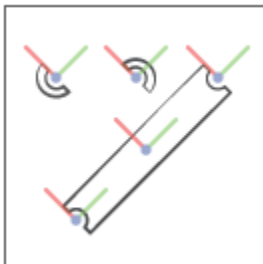
“Verify Body Subsystems” on page 2-3

### Study the Bodies to Model



Break down each body into shapes that you can model using the Solid block. Obtain the dimensions, inertia, and color of each solid.

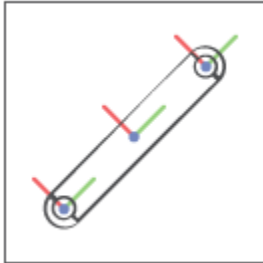
### Model the Solids in Each Body



Specify the properties of each solid using a Solid block. Create solid frames to more easily connect the various solids. See “Model an Extruded Solid” on page 2-57 for a solid

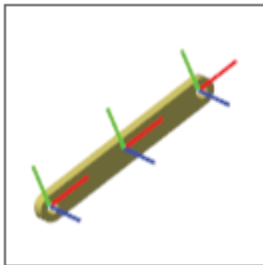
property example. See “Add Frames to Solids” on page 2-34 for a frame creation example.

## Connect the Solids Through Frames



Connect the various frames to assemble the solids into a body. Use the Rigid Transform block to rotate and translate frames relative to each other. See “Model a Compound Body” on page 2-63 for an example.

## Verify Body Subsystems



Visualize each body and verify its geometry, color, and frames. You must connect the body subsystem to a Solver Configuration block and update the diagram in order to open the Mechanics Explorer visualization tool. See the “Visualize Solid Model” on page 2-60 section of “Model a Compound Body” on page 2-63 for an example.

## Simscape Multibody Bodies

### In this section...

“Rigid Body Overview” on page 2-4

“Rigid Body Properties” on page 2-5

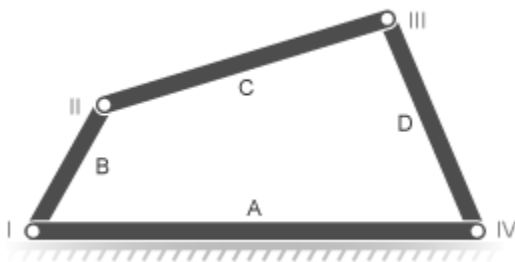
“Rigid Body Frames” on page 2-6

“Rigid Body Delimitation” on page 2-7

“Simple and Compound Rigid Bodies” on page 2-9

### Rigid Body Overview

Bodies are the basic components of a Simscape Multibody model. They are the parts that you interconnect with joints and constraints to model an articulated mechanism or machine. As an example, a four-bar linkage contains four bodies, each a binary link, which interconnect via four revolute joints. The figure shows the four bodies, labeled A–D.

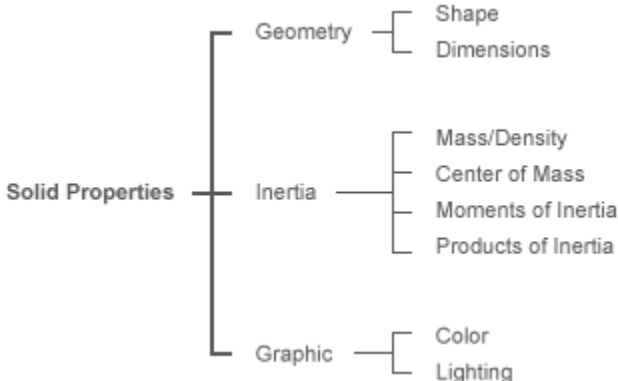


In a Simscape Multibody model, all bodies are rigid. They are idealizations in which internal strains always equal zero. True rigid bodies do not exist in nature but, under normal operating conditions, many engineered components behave as approximately rigid bodies—that is, with negligible deformation. In general, the rigid-body approximation provides accurate modeling results whenever the expected deformation is much smaller than the characteristic length of the modeled system.



## Rigid Body Properties

Solid properties determine the appearance and behavior of a rigid body. For example, the moments and products of inertia determine the angular acceleration of a free rigid body in response to an applied torque. In Simscape Multibody, solid properties fall into three groups—geometry, inertia, and graphic—each with group-specific parameters. The figure lists these properties.



To specify the solid properties of a rigid body, you use the blocks in the Body Elements library. The library contains three blocks, of which Solid is the most frequently used. This block enables you to specify all the solid properties of a rigid body in a single place. The remaining blocks, Inertia and Graphic, serve special cases, such as visualizing certain frames and modeling mass disturbances.

The table summarizes the primary purposes of the Body Elements blocks.

Block	Purpose
Solid	Specify the solid properties—geometry, inertia, graphic—of a simple rigid body or of part of a compound rigid body.
Inertia	Specify the inertial properties of a mass element, such as a mass disturbance present in rigid bodies.
Graphic	Select a graphical icon to visualize any Simscape Multibody frame in a model.

### Rigid Body Frames

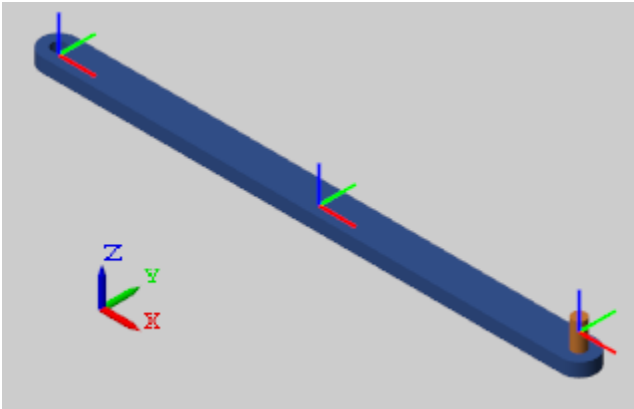
In Simscape Multibody, rigid bodies have frames, each identifying a position and orientation in 3-D space. These frames are important to the Simscape Multibody modeling workflow. They enable you to specify the correct position and orientation for each of the following tasks:

- Connect joints and constraints between rigid bodies. For example, you always connect a revolute joint between two frames in separate rigid bodies (or, alternatively, between a rigid body frame and the world frame).
- Apply forces and torques to or between rigid bodies. For example, you always apply an external force and torque to a single frame in a rigid body.
- Sense motion, forces, and torques between rigid bodies. For example, you always sense the relative position coordinates between two frames on different rigid bodies (or, alternatively, between a rigid body frame and the world frame).

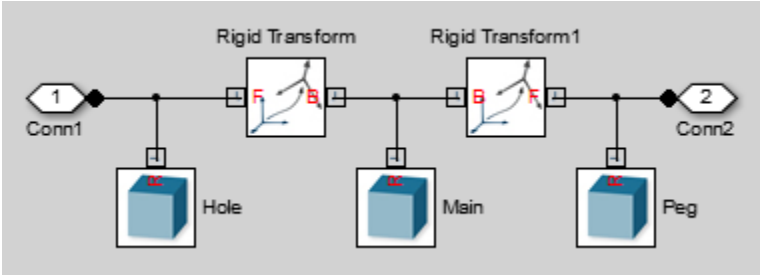
The Solid block, the main component of any body subsystem, provides a reference frame through frame port R. You can create additional frames in the Solid block dialog box using its frame-creation interface. This interface is accessible from the **Frames** area of the Solid block dialog box. The Solid block adds a frame port for every frame that you create.

Drawing a frame connection line between frame ports on different Solid blocks makes the port frames coincident in space. You can translate and rotate these frames relative to each other by adding a Rigid Transform block to the connection line. This block enables you to specify the pose of the follower frame relative to the base frame.

The figure shows an example of a rigid body subsystem in Mechanics Explorer. The rigid body is a binary link with three frames, each associated with a solid section of the link—hole, main, and peg. Rigid transforms specify the translational offset between each pair of frames.



In the binary-link block diagram, Rigid Transform blocks specify the translation transforms between the three frames. A total of two such blocks are needed, one between each pair of frames. The following figure shows the binary link block diagram. To view this subsystem, at the MATLAB command prompt, enter `smdoc_binary_link_a`.



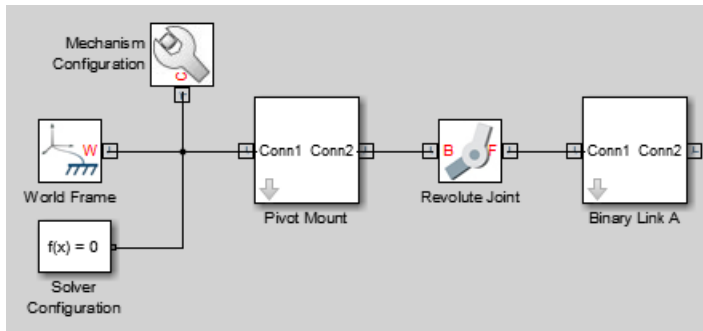
### Rigid Body Delimitation

In a Simscape Multibody model, a set of Solid and Rigid Transform blocks between two joint blocks or between one joint block and the World Frame block constitutes a rigid body. During simulation, Simscape Multibody software computes the center of mass for each such block subset. Gravitational Field blocks in your model, if any, then apply a gravitational force at the calculated centers of mass.

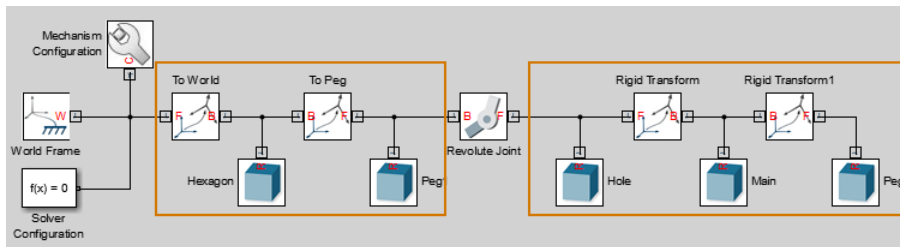
If you connect two halves of a rigid body using a Weld Joint block, the Simscape Multibody model treats the two halves as rigidly connected but independent rigid bodies. Any Gravitational Field blocks in your model then exert a gravitational force at the

center of mass of each half. This strategy enables you to account for gravitational torques acting on a rigid component, such as an asteroid orbiting the Sun.

The figure shows a simple-pendulum model. In this model, a subsystem block neatly encapsulates each rigid body. The model contains two rigid bodies: a pivot mount and a binary link. A joint block separates the mount and the link.



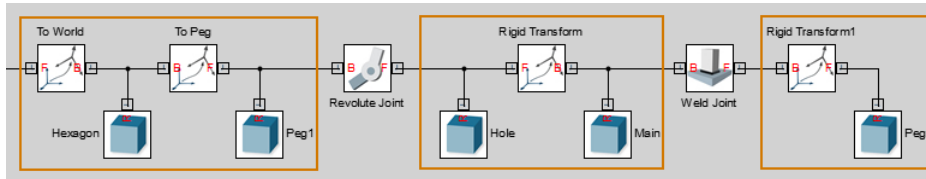
The following figure shows the same model without subsystem blocks. The model treats the blocks on either side of the Revolute Joint block as separate rigid bodies. The blocks to the left of the joint block represent the pivot mount, while the blocks to the right of the joint block represent the binary link.



If you connect a Weld Joint block between the Main and Rigid Transform1 blocks, the Simscape Multibody model recognizes three rigid bodies:

- Rigid body I to the left of the Revolute Joint block.
- Rigid body II between the Revolute Joint and Weld Joint blocks.
- Rigid body III to the right of the Weld Joint block.

The following figure shows the modified model with the Weld Joint block. The Mechanism Configuration, World Frame, and Solver Configuration blocks are omitted to conserve space.



## Simple and Compound Rigid Bodies

Rigid bodies can be simple or compound. The difference between the two rigid body types lies in their complexity. Simple rigid bodies typically have basic shapes, uniform mass distributions, and a single color. Compound rigid bodies have more complex shapes and, occasionally, segmented mass distributions that require multiple Solid blocks to model.

Consider a binary link, the basic component of mechanical linkages such as the double pendulum and the four-bar mechanism. Depending on the level of detail you want to incorporate in your model, you can treat the binary link as a simple rigid body or as a compound rigid body:

- **Simple** — Approximate the rigid-body geometry using a standard Simscape Multibody shape. For example, you can model the binary link using a brick shape with a uniform mass distribution and a single color. The tutorial “Model a Simple Link” shows this approach.
- **Compound** — Model the rigid-body geometry accurately using multiple standard shapes. For example, you can model the binary-link main and hole sections using separate Solid blocks, each with its own shape. The tutorial “Model a Compound Body” on page 2-63 shows this approach.

## Solid Geometry

In this section...
“Geometry Overview” on page 2-10
“Compound Shapes” on page 2-10
“Basic Shapes” on page 2-11
“General Extrusion and Revolution Shapes” on page 2-12
“Imported Shapes” on page 2-13

### Geometry Overview

Geometry is one of three solid properties that you can specify in a Simscape Multibody model. This property includes the shape of a rigid body and its size. For example, the geometry of Earth consists of a spherical shape and an approximate radius of 6,370 km. Specifying these parameters enables Simscape Multibody to perform two tasks:

- Set the visual appearance of a rigid body, excluding color and lighting, in Mechanics Explorer.
- Automatically calculate the inertial properties of a rigid body, including the center of mass, the moments of inertia, and the products of inertia.

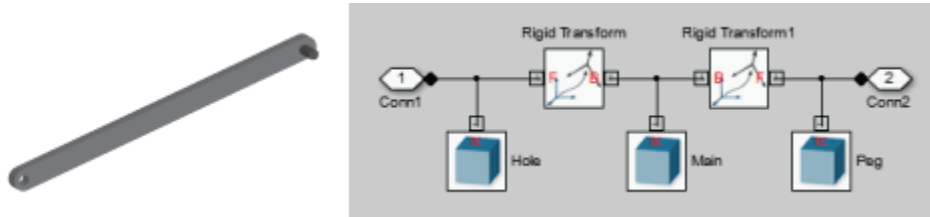
You specify solid geometry using the Solid block. Solid shapes that you can specify range from basic, such as a cylinder, to more sophisticated, such as a general extrusion. For intricate shapes, you can also load solid geometry from external files. The following figure shows examples of the different types of shapes.



### Compound Shapes

Solid shapes are versatile but, used individually, limited. To make the most of solid shapes, you must combine them using multiple Solid blocks. This approach yields a





compound rigid body with an aggregate shape that can be more complex than a single solid shape would allow. The binary link shown in the following figure is one example.




In the figure, three Solid blocks provide the geometries of the binary-link sections—the main body and the two ends. Two Rigid Transform blocks provide the spatial relationships between the three binary-link sections, including their relative positions and orientations.

## Basic Shapes

Often, you can model a rigid body using basic shapes. These are common, simple shapes, such as sphere and cylinder, with parameterizations based on dimensions such as radius and length. Such shapes enable you to quickly model a rigid body approximately—e.g., for quick, proof-of-concept models. The table lists the basic shapes that you can model using the Solid block.

Shape	Example	Parameters
Cylinder		<ul style="list-style-type: none"> <li>Length</li> <li>Radius</li> </ul>
Sphere		<ul style="list-style-type: none"> <li>Radius</li> </ul>
Brick		<ul style="list-style-type: none"> <li>Length</li> <li>Width</li> <li>Thickness</li> </ul>
Ellipsoid		<ul style="list-style-type: none"> <li>Ellipsoid radii along semi-principal axes</li> </ul>



Shape	Example	Parameters
Regular Extrusion		<ul style="list-style-type: none"> <li>• Length</li> <li>• Outer radius</li> <li>• Number of sides</li> </ul>

As an example, you can model a binary link using a brick shape. By using this approximation, you can quickly move on to more important aspects of the modeling workflow, such as multibody assembly. Once you have a working model, you can add detail to the rigid bodies, e.g., by treating the binary links as compound rigid bodies with more complex shapes.

For a tutorial showing how to model a binary link using a brick approximation, see “Model a Simple Link”.

## General Extrusion and Revolution Shapes

To model more complex rigid bodies, the Solid block provides general-extrusion and revolution shapes. These shapes enable you to model rigid bodies with arbitrary cross-sections, such as I-shaped beams and circular domes. The parameterizations for these shapes are more advanced and require detailed knowledge of the cross-section coordinates. The table summarizes the differences between two shapes.

Shape	Description	Example	Parameters
General extrusion	Shape with a general cross-section that remains constant along an extrusion axis.		<ul style="list-style-type: none"> <li>• Cross-section coordinates</li> <li>• Extrusion length</li> </ul>
Revolution	Shape with a general cross-section that remains constant about a revolution axis.		<ul style="list-style-type: none"> <li>• Cross-section coordinates</li> <li>• Revolution angle</li> </ul>

For tutorials showing how to model general-extrusion and revolution shapes, see:

- “Model a Solid of Revolution” on page 2-51
- “Model an Extruded Solid” on page 2-57



## Imported Shapes

Instead of modeling a complex geometry manually, you can import it using one of two file formats: STL and STEP. Files in these formats specify the surface geometries of 3-D solids, albeit using different approaches. The table summarizes the key differences between the two formats.

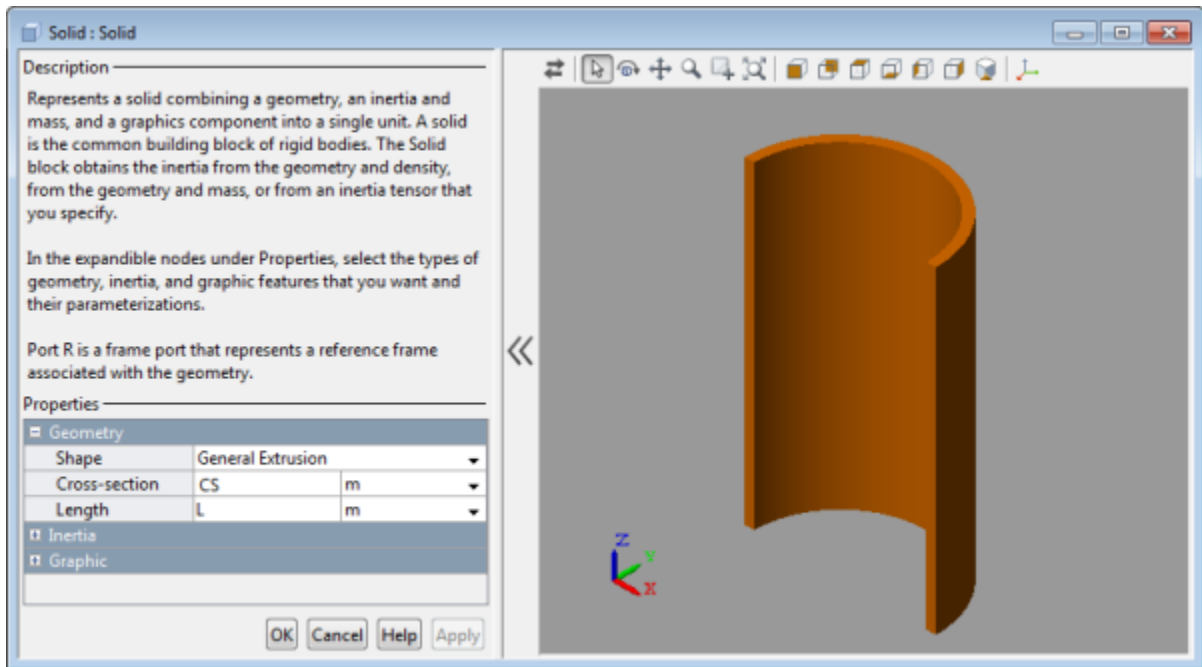
Geometry File Format	Description
STL (Standard Tessellation Language)	Provides the vertex coordinates and normal-vector components for each triangle segment in a tessellated 3-D surface.
STEP (Standard for Exchange of Product Data)	Provides the analytical curves that describe a 3-D surface. STEP files enable the Solid block to automatically compute inertial properties.

Note that these files do not specify graphic or inertial properties. You must specify those properties separately. To generate the STL or STEP files, you must use an external application, such as a CAD tool.

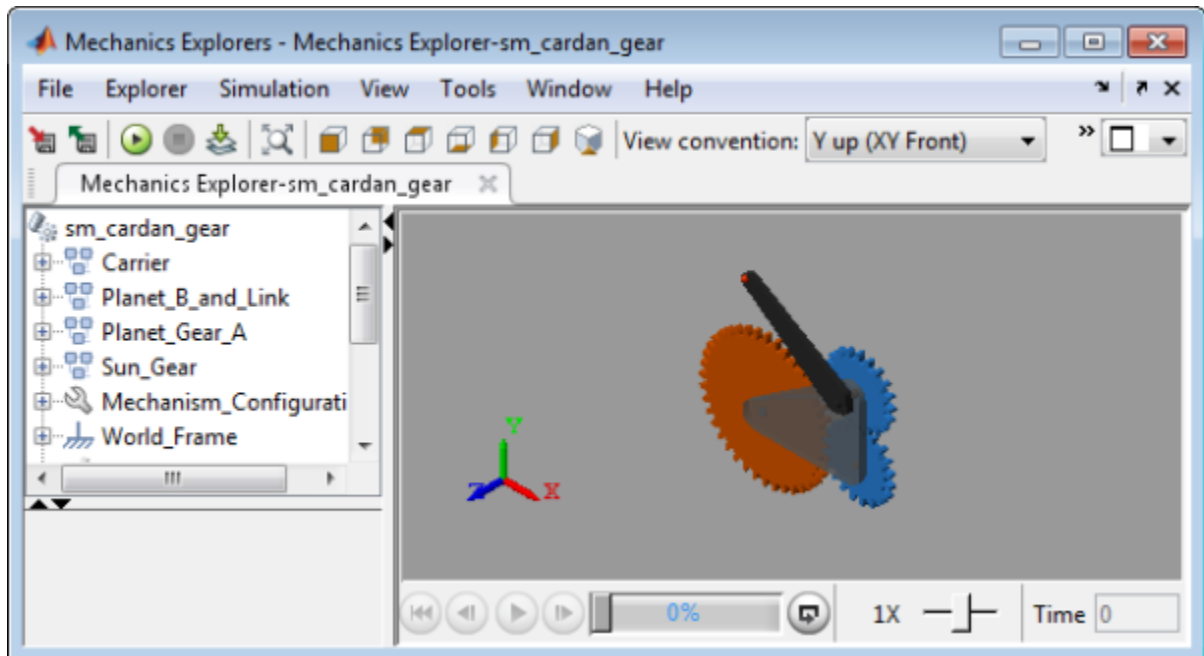
## Solid and Multibody Visualization

Simscape Multibody software provides two visualization utilities that you can use to examine the contents of a model:

- Solid block visualization pane — Visualize an individual solid in a model that is still in progress. Use the contents of the **Solid** block visualization pane to ensure that the solid geometry, color, and frames are correct. You do not have to update or simulate the model.



- Mechanics Explorer — Visualize a complete multibody model. Use the contents of the visualization pane to ensure that the body dimensions and spatial arrangements are correct or reasonable. You must update or simulate the model in order to open Mechanics Explorer.

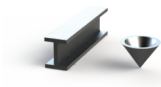


## Revolution and General Extrusion Shapes

In this section...
“Shape Cross Sections” on page 2-16
“Coordinate Matrices” on page 2-16
“Hollow Cross Sections” on page 2-17
“3-D Shape Generation” on page 2-18

### Shape Cross Sections

Using **General Extrusion** and **Revolution** shapes, you can model solids with arbitrary cross sections that remain constant along or about some axis. I-beam and cone shapes are two examples. These shapes differ from simple shapes such as **Brick** and **Sphere** in their requirement that you specify the cross-section coordinates explicitly. You specify these coordinates in the Solid block dialog box in a matrix format.



### Extrusion and Revolution Shapes

#### Coordinate Matrices

Coordinate matrices are  $M \times 2$  in size. Each row corresponds to a point on the cross-section outline and each column corresponds to a coordinate on the cross-section plane. If you specify a cross-section shape using ten points, the resulting coordinate matrix is  $10 \times 2$  in size.

The cross-section plane differs between **General Extrusion** and **Revolution** shapes. That plane is the XY plane for general extrusion shapes and the XZ plane for revolution shapes. Cross-section coordinates are therefore [X, Y] pairs for general extrusion shapes and [X, Z] pairs for revolution shapes.

The Solid block generates the cross-section shape from your coordinate matrix by connecting each point to the next with a straight line. If the last point is different from the first point, the block connects the two in order to close the cross-section outline.

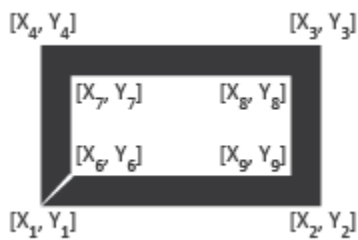
The cross-section outline divides the solid region of your shape from the empty region outside of it. To decide where the solid region is, the Solid block uses a special rule: looking from one point to the next, the solid region must lie to the left of the connecting line. The figure shows the application of this rule to an I-beam cross-section shape.



## Hollow Cross Sections

Bodies are often hollow. A box beam is one example. You can model such bodies using General Extrusion and Revolution shapes. As before, you must specify the coordinate matrix as a continuous path so that, looking from one coordinate pair to the next, the solid region lies to your left and the empty region lies to your right.

However, the path must now traverse not only the outer outline of the cross-section, but also the outline of its hollow region. To do this, the path must cut across the solid portion of the cross section. The figure shows such a cut.



The coordinate matrix for the outer cross-section outline is

$$\text{outerCS} = [X_1, Y_1; X_2, Y_2; X_3, Y_3; X_4, Y_4; X_1, Y_1].$$

Similarly, the coordinate matrix for the inner cross-section outline is

$$innerCS = [X_6, Y_6; X_7, Y_7; X_8, Y_8; X_9, Y_9; X_6, Y_6].$$

The complete coordinate matrix is then

$$CS = [outerCS; innerCS].$$

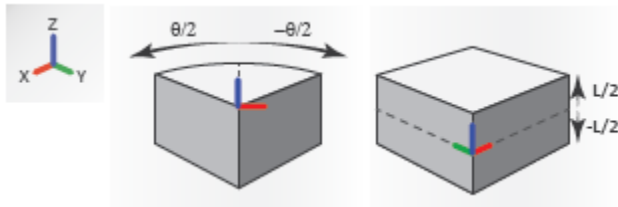
Note that *outerCS* traces the outer profile counterclockwise, while *innerCS* traces the inner profile clockwise. You must reverse the order of the coordinates as you transition between the two outlines in order to keep the solid region to the left of the cross-section line.

Note also that, taken individually, *outerCS* and *innerCS* each trace a closed outline. You must close each outline by ending its coordinate matrix on the first coordinate pair for that outline. The Solid block automatically closes the overall cross-section profile by connecting the last coordinate pair to the first. In doing so, the Solid block traces the first cut at the same location and in reverse, ensuring that the cut thickness is zero.

### 3-D Shape Generation

The Solid block produces a 3-D shape from a cross-section outline by sweeping the outline along or about the reference frame Z axis. The sweep amount is the same in the positive and negative directions of the Z axis. The figure shows the directions and amounts of sweep for a revolution shape and a general extrusion shape:

- For a general extrusion of thickness  $L$ , the block sweeps the cross-section outline by  $L/2$  along the positive and negative directions of the Z axis.
- For a revolution with a sweep angle of  $\theta$ , the block sweeps the cross-section outline by  $\theta/2$  about the positive and negative directions of the Z axis.



The reference port frame of revolution and general extrusion solids has its origin at the  $[0, 0]$  coordinate. This coordinate lies in the cross-section plane for a general extrusion solid and in the revolution axis for a revolution solid.

## Solid Inertia

In this section...
“Inertial Properties” on page 2-20
“Blocks with Inertia” on page 2-20
“Inertia in a Model” on page 2-21
“Inertia Parameterizations” on page 2-22
“Custom Inertia” on page 2-22
“Moments and Products of Inertia” on page 2-23
“Complex Inertias” on page 2-25

### Inertial Properties

Inertia is the resistance of matter to acceleration due to applied forces and torques. The inertial properties of a body include its mass and inertia tensor—a symmetric  $3 \times 3$  matrix that contains the moments and products of inertia. Mass resists translational acceleration while the moments and products of inertia resist rotational acceleration.

Among the solid properties of a model, the inertial properties have the greatest impact on multibody dynamics. Those that you must specify depend on the type of inertia you are modeling—a point mass or a body with a 3-D mass distribution. They include one or more of the following:

- Mass or density
- Center of mass
- Moments of inertia
- Products of inertia

In a Simscape Multibody model, these properties are time-invariant. Rigid bodies cannot gain or lose mass nor can they deform in response to an applied force or torque. The mass distribution of a body—and therefore its inertia tensor and center of mass—remain constant throughout simulation.

### Blocks with Inertia

You can model inertia using the following blocks:



- **Solid** — Model a complete solid element with geometry, inertia, and color. This block can automatically compute the moments of inertia, products of inertia, and center of mass based on the solid geometry and mass or mass density. During simulation, Mechanics Explorer renders the solid using the geometry and color specified.
- **Inertia** — Model only the inertial properties of a solid element. You must specify the moments of inertia, products of inertia, and center of mass explicitly. During simulation, Mechanics Explorer identifies the center of mass using the inertia icon



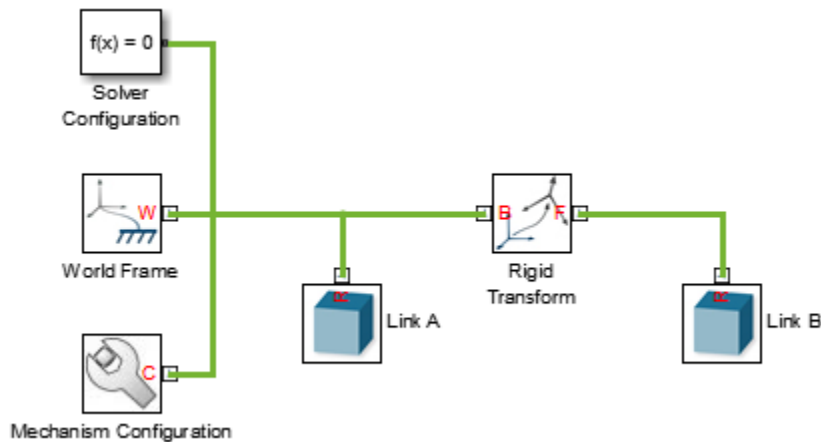
## Inertia in a Model

To add a Solid or Inertia block to your model, connect its frame port to another frame entity in the model. Frame entities include frame lines, nodes, and ports. The frame entity to which you connect the block determines the position and orientation of the inertia within the model. See “Representing Frames” on page 1-11.

The Solid and Inertia blocks each provide a reference frame port. The Solid block enables you to create additional frames, each of which adds a new frame port to the block. You can use any of these frames to connect a Solid block in a model.

The figure shows an example. The model shown contains two Solid blocks, labeled Link A and Link B. The reference frame port of Link A connects directly to the World Frame block. Its reference frame is therefore coincident with the World frame.

The reference frame port of Link B connects to the follower frame port of the Rigid Transform block. This block applies a spatial transform between the World frame the reference frame of the Link B block. The spatial transform translates and/or rotates the two frames relative to each other.



For examples showing how to position solid elements in a model, see:

- “Model a Compound Body” on page 2-63

### Inertia Parameterizations

Once you have connected the Solid or Inertia blocks in a model, you must specify their inertial parameters. These depend on the inertia parameterization that you select. The blocks provide three optional parameterizations:

- **Calculate from Geometry** — Specify mass or density. The Solid block automatically computes the remaining inertial properties based on the solid geometry. Only the Solid block provides this parameterization.
- **Point Mass** — Specify the mass and ignore the remaining inertial parameters. The inertia behaves as point mass with no rotational inertia.
- **Custom** — Manually specify every inertial parameter. You must obtain each parameter through direct calculation or from an external modeling platform.

### Custom Inertia

If you select the **Custom Inertia** parameterization, you must specify the moments of inertia, products of inertia, and center of mass explicitly. These parameters depend

closely on the reference frame used in their calculations, so you must ensure that frame matches the one used in Simscape Multibody:

- Moments and products of inertia — Enter with respect to a frame parallel to the reference port frame but with origin at the center of mass.
- Center of mass — Enter with respect to the reference port frame.

Consider the main section of the binary link in “Model a Compound Body” on page 2-63. You model this solid using a single solid block with a **General Extrusion** shape. As described in the Solid block documentation, the reference port frame for a general extrusion has its origin in the XY plane at the [0,0] cross-section coordinate.

The figure shows the solid reference port frame, labeled R. The center-of-mass coordinates must be with respect to this frame. The moments and products of inertia must be with respect to a parallel frame offset so that its origin coincides with the center of mass. This frame is virtual, as it does not correspond to any frame port, line, or node in the model. It is labeled R\* in the figure.

## Moments and Products of Inertia

You can extract the moments and products of inertia directly from the inertia tensor. This tensor is symmetric: elements with reciprocal indices have the same magnitude. That is:

- $I_{xy} = I_{yx}$
- $I_{yz} = I_{zy}$
- $I_{zx} = I_{xz}$

This symmetry reduces the number of unique tensor elements to six—three moments of inertia and three products of inertia. The complete inertia tensor has the form:

$$\begin{bmatrix} I_{xx} & I_{xy} & I_{zx} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{zx} & I_{yz} & I_{zz} \end{bmatrix}$$

The moments of inertia are the diagonal elements:

$$\begin{bmatrix} I_{xx} & & \\ & I_{yy} & \\ & & I_{zz} \end{bmatrix}$$

Simscape Multibody defines these elements as follows:

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

The products of inertia are the unique off-diagonal elements, each of which appears in the inertia tensor twice:

$$\begin{bmatrix} & I_{xy} & I_{zx} \\ I_{xy} & & I_{yz} \\ I_{zx} & I_{yz} & \end{bmatrix}$$

Simscape Multibody defines these elements as follows:

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

The inertia tensor is simplest when it is diagonal. Such a tensor provides the moments of inertia about the principal axes of the solid or inertia element—known as the principal moments of inertia. The products of inertia reduce to zero:

$$\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

For more information, see the **Solid** and **Inertia** block reference pages.

## Complex Inertias

Bodies often comprise different materials, have complex shapes, or contain material imperfections that alter their centers of mass and principal axes. One example is an imbalanced automobile wheel after driving through a pothole. You can model complex inertias such as these using two approaches:

- Use a divide-and-conquer approach. Break up the complex solid or inertia into simpler chunks and model each using a separate **Solid** or **Inertia** block. The resulting set of **Solid** and **Inertia** blocks constitute a compound inertia. You use a similar approach to model complex geometries, such as the binary link geometry in “Model a Compound Body” on page 2-63.
- Manually specify the complete inertial properties using a single **Solid** or **Inertia** block with the inertia parameterization set to **Custom**. You must obtain the moments of inertia, products of inertia, and center of mass through direct calculation, from another modeling platform, or from another external source.

For bodies with complex shapes but uniform mass distributions, you can also import a **STEP** file containing the solid geometry and set the inertia parameterization to **Calculate from Geometry**.

## Specify Custom Inertia

In this section...
“Custom Inertia” on page 2-26
“Model Overview” on page 2-26
“Inertia Parameters” on page 2-27
“Build Model” on page 2-29
“Specify Inertia” on page 2-29
“Add Motion Sensing” on page 2-31
“Run Simulation” on page 2-32

### Custom Inertia

You can specify the inertia tensor and center of mass of a solid explicitly. To do this, in the **Solid** or **Inertia** block, you set the inertia parameterization to **Custom**. This option exposes additional fields so that you can enter the moments of inertia, products of inertia, and center of mass.

Before entering the inertia parameters, you must ensure that they are defined with respect to the correct frame. This frame typically coincides with the reference port frame for standard shapes such as **Sphere**, **Brick**, and **Cylinder**, but not for the more advanced **Revolution** and **General Extrusion** shapes or imported shapes.

This tutorial shows how you can specify the inertial parameters of a solid explicitly, clarifying along the way the frames that the moments of inertia, products of inertia, and center of mass must refer to.

### Model Overview

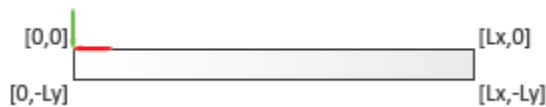
The model in this tutorial is simple. It contains a single **Solid** block through which you specify the inertial parameters of your solid. This block connects to the World frame through a **Revolute Joint** block, providing it one rotational degree of freedom. You examine the effect of the inertial parameters by simulating the angular motion in the model.

The solid that you model has a brick shape. To illustrate the challenges associated with the **Custom Inertia** parameterization, the tutorial represents the brick shape as a

general extrusion. The result is a brick shape with the reference port frame origin located at the  $[0,0]$  cross-section coordinates—which in this tutorial do not coincide with the solid center of mass.

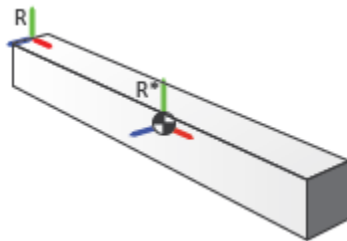
The figure shows the cross section of the general extrusion and its coordinates. The  $[0,0]$  coordinates, and therefore the reference port frame origin, coincide with the upper left vertex in the cross section. Variables  $L_x$ ,  $L_y$ , and  $L_z$  refer to the length, width, and thickness of the solid, respectively. Their values are:

- $L_x = 20$  cm
- $L_y = 1$  cm
- $L_z = 1$  cm



You specify the center of mass, but not the moments or products of inertia, with respect to the reference port frame. To specify the moments and products of inertia, you must use a different frame—one whose axes are parallel to the reference port frame axes but whose origin is coincident with the solid center of mass.

The figure shows the extruded solid, its reference port frame (R), and the parallel frame with origin at the center of mass ( $R^*$ ). This frame does not correspond to any frame entity in the model. It is said to be virtual.



## Inertia Parameters

To fully specify the inertia of a solid, you must specify four parameters:

- Mass
- Center of mass
- Moments of inertia
- Products of inertia

The mass of the solid is the product of its density and volume. For an aluminum solid, the density is  $\rho = 2.7 \frac{\text{g}}{\text{cm}^3}$ . The volume is the product of the side lengths

$V = L_x L_y L_z = 20 \text{ cm}^3$ . The solid mass is therefore:

$$M = \rho V = 54 \text{ g.}$$

Assuming the mass distribution is uniform, the center of mass must coincide with the center of geometry, which in frame  $R$  has the coordinates:

$$CM = \left[ \frac{L_x}{2}, -\frac{L_y}{2}, 0 \right] = [10, -0.5, 0] \text{ cm.}$$

The axes of frame  $R^*$  align with the principal axes of the solid. Taken with respect to this frame, the moments of inertia are:

- $I_{xx} = \frac{M}{12} (L_y^2 + L_z^2) = 9 \text{ g cm}^2$
- $I_{yy} = \frac{M}{12} (L_z^2 + L_x^2) = 1804.5 \text{ g cm}^2$
- $I_{zz} = \frac{M}{12} (L_x^2 + L_y^2) = 1804.5 \text{ g cm}^2$

The axes of frame  $R^*$  are the solid principal axes. The products of inertia are therefore zero:

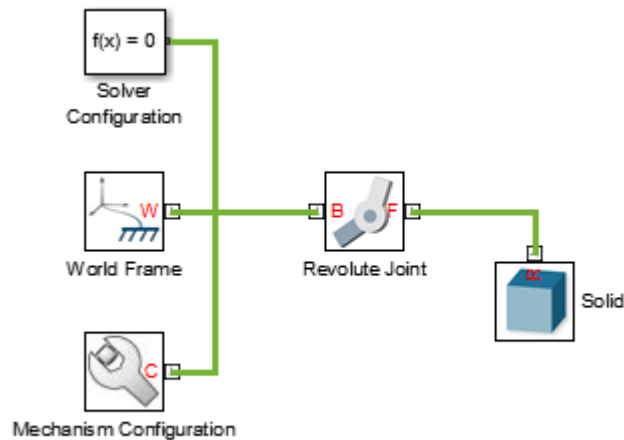
- $I_{yz} = 0$
- $I_{zx} = 0$



- $I_{xy} = 0$

## Build Model

- 1 At the MATLAB command prompt, enter `smnew`. Simscape Multibody opens a new model with some commonly used blocks. The Simscape Multibody library also opens up.
- 2 From the Joints library, drag a **Revolute Joint** block.
- 3 Connect the blocks as shown in the figure. You can remove any unused blocks.



- 4 In the Mechanism Configuration block, set **Gravity** to  $[0 \ -9.81 \ 0]$ . The new gravity vector is perpendicular to the revolute joint axis, ensuring that the solid you modeled oscillates due to gravity when displaced from its equilibrium position.

## Specify Inertia

- 1 In the Solid block dialog box, under **Geometry**, specify the following parameters. These parameters define the shape and size of the solid.

Parameter	Value	Units
Shape	General Extrusion	

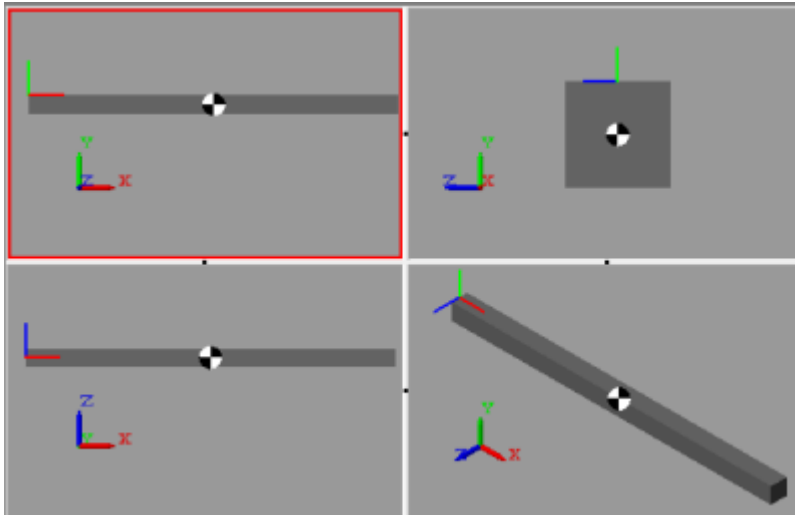
Parameter	Value	Units
<b>Cross-Section</b>	[0 0; 0 -1; 20 -1; 20 0]	cm
<b>Length</b>	1	cm

- 2 Under **Inertia**, specify the following parameters. These are the inertia parameters that you calculated at the beginning of the tutorial.

Parameter	Value	Units
<b>Type</b>	Custom	
<b>Mass</b>	54	g
<b>Center of mass</b>	[10, -0.5, 0]	cm
<b>Moments of Inertia</b>	[9, 1804.5, 1804.5]	$\text{g}\cdot\text{cm}^2$
<b>Products of Inertia</b>	[0, 0, 0]	$\text{g}\cdot\text{cm}^2$

- 3 Update the block diagram—for example, by selecting **Simulation > Update Diagram**. Mechanics Explorer opens with a static 3-D view of the model in its initial configuration.
- 4 In the Mechanics Explorer menu bar, select:
- **View > Layout > Four Standard Views**. This option splits the visualization window into four panes, each with a different view.
  - **View > Show Frames**. This option exposes all the frames in the model.
  - **View > Show COMs**. This option exposes the center of mass for each rigid body in the model.

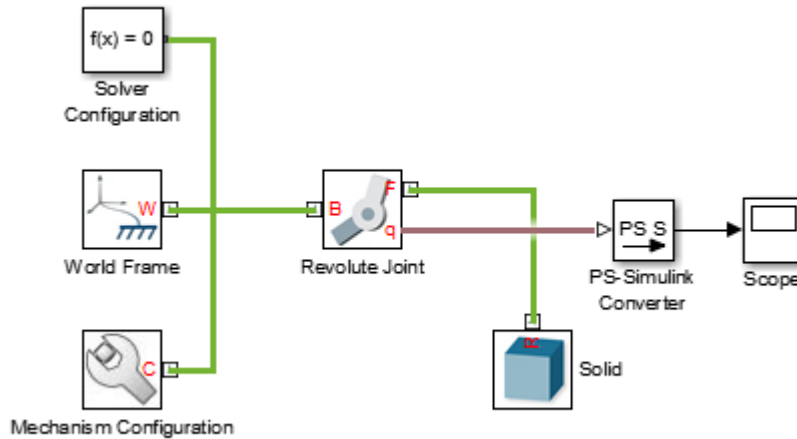
The figure shows the resulting view in Mechanics Explorer.



- 5 Examine the solid reference frame and center of mass. Verify that the center of mass appears at the geometric center of the solid and that the solid reference frame origin coincides with the upper left corner of the solid cross section.

## Add Motion Sensing

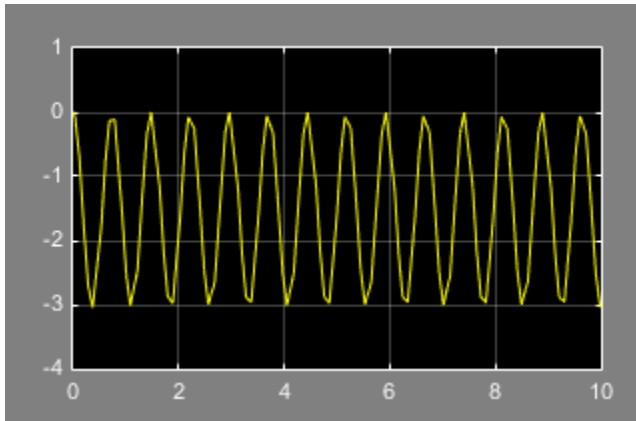
- 1 In the Revolute Joint block dialog box, under **Sensing**, select **Position** and click **OK**. The block exposes the physical signal output port  $q$ .
- 2 From the Simscape Utilities library, drag a **PS-Simulink Converter** block. This block enables you to convert the physical signal output from the joint block into a Simulink signal.
- 3 From the Simulink<sup>®</sup> Sinks library, drag a **Scope** block. This block enables you to plot the joint position output as a function of time.
- 4 Connect the blocks as shown in the figure.



### Run Simulation

Simulate the model—for example, by selecting **Simulation** > **Run**. Mechanics Explorer plays an animation of the model. To ensure that the gravity vector aligns vertically on your screen, in the Mechanics Explorer toolstrip, set **View Convention** to **Y up (XY Front)**.

Double-click the Scope block and examine the oscillation period of the solid. The figure shows the resulting plot.



Try changing the inertia parameterization in the Solid block to **Calculate from Geometry** and simulate the model once again. Compare the plot from the second simulation to the first. The results are identical.

## Add Frames to Solids

In this section...
“Solid Frames” on page 2-34
“Frame-Creation Interface” on page 2-35
“Model Solid Shape” on page 2-36
“Create New Frame” on page 2-37
“Specify Frame Origin” on page 2-38
“Specify Primary Axis” on page 2-40
“Specify Secondary Axis” on page 2-41
“Save New Frame” on page 2-42

### Solid Frames

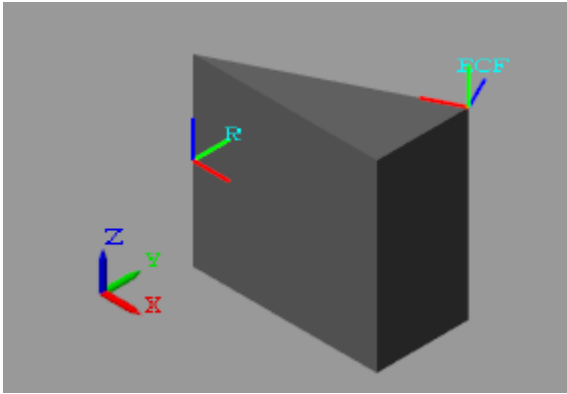
By default, the Solid block provides only a reference frame port, labeled R. In simple shapes, such as bricks, cylinders, and spheres, the reference frame origin coincides with the solid center of mass. In more complex shapes, such as extrusions and revolutions, the reference frame can be anywhere relative to the solid.

In many applications, the reference frame of a solid is inadequate for connecting joints and constraints or for applying forces and torques. In such cases, you can create new frames external to the Solid block using the Rigid Transform block. This block enables you to define the new frame by specifying translation and rotation transforms numerically.



An alternative approach, and one that is often more intuitive, is to create new frames directly in the Solid block dialog box using the frame-creation interface. This interface enables you to define new frames interactively by aligning the frame origin and axes with geometric features such as planes, lines, and points.

In this example, you create a new frame in a solid using the frame-creation interface. The solid shape is a general extrusion with three unequal sides. This shape helps to demonstrate the difference between the primary and secondary frame axes that you specify in the frame creation interface.

The figure shows the solid shape, its default reference frame (R), and the new frame that you create (ECF).



## Frame-Creation Interface

The frame-creation interface is accessible through the Solid block dialog box. To open the interface, in the **Frames** expandable area, select the **Create** button . If you change any of the block parameters, you must first update the solid visualization by selecting the **Update Visualization** button .

You can define frames based on geometric features of the solid or a choice of two frames—reference and principal inertia frames. The reference frame is the default frame of the solid. The principal inertia frame is one whose origin coincides with the center of mass and whose axes coincide with the principal axes of the solid.

Frames that you define by geometric features are specific to the shapes the features belong to. If you make the frame origin coincident with the vertex of a brick, the new frame is valid only for that particular brick shape. If you change shapes, you must edit or delete the new frame, as the geometric features it depends on no longer exist.

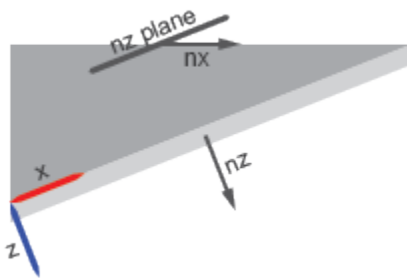
The frame-creation interface has three sections for specifying the following:

- Frame origin
- Primary axis
- Secondary axis

The primary axis constrains the possible directions of the remaining two axes. These axes must lie in the normal plane of the primary axis. If the axis or geometric feature

used to define the secondary axis does not lie on this plane, the secondary axis is the projection of that axis or feature onto the normal plane.


The figure shows a top view of the three-sided extrusion you model in this tutorial. You align the primary axis ( $z$ ) with the surface normal vector  $n_z$  and the secondary axis ( $x$ ) with the line vector  $n_x$ . Because  $n_x$  is not normal to the primary axis, the secondary axis is the projection of  $n_x$  onto the normal plane of the primary axis.



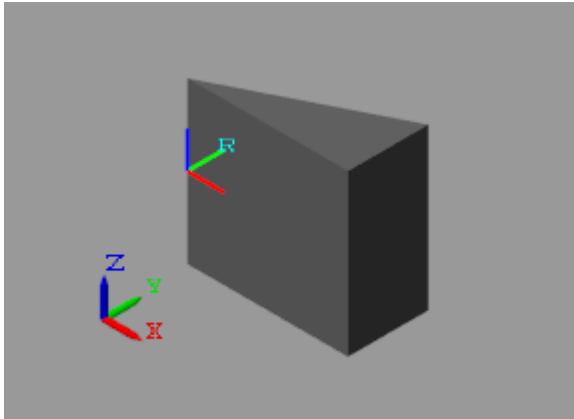
### Model Solid Shape

- 1 From the Body Elements library, add one Solid block to a new model. The Solid block provides its own visualization utility. You do not need to update the block diagram to visualize the solid shape or its frames.
- 2 In the Solid block dialog box, specify these parameters.


Parameter	Value
Geometry > Shape	General Extrusion
Geometry > Cross-section	[0,0;1,0;1,0.5]

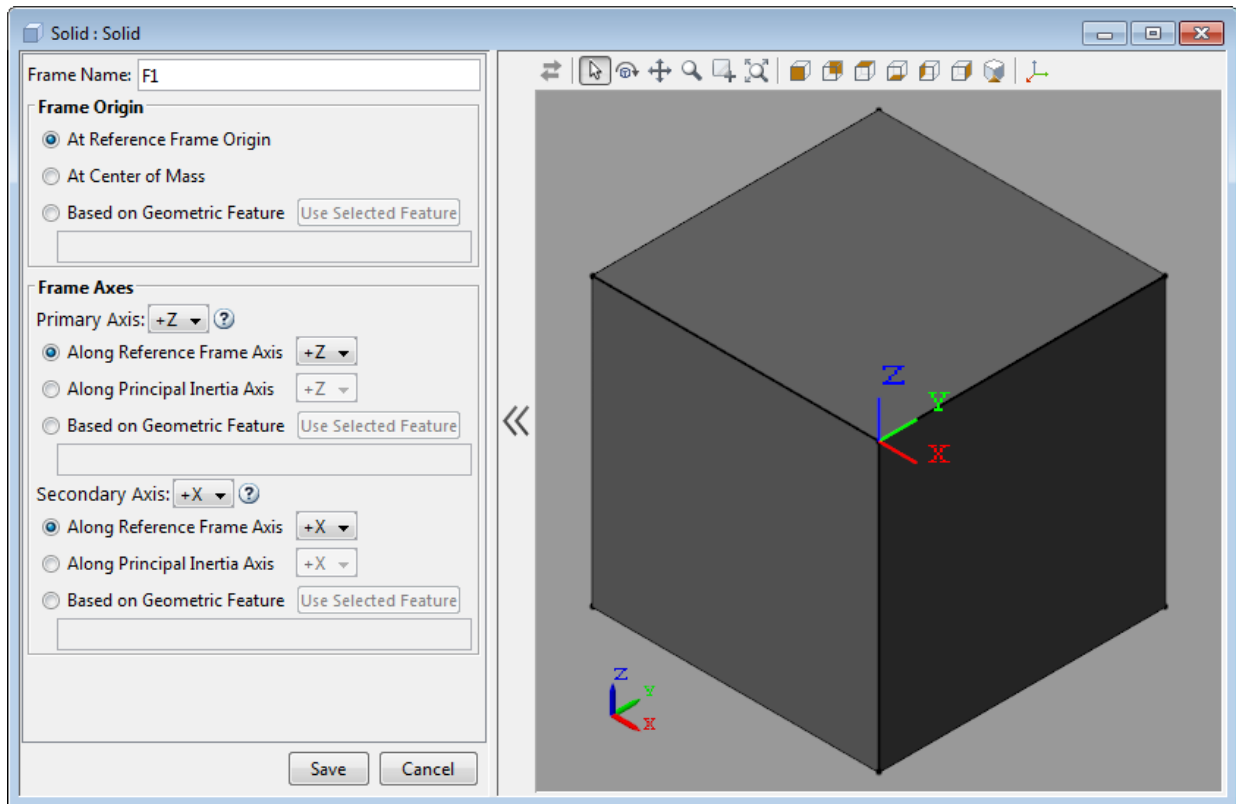
- 3 In the visualization toolstrip, select the **Update Visualization** button . The visualization pane updates with the three-sided extrusion that you specified.
- 4 Select the **Toggle Frames** button. The visualization pane shows all the frames in the solid. At this point, the solid has a single frame—its reference frame. The reference frame origin coincides with the [0,0] cross-section coordinate in the midplane of the extrusion.





## Create New Frame

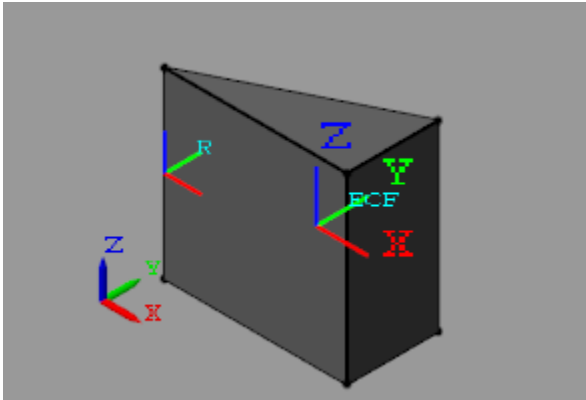
In the **Frames** expandable area of the Solid block dialog box, select the **Create** button . The Solid block opens the frame-creation interface.



In the **Frame Name** parameter, enter ECF (short for Extrusion Corner Frame). The frame name identifies the new frame in the Solid block visualization pane. It also appears as the frame port label on the Solid block.

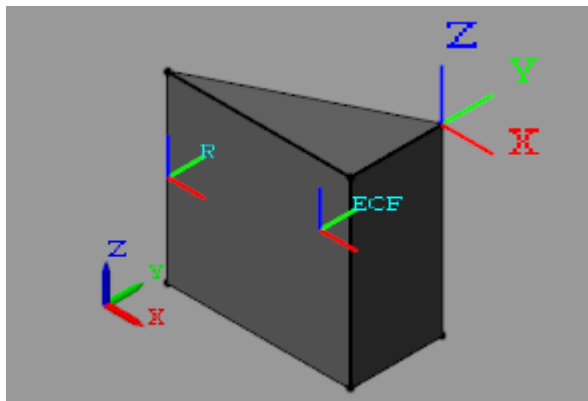
### Specify Frame Origin

Under **Frame Origin**, select **At Center of Mass**. The visualization pane updates with the new frame at the center of mass of the solid. This frame has the default frame orientation, that of the reference frame. The label ECF identifies the new frame.



Experiment with other frame origin locations. Define the origin location using one of the extrusion vertices.

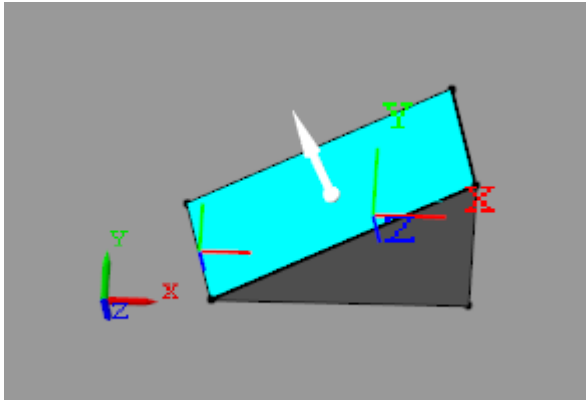
- 1 Under **Frame Origin**, select **Based on Geometric Feature**. This option enables you to select a point or the center of a plane or line as the frame origin.
- 2 In the visualization pane, select the vertex shown in the figure. The vertex is in the top plane of the extrusion. Ensure the view point is set to **Isometric**. In the **Frame Origin** area, ensure the vertex is named **Location of top point 3**.
- 3 Under **Frame Origin**, select the **Use Selected Feature** button. The visualization pane updates with the frame origin at the selected corner.



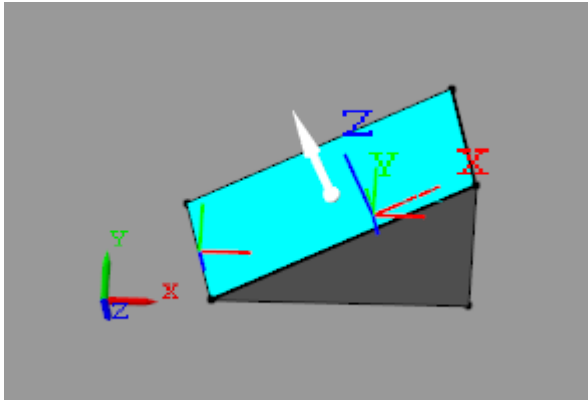
### Specify Primary Axis

The primary axis constrains the remaining two axes to lie on its normal plane. In this sense, the primary axis plays the dominant role in setting the orientation of the frame. Make the primary axis normal to the surface that contains the cross-section hypotenuse:

- 1 In the **Frame Axes** area under **Primary Axis**, select **Based on Geometric Feature**. The direction you specify in the next steps is that of the default primary axis, +Z.
- 2 In the visualization pane, rotate the solid and select the surface shown. The visualization pane highlights the surface and shows its normal vector. In the **Frame Axes** area under **Primary Axis**, ensure the surface is named **Surface normal of side surface 3**.



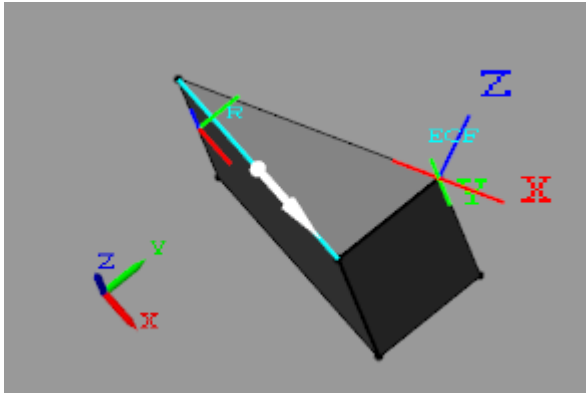
- 3 In the **Frame Axes** area under **Primary Axis**, select the **Use Selected Feature** button. The visualization pane updates with the z axis of the ECF frame, shown in dark blue, parallel to the normal vector of the selected surface.



## Specify Secondary Axis

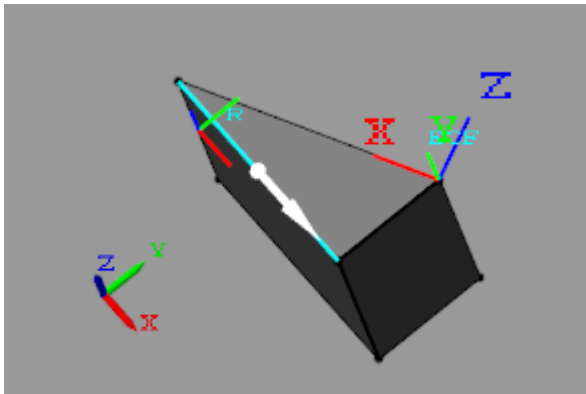
The secondary axis completes the definition of the new frame. In conjunction with the primary axis, the secondary axis fully constrains the direction of the third axis. The secondary axis is itself constrained to lie on the normal plane of the primary axis. To see the effects of this constraint, define the secondary axis based on a line not normal to the primary axis:

- 1 In the **Frame Axes** area, set the **Secondary Axis** parameter to **-X**. The direction you specify in the following steps is that of the **-X** axis.
- 2 In the **Frame Axes** area, under **Secondary Axis**, select **Based on Geometric Feature**.
- 3 In the visualization pane, rotate the solid and select the line shown. In the **Frame Axes** area, under **Secondary Axis**, ensure this line is named **Curve direction of top curve 1**.



- 4 Select the **Use Selected Feature** button. The visualization pane updates with the x axis of the frame, shown in red, partially aligned with the selected line.

The two are not completely aligned as the selected line does not lie on the normal plane of the primary axis. The secondary axis is therefore the projection of the selected line onto the normal plane of the primary axis.

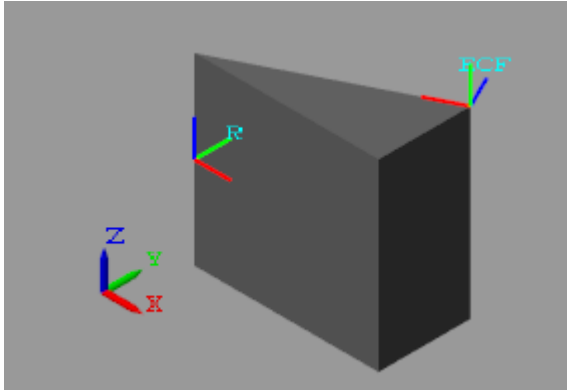


### Save New Frame

To save the frame you defined and commit it to the model:

- 1 Select the **Save** button. The visualization pane shows the solid with the final version of the frame you defined.

- 2 In the main interface of the Solid block dialog box, select **OK** or **Apply**. The Solid block commits the new frame to the model and exposes a new frame port labeled with the frame name you specified.



## Solid Visual Properties

### In this section...

“Visual-Property Parameterizations” on page 2-44

“RGB and RGBA Vectors” on page 2-45

“Simple Visual Properties” on page 2-46

“Advanced Visual Properties” on page 2-46

“Adjust Solid Opacity” on page 2-47

“Adjust Highlight Color” on page 2-48

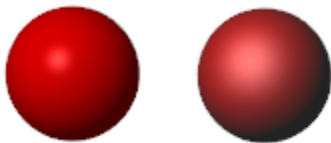
“Adjust Shadow Color” on page 2-48

“Adjust Self-Illumination Color” on page 2-49

### Visual-Property Parameterizations

The **Solid** block provides two parameterizations, **Simple** and **Advanced**, that you can use to specify solid visual properties. The **Simple** parameterization provides control over the solid color and opacity. The **Advanced** parameterization adds control over highlight, shadow, and self-illumination colors. You can use the **Advanced** visual properties to model emissive solids such as the sun and glossy solids such as polished metal parts.

You select a visual-property parameterization through the **Graphic > Visual Properties** block parameter. The figure contrasts the two parameterizations. On the left is a solid with **Simple** visual properties. On the right is the same solid with **Advanced** visual properties—including **Specular Color** and **Shininess** parameters, which impart to the solid a slight metallic sheen.



### Try It

Set the visual-property parameterization of a solid to **Advanced**:



- 1 Add a Solid block to a new model canvas. The block provides its own visualization pane. You can use this pane to visualize the solid even if the model is not topologically valid.
- 2 Set the **Geometry > Shape** parameter to **Sphere**. In later examples, the curved spherical surface makes the specular highlights and ambient shadows easier to see.
- 3 In the Solid block dialog box, set the **Graphic > Visual Properties** parameter to **Advanced**. You set the visual-property parameterization individually for each solid.

## RGB and RGBA Vectors

You can specify colors directly as [R,G,B] and [R,G,B,A] vectors on a normalized scale of 0–1. The R, G, and B elements provide the red, green, and blue contents of the specified color. The A element provides the color opacity—the degree to which the solid obstructs other components located behind it. Omitting the A element is equivalent to setting its value to 1.

### Try It

Identify the color and opacity given by the [R,G,B,A] vectors below:

- [0,0,1,1] — Denotes a solid color with no red or blue contents, maximum green content, and maximum opacity. The solid is bright green and fully opaque.



- [1,0,0,0.5] — Denotes a solid color with maximum red content, no green or blue contents, and partial opacity. The solid is bright red and transparent.



## Simple Visual Properties

The **Simple** parameterization enables you to set the solid color and opacity. You can select a color using an interactive color picker or specify a color as an [R,G,B] vector. The **Color** parameter in the **Simple** parameterization is the same as the **Diffuse Color** parameter in the **Advanced** parameterization.

### See It

See the parameters that comprise the **Simple** parameterization:

- 1 In the Solid block dialog box, set the **Visual Properties** parameter to **Simple**. This setting is the parameter default.
- 2 Expand the **Visual Properties** node. **Color** and **Opacity** appear as the active visual property parameters.

Graphic	
Type	From Geometry
Visual Properties	Simple
Color	[0.5 0.5 0.5]
Opacity	1.0

## Advanced Visual Properties

The **Advanced** parameterization adds control over the highlight, shadow, and self-illumination colors as well as the size of the highlight areas. You must specify the colors directly as [R,G,B,A] vectors. The optional A element serves the same purpose as the **Opacity** parameter in the **Simple** parameterization.

### See It

See the parameters that comprise the **Advanced** parameterization:

- 1 In the Solid block dialog box, set the **Visual Properties** parameter to **Advanced**.
- 2 Expand the **Visual Properties** node. The solid colors and shininess appear as the active visual property parameters.


Graphic	
Type	From Geometry ▼
Visual Properties	Advanced ▼
Diffuse Color	[0.5 0.5 0.5]
Specular Color	[0.5 0.5 0.5 1.0]
Ambient Color	[0.15 0.15 0.15 1.0]
Emissive Color	[0.0 0.0 0.0 1.0]
Shininess	75

## Adjust Solid Opacity

You can make a solid transparent using either parameterization. If using the **Simple** parameterization, set the **Opacity** parameter to a value less than one. If using the **Advanced** parameterization, set the optional fourth element of the **Diffuse Color** [R,G,B,A] vector to a value less than one.

### Try It

Model a transparent red solid using the **Advanced** visual-property parameterization:

- 1 Under the **Graphic > Visual Properties** node, change the **Diffuse Color** parameter to [1,0,0,0.5].
- 2 In the visualization pane, click the  button to refresh the solid visualization.



Delete the fourth vector element in the **Diffuse Color** parameter or set its value to 1 in order to make the solid opaque again.




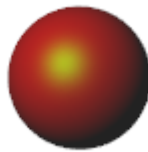
### Adjust Highlight Color

You can control the size and color of specular highlights by adjusting the **Shininess** and **Specular Color** parameters in the **Advanced** visual-property parameterization. Lower the shininess value for large but soft highlights. Increase its value for small but sharp highlights.

#### Try It

Give the specular highlights a bright green hue. Set the **Diffuse Color** vector to  $[1, 0, 0, 1]$  in order to make the solid opaque. Then:

- 1 In the **Graphic > Visual Properties** node, lower the **Shininess** parameter to 10. This value increases the highlight size, making the specular color easier to see.
- 2 Change the **Specular Color** parameter to  $[0, 1, 0, 1]$ . This vector sets the highlight color to bright green.
- 3 In the visualization pane, click the  button to refresh the solid visualization. The specular color combines with the diffuse color to give highlight areas a green hue.




### Adjust Shadow Color

You can control the color of shadow areas by adjusting the **Ambient Color** parameter in the **Advanced** visual-property parameterization.

### Try It

Give the shadow areas a slight blue hue:

- 1 In the **Graphic > Visual Properties** node, set the **Ambient Color** parameter to  $[0.15, 0.15, 0.3]$ . This vector sets the shadow color to dark blue.
- 2 In the visualization pane, click the  button to refresh the solid visualization. The ambient color combines with the diffuse color to give shadow areas a blue hue.




### Adjust Self-Illumination Color

You can model self-illuminating solids such as the sun by adjusting the **Emissive Color** parameter in the **Advanced** visual-property parameterization.

### Try It

Give the solid surface a red emissive color:

- 1 Under the **Graphic > Visual Properties** node, change the **Emissive Color** parameter to  $[1, 0, 0, 1]$ .
- 2 In the visualization pane, click the  button to refresh the solid visualization.



### **More About**

- “Solid Geometry” on page 2-10
- “Solid Inertia” on page 2-20
- “Simscape Multibody Bodies” on page 2-4

## Model a Solid of Revolution

### In this section...

“Tutorial Overview” on page 2-51

“Obtain the Cross-Section Coordinates” on page 2-51

“Create the Dome Model” on page 2-52

“Specify the Dome Properties” on page 2-53

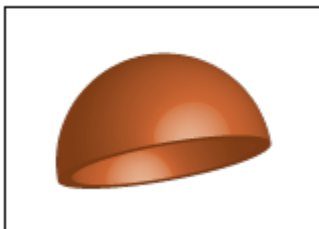
“Visualize the Dome Geometry” on page 2-54

### Tutorial Overview

This tutorial shows how to model a solid with a rotational axis of symmetry using the **Solid** block **Revolution** shape. The solid considered is a circular dome with a quarter circle for cross section. You model the dome geometry by specifying the cross-section coordinates in the solid  $xz$  plane. The **Solid** block then obtains the dome geometry by revolving the cross section specified about the reference frame  $z$  axis.

In this tutorial, you:

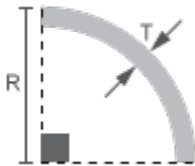
- 1 Parameterize the cross-section coordinates in terms of relevant dimensions.
- 2 Specify the cross-section coordinates in the **Solid** block dialog box.
- 3 Generate the dome geometry in the **Solid** block visualization pane.



### Obtain the Cross-Section Coordinates

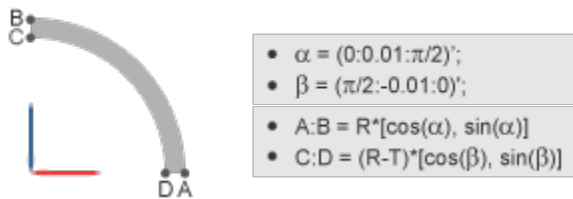
To represent the dome geometry, first identify its cross-section shape. This is the 2-D shape that Simscape Multibody revolves to obtain the 3-D dome. You can then specify the

cross-section coordinates in the **Solid** block dialog box. These coordinates must satisfy certain restrictions. See “Revolution and General Extrusion Shapes” on page 2-16.



### Dome Cross Section

The [0 0] cross-section coordinate identifies the reference frame origin for this solid. To place the solid reference frame at the dome base center, you specify the coordinates so that the [0 0] coordinate coincides with the base center. By parameterizing the cross-section coordinates in terms of the relevant dome dimensions, you can quickly change the dome dimensions without having to reenter the cross-section coordinates. The figure shows the parameterized cross-section coordinate points.



To define the dome cross-section, first define two angle arrays—one in counterclockwise order, running from 0–90°; the other in a clockwise order running from 90–0°. You can then use the first array to define the outer cross-section coordinates, and the second array to define the inner cross-section coordinates. You do that using the MATLAB `cos` and `sin` functions.

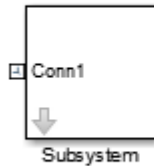
### Create the Dome Model

- 1 At the MATLAB command prompt, enter `smnew`. A new Simscape Multibody model opens with some commonly used blocks. Delete all but the **Solid** block.
- 2 In the **Solid** block dialog box, specify the following parameters. You later initialize the different MATLAB variables in a subsystem mask.




Parameter	Select or Enter
Geometry > Shape	Revolution
Geometry > Cross-Section	CS, units of cm
Inertia > Density	Rho
Graphic > Visual Properties > Color	RGB

- 3 Select the Solid block and generate a subsystem, e.g., by pressing **Ctrl+G**.



## Specify the Dome Properties

- 1 Select the Subsystem block and create a subsystem mask, e.g., by pressing **Ctrl+M**.
- 2 In the **Parameters & Dialog** tab of the Mask Editor, drag four Edit boxes  into the **Parameters** group and specify these parameters.

Prompt	Name
Base Radius	R
Wall Thickness	T
Density	Rho
Color	RGB

- 3 In the **Initialization** tab of the Mask Editor, define the cross-section coordinates and assign them to the MATLAB variable **CS**:

```
% Circular dome outer coordinates:
```

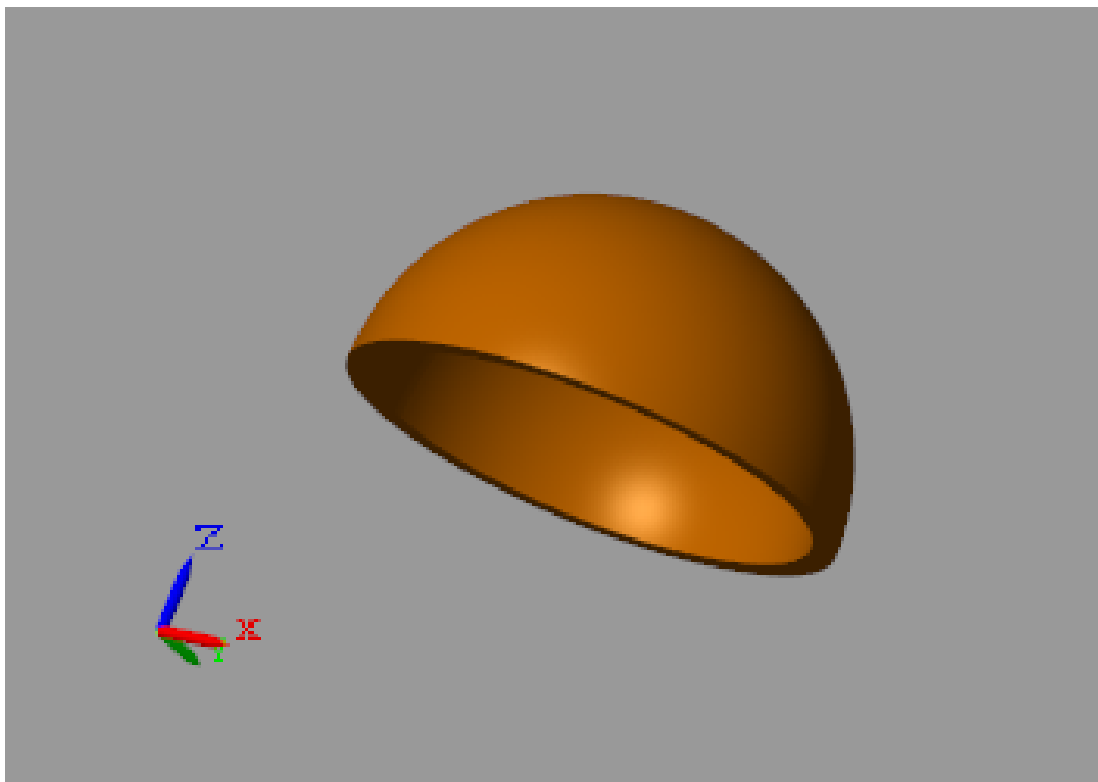
```
Alpha = (0:0.01:pi/2)';  
OuterCS = R*[cos(Alpha), sin(Alpha)];  
  
% Circular dome inner coordinates:  
Beta = (pi/2:-0.01:0)';  
InnerCS = (R-T)*[cos(Beta), sin(Beta)];  
  
CS = [OuterCS; InnerCS];
```

- 4 In the Subsystem block dialog box, specify the numerical values of the solid properties. The table shows some values that you can enter.

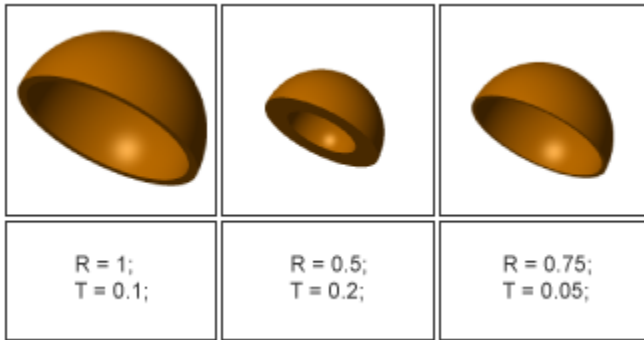
Parameter	Enter
Base Radius	1
Wall Thickness	0.1
Density	2700
Color	[0.85 0.45 0]

### Visualize the Dome Geometry

You can now visualize the dome solid. To do this, look under the Subsystem mask—e.g., by selecting the Subsystem block and pressing **Ctrl+U**—and open the Solid block dialog box. The solid visualization pane shows the solid that you modeled.



Parameterizing the solid dimensions in terms of MATLAB variables enables you to modify the solid shape without having to redefine its cross-section coordinates. You can change the solid size and proportions simply by changing their values in the Subsystem block dialog box. The figure shows some examples.



# Model an Extruded Solid

## In this section...

“Model Overview” on page 2-57

“Modeling Approach” on page 2-57

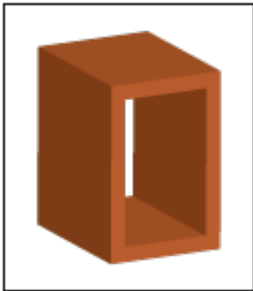
“Build Solid Model” on page 2-58

“Define Solid Properties” on page 2-59

“Visualize Solid Model” on page 2-60

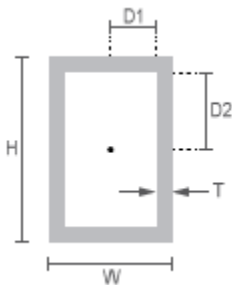
## Model Overview

You can model an extrusion with a hole. One example is the box beam. Specifying hollow cross-sections must satisfy the cross-section guidelines. See “Revolution and General Extrusion Shapes” on page 2-16. In this example, you specify the cross-section coordinates of a box beam.

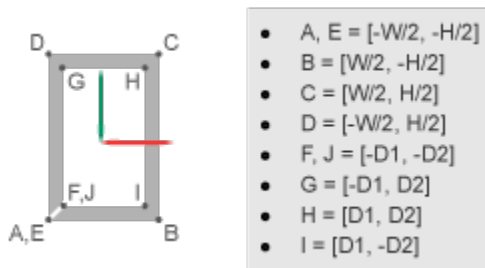


## Modeling Approach

To represent the box beam geometry, first identify its cross-section. This is the 2-D area that you sweep along an axis to obtain the 3-D box beam. You can then specify the cross-section coordinates using the **Solid** block. The figure shows the box beam cross-section that you specify in this example.



The [0 0] coordinate identifies the solid reference frame origin. To place the reference frame at the center of the box beam, specify the coordinates so that the [0 0] coordinate is at the cross-section center. By parameterizing the cross-section coordinates in terms of relevant box beam dimensions, you can later change the box beam dimensions without having to reenter the cross-section coordinates. The figure shows the cross-section dimensions and coordinates that you must specify to represent the box beam.



Using the cross-section points that the figure shows, you define the coordinate matrix as:

```
OuterCS = [A, B, C, D, E];
```

```
InnerCS = [F, G, H, I, J];
```

```
CS = [OuterCS; InnerCS];
```

For more information about specifying the hollow cross-section coordinates, see “Hollow Cross Sections” on page 2-17.

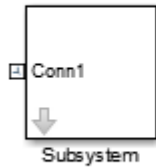
### Build Solid Model

- 1 At the MATLAB command prompt, enter `smnew`. A new Simscape Multibody model opens with some commonly used blocks. Delete all but the `Solid` block.

- In the Solid block dialog box, specify the following parameters. You later initialize the different MATLAB variables in a subsystem mask.


Parameter	Select or Enter
Geometry > Shape	General Extrusion
Geometry > Cross-Section	CS, units of cm
Geometry > Length	L, units of cm
Inertia > Density	Rho
Graphic > Visual Properties > Color	RGB

- Select the Solid block and generate a new subsystem, e.g., by pressing **Ctrl+G**.



## Define Solid Properties

In the subsystem mask, initialize the solid parameters. Then, in the subsystem dialog box, specify their values.

- Select the Subsystem block and create a subsystem mask, e.g., by pressing **Ctrl+M**.
- In the **Parameters & Dialog** tab of the Mask Editor, drag six Edit boxes  into the **Parameters** group and specify these parameters.

Prompt	Name
Length	L
Height	H

Prompt	Name
Width	W
Thickness	T
Density	Rho
Color	RGB

- 3 In the **Initialization** tab of the Mask Editor, define the cross-section coordinates and assign them to MATLAB variable CS:

```
D1 = W/2-T;
D2 = H/2-T;
OuterCS = [-W/2,-H/2; W/2,-H/2; W/2,H/2; ...
-W/2,H/2; -W/2,-H/2];
InnerCS = [-D1,-D2; -D1,D2; D1,D2; D1 -D2; -D1,-D2];
CS = [OuterCS; InnerCS];
```

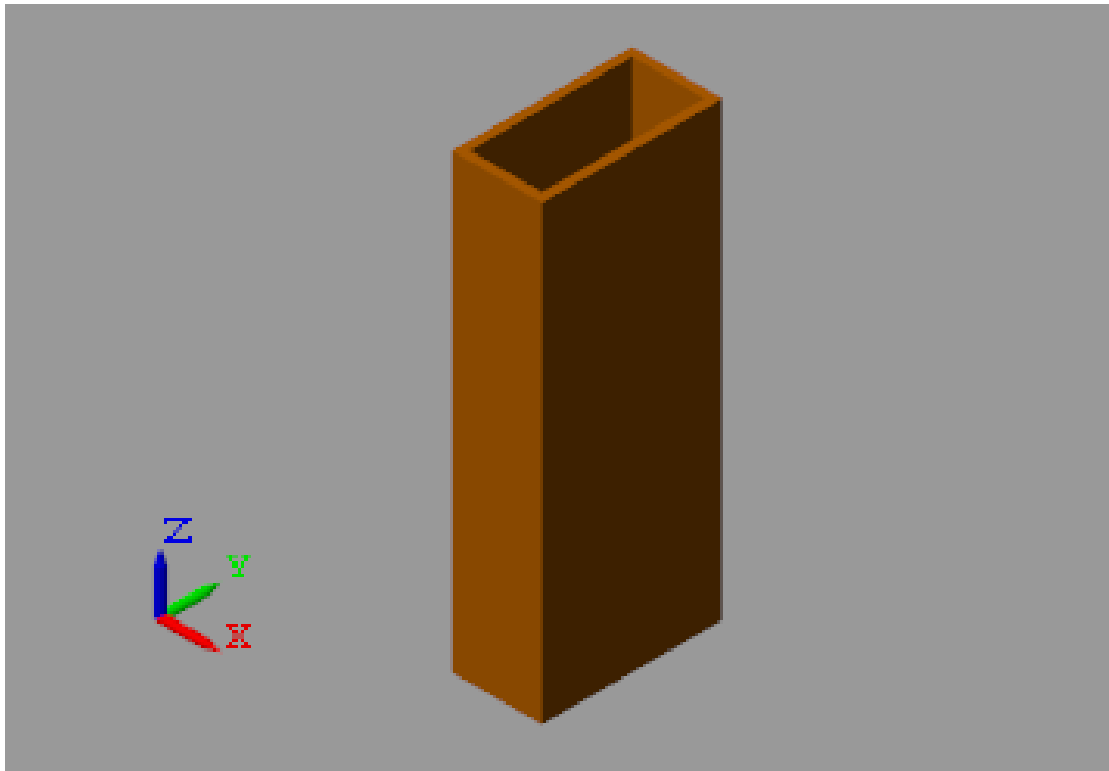
- 4 In the Subsystem block dialog box, specify the numerical values of the solid properties. The table shows some values that you can enter.

Parameter	Enter
Length	10
Height	4
Width	2
Thickness	0.2
Density	2700
Color	[0.85 0.45 0]

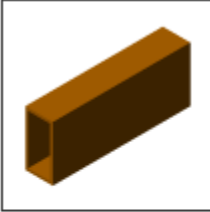
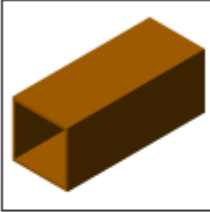
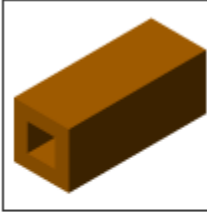
## Visualize Solid Model

You can now visualize the box-beam solid. To do this, look under the Subsystem mask —e.g., by selecting the Subsystem block and pressing **Ctrl+U**—and open the Solid block dialog box. The solid visualization pane shows the solid that you modeled.





Parameterizing the solid dimensions in terms of MATLAB variables enables you to modify the solid shape without having to redefine its cross-section coordinates. You can change the solid size and proportions simply by changing their values in the Subsystem block dialog box. The figure shows some examples.

		
$L = 10;$ $h = 4;$ $w = 2;$ $t = 0.3;$	$L = 10;$ $h = 4;$ $w = 4;$ $t = 0.2;$	$L = 10;$ $h = 4;$ $w = 4;$ $t = 1;$

## Model a Compound Body

### In this section...

“Model Overview” on page 2-63

“Build Model” on page 2-63

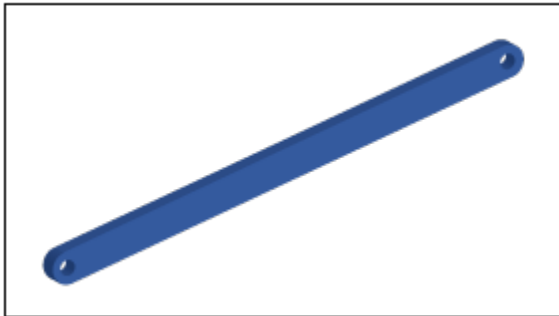
“Generate Subsystem” on page 2-65

“Visualize Model” on page 2-67

“Open Reference Model” on page 2-67

### Model Overview

In this example, you model a two-hole binary link as a rigid body. Three Solid blocks represent the main body and hole sections of the link. Two Rigid Transform blocks define the spatial relationships between the three solids.



### Build Model

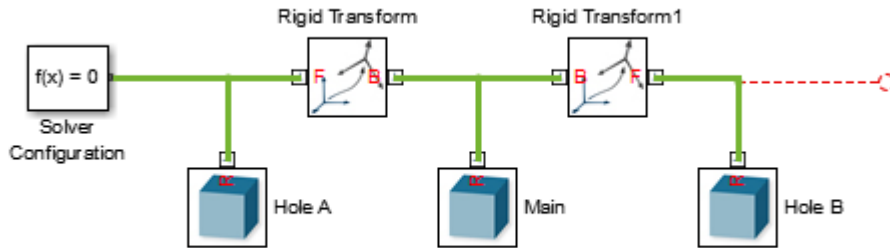
- 1 Start a new model.
- 2 Drag the following blocks to the model.

Library	Block	Quantity
Simscape > Utilities	Solver Configuration	1

Library	Block	Quantity
Simscape > Multibody > Frames and Transforms	Rigid Transform	2
Simscape > Multibody > Body Elements	Solid	3

3 Connect and name the blocks as shown in the figure.

Be sure to flip the Rigid Transform block. Its B frame port must face the Main Solid block. Also include the broken line extending from the Hole B block (right click the existing connection line and drag).



4 In the solid block dialog boxes, specify these parameters.

Parameter	Hole A	Main	Hole B
Geometry > Shape	Select General Extrusion.	Select General Extrusion.	Select General Extrusion.
Geometry > Cross-section	Enter HoleACS. Select units of cm.	Enter MainCS. Select units of cm.	Enter HoleBCS. Select units of cm.
Geometry > Length	Enter T. Select units of cm.	Enter T. Select units of cm.	Enter T. Select units of cm.
Inertia > Density	Enter Rho.	Enter Rho.	Enter Rho.
Graphic > Visual Properties > Color	Enter LinkRGB.	Enter LinkRGB.	Enter LinkRGB.

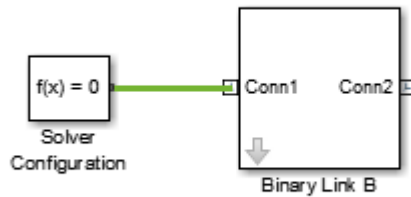
- 5 In the rigid transform block dialog boxes, specify these parameters.

Parameter	Rigid Transform	Rigid Transform 1
Translation > Method	Select Standard Axis.	Select Standard Axis.
Translation > Axis	Select +X.	Select +X.
Translation > Offset	Enter $-L/2$ . Select units of cm.	Enter $+L/2$ . Select units of cm.


## Generate Subsystem

Enclose the binary link blocks in a Subsystem block, define the general extrusion coordinates, and specify the relevant parameter values:

- 1 Select all blocks excluding Solver Configuration and press **Ctrl+G**. Simulink encloses the selected blocks in a new subsystem block. Rename the subsystem block as shown in the figure.



- 2 Select the subsystem block and press **Ctrl+M**. Simulink adds a parameter mask to the subsystem block.

- 3 In the **Parameters & Dialog** tab of the Mask Editor, drag six edit boxes  into the **Parameters** group and specify the following parameters.

Prompt	Name
Length	L
Width	W

Prompt	Name
Thickness	T
Peg Hole Radius	R
Mass Density	Rho
Link Color	LinkRGB

- 4 In the **Initialization** tab of the Mask Editor, define the extrusion cross sections and click **OK**:

```
% Cross-section of Main:
Alpha = (pi/2:-0.01:-pi/2)';
Beta = (3*pi/2:-0.01:pi/2)';

EndACS = [-L/2 W/2; -L/2+R*cos(Alpha)...
R*sin(Alpha); -L/2 -W/2];

EndBCS = [L/2 -W/2; L/2+R*cos(Beta)...
R*sin(Beta); L/2 W/2];

MainCS = [EndACS; EndBCS];

% Cross-section of HoleA:
Alpha = (pi/2:0.01:3*pi/2)';
Beta = (3*pi/2:-0.01:pi/2)';
HoleACS = [W/2*cos(Alpha) W/2*sin(Alpha);...
R*cos(Beta) R*sin(Beta)];


% Cross-section of HoleB:
Alpha = (-pi/2:0.01:pi/2)';
Beta = (pi/2:-0.01:-pi/2)';
HoleBCS = [W/2*cos(Alpha) W/2*sin(Alpha);...
R*cos(Beta) R*sin(Beta)];
```

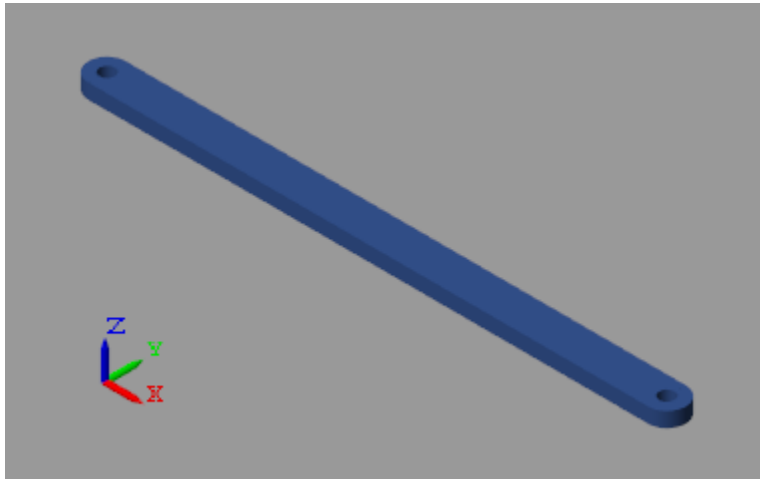
- 5 In the dialog box of the Binary Link B subsystem block, specify these parameters.

Parameter	Value
Length	30
Width	2
Thickness	0.8
Peg Hole Radius	0.4

Parameter	Value
Mass Density	2700
Link Color [R G B]	[0.25 0.4 0.7]

## Visualize Model

Update the block diagram. You can do this by pressing **Ctrl+D**. Mechanics Explorer opens with a static display of the binary link rigid body. To obtain the view shown in the figure, in the Mechanics Explorer toolstrip select the isometric view button .



You can open a copy of the resulting model. At the MATLAB command line, enter `smdoc_binary_link_b`.

## Open Reference Model

To open a completed version of the binary link model, at the MATLAB command prompt, enter `smdoc_binary_link_b`.

## Related Examples

- “Model a Closed-Loop Kinematic Chain” on page 3-21





# Multibody Systems

---

- “Assembling Multibody Systems” on page 3-2
- “Joint Connections” on page 3-4
- “Multibody Assembly” on page 3-9
- “Mechanism Degrees of Freedom” on page 3-15
- “Model an Open-Loop Kinematic Chain” on page 3-16
- “Model a Closed-Loop Kinematic Chain” on page 3-21
- “Troubleshoot Assembly Errors in Aiming Mechanism Model” on page 3-27
- “Gear Constraints” on page 3-36
- “Model a Rack-and-Pinion Constraint” on page 3-45
- “Model a Common Gear Constraint” on page 3-57
- “Model a Cam Constraint” on page 3-73

## Assembling Multibody Systems

### In this section...

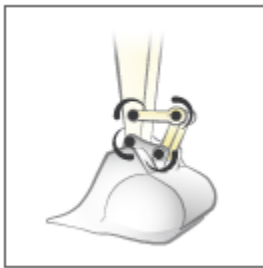
“Step 1: Study the Joints and Constraints to Model” on page 3-2

“Step 2: Assemble Bodies Using Joints and Constraints” on page 3-2

“Step 3: Guide Model Assembly” on page 3-3

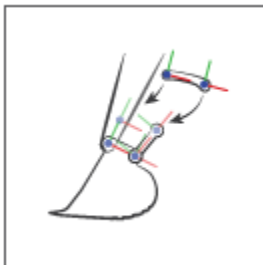
“Step 4: Verify Model Assembly” on page 3-3

### Step 1: Study the Joints and Constraints to Model



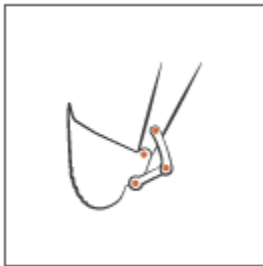
Identify the joints and constraints between the various bodies. Joints can be real, such as that between a piston and its case, or virtual, such as that between two planets. See “Model a Closed-Loop Kinematic Chain” on page 3-21 for an example.

### Step 2: Assemble Bodies Using Joints and Constraints



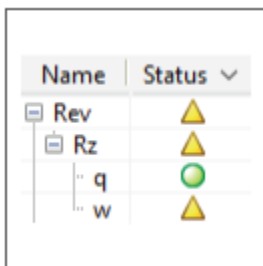
Model the degrees of freedom between bodies by connecting their frames through joints. You can further constrain these degrees of freedom through specialized constraints, such as those between gears. See “Model a Rack-and-Pinion Constraint” on page 3-45 for an example.

### Step 3: Guide Model Assembly



Specify the state targets of the various joints. You can specify the desired position and velocity of a joint at time zero. If the state targets are valid and compatible, the joints assemble in the states specified. See the “Guide Assembly and Visualize Model” on page 3-24 section of “Model a Closed-Loop Kinematic Chain” on page 3-21 for an example.

### Step 4: Verify Model Assembly

A screenshot of a table showing the status of joints in a model. The table has two columns: 'Name' and 'Status'. The 'Status' column contains icons: a yellow triangle for 'Rev', 'Rz', and 'w', and a green circle for 'q'.

Name	Status
Rev	▲
Rz	▲
q	●
w	▲

Update the block diagram. Examine the model visualization for assembly issues. Open the Simscape Variable Viewer or the Simscape Multibody Model Report to see if all state targets have been satisfied. See the “Verify Model Assembly” on page 3-25 section of “Model a Closed-Loop Kinematic Chain” on page 3-21 for an example.

## Joint Connections

### In this section...

“Role of Joints in a Model” on page 3-4

“Joint Degrees of Freedom” on page 3-4

“Joint Primitives” on page 3-6

“Joint Inertia” on page 3-8

### Role of Joints in a Model

Joints impose between bodies the primary kinematic constraints that determine how they can move relative to each other. A joint can be a physical connection, such as that between the case and shaft of a linear hydraulic actuator, or a virtual connection, such as that between the Earth and the moon. In Simscape Multibody, you model both connection types using Joint blocks.



### Examples of physical and virtual connections between bodies

Gear and Constraint blocks too impose kinematic constraints between bodies. How are joint blocks different? While Gear and Constraint blocks are parameterized in terms of the DoFs they remove between bodies, Joint blocks are parameterized in terms of the DoFs they provide, through modules called joint primitives.

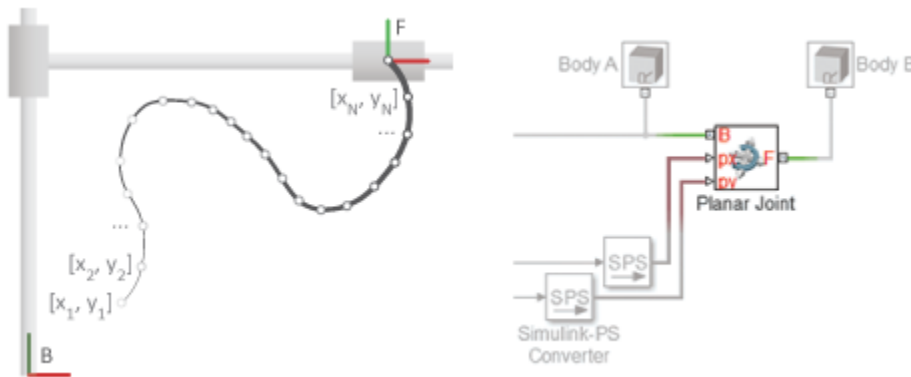
### Joint Degrees of Freedom

Each Joint block connects exactly two bodies. Such a connection determines the maximum degrees of freedom, or DoFs, that the adjoining bodies can share. These DoFs

range from zero in the Weld Joint block to six—three translational and three rotational—in 6-DOF Joint and Bushing Joint blocks. Translation refers to a change in position and rotation to a change in orientation.

Joint DoFs are a measure of joint mobility. Precluding other constraints in a model, a joint with more DoFs allows greater freedom of motion between the adjoining bodies. Joint DoFs also have a mathematical interpretation. They are the minimum number of state variables needed to fully determine the configuration of a joint at each time step during simulation.

Consider a rectangular joint. This joint allows translation in a plane and it therefore has two translational DoFs—one for each spatial dimension. At each time step, the joint configuration is fully determined by two state variables, the position coordinates in the plane of motion  $[x(t), y(t)]$ . This means, for example, that you can fully prescribe motion at this joint using two position input signals.



The table summarizes the DoFs that the various Joint blocks provide.

Joint Block	Translational DoFs	Rotational DoFs	Total DoFs
6-DOF Joint	3	3	6
Bearing Joint	1	3	4
Bushing Joint	3	3	6
Cartesian Joint	3	0	3
Constant Velocity Joint	0	2	2
Cylindrical Joint	1	1	2
Gimbal Joint	0	3	3
Leadscrew Joint	1 (coupled translational-rotational)		1
Pin Slot Joint	1	1	2
Planar Joint	2	1	3
Prismatic Joint	1	0	1
Rectangular Joint	2	0	2
Revolute Joint	0	1	1
Spherical Joint	0	3	3
Telescoping Joint	1	3	4
Universal Joint	0	2	2
Weld Joint	0	0	0

The actual DoFs at a joint are often fewer in number than the joint alone would allow. This happens when kinematic constraints elsewhere in the model limit the relative motion of the adjoining bodies. Such constraints can arise from gears in mesh, forbidden DoFs due to other joints in closed kinematic loops, and fixed distances and angles between bodies, among other factors.

### Joint Primitives

Joint blocks are assortments of joint primitives, basic yet complete joints of various kinds you cannot decompose any further—at least without losing behavior such as the rotational-translational coupling of the lead screw joint. Joint primitives range in number from zero in the Weld Joint block to six in the Bushing Joint block. There are five joint primitives:

- Prismatic — Allows translation along a single standard axis (x, y, or z). Joint blocks can contain up to three prismatic joint primitives, one for each translational DoF. Prismatic primitives are labeled P\*, where the asterisk denotes the axis of motion, e.g., Px, Py, or Pz.

- Revolute — Allows rotation about a single standard axis (x, y, or z). Joint blocks can contain up to three revolute joint primitives, one for each rotational DoF. Revolute primitives are labeled R\*, where the asterisk denotes the axis of motion, e.g., Rx, Ry, or Rz.
- Spherical — Allows rotation about any 3-D axis, [x, y, z]. Joint blocks contain no more than one spherical primitive, and never in combination with revolute primitives. Spherical primitives are labeled S.
- Lead Screw Primitive — Allows coupled rotation and translation on a standard axis (e.g., z). This primitive converts between rotation at one end and translation at the other. Joint blocks contain no more than one lead screw primitive. Lead screw primitives are labeled LS\*, where the asterisk denotes the axis of motion.
- Constant Velocity Joint — Allows rotation at constant velocity between intersecting though arbitrarily aligned shafts. Joint blocks contain no more than one constant velocity primitive. Constant velocity primitives are labeled CV.

The table summarizes the joint primitives and DoFs that the various Joint blocks provide.

Joint Block	Joint Primitives								
6-DOF Joint	Px	Py	Pz	.	.	.	S	.	.
Bearing Joint	.	.	Pz	Rx	Ry	Rz	.	.	.
Bushing Joint	Px	Py	Pz	Rx	Ry	Rz	.	.	.
Cartesian Joint	Px	Py	Pz	.	.	.	.	.	.
Constant Velocity Joint	.	.	.	.	.	.	.	CV	.
Cylindrical Joint	.	.	Pz	.	.	Rz	.	.	.
Gimbal Joint	.	.	.	Rx	Ry	Rz	.	.	.
Leadscrew Joint	.	.	.	.	.	.	.	.	LSz
Pin Slot Joint	Px	.	.	.	.	Rz	.	.	.
Planar Joint	Px	Py	.	.	.	Rz	.	.	.
Prismatic Joint	.	.	Pz	.	.	.	.	.	.
Rectangular Joint	Px	Py	.	.	.	.	.	.	.
Revolute Joint	.	.	.	.	.	Rz	.	.	.
Spherical Joint	.	.	.	.	.	.	S	.	.
Telescoping Joint	.	.	Pz	.	.	.	S	.	.
Universal Joint	.	.	.	Rx	Ry	.	.	.	.
Weld Joint	.	.	.	.	.	.	.	.	.

Why use Joint blocks with spherical primitives? Those with three revolute primitives are susceptible to gimbal lock—the natural but often undesired loss of one rotational DoF when any two rotation axes become aligned. Gimbal lock leads to simulation errors due to numerical singularities. Spherical primitives eliminate the risk of gimbal-lock errors by representing 3-D rotations using 4-D quantities known as quaternions.

### Joint Inertia

Simscape Multibody joints are idealizations. They differ from real joints in that they have no inertia—a suitable approximation in most models, where the impact of joint inertia on system dynamics is often negligible. This is the case, for example, in the constant-velocity joints of automobile driveline systems, where shaft inertia can dwarf joint inertia.

If joint inertia is important in your model, you can account for it using Solid or Inertia blocks. Connect the block reference frame ports to the appropriate joint frames and specify the joint inertial properties in the block dialog boxes. You can specify joint mass or density, products of inertia, moments of inertia, and center of mass. For more information on how to specify inertia, see “Solid Inertia” on page 2-20.



# Multibody Assembly

## In this section...

“Model Assembly” on page 3-9

“Connecting Joints” on page 3-9

“Orienting Joints” on page 3-10

“Guiding Assembly” on page 3-11

“Verifying Model Assembly” on page 3-11

## Model Assembly

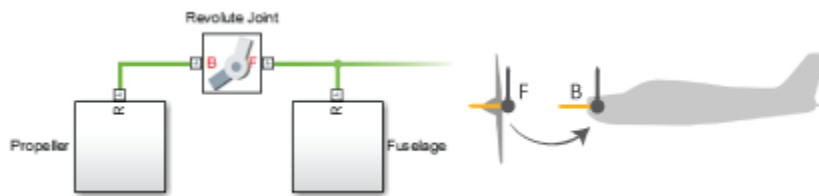
You model an articulated system by interconnecting bodies through joints and occasionally gears and other constraints. Bodies contribute their inertias to the model, while joints, gears, and constraints determine the relative degrees of freedom that exist between the bodies. You interconnect the two component types by linking frame ports on Joint, Gear, and Constraint blocks to frame ports on body subsystems.

Simscape Multibody automatically assembles your model when you update the block diagram—for example, by selecting **Simulation > Update Diagram** from the Simulink menu bar.

During model update, Simscape Multibody determines the initial states of joints—their positions and velocities—so that the resulting assembly satisfies all kinematic constraints in the model. This process occurs in two phases, with the assembly algorithm first computing the joint positions and then the joint velocities. The complete process is called model assembly.

## Connecting Joints

Joints connect to bodies through frames. Each Joint block contains two frame ports, base (B) and follower (F), identifying the connection points in the adjoining bodies and the relative directions they can move in. When you connect these ports to frames in the body subsystems, you determine how the bodies themselves connect upon model assembly.



#### Joint Frames Identifying Connection Points and Rotation Axis of Aircraft Propeller

If a joint has no actuation and no sensing outputs, its frame ports are fully interchangeable. In this case, you can switch the bodies that the ports connect to without affecting model dynamics or joint sensing outputs. If the joint does have actuation inputs or sensing outputs, you may need to reverse the actuation or sensing signals to obtain the same dynamic behavior and simulation results.

To change the connection points of a joint, you must modify the connection frames in the adjoining body subsystems. You do this by specifying a translation transform using a **Rigid Transform** block. You can add new Rigid Transform blocks to the body subsystems or, if appropriate, change the translation transforms in existing Rigid Transform subsystems.

For more information on how Simscape Multibody software interprets frame ports, nodes, and lines, see “Representing Frames” on page 1-11.

#### Orienting Joints

To obtain the motion expected in a model, you must align its various joint motion axes properly. This means aligning the joints themselves as observed or anticipated in the real system. Misaligning the joint axes may lead to unexpected motion but it often leads to something more serious, such as a failure to assemble and simulate.

You can specify and change joint alignment by rotating the connection frames local to the adjoining body subsystems. For this purpose, you specify rotation transforms using Rigid Transform blocks, either by adding new blocks to the body subsystems or, if appropriate, by changing the rotation transforms in existing blocks within the subsystems.

Why change the orientation of joints through body subsystem frames? The primitives in a Joint block each have a predetermined motion axis, such as  $x$  or  $z$ . The axis definition is fixed and cannot be changed. Realigning the connection frames local to the adjoining body subsystems provides a natural way to reorient joints while avoiding confusion over which axis a particular joint uses.

## Guiding Assembly

Joints can start simulation from different states. For example, the crank joint of a crank-rocker linkage can start at any angle from  $0^\circ$  to  $360^\circ$ . As a result, during model assembly, Simscape Multibody must choose from many equally valid states. You can guide the states chosen by specifying state targets in the Joint block dialog boxes.



### Crank-Slider Mechanism in Fully Extended and Fully Retracted Initial Configurations

State targets need not be exact values. If Simscape Multibody cannot achieve a state target exactly, it searches for the joint state nearest to the state target. For example, if you specify a position state target of  $60^\circ$  but the joint can only reach angles of  $0^\circ$  to  $45^\circ$ , Simscape Multibody attempts to assemble the joint at  $45^\circ$ .

How close the actual joint state is to the state target depends on the kinematic constraints in your model, any conflicts with other state targets, and the state target priority level—a ranking that determines which of two state targets to satisfy if they prove to be mutually incompatible. You can set the priority level to **Low** or **High**.

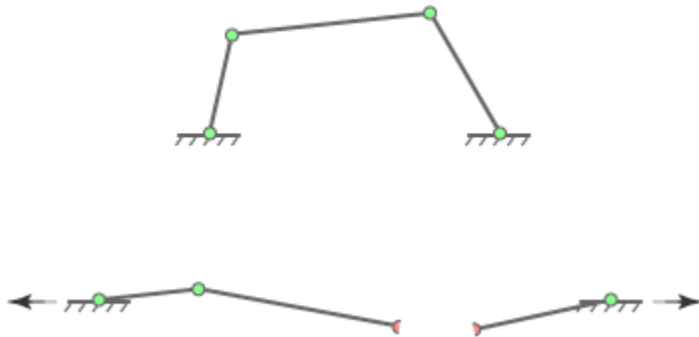
Simscape Multibody first attempts to satisfy all state targets exactly. If a state target conflict arises, Simscape Multibody ignores the low-priority state targets and attempts to satisfy only the high-priority state targets. If a state target conflict still exists, Simscape Multibody ignores also the high-priority state targets and attempts to assemble the model in the nearest valid configuration.

You can specify state targets for all joints in an open kinematic chain. However, to avoid simulation errors, every closed chain must contain at least one joint without state targets.

## Verifying Model Assembly

A model assembles successfully only if the connections between its bodies are congruous with each other. If in satisfying one kinematic constraint, Simscape Multibody must

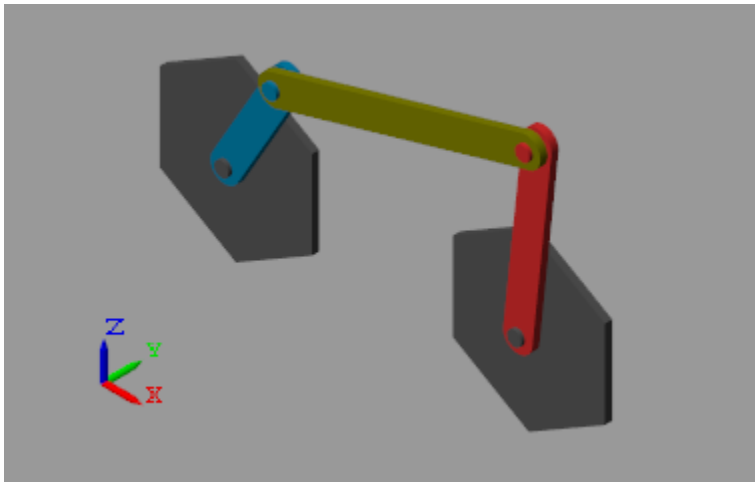
violate another kinematic constraint, the model is kinematically invalid and assembly fails. This happens, for example, when the ground link of a four-bar assembly exceeds the combined length of the remaining three links, preventing at least one joint from assembling.



#### Joint Assembly Failure in Four-Bar Linkage with Exceedingly Long Ground Link

To ensure that your model has assembled correctly, use these Simscape Multibody and Simscape utilities:

- **Mechanics Explorer** — Simscape Multibody visualization utility. Visually examine your model from different points of view to ensure that its bodies connect at the expected locations and with the proper orientations.



- Variable Viewer — Simscape state-reporting utility. Check the assembly status of individual joints and constraints and compare your state targets to the actual joint states achieved during assembly.

Name	Status	Priority	Target	Start	Unit
Base_Crank_Revolute	●				
Rz	●				
q	●	High	150.0	150.0	deg
w	●	High	-360.0	-360.0	deg/s
Base_Rocker_Revolute	●				
Rz	●				
q	●			3.0338	rad
w	●			-3.13757	rad/s
Connector_Rocker_Revolute	●				
Rz	●				
q	●			1.1814	rad
w	●			-4.35682	rad/s
Crank_Connector_Revolute	▲				
Rz	▲				
q	▲	Low	-45.0	-43.86525496465046	deg
w	●			7.50244	rad/s

▲ All high priority targets satisfied but some low priority targets not satisfied      Variables at start ▼

- Statistics Viewer — Simscape metrics-reporting utility. Check, among other metrics, the degrees of freedom, number of joints, and number of constraints in your model.

Name	Value
3-D Multibody System	
Number of rigidly connected components (excluding ...	3
Number of joints (total)	4
Number of explicit tree joints	3
Number of implicit 6-DOF tree joints	0
Number of cut joints	1
Number of constraints	0
Number of tree degrees of freedom	3
Number of position constraint equations (total)	5
Number of position constraint equations (non-redund...	2
Number of mechanism degrees of freedom (minimum)	1
State vector size	8
Average kinematic loop length	4

## Mechanism Degrees of Freedom

The number and types of joints, gears, and constraints in a mechanism partially determine its mobility—the total number of degrees of freedom, or DoFs, that the mechanism provides and therefore the minimum number of input variables needed to fully constrain its configuration. The mobility  $F$  of a mechanism with  $N$  bodies and  $j$  joints, each with  $f$  DoFs follows from expressions such as the Kutzbach criterion, which for a planar mechanism states:

$$F = 3(N - 1) - \sum_{i=1}^j (3 - f_i)$$

Applying this criterion to a four-bar linkage, an assembly of four bodies ( $n = 4$ ) and four joints ( $j = 4$ ) with one rotational DoF each ( $f_i = 1$ ), yields a mobility of one DoF—indicating that a single input variable suffices to fully control the linkage configuration. As mechanisms grow in complexity, manually calculating total DoFs becomes more time-consuming, so Simscape Multibody automatically computes them for you.

You can view the mechanism DoFs through the Simscape Statistics Viewer, shown below for the four-bar featured example. You open the Statistics Viewer from the Simulink Editor menu bar by selecting **Analysis > Simscape > Statistics Viewer**. Enter `sm_four_bar` at the MATLAB command prompt to open the four-bar model and view its DoFs through the Statistics Viewer.

Name	Value
Number of implicit 6-DOF tree joints	0
Number of cut joints	1
Number of constraints	0
Number of tree degrees of freedom	3
Number of position constraint equations (total)	5
Number of position constraint equations (non-redundant)	2
Number of mechanism degrees of freedom (minimum)	1
State vector size	8
Average kinematic loop length	4

## Model an Open-Loop Kinematic Chain

### In this section...

“Model Overview” on page 3-16

“Build Model” on page 3-17

“Guide Model Assembly” on page 3-18

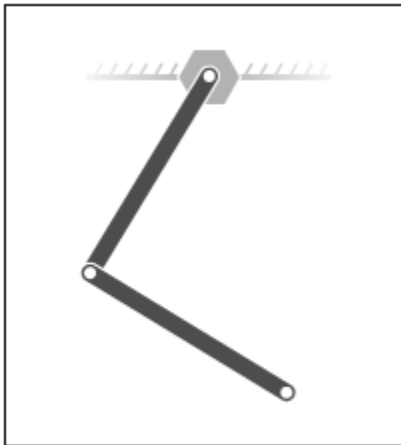
“Visualize Model and Check Assembly Status” on page 3-18

“Simulate Model” on page 3-20

“Open Reference Model” on page 3-20

### Model Overview

This example shows how to model a double pendulum—a simple kinematic chain comprising two moving bodies connected in series via two revolute joints. A third body represents a mechanical ground and is rigidly connected to the inertial World frame. The custom `smdoc_compound_rigid_bodies` library provides the rigid body subsystem blocks used in the example.



Revolute Joint blocks enable you to model the joints connecting adjacent bodies and help set their initial states. Simscape Multibody software satisfies a joint state target



precisely if it is kinematically valid and not in conflict with other state targets. A **Priority** parameter lets you specify which targets to attempt to satisfy first.

## Build Model

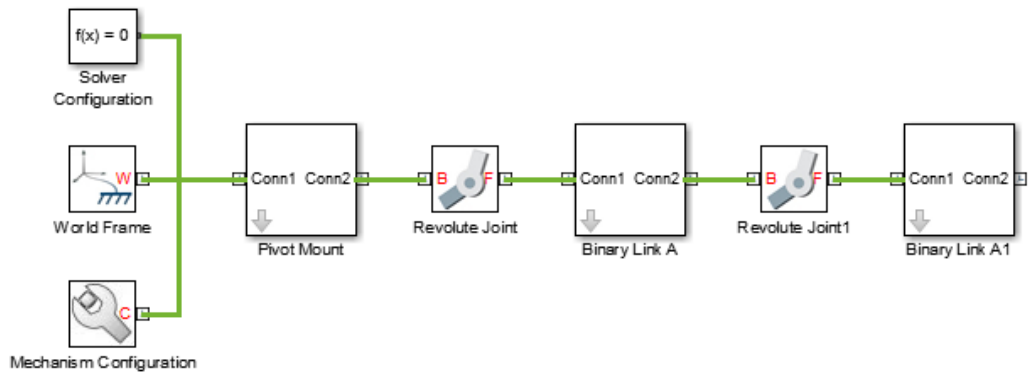
- 1 Start a new model.
- 2 Drag these blocks into the model. The two Revolute Joint blocks provide the double pendulum two rotational degrees of freedom.

Library	Block	Quantity
Simscape > Utilities	Solver Configuration	1
Simscape > Multibody > Utilities	Mechanism Configuration	1
Simscape > Multibody > Frames and Transforms	World Frame	1
Simscape > Multibody > Joints	Revolute Joint	2

- 3 At the MATLAB command prompt, enter `smdoc_compound_rigid_bodies`. A custom block library with the same name opens up.
- 4 Drag these custom blocks into the model. Each block represents a rigid body in the double pendulum.

Block	Quantity
Pivot Mount	1
Binary Link A	2

- 5 Connect the blocks as shown in the figure.



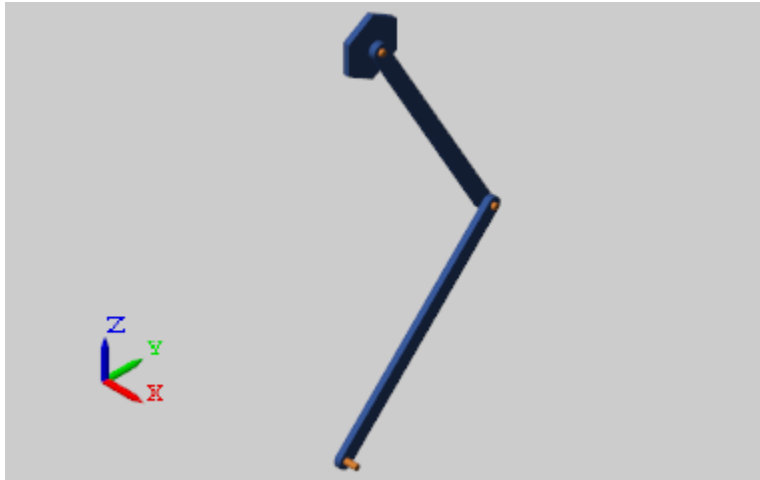
## Guide Model Assembly

- 1 In the Revolute Joint block dialog boxes, select **State Targets > Specify Position Target**. You can now specify the desired starting positions of the two joints.
- 2 In **Value**, enter these joint angles.

Block Name	Value (degrees)
Revolute Joint	30
Revolute Joint1	-75

## Visualize Model and Check Assembly Status

To visualize the model, update the block diagram. You can do this from the menu bar by selecting **Simulation > Update Diagram**. Mechanics Explorer opens with a 3-D view of the double pendulum assembly. Click the isometric view button to obtain the perspective in the figure.



To check the assembly status of the revolute joints, use the Model Report utility. You can open this utility from the Mechanics Explorer menu bar by selecting **Tools > Model Report**. The figure shows the assembly information for the double pendulum.

Model Report - double\_pendulum\_model

Assembly status: ●  
 Joints: ●  
 Constraints: ●

Joints  
  Constraints  
  Statistics

Joint	Asse...	Primit...	Position					Velocity				
			Actual	Specif...	Unit	Priority	Status	Actual	Specif...	Units	Priority	Status
Revolute_Joint	<span style="color: green;">●</span>	Rz	+30	+30	deg	High	<span style="color: green;">●</span>	+0		deg/s		
Revolute_Joint1	<span style="color: green;">●</span>	Rz	-75	-75	deg	High	<span style="color: green;">●</span>	+0		deg/s		

OK

### Simulate Model

Run the simulation, e.g., by selecting **Simulation > Run**. Mechanics Explorer shows a 3-D animation of the double pendulum assembly. The assembly moves due to gravity, specified in the Mechanism Configuration block.

### Open Reference Model

To see a complete model of the double pendulum assembly, at the MATLAB command prompt enter:

- `smdoc_double_pendulum`

## Model a Closed-Loop Kinematic Chain

### In this section...

“Model Overview” on page 3-21

“Build Model” on page 3-22

“Specify Block Parameters” on page 3-24

“Guide Assembly and Visualize Model” on page 3-24

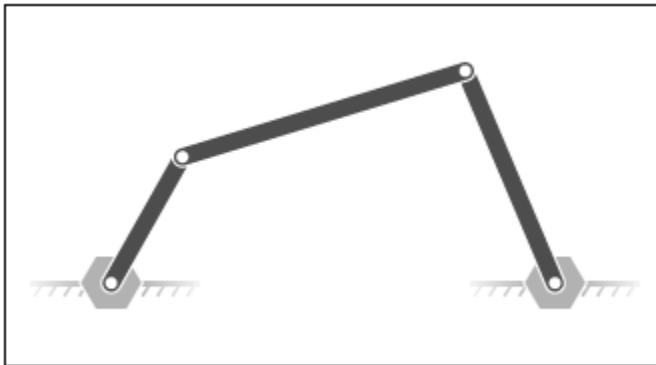
“Verify Model Assembly” on page 3-25

“Simulate Model” on page 3-26

“Open Reference Model” on page 3-26

### Model Overview

This example shows how to model a four bar—a closed kinematic chain comprising four bodies that connect through revolute joints. One of the links is fixed to the World frame and acts as a ground. This link is replaced here by two pivot mounts connected through a rigid translation transform. The custom `smdoc_compound_rigid_bodies` library provides the rigid body subsystem blocks used in the example.



Revolute Joint blocks enable you to model the joints connecting adjacent bodies and help set their initial states. Simscape Multibody software satisfies a joint state target precisely if it is kinematically valid and not in conflict with other state targets. A **Priority** parameter lets you specify which targets to attempt to satisfy first.

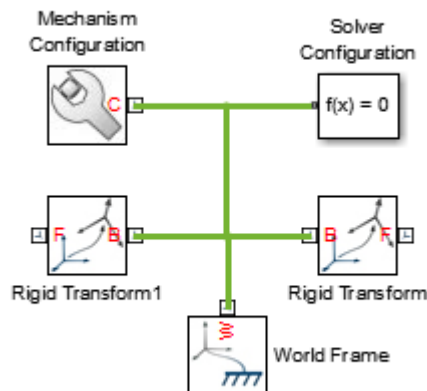
## Build Model

To model the four-bar linkage:

- 1 Start a new model.
- 2 Drag these blocks to the model. The Rigid Transform blocks specify the distance between the two pivot mounts. This distance is the length of the implicit ground link.

Library	Block	Quantity
Simscape > Utilities	Solver Configuration	1
Simscape > Multibody > Utilities	Mechanism Configuration	1
Simscape > Multibody > Frames and Transforms	World Frame	1
Simscape > Multibody > Frames and Transforms	Rigid Transform	2

- 3 Connect and name the blocks as shown in the figure. The base frame ports of the Rigid Transform blocks must connect to the World Frame block.

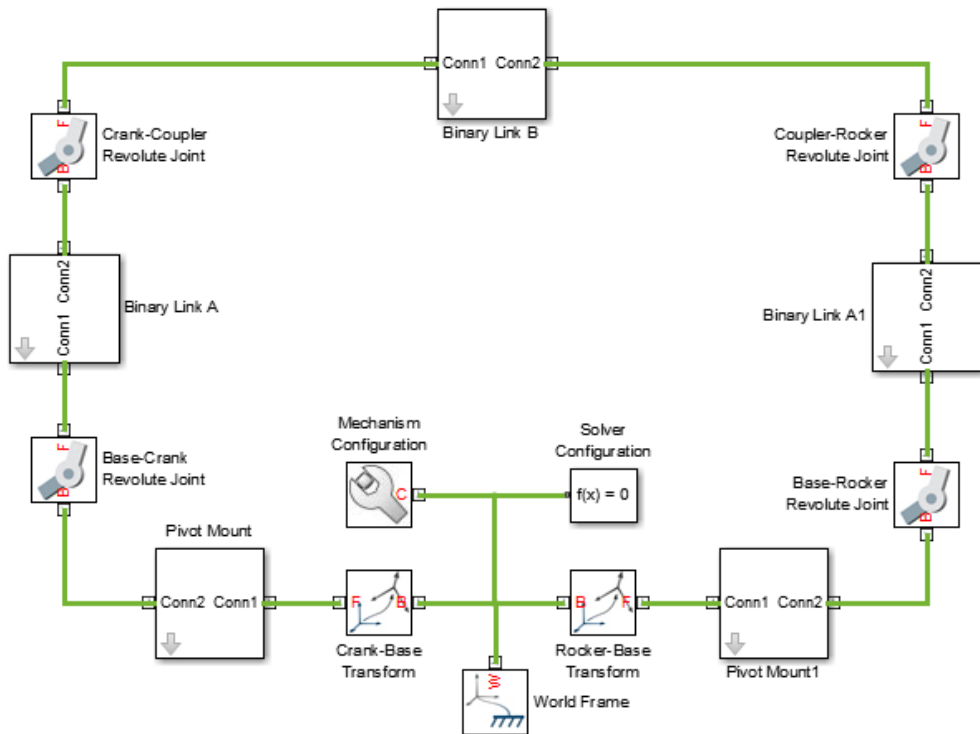


- 4 From the **Simscape > Multibody > Joints** library, drag four Revolute Joint blocks into the model.

- 5 At the MATLAB command prompt, enter `smdoc_compound_rigid_bodies`. A custom library with compound rigid body blocks opens up.
- 6 From the `smdoc_compound_rigid_bodies` library, drag these blocks. Each block represents a rigid body present in the four bar assembly.

Block	Quantity
Pivot Mount	2
Binary Link A	2
Binary Link B	1

- 7 Connect and name the blocks as shown in the figure. You must position the frame ports of the custom rigid body blocks exactly as shown.



## Specify Block Parameters

- 1 In the Rigid Transform block dialog boxes, specify the offset between the pivot mounts and the world frame. The pivot mounts are assumed to be symmetrically positioned about this frame.

Parameter	Crank-Base Transform	Rocker-Base Transform
Translation > Method	Standard Axis	Standard Axis
Translation > Axis	-Y	+Y
Translation > Offset	15 in units of cm	15 in units of cm

- 2 In each binary link block dialog box, specify the length parameter.

Block	Length (cm)
Binary Link A	10
Binary Link B	35
Binary Link A1	20

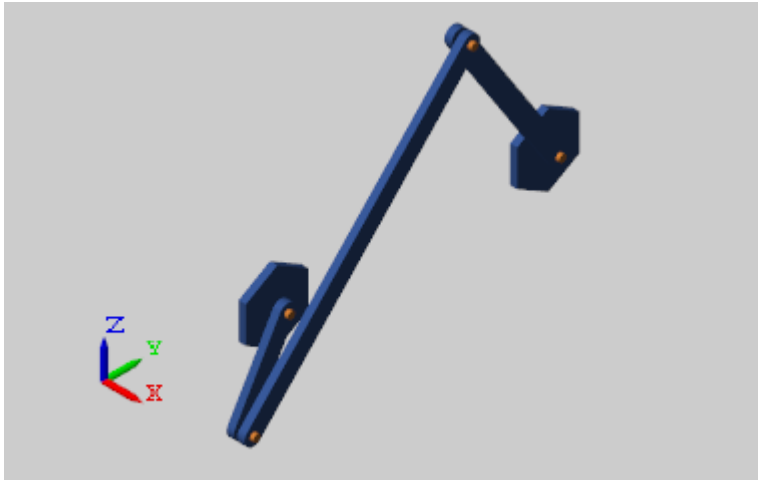
## Guide Assembly and Visualize Model

Guide model assembly by specifying the desired initial state for one or more joints in the model. To specify an initial angle of 30° for the Base-Crank joint:

- 1 In the Base-Crank Revolute Joint block dialog box, expand **State Targets** and select **Specify Position Target**.
- 2 In **Value**, enter -30 and press **OK**.
- 3 In the menu bar, select **Simulation > Update Diagram**

Mechanics Explorer opens with a static display of the four-bar linkage in its initial configuration. If the joint state targets that you specified are valid and compatible, the initial configuration matches those state targets precisely.





## Verify Model Assembly

To check whether—and how precisely—your state targets were met, you can use the Simscape Variable Viewer or the Simscape Multibody Model Report.

To open the Simscape Variable Viewer, in the Simulink menu bar, select **Analysis > Simscape > Variable Viewer**. To open the Simscape Multibody Model Report, update the diagram and, in the Mechanics Explorer menu bar, select **Tools > Model Report**.

The figure shows a Model Report example. The yellow marker indicates that the Base-Rocker Revolute Joint state target was satisfied approximately only. The remaining green marker indicates that the remaining state target was satisfied precisely.

The screenshot shows a window titled "Model Report - four\_bar". At the top, it indicates the assembly status: Assembly status: ●, Joints: ●, and Constraints: ●. Below this, there are three tabs: "Joints", "Constraints", and "Statistics". The "Joints" tab is active, displaying a table with the following data:

Joint	Assembled	Primitive	Position					Velocity				
			Actual	Specified	Unit	Priority	Status	Actual	Specified	Units	Priority	Status
Base_Cran...	<span style="color: green;">●</span>	Rz	-30	-30	deg	High	<span style="color: green;">●</span>	+0		deg/s		
Base_Rock...	<span style="color: green;">●</span>	Rz	-5.33164	+0	deg	Low	<span style="color: yellow;">▲</span>	+0		deg/s		
Connecto...	<span style="color: green;">●</span>	Rz	+103.423		deg			+0		deg/s		
Crank_Co...	<span style="color: green;">●</span>	Rz	-78.7549		deg			+0		deg/s		

An "OK" button is located at the bottom right of the window.

## Simulate Model

Run the simulation, e.g., by selecting **Simulation > Run**. Mechanics Explorer shows a 3-D animation of the four bar assembly. The assembly moves due to gravity, specified in the Mechanism Configuration block.

## Open Reference Model

To see a complete model of the four-bar assembly, at the MATLAB command prompt enter:

- `smdoc_four_bar`

## Troubleshoot Assembly Errors in Aiming Mechanism Model

### In this section...

“Model Overview” on page 3-27

“Explore Model” on page 3-28

“Update Model” on page 3-31

“Troubleshoot Assembly Error” on page 3-31

“Correct Assembly Error” on page 3-33

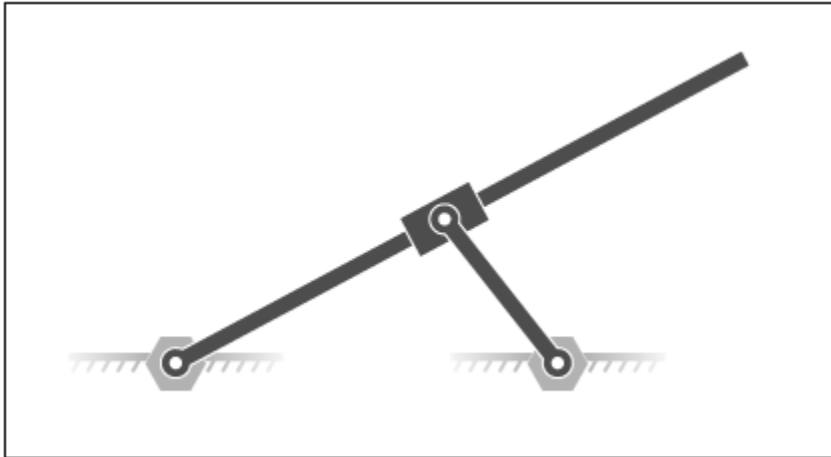
“Simulate Model” on page 3-34

### Model Overview

In closed-loop systems, joints and constraints must be mutually compatible. For example, in a four-bar linkage, all revolute joints must spin about parallel axes. If one of the joints spins about a different axis, assembly fails and the model does not simulate.

To simplify the troubleshooting process, Simscape Multibody provides Model Report. This tool helps you pinpoint the joints and constraints that caused assembly to fail. Once you identify these joints and constraints, you can then determine which of their frames to correct—and how to correct them.

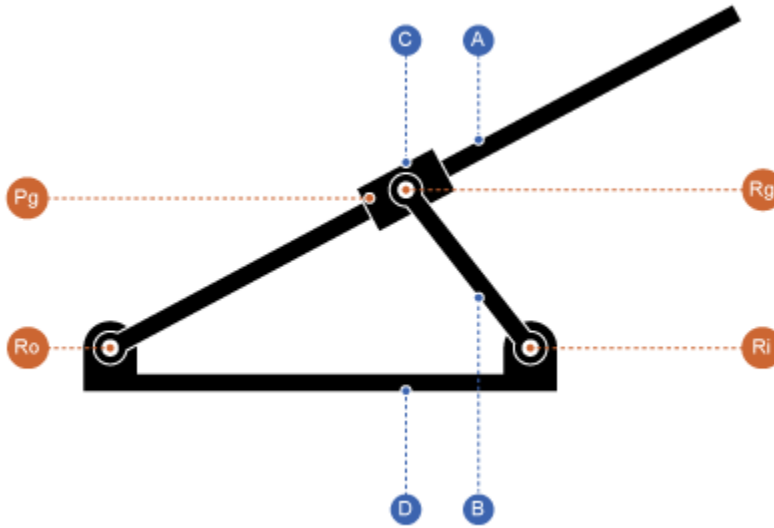
In this example, you identify the assembly error source in an aiming mechanism model using Model Report. Then, using Mechanics Explorer, you determine how to correct that error source. The `sm_dcrankaim_assembly_with_error` featured example provides the basis for this example.



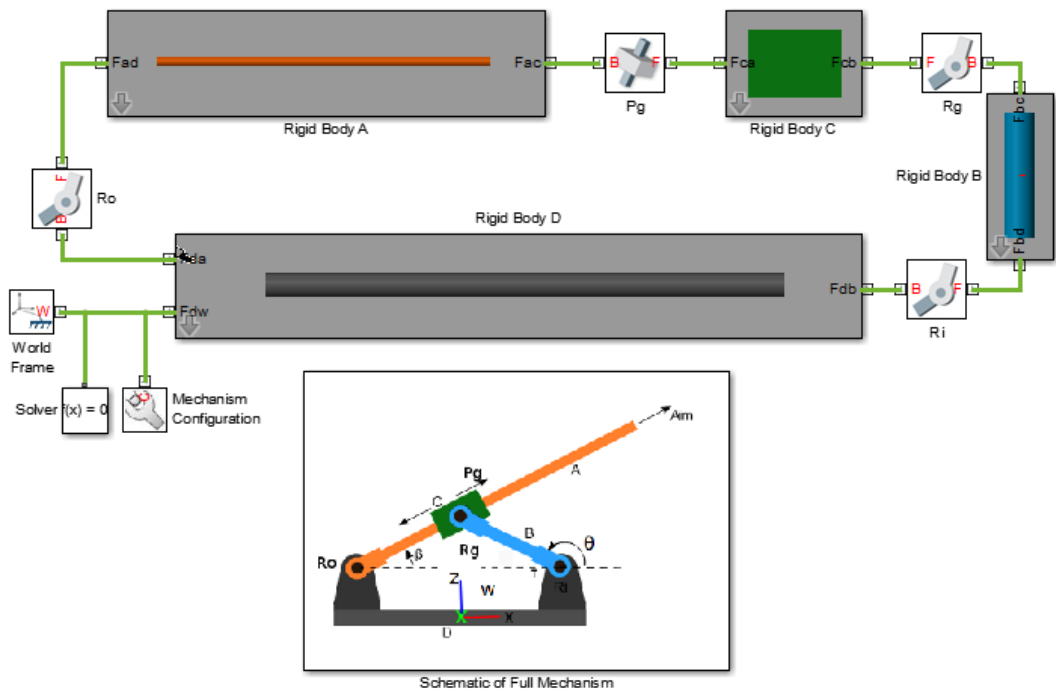
### Explore Model

To open the model, at the MATLAB command line, enter `sm_dcrankaim_assembly_with_error`. The model opens in a new window.

The figure shows a schematic of the system that the model represents. This system contains four rigid bodies, labeled A-D. These rigid bodies connect in a closed loop via four joints, labeled Ri, Ro, Rg, and Pg. When connected to each other, these components form a system with one degree of freedom.

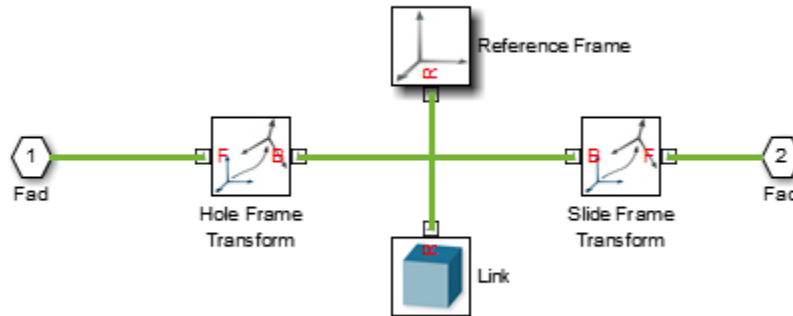


The model represents the components of this system using blocks. Each block represents a physical component. A World Frame block provides the ultimate reference frame in the model. The figure shows the block diagram that the model uses to represent the double-crank aiming mechanism.



To represent the rigid bodies, the model contains four subsystem blocks, labeled Rigid Body A-D. Each subsystem contains one **Solid** block and multiple **Rigid Transform** blocks. The **Solid** block provides geometry, inertia, and color to the rigid body subsystem. The **Rigid Transform** blocks provide the frames that you connect the joints to. A **Reference Frame** block identifies the ultimate reference frame in the subsystem block. A

The model labels the rigid body subsystem blocks Rigid Body A-D. To examine the block diagram for a rigid body subsystem, right-click the subsystem block and select **Mask > Look Under Mask**. The figure shows the block diagram for Rigid Body A.



To represent the joints, the model contains four joint blocks. Three joints provide one rotational degree of freedom between a pair of rigid bodies. You represent each of these joints with a **Revolute Joint** block. A fourth joint provides one translational degree of freedom between a pair of rigid bodies. You represent this joint with a **Prismatic Joint** block. The model labels the revolute joint blocks Ro, Rg, and Ri, and the prismatic joint block Pg.

## Update Model

As the model name suggests, this model contains an error. The error prevents the model from assembling successfully, which causes simulation to fail. To update the model and investigate the assembly error:

- On the Simulink menu bar, select **Simulation > Update Diagram**.

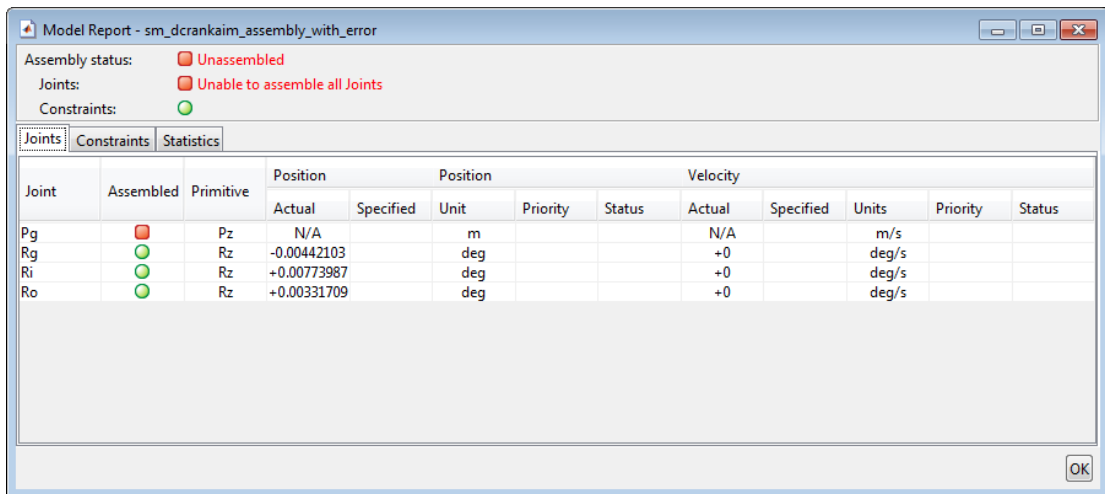
Mechanics Explorer opens with a static display of your model in its initial state. Because the model contains an assembly error, Simscape Multibody issues an error message. Ignore that message for now.

## Troubleshoot Assembly Error

Mechanics Explorer provides access to Model Report, a Simscape Multibody utility that summarizes the assembly status of each joint and constraint in a model. Open this utility

to determine which joint has failed to assemble. To do this, in the Mechanics Explorer menu bar, select **Tools > Model Report**.

Model Report opens in a new window. A red square indicates that the model, as expected, has failed to assemble. A second red square indicates that an unassembled joint, Pg, is the only contributing factor in the model assembly error. This information enables you to concentrate your troubleshooting efforts on a small block diagram region—that surrounding the Pg joint block.

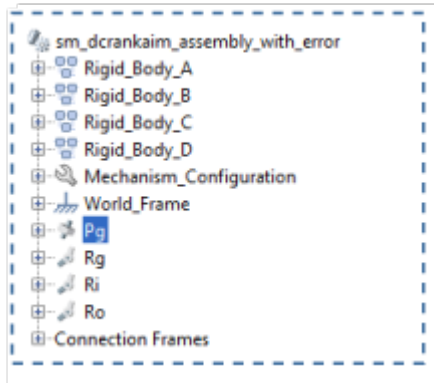


### Identifying Error Root Cause

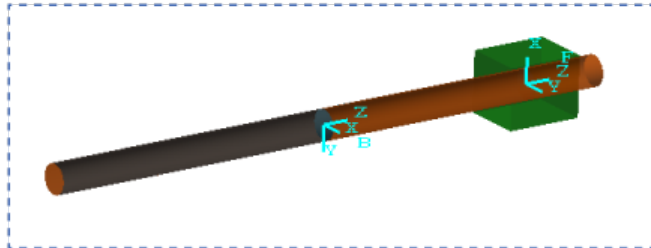
The error message that Simscape Multibody issued during model update identifies position violation as the root cause of assembly failure. This suggests that the frames connected by joint Pg are improperly aligned. To confirm this hypothesis, check the orientation of these frames in Mechanics Explorer.

- 1 In the Mechanics Explorer tree pane, select **Pg**.





- 2 In the Mechanics Explorer visualization pane, examine the position and orientation of the highlighted frames. These are the frames that appear in a light turquoise blue color.



The two frames are offset along the Z axis. This offset is valid, since joint Pg contains a prismatic primitive aligned with the Z axis, providing the frames with one translational degree of freedom along that axis. However, the two frames are also rotated with respect to each other about the common Z axis. This offset is invalid, since joint Pg contains no Revolute or Spherical primitives, and hence no rotational degrees of freedom about any axis. To correct the model assembly error, you must rotate either of the two frames so that all of their axes are parallel to each other.

### Correct Assembly Error

In this example, you apply a rotation transform to the follower frame so that its axes lie parallel to the base frame axes. Alternatively, you could apply an equivalent rotation

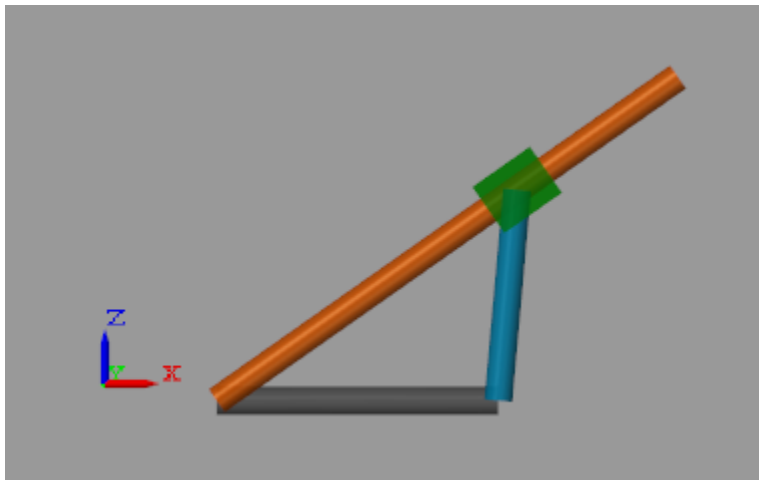
transform to the base frame. This step enables joint Pg, and hence the model itself, to assemble successfully.

- 1 In the tree pane of Mechanics Explorer, right-click the Pg node and select **Go To Block**. Simscape Multibody brings the block diagram to the front and highlights the Pg block.
- 2 Right-click the Rigid Body C subsystem block and select **Mask > Look Under Mask**.
- 3 Double-click the **Slide Frame Transform** block and select the new parameter values that the table provides. Select **OK**.

Parameter	New Value
Rotation > Pair 2 > Follower	+X
Rotation > Pair 2 > Base	+Y

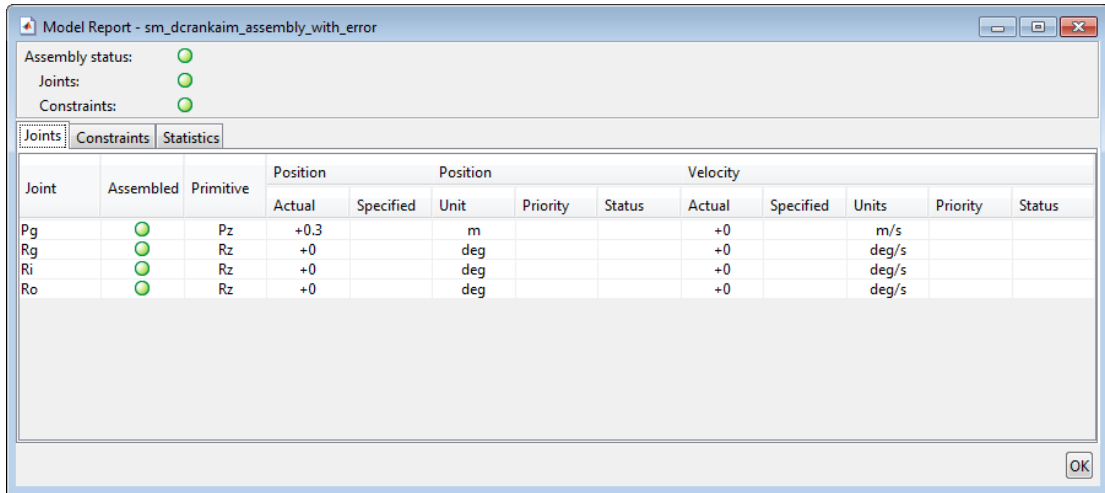
## Simulate Model

You can now simulate the model. On the Simulink menu bar, select **Simulation > Run**. Mechanics Explorer opens with a 3-D animation of your model. The figure shows a snapshot of the animation. Rotate, roll, pan, and zoom to explore.



You can use the Model Report tool to verify the assembly status. To do this, in the Mechanics Explorer menu bar, select **Tools > Model Report**. In Model Report, check

that the assembly status icons for the model and its joints are green circles. The green circles indicate that the model has assembled correctly.



## Related Examples

- “Model an Open-Loop Kinematic Chain” on page 3-16
- “Model a Closed-Loop Kinematic Chain” on page 3-21

## More About

- “Joint Connections” on page 3-4

## Gear Constraints

**In this section...**

- “Gear Types” on page 3-36
- “Featured Examples” on page 3-37
- “Inertia, Geometry, and Efficiency” on page 3-37
- “Using Gear Blocks” on page 3-37
- “Assembling Rigid Bodies with Gear Constraints” on page 3-39
- “Common Gear Assembly and Simulation” on page 3-40
- “Rack and Pinion Assembly and Simulation” on page 3-42

You can represent gear constraints in a multibody model. To do this, Simscape Multibody provides a Gears and Couplings library. This library contains gear blocks that you can use to constrain the motion of two rigid body frames. The figure shows the gear blocks that the library provides.



## Gear Types

The **Gears and Couplings > Gears** library provides blocks for modeling gears. The table summarizes the gears you can model with these blocks.

Block	Description
Common Gear Constraint	Transfer rotational motion between two frames spinning about parallel axes

Block	Description
Rack and Pinion Constraint	Transfer rotational motion at a pinion into translational motion at a rack and vice-versa.
Bevel Gear Constraint	Transfer rotational motion between two frames spinning about arbitrarily aligned axes.

## Featured Examples

Simscape Multibody provides various examples showcasing the use of gear blocks. The table lists some of these examples. To open an example model, at the MATLAB command line, enter the model name, e.g., `sm_cardan_gear`.

Featured Example	Model Name	Gear Blocks Used
Cardan gear	<code>sm_cardan_gear</code>	Common Gear Constraint
Windshield wiper	<code>sm_windshield_wiper</code>	Rack and Pinion Constraint
Robotic wrist	<code>sm_robotic_wrist</code>	Bevel Gear Constraint

Open the models and examine the blocks for examples of how to connect the gear blocks and specify their parameters.

## Inertia, Geometry, and Efficiency

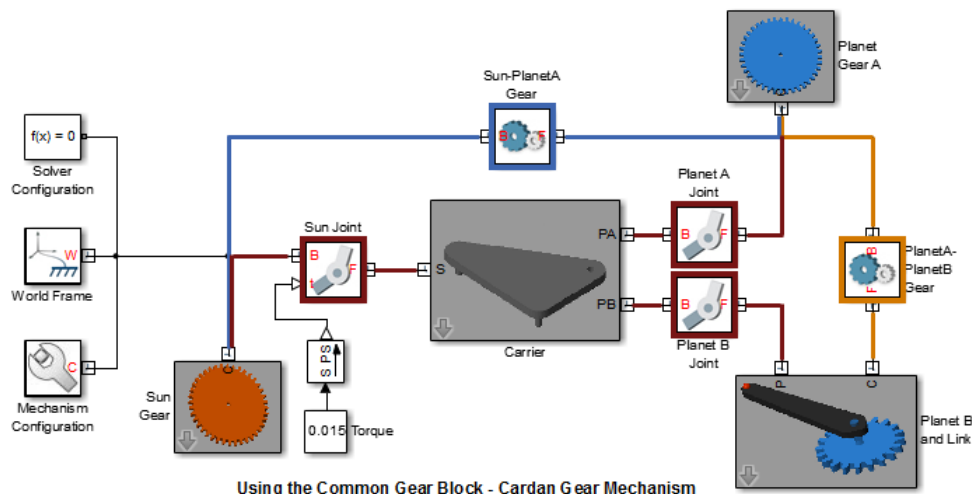
Each gear block represents a kinematic constraint between two rigid body frames. This constraint does not account for the effects of inertia or power transmission losses. It also does not provide gear visualization. If necessary, consider modeling these effects using other Simscape Multibody and Simscape blocks. To represent gear inertia and geometry, use the `Solid` block.

## Using Gear Blocks

To apply a gear constraint between two rigid bodies, connect the base and follower frames of the gear block to the rigid body frames that you want to constrain. Then, open

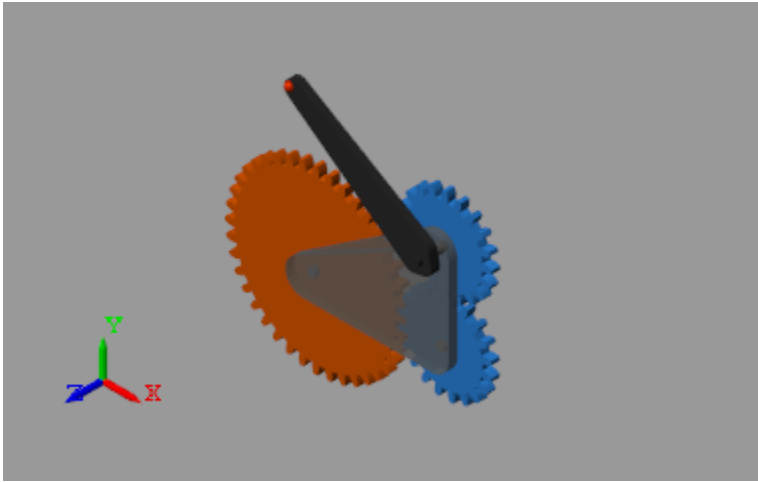
the gear block dialog box and specify the gear parameters. Parameters can include gear dimensions and ratio.

Featured example `sm_cardan_gear` illustrates an application of the Common Gear block. In this model, two Common Gear blocks connect three gear rigid bodies. Subsystems Planet Gear A, Planet B and Link, and Sun Gear represent these rigid bodies. One Common Gear block constrains the motion of subsystem Planet Gear A with respect to subsystem Sun Gear. The other Common Gear block constrains the motion of subsystem Planet B and Link with respect to subsystem Planet Gear A. The figure shows the block diagram of this model.



This example shows the Cardan Gear mechanism that converts rotational motion into reciprocating linear motion without using linkages or sliders. The mechanism uses three gears - one sun and two planet gears. The sun gear is twice as large as the planet gears (which are of the same size). The red pointer on the link traces a straight line as the gears rotate.

So that the three gear subsystems can rotate with respect to each other, the model includes three Revolute Joint blocks. Each Revolute Joint block provides one rotational degree of freedom between one gear subsystem and the gear carrier—a rigid body that holds the three rotating gears. The figure shows the Mechanics Explorer display of this model.



## Assembling Rigid Bodies with Gear Constraints

To assemble successfully, a model must satisfy the constraints that a gear block imposes. These include distance and orientation constraints that are specific to each block. The table summarizes these constraints.

Gear Constraint	Description
Frame Distance	The model must maintain a fixed distance between the base and follower gear frames. The value of this distance depends on the gear block that you use.
Frame Orientation	The model must orient the base and follower gear frames according to rules that are specific to each block.

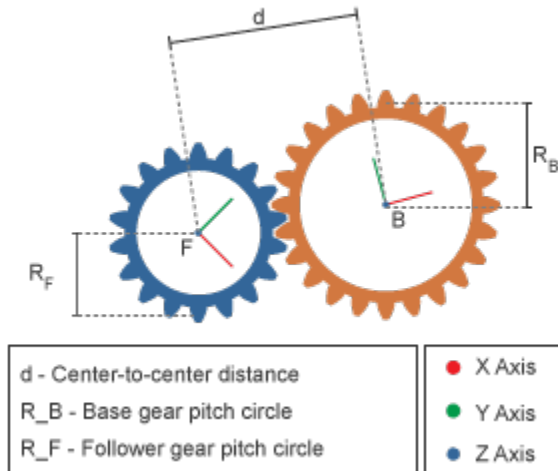
The rigid body frames that the gear block connects must have the proper number and type of degrees of freedom. For a Common Gear block, the frames must have two rotational degrees of freedom with respect to each other. For a Rack and Pinion block, the frames must have one translational and one rotational degree of freedom with respect to each other. You provide these degrees of freedom using joint blocks.

- Use joint blocks with revolute primitives to provide the rotational degrees of freedom.

- Use joint blocks with prismatic primitives to provide the translational degrees of freedom.

## Common Gear Assembly and Simulation

During assembly, the Common Gear block requires that the base and follower frame Z axes align. These are the rotation axes of the two gear frames. Failure to align the Z axes of the two gear frames results in assembly failure during model update. The figure illustrates the common gear rigid bodies, frames, and distance constraints.

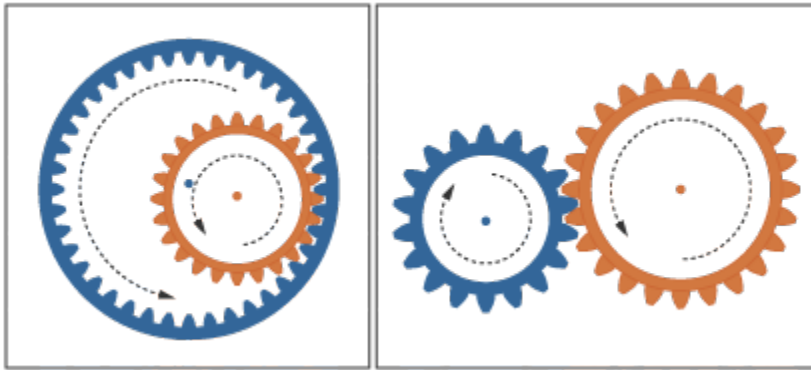


Connect the gear rigid bodies to joints possessing one (or more) revolute joint primitives. The rotational axis of the revolute primitive must align with the Z axis of the gear frame that it connects to. This ensures that the gear frames possess a rotational degree of freedom about the correct axis (Z).

### Common Gear Types

With the Common Gear block, you can represent internal and external gear constraints. If the gear constraint is internal, the gear frames rotate in the same direction. If it is external, the gear frames rotate in opposite directions. The figure illustrates the two common gear types that you can represent and their relative rotation senses.





### Gear Dimensions

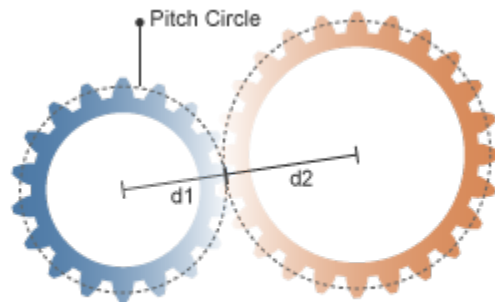
In the block dialog box, you specify the gear dimensions. Depending on the specification method that you choose, you can specify the center-to-center distance between gears or the pitch circle radii. During model assembly, the Common Gear block imposes this distance constraint between the two gear frames. This ensures that the gear assembles properly or, if issues arise, that you can correct any assembly issues early on.

You specify the gear relative sizes in the block dialog box. If you select the **Center Distance and Ratio** specification method, the gear ratio specifies which of the two gears is the larger one. If the gear ratio is greater than one, the follower gear is the larger gear. If the gear ratio is smaller than one, the base gear is the larger gear.

If you specify an internal gear type, the larger gear is the ring gear. A gear ratio greater than unity makes the follower gear the ring gear. A gear ratio smaller than unity makes the base gear the ring gear.

### Gear Pitch Circles

The pitch circle of a gear is an imaginary circle that passes through the contact point between gears. The pitch radius of a gear is the radius of this imaginary circle. The figure illustrates the pitch circles of two meshing gears and their pitch radii. These are the gear radii that you enter in the block dialog box when you select the **Pitch Circle Radii** specification method.



d1 - First gear pitch radius    d2 - Second gear pitch radius

### Simulation

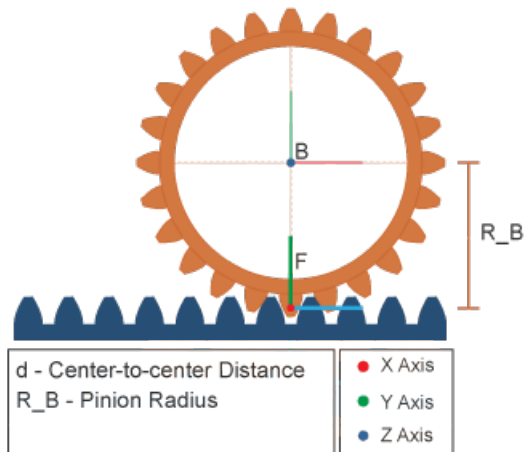
During simulation, the Common Gear block requires that the model maintain the proper distance between gear frames. This distance must equal either the center-to-center distance or the sum of base and follower gear pitch radii that you specify in the block dialog box. The structure of the model must be such that the gears maintain this distance between them. Failure to maintain this distance results in an error during simulation.

In the Cardan Gear example, the Carrier rigid body fixes the distances between the three gears. As long as these distances match the gear dimensions that you specify in the block dialog box, the model should simulate without an issue.

### Rack and Pinion Assembly and Simulation

The base frame of the Rack and Pinion block represents the pinion. It can rotate about its Z axis. The follower frame of the same block represents the rack. It can translate along its Z axis. During assembly, the Rack and Pinion block requires that the base and follower frame Z axes be mutually orthogonal.

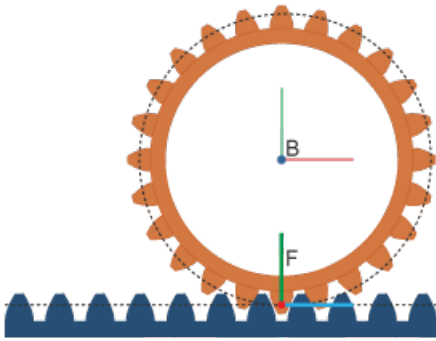
When the gear is in its zero configuration—a configuration in which the angle and displacement between base and follower frames are taken as zero—the follower frame Z axis is also parallel to the base frame X axis, and base and follower frame Y axes are parallel to each other. The follower frame origin lies along the base frame -Y axis, at a distance equal to the base gear pitch radius. The figure illustrates these constraints.



To ensure the rack and pinion can move with respect to each other, you must connect the rack and pinion rigid bodies to joints blocks. The joint block on the rack side must have one (or more) prismatic primitives. At least one primitive axis must align with the Z axis of the follower gear frame. The joint block on the pinion side must have one (or more) revolute primitives. At least one revolute axis must align with the Z axis of the base gear frame.

### **Gear Pitch Circles**

The pitch circle of a rack and pinion gear is the imaginary circle that passes through the contact point between the pinion and the rack. The pitch radius is the radius of this imaginary circle. The figure illustrates the pitch circle for a rack and pinion. This is the circle whose radius you enter in the block dialog box.



#### Simulation

During simulation, the Rack and Pinion block requires that the model maintain the proper distance between gear frames. The distance between the base frame origin (pinion) and the follower frame Z axis must equal the pinion radius. Failure to maintain this distance between gear frames results in a simulation error.

## Model a Rack-and-Pinion Constraint

### In this section...

“Model Overview” on page 3-45

“Model Pinion” on page 3-47

“Model Rack” on page 3-49

“Add Rack and Pinion Constraint” on page 3-52

“Actuate Model” on page 3-53

“Simulate Model” on page 3-55

“Open Complete Model” on page 3-56

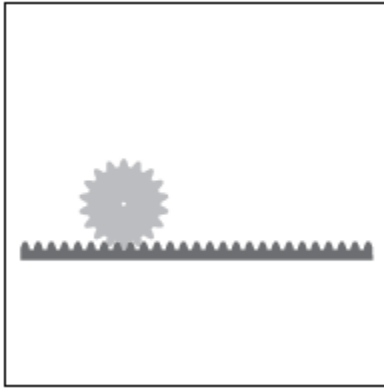
### Model Overview

In this tutorial, you model a kinematic constraint between rack and pinion components. The constraint causes the two components to move in sync such that a pinion rotation corresponds to a rack translation:

$$V_F = \omega_B \cdot R_B,$$

where:

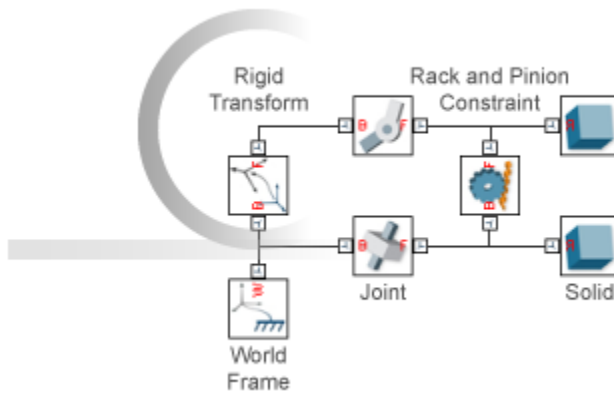
- $V_F$  is the rack translational velocity.
- $\omega_B$  is the pinion rotational velocity.
- $R_B$  is the radius of the pinion pitch circle, an imaginary circle intersecting the contact point between rack and pinion teeth.



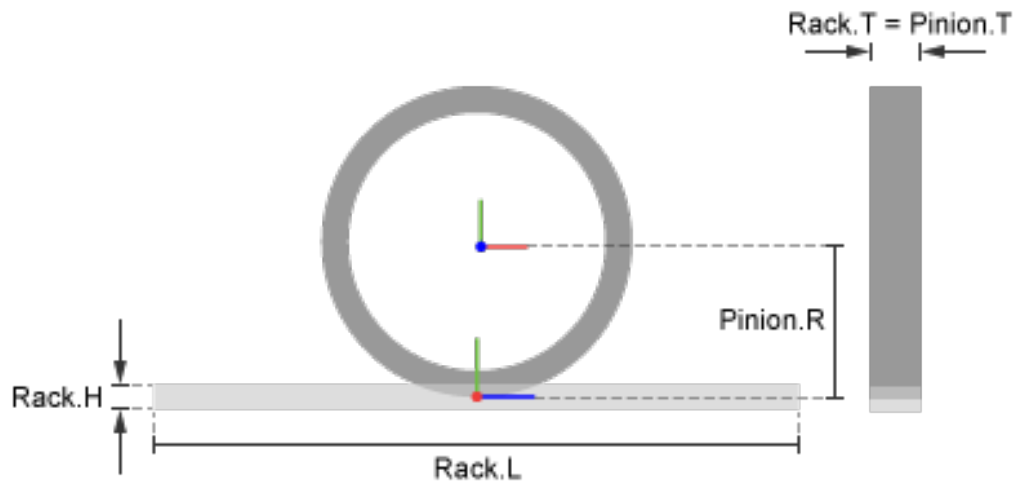
The model uses three key blocks:

- Solid — Specify rack and pinion geometry, inertia, and color
- Joint — Provide motion degrees of freedom to the rack and pinion components. These degrees of freedom enable the rack to translate and the pinion to rotate with respect to the world frame.
- Rack and Pinion Constraint — Constrain the motion of the rack and pinion components so that they move in a meshed configuration.

The figure shows how these blocks connect in the model.



For simplicity, the rack has a brick shape and the pinion has a cylinder shape. These shapes depend on several dimensions, shown in the figure. You specify each dimension using a MATLAB variable. After model assembly, you can add detail to the component shapes. For example, you can specify an involute tooth profile for the rack and pinion.



## Rack and Pinion Dimensions

### Model Pinion

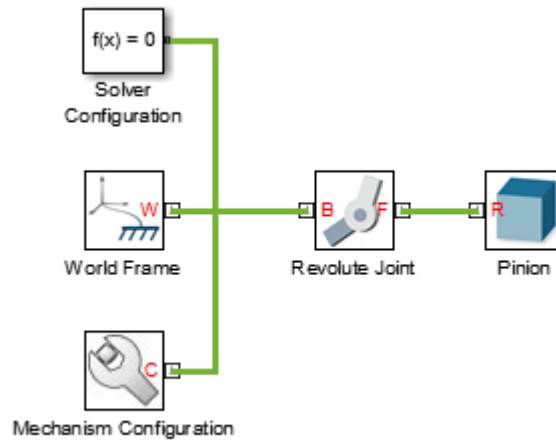
- 1 Start a new model.
- 2 Add these blocks to the model.

Library	Block
Simscape > Utilities	Solver Configuration
Simscape > Multibody > Frames and Transforms	World Frame
Simscape > Multibody > Utilities	Mechanism Configuration
Simscape > Multibody > Joints	Revolute Joint

Library	Block
Simscape > Multibody > Body Elements	Solid

The Solid block specifies the component geometry, inertia, and color. The joint block provides the component its motion degrees of freedom—in this case, one rotational degree of freedom with respect to the world frame.

- 3 Connect and name the blocks as shown in the figure. Port frames joined by a connection line are coincident in space.



- 4 In the Pinion block dialog box, specify geometry, inertia, and color.

Parameter	Enter or Select
Geometry > Shape	Cylinder
Geometry > Radius	Pinion.R, units of cm
Geometry > Length	Pinion.T, units of cm
Inertia > Density	Rho
Graphic > Visual Properties > Color	Pinion.RGB

- 5 In the model workspace, initialize the MATLAB variables you entered in the block dialog boxes:




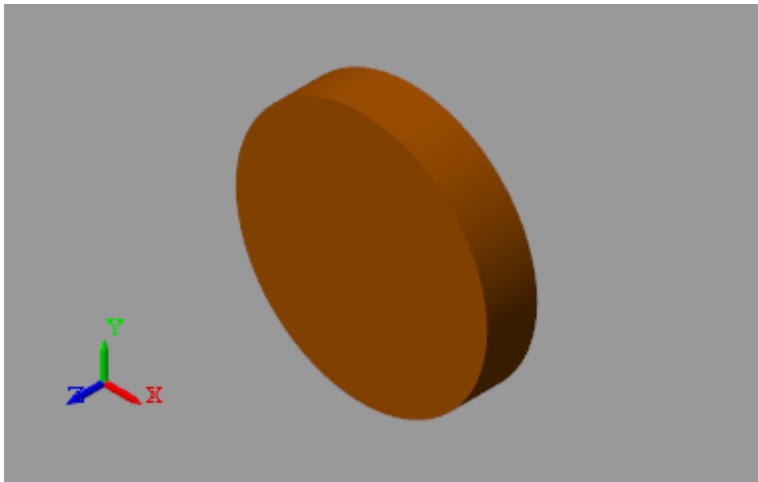
```

% Common Parameters
Rho = 2700; % Mass density of both rack and pinion components

% Pinion Parameters
Pinion.R = 10;
Pinion.T = 4;
Pinion.RGB = [0.8, 0.4, 0];

```

- 6 Update the block diagram. You can do this by selecting **Simulation > Update Diagram**. Mechanics Explorer opens with a 3-D view of the pinion gear. To obtain the view shown in the figure, in the Mechanics Explorer toolstrip set the **View convention** parameter to Y up (XY Front). Then, select the isometric view button .



## Model Rack

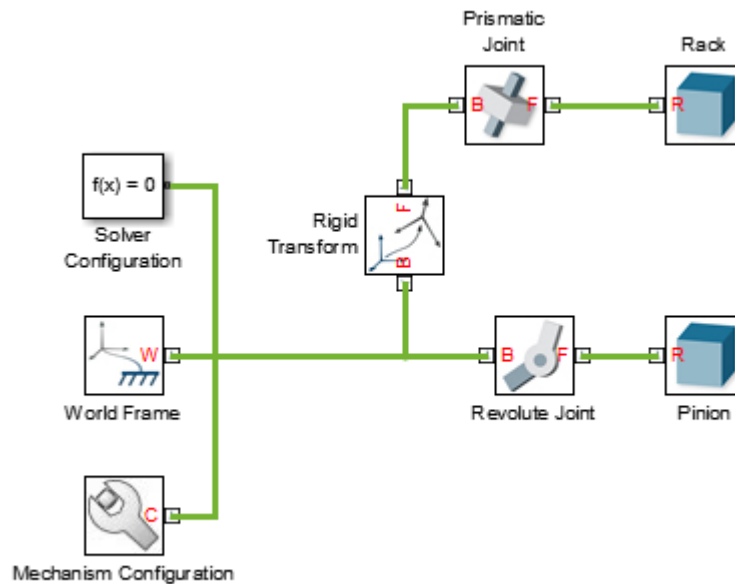
- 1 Add these blocks to the model.

Library	Block
Simscape > Multibody > Frames and Transforms	Rigid Transform
Simscape > Multibody > Joints	Prismatic Joint

Library	Block
Simscape > Multibody > Body Elements	Solid

The Rigid Transform block sets the rack position and pose with respect to the pinion. These quantities must satisfy the assembly conditions later imposed by the Rack and Pinion Constraint block.

- 2 Connect and name the blocks as shown in the figure.



- 3 In the Rack block dialog box, specify geometry, inertia, and color.

Parameter	Select or Enter
Geometry > Shape	Brick
Geometry > Dimensions	[Rack.T, Rack.H, Rack.L], units of cm
Inertia > Density	Rho

Parameter	Select or Enter
Graphic > Visual Properties > Color	Rack.RGB


- 4 In the Rigid Transform block dialog box, specify the rack position and pose with respect to the pinion.

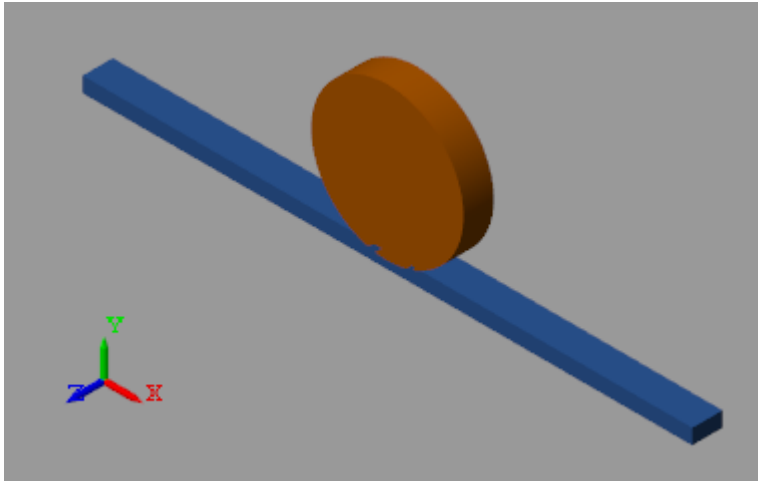
Parameter	Select or Enter
Rotation > Method	Standard Axis
Rotation > Axis	+Y
Rotation > Angle	90
Translation > Method	Standard Axis
Translation > Axis	-Y
Translation > Offset	Pinion.R in units of cm

The rotation transform makes the rack and pinion Z axes mutually orthogonal while keeping the Y axes parallel. The translation transform separates the rack and pinion frame origins by a distance equal to the pinion pitch radius. These transforms satisfy the assembly conditions imposed by the Rack and Pinion Constraint block.

- 5 In the model workspace, initialize the new MATLAB variables entered in the block dialog boxes:

```
% Rack Parameters
Rack.L = 80;
Rack.H = 2;
Rack.T = Pinion.T;
Rack.RGB = [0.2, 0.4, 0.7];
```

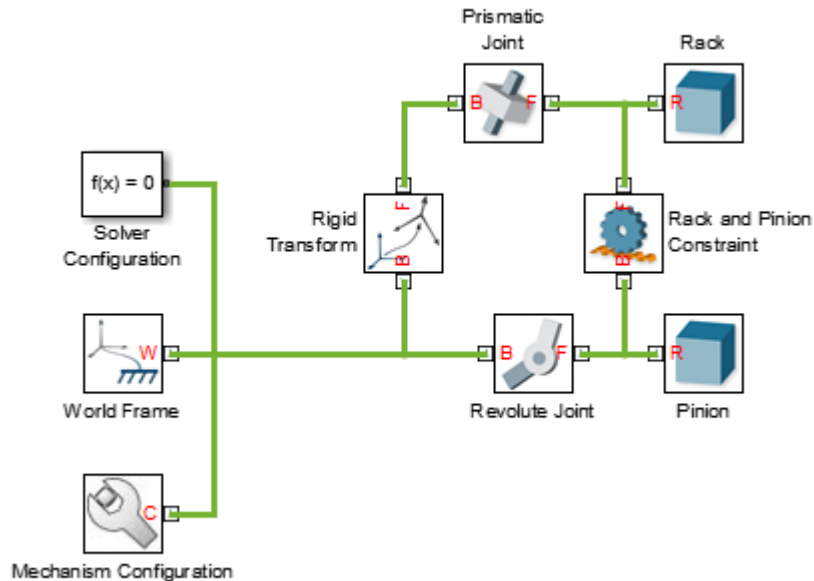
- 6 Update the block diagram. Mechanics Explorer displays a 3-D view of the rack and pinion assembly. Examine the assembly from different viewpoints and verify it is accurate. You can view the rack and pinion frames by clicking the frame button  in the Mechanics Explorer tool bar.



## Add Rack and Pinion Constraint

The model is nearly complete. It remains to constrain the motion of the rack and pinion components. You add this kinematic constraint using the Rack and Pinion Constraint block.

- 1 From the **Gears and Couplings > Gears** library, drag a **Rack and Pinion Constraint** block to the model.
- 2 Connect the block as shown in the figure. The follower frame port connects to the Rack block, while the base frame port connects to the Pinion block.



- 3 In the dialog box of the Rack and Pinion Constraint block, enter `Pinion.R` in **Pinion Radius**.
- 4 Update the block diagram. Mechanics Explorer shows a 3-D display of the updated rack and pinion assembly. Assembly errors due to gear constraints become evident at this stage. If Simscape Multibody issues an error message, correct the model before attempting to run the simulation.

## Actuate Model

- 1 In the Revolute Joint block dialog box, for **Z Revolute Primitive (Rz) > Actuation > Torque**, select **Provided by Input**.

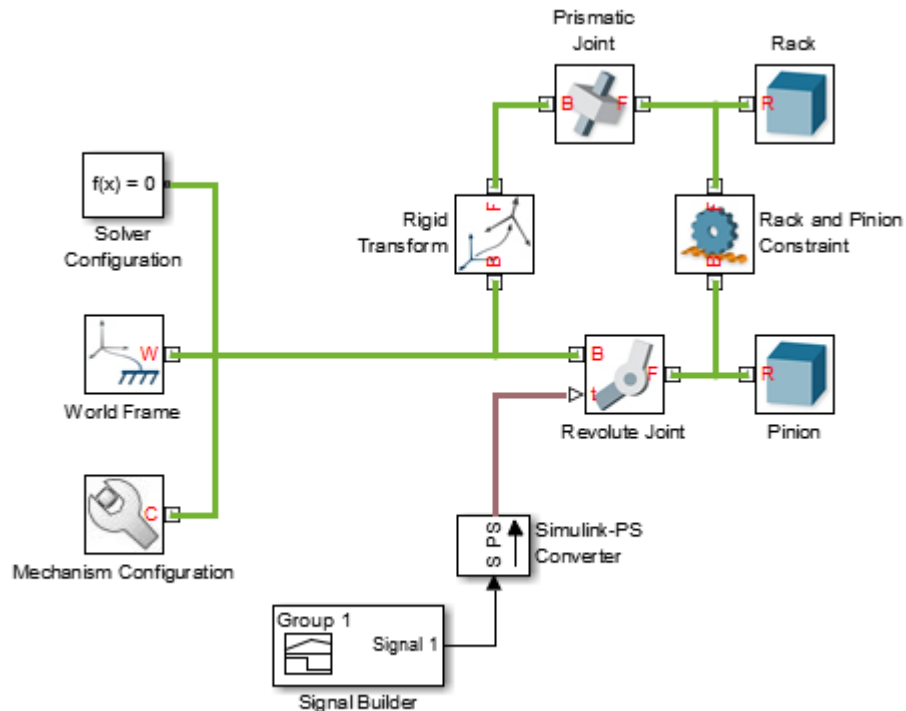
The block exposes a physical signal input port. You use this port to specify a driving torque acting on the pinion. During simulation, this torque will be the source of motion in the model.

- 2 Drag these blocks to specify and process the input torque signal.

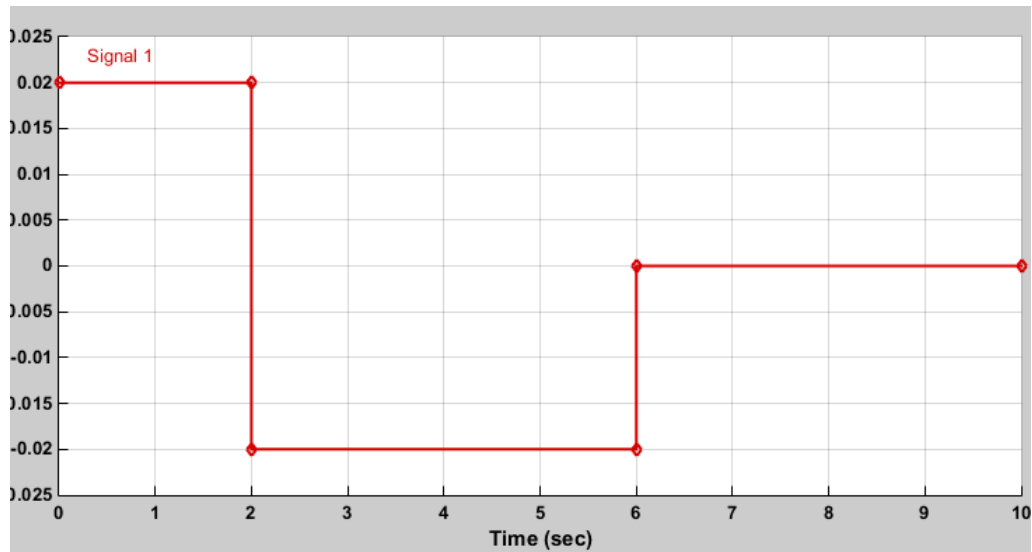
Library	Block
Simulink > Sources	Signal Builder
Simscape > Utilities	Simulink-PS Converter

The Simulink-PS Converter block converts the Simulink input signal into a physical signal compatible with Simscape Multibody blocks. It also provides signal filtering, which enables you to smooth discontinuous signals.

- 3 Connect the blocks as shown in the figure.




- 4 In the Signal Builder block dialog box, draw the input signal as shown in the figure. This signal starts with a positive torque followed by a negative torque. The positive torque causes the pinion to rotate counterclockwise about the base frame +Z axis and the rack to translate along the follower frame +Z axis.

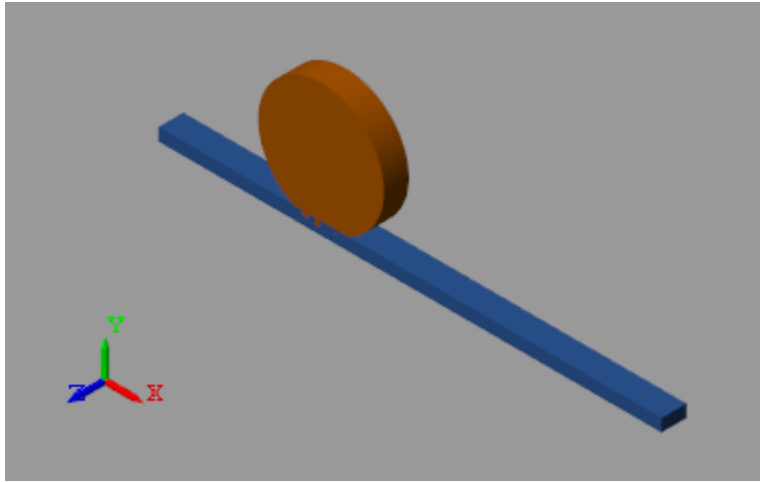


- 5 In the Simulink-PS Converter block dialog box, in the **Input Handling** tab, specify second-order filtering with a time constant of 0.1 s. This filter helps to smooth the discontinuities of the input signal.

Parameter	Select or Enter
Filtering and derivatives	Filter input
Input filtering order	Second-order filtering
Input filtering time constant (in seconds)	0.1

## Simulate Model

Run the simulation. You can do this by selecting **Simulation > Run**. Mechanics Explorer plays a physics-based animation of the rack and pinion assembly. To better see motion during playback, select the frame button  in the Mechanics Explorer tool bar.



## Open Complete Model

To view a complete model of the rack and pinion mechanism, at the MATLAB command prompt enter:

```
smdoc_rack_pinion_c
```

For an example using a helper function to generate a rough involute tooth profile, enter

```
smdoc_rack_pinion_d
```



# Model a Common Gear Constraint

**In this section...**

“Model Overview” on page 3-57  
“Model Sun-Planet Gear Set” on page 3-58  
“Constrain Sun-Planet Gear Motion” on page 3-61  
“Add Ring Gear” on page 3-63  
“Add Gear Carrier” on page 3-66  
“Add More Planet Gears” on page 3-71

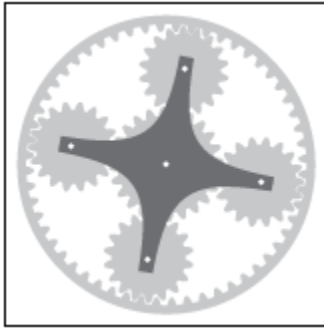
## Model Overview

Planetary gear trains are common in industrial, automotive, and aerospace systems. A typical application is the automatic transmission system of car. From a kinematic point of view, what sets this mechanism apart is the kinematic constraint set between gear pairs. These constraints fix the angular velocity ratios of the gear pairs, causing the gears in each pair to move in sync.

In Simscape Multibody, you represent the kinematic constraint between meshed gears using blocks from the Gears sublibrary. This tutorial shows you how to use these blocks to model a planetary gear train. The gear train contains four rigid bodies:

- Sun gear
- Planet gear
- Ring gear
- Planet carrier

Each rigid body, including the planet carrier, can spin about its central axis. In addition, each planet gear can revolve about the sun gear. Joint blocks provide the required degrees of freedom, while gear constraint blocks ensure the gears move as if they were meshed.



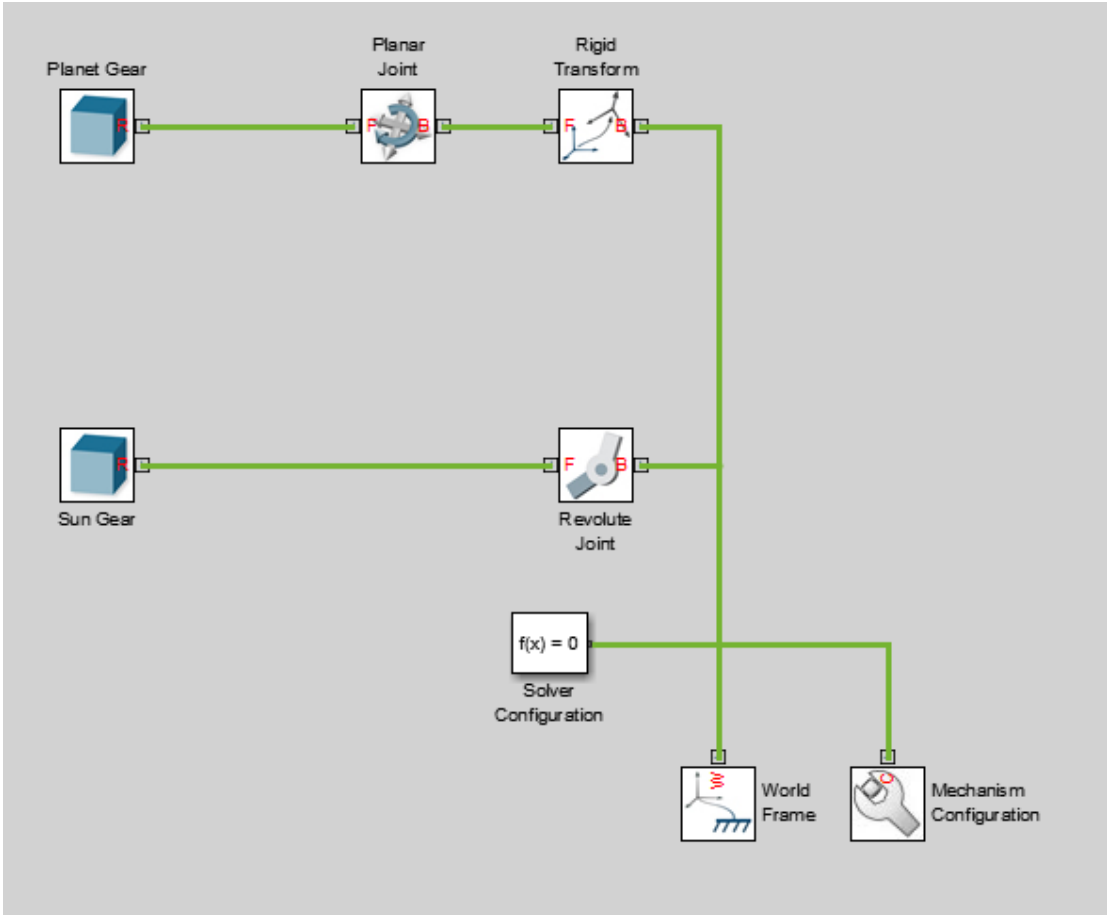
### Model Sun-Planet Gear Set

Model the gear rigid bodies and connect them with the proper degrees of freedom. In a later step, you add gear constraints to this model.

- 1 Drag these blocks to a new model.

Library	Block	Quantity
Body Elements	Solid	2
Joints	Revolute Joint	1
Joints	Planar Joint	1
Frames and Transforms	Rigid Transform	1
Frames and Transforms	World Frame	1
Utilities	Mechanism Configuration	1
Simscape > Utilities	Solver Configuration	1

- 2 Connect and name the blocks as shown.



3 In the Sun Gear block dialog box, specify these parameters.

Parameter	Setting
Geometry > Shape	Select General Extrusion.
Geometry > Cross-Section	Enter <code>simmechanics.demohelpers.gear_profile(2*S</code> Select units of cm.
Geometry > Length	Enter T. Select units of cm.

Parameter	Setting
Inertia > Density	Enter Rho.
Graphic > Visual Properties > Color	Enter Sun . RGB.

The function `simmechanics.demohelpers.gear_profile` produces a rough approximation of an involute gear profile.

- 4 In the Planet Gear block dialog box, specify these parameters.

Parameter	Setting
Geometry > Shape	Select General Extrusion.
Geometry > Cross-Section	Enter <code>simmechanics.demohelpers.gear_profile(2*P</code> Select units of cm.
Geometry > Length	Enter T. Select units of cm.
Inertia > Density	Enter Rho.
Graphic > Visual Properties > Color	Enter Planet . RGB.

- 5 In the Rigid Transform block dialog box, specify these parameters.

Parameter	Setting
Translation > Method	Select Standard Axis.
Translation > Axis	Select +Y.
Translation > Offset	Enter <code>Sun.R + Planet .R</code> . Select units of cm.

- 6 In the model workspace, define the block parameters using MATLAB code:

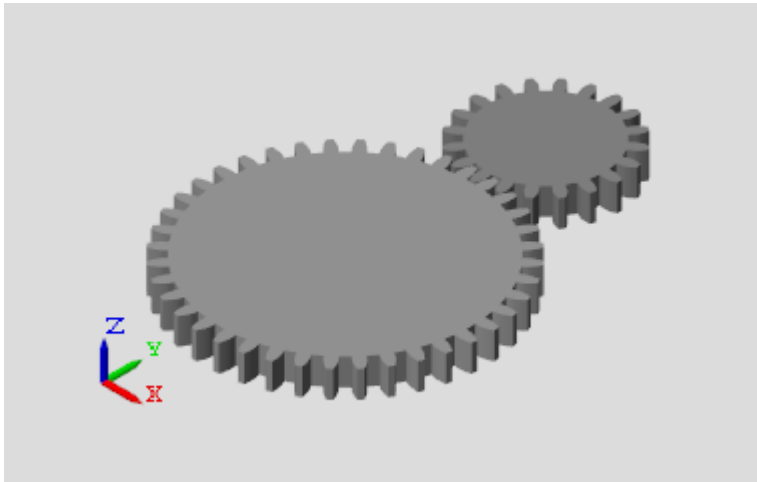
```
% Common Parameters
Rho = 2700;
T = 3;
A = 0.8; % Gear Addendum

% Sun Gear Parameters
Sun.RGB = [0.75 0.75 0.75];
Sun.R = 15;
Sun.N = 40;

% Planet Gear Parameters
```

```
Planet.RGB = [0.65 0.65 0.65];
Planet.R = 7.5;
Planet.N = Planet.R/Sun.R*Sun.N;
```

- 7 Simulate the model. To induce motion, try adjusting the velocity state targets in the joint block dialog boxes. Notice that the sun and planet gears move independently of each other. To constrain gear motion, you must add a gear constraint block between the gear solid blocks.



You can open a copy of the resulting model. At the MATLAB command line, enter `smdoc_planetary_gear_a`.

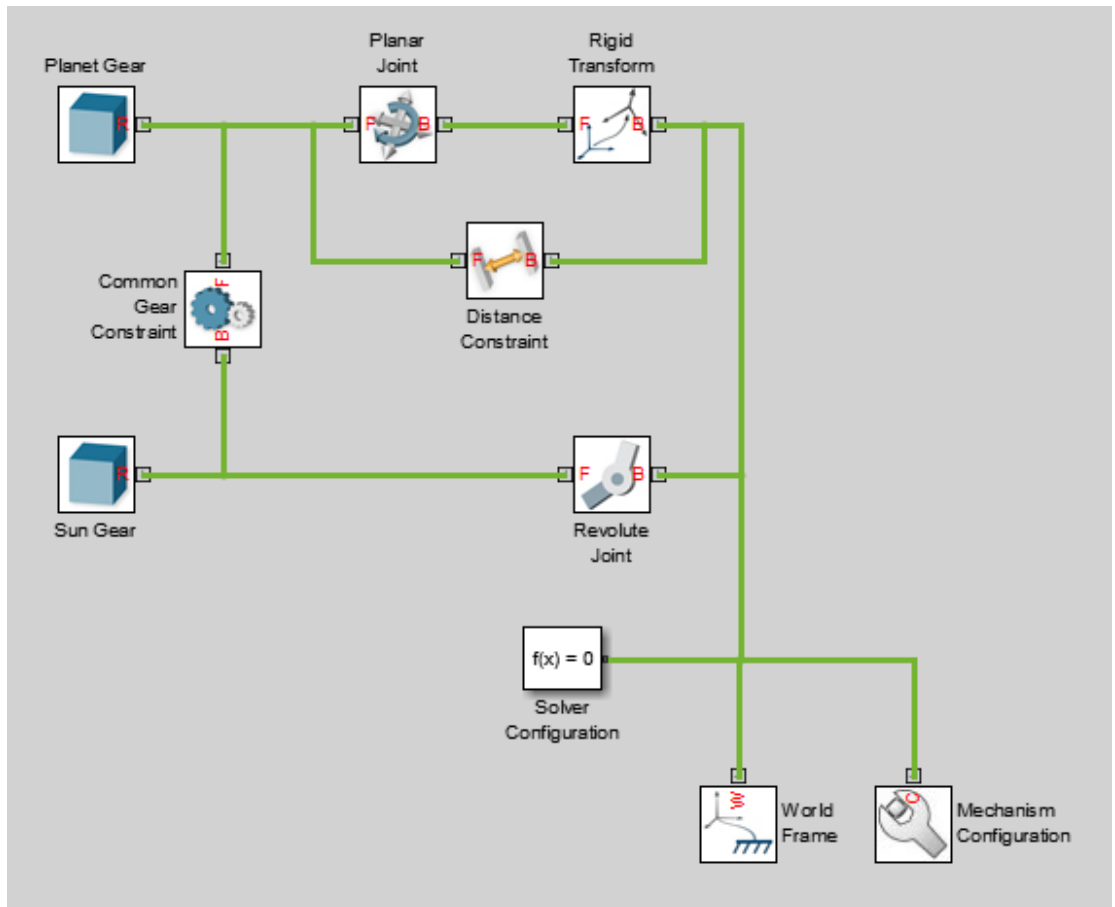
## Constrain Sun-Planet Gear Motion

Specify the kinematic constraints acting between the sun and planet gears. These constraints ensure that the gears move in a meshed fashion.

- 1 Drag these blocks to the sun-planet gear model.

Library	Block
Constraints	Distance Constraint
Gears and Couplings > Gears	Common Gear Constraint

- 2 Connect the blocks as shown. The new blocks are highlighted.



3 In the Common Gear Constraint block dialog box, specify these parameters.

Parameter	Setting
<b>Specification Method</b>	Select <b>Pitch Circle Radii</b> .
<b>Specification Method &gt; Base Gear Radius</b>	Enter Sun .R. Select units of cm.
<b>Specification Method &gt; Follower Gear Radius</b>	Enter Planet .R. Select units of cm.

- 4 In the Distance Constraint block dialog box, specify this parameter:
  - **Distance** — Enter `Sun.R + Planet.R`. Select units of `cm`.
- 5 Simulate the model. To induce motion, try adjusting the velocity state targets in the joint block dialog boxes. Notice that the sun and planet gears now move in sync.

You can open a copy of the resulting model. At the MATLAB command line, enter `smdoc_planetary_gear_b`.

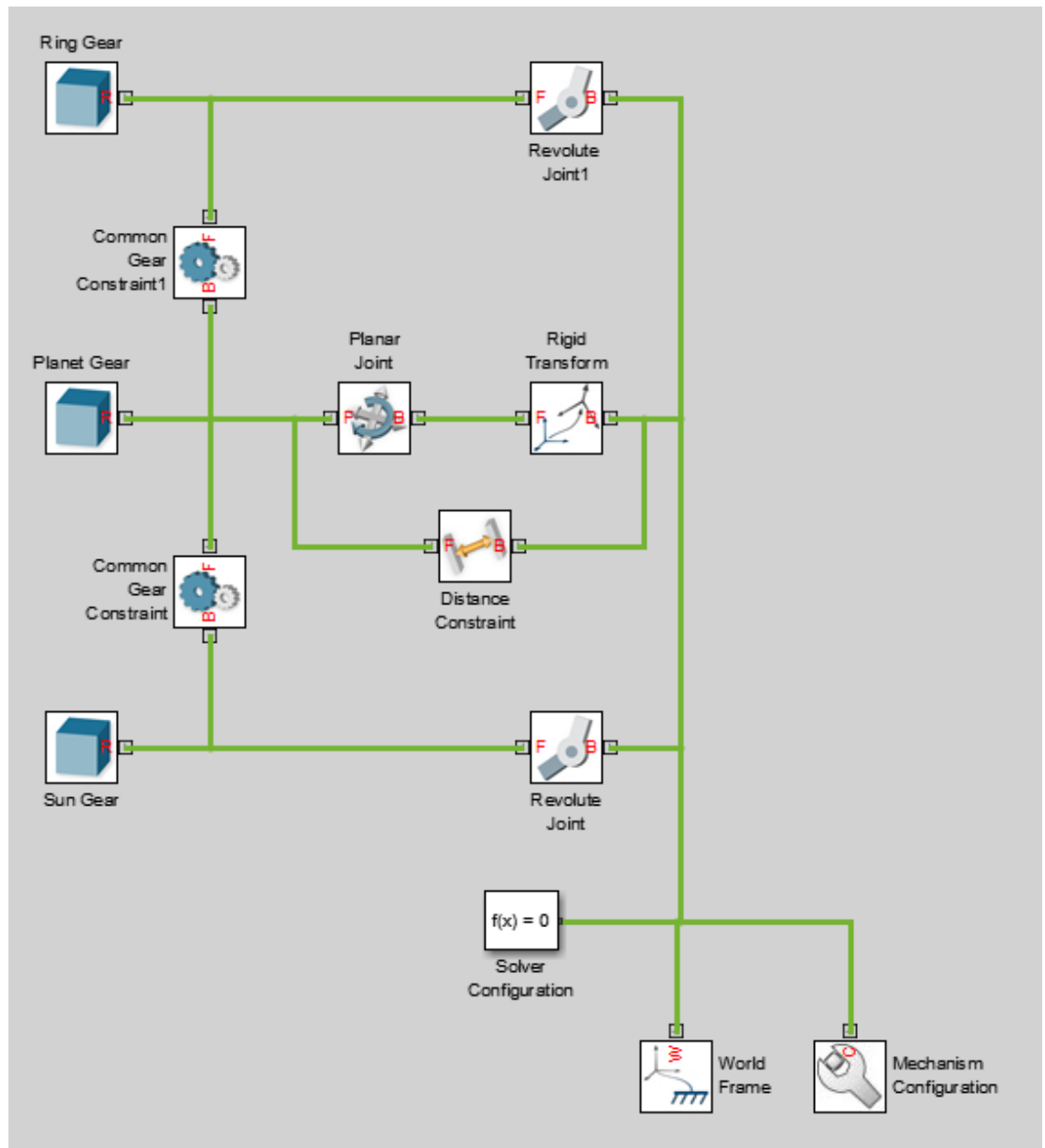
## Add Ring Gear

Model the ring gear rigid body, connect it with the proper degrees of freedom, and constrain its motion with respect to the planet gear.

- 1 Add these blocks to the sun-planet gear model.

Library	Block
Body Elements	Solid
Joints	Revolute Joint
Gears and Couplings > Gears	Common Gear Constraint

- 2 Connect and name the blocks as shown. The new blocks are highlighted.





- 3 In the Ring Gear block dialog box, specify these parameters.

Parameter	Setting
<b>Geometry &gt; Shape</b>	Select <b>General Extrusion</b> .
<b>Geometry &gt; Cross-Section</b>	Enter <b>Ring.CS</b> . Select units of <b>cm</b> .
<b>Geometry &gt; Length</b>	Enter <b>T</b> .
<b>Inertia &gt; Density</b>	Enter <b>Rho</b> .
<b>Graphic &gt; Visual Properties &gt; Color</b>	Enter <b>Ring.RGB</b> .

- 4 In the Common Gear Constraint1 block dialog box, specify these parameters.

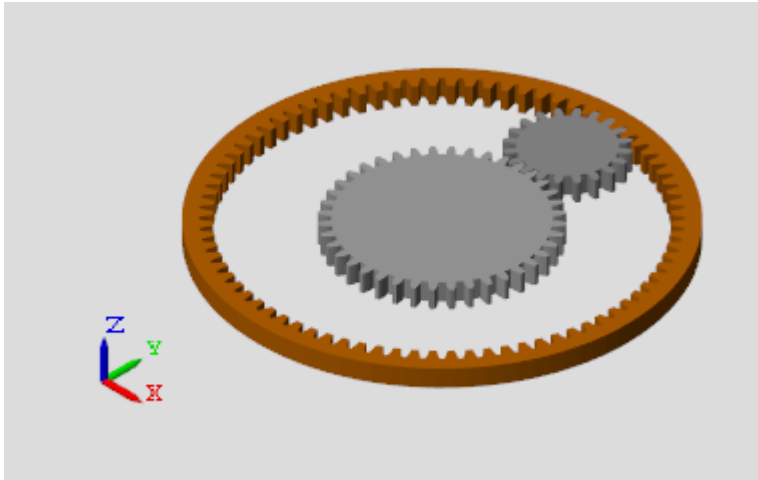
Parameter	Setting
<b>Type</b>	Select <b>Internal</b> .
<b>Specification Method</b>	Select <b>Pitch Circle Radii</b> .
<b>Specification Method &gt; Base Gear Radius</b>	Enter <b>Planet.R</b> . Select units of <b>cm</b> .
<b>Specification Method &gt; Follower Gear Radius</b>	Enter <b>Ring.R</b> . Select units of <b>cm</b> .

- 5 In the model workspace, define the Ring Gear block parameters using MATLAB code:

```
% Ring Gear Parameters
Ring.RGB = [0.85 0.45 0];
Ring.R = Sun.R + 2*Planet.R;
Ring.N = Ring.R/Planet.R*Planet.N;

Ring.Theta = linspace(-pi/Ring.N,2*pi-pi/Ring.N,100)';
Ring.RO = 1.1*Ring.R;
Ring.CS0 = [Ring.RO*cos(Ring.Theta) Ring.RO*sin(Ring.Theta)];
Ring.CSI = simmechanics.demohelpers.gear_profile(2*Ring.R, Ring.N,A);
Ring.CSI = [Ring.CSI; Ring.CSI(1,:)];
Ring.CS = [Ring.CS0; flipud(Ring.CSI)];
```

- 6 Simulate the model. To induce motion, try adjusting the velocity state targets in the joint block dialog boxes. Notice that the sun, planet, and ring gears move in a meshed fashion.



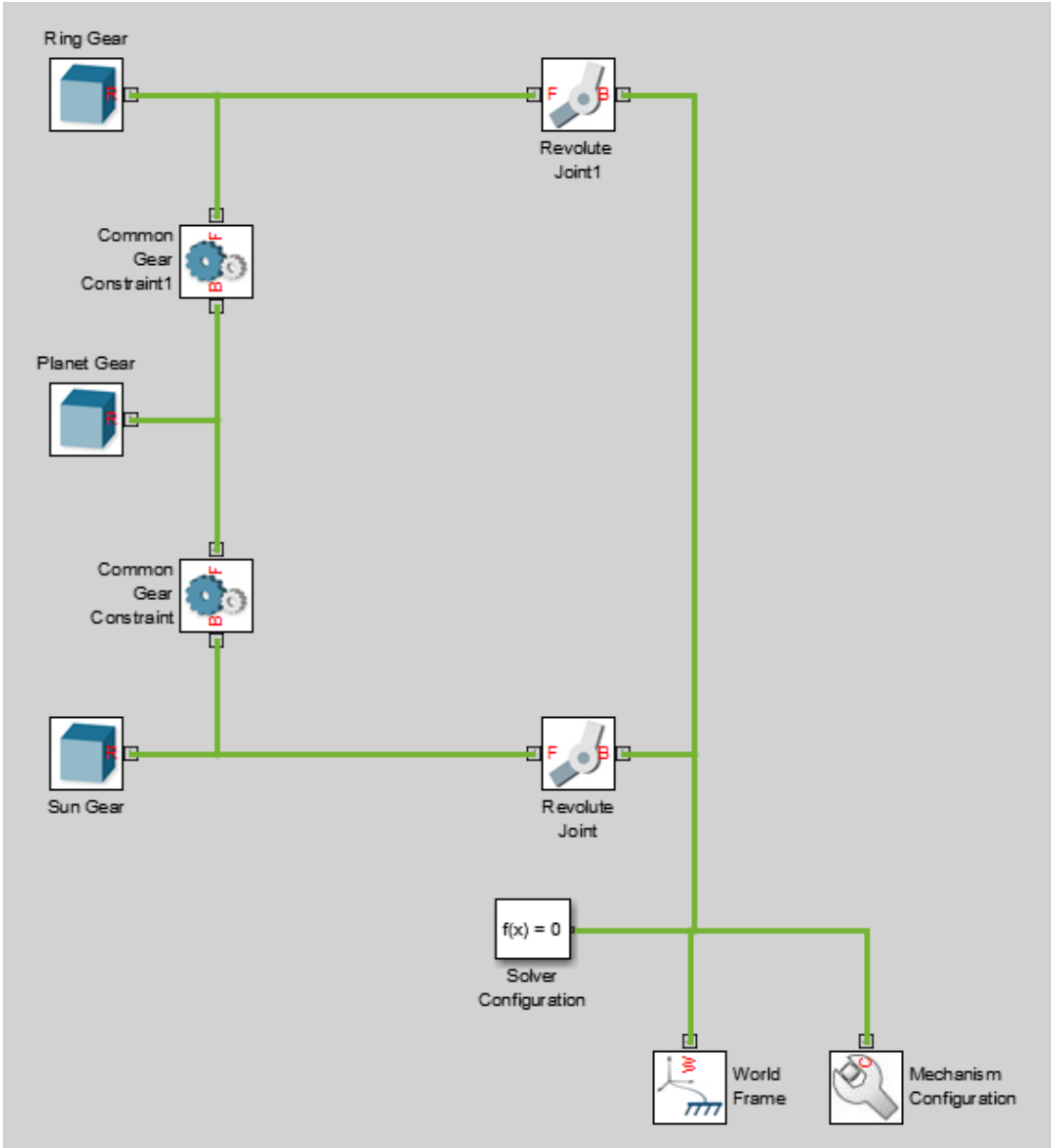
You can open a copy of the resulting model. At the MATLAB command line, enter `smdoc_planetary_gear_c`.

## Add Gear Carrier

Up to now, you have kept the sun and planet gears at a fixed distance using a Distance Constraint block. In an actual planetary gear, a gear carrier enforces this constraint. Model the gear carrier and connect it between the sun and planet gears.

1 Remove these blocks from the planetary gear model:

- Planar Joint
- Rigid Transform
- Distance Constraint

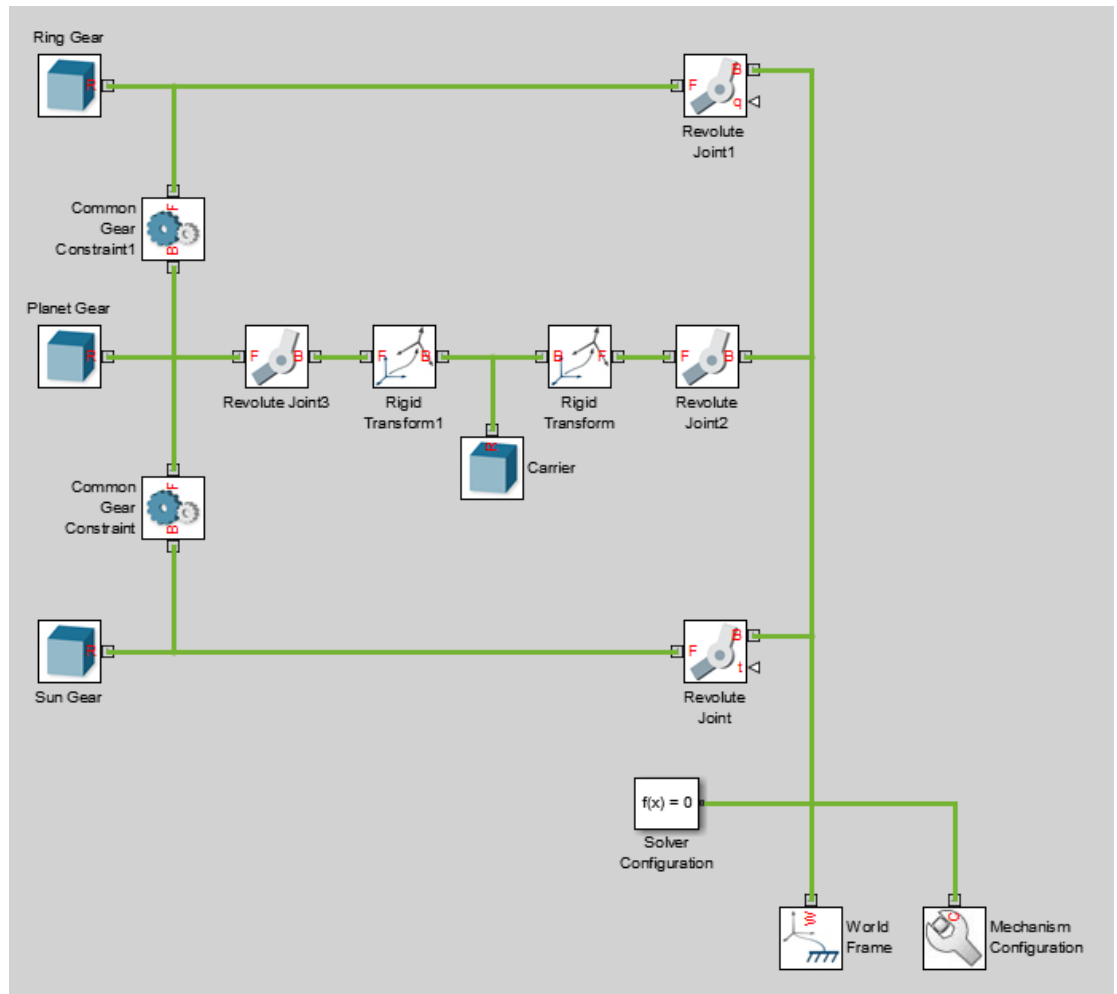


- 2 Add these blocks to the planetary gear model.

<b>Library</b>	<b>Block</b>	<b>Quantity</b>
<b>Body Elements</b>	Solid	1
<b>Joints</b>	Revolute Joint	2
<b>Frames and Transforms</b>	Rigid Transform	2

- 3 Connect and name the blocks as shown.

Pay close attention to the Rigid Transform block orientation: the B frame ports should face the Solid block. The new blocks are highlighted.



- 4 In the Carrier block dialog box, specify these parameters.

Parameter	Setting
Geometry > Shape	Select General Extrusion.
Geometry > Cross-Section	Enter Carrier.CS. Select units of cm.
Geometry > Length	Enter Carrier.T.

Parameter	Setting
Inertia > Density	Enter Rho.
Graphic > Visual Properties > Color	Enter Carrier.RGB.

- 5 In the Rigid Transform block dialog box, specify these parameters.

Parameter	Setting
Translation > Method	Select Cartesian.
Translation > Offset	Enter [Carrier.L/2 0 -(Carrier.T +T)/2]. Select units of cm.

- 6 In the Rigid Transform1 block dialog box, specify these parameters.

Parameter	Setting
Translation > Method	Select Cartesian.
Translation > Offset	Enter [-Carrier.L/2 0 -(Carrier.T +T)/2]. Select units of cm.

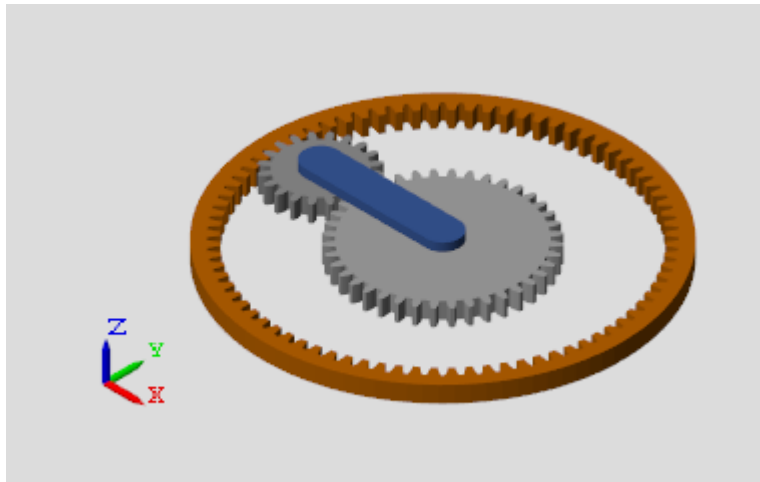
- 7 In the model workspace, define the Carrier block parameters using MATLAB code:

```
% Gear Carrier Parameters
Carrier.RGB = [0.25 0.4 0.7];
Carrier.L = Sun.R + Planet.R;
Carrier.W = 2*T;
Carrier.T = T/2;

Theta = (90:1:270)'*pi/180;
Beta = (-90:1:90)'*pi/180;

Carrier.CS = [-Carrier.L/2 + Carrier.W/2*cos(Theta) ...
Carrier.W/2*sin(Theta); Carrier.L/2 + Carrier.W/2*cos(Beta), ...
Carrier.W/2*sin(Beta)];
```

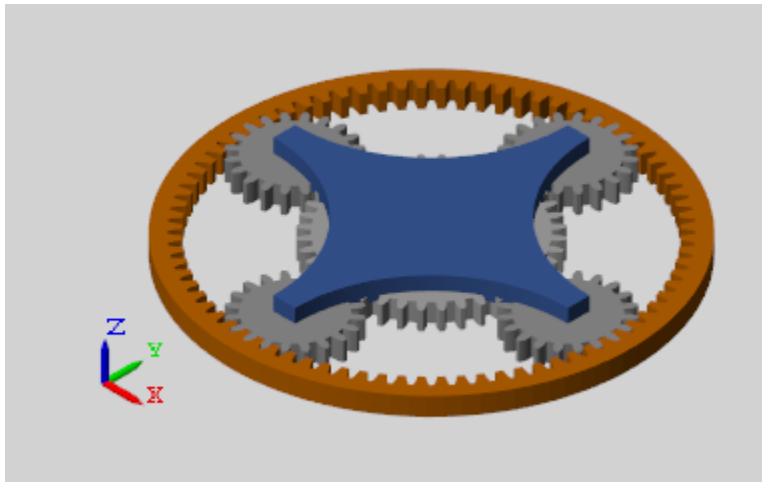
- 8 Simulate the model. To induce motion, try adjusting the velocity state targets in the joint block dialog boxes. Notice that the gear carrier now performs the task of the Distance Constraint block.



You can open a copy of the resulting model. At the MATLAB command line, enter `smdoc_planetary_gear_d`.

## Add More Planet Gears

Experiment with the model by adding more planet gears. Remember that you must change the Carrier rigid body to accommodate any additional planet gears. To see an example with four planet gears, at the MATLAB command line enter `smdoc_planetary_gear_e`.





## Model a Cam Constraint

### In this section...

“Model Overview” on page 3-73

“Geometry Ports” on page 3-74

“Spline Curves” on page 3-76

“Point On Curve Constraints” on page 3-77

“Model Eccentric Cam” on page 3-77

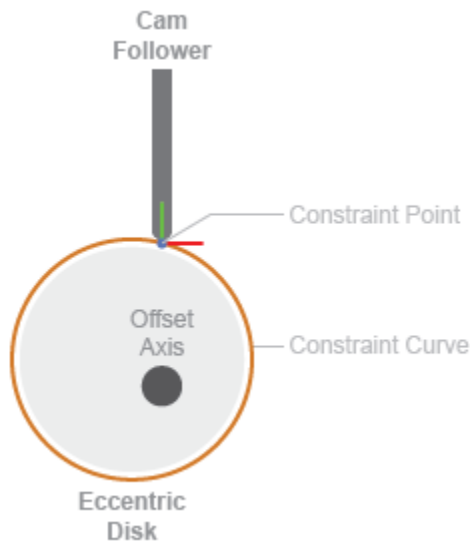
“Model Cam Follower” on page 3-83

“Interactively Create Frame at Follower Tip” on page 3-85

“Constrain Cam and Follower” on page 3-89

### Model Overview

This tutorial shows how to model an eccentric cam mechanism. The mechanism consists of an eccentric disk (the cam) with a lever (the cam follower) mounted on its periphery. The distance between the rotation axis and perimeter of the cam varies with rotation angle, causing the follower to translate in a reciprocating motion.



#### Cam Mechanism Schematic

To work, the cam mechanism must constrain the follower tip (a point) to lie on the cam periphery (a curve). This type of constraint is known as point-on-curve. The same constraint is at work, for example, in a roller coaster cart bound to the perimeter of a track. You model this constraint using the **Point On Curve Constraint** block.

Any frame origin associated with a frame port can be a constraint point. Any curve associated with a geometry port can be a constraint curve. In this example, a frame origin positioned at the follower tip provides the constraint point. A circular spline curve defined in a **Spline** block provides the constraint curve.

#### Geometry Ports

Geometry ports are analogues of frame ports. In the same way that frame ports identify frames on bodies, geometry ports identify curves and surfaces. You use these ports to apply kinematic constraints between the frames, curves, and surfaces that the ports represent.

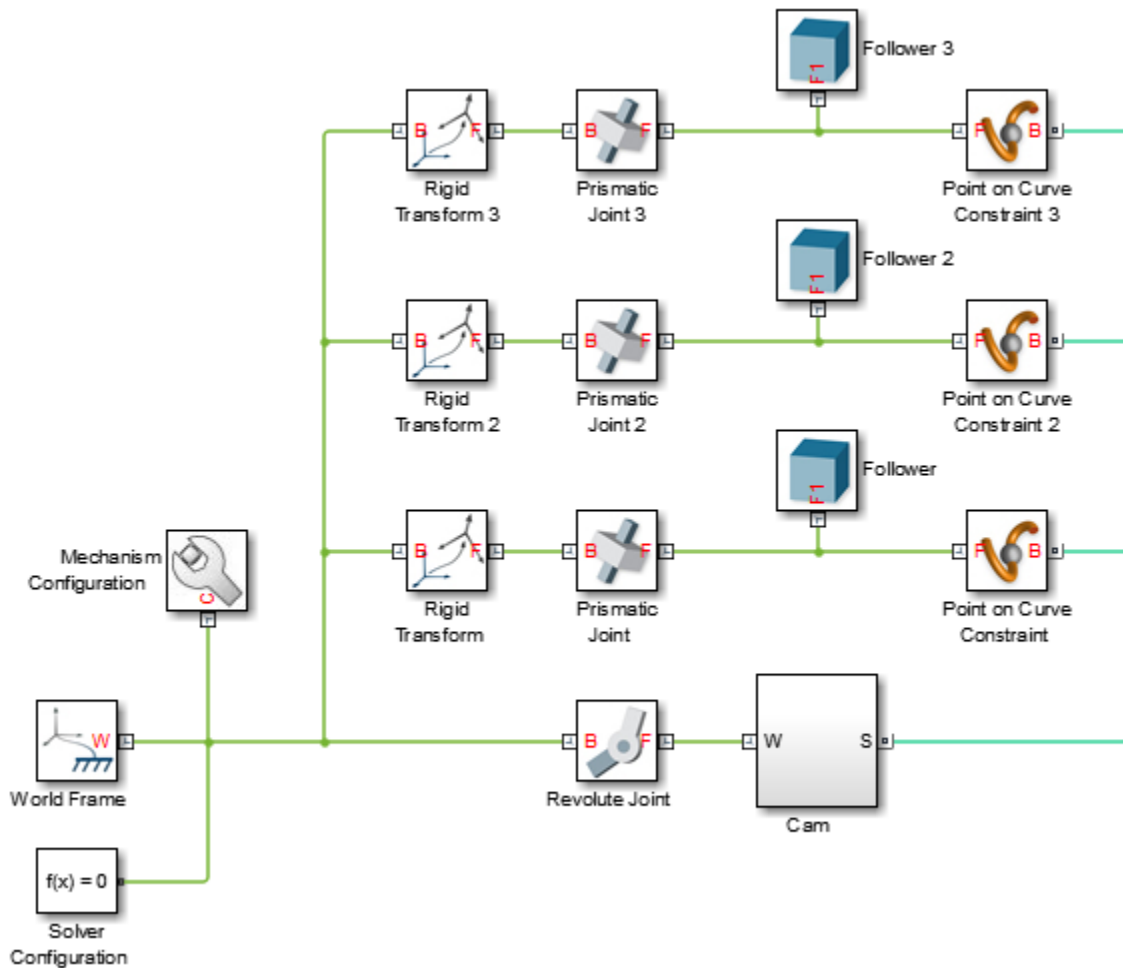
You can then apply kinematic constraints between the frames, curves, and surfaces given by these ports.

If a block has an intrinsic curve or surface definition, its geometry port makes that definition available to other blocks. If a block does not have such a definition, its geometry port enables you to reference one through a geometry connection line.

In this example, the Spline block provides an intrinsic curve definition. You specify this curve in the Spline block dialog box. The Point On Curve Constraint block, which does not have an intrinsic curve definition, then references this curve through a geometry connection line to the Spline block.

You can branch a geometry connection line, for example, so that it joins one Spline block to several Point On Curve Constraint blocks. Such a connection enables you to constrain various cam followers to the same cam or roller coaster carts to the same track.

However, branched or not, a geometry connection line must have exactly one geometry definition. If two blocks with intrinsic geometry definitions attach to the same geometry connection line, Simscape Multibody ignores one. If no such block connects to a geometry connection line, the model does not simulate. The figure shows a valid branched geometry line between one cam and three point-on-curve constraints.



## Spline Curves

The Spline block enables you to model a smooth, continuous curve. In mechanical systems, such curves are generally limited to contours on bodies. To preserve the parallel between your model and the system it represents, use the Spline block in the body subsystem that the spline curve is based on.

In this example, the spline curve represents the cam profile, a 2-D circle on the periphery of the cam. For this reason, you place the Spline block in the subsystem that represents the cam body. This approach treats the spline curve as part not of the constraint definition but of the body definition.

Treating the constraint curve as part of the cam body subsystem enables you to replace one cam subsystem for another without having to change anything else in the model. It also enables you to parameterize both the cam solid properties and spline curve in terms of MATLAB variables defined in a common subsystem mask.

To create the spline curve, the Spline block applies smooth interpolation between the data points you specify in the block dialog box. The interpolation ensures that the curve and its first two derivatives are continuous at each point. These constraints enable you to specify relatively complex curves with a only small number of interpolation points.

Because spline curves need only a small number of interpolation points, they are more computationally efficient than other curve types. However, if you specify a sufficiently large number of interpolation points, spline curves can slow down simulation. Try starting with a small number of interpolation points and gradually adding more until you reach the curve precision you need.

## Point On Curve Constraints

The Point On Curve Constraint block merely applies a constraint between a point and a curve. It defines neither the constraint point nor the constraint curve themselves. You must identify the constraint point externally by connecting the frame port of the Point On Curve Constraint block to another frame port in your model. The frame origin associated with the frame port or line is the constraint point.

Similarly, you must identify the constraint curve externally by connecting the geometry port of the Point On Curve Constraint block to another geometry port in your model. The geometry connection must be to a block with intrinsic curve definition, such as Spline. The curve associated with the geometry port is the constraint curve.

## Model Eccentric Cam

Add and connect blocks.

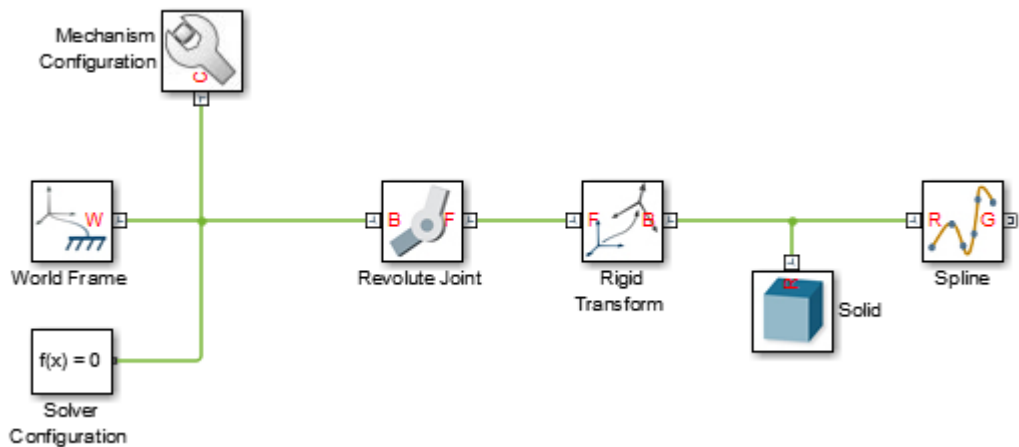
- 1 At the MATLAB command prompt, enter `smnew`. MATLAB opens the Simscape Multibody library and a model template with commonly used blocks. Remove all but

the Mechanism Configuration, World Frame, and Solver Configuration blocks.

- 2 Add these blocks to the model canvas.

Library	Block	Purpose
Joints	Revolute Joint	Provide the cam one rotational degree of freedom relative to the World frame.
Frames and Transforms	Rigid Transform	Specify translational offset between cam center of mass and rotation axis.
Body Elements	Solid	Provide the solid properties of the cam body, including its geometry, inertia, and color.
Curves and Surfaces	Spline	Provide the cam profile for referencing in the Point On Curve Constraint block.

- 3 Connect the blocks as shown. Ensure that the base frame port of the Rigid Transform block faces the reference frame port of the Solid block.



The Spline block provides the curve coordinates relative to its reference frame port. The origin of this frame coincides with the [0, 0] coordinate.

Specify the block parameters.

- 1 In the Rigid Transform block dialog box, specify these parameters.

Parameter	Value
<b>Translation &gt; Method</b>	Standard Axis
<b>Translation &gt; Axis</b>	+Y
<b>Translation &gt; Offset</b>	camOffset, units of in

The string `camOffset` is the translational offset between the cam center of mass and rotation axis specified as a MATLAB variable. You later initialize this and other MATLAB variables in the model workspace.

- 2 In the Solid block dialog box, specify these parameters.

Parameter	Value
<b>Geometry &gt; Shape</b>	Cylinder
<b>Geometry &gt; Radius</b>	r, units of in
<b>Geometry &gt; Length</b>	t, units of in
<b>Inertia &gt; Density</b>	rho
<b>Graphic &gt; Visual Properties &gt; Color</b>	rgbCam

The strings `r`, `t`, `rho`, `rgbCam` are the cam properties specified as MATLAB variables.

- 3 In the Spline curve block, specify these parameters.

Parameter	Value
<b>Interpolation Points</b>	camProfile, units of in
<b>Graphic &gt; Visual Properties &gt; Color</b>	rgbSpline

The string `camProfile` is the spline curve specified as a MATLAB variable. The string `rgbSpline` is the color of the spline curve.

- 4 In the model workspace, define the MATLAB variables referenced in the block dialog boxes by entering this code:

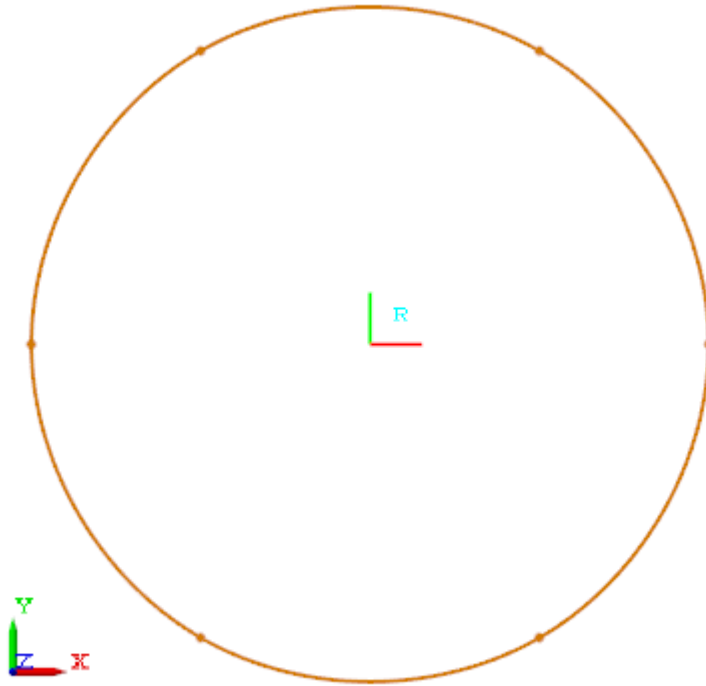
```
% Cam parameters
r = 1; % Cam radius
t = 0.3; % Cam thickness
rho = 2700; % Aluminum density, kg/m^3
rgbCam = [1,1,1]; % Cam color
camOffset = r/3; % Distance from rotation axis to CM

% Spline parameters
rgbSpline = [210,120,0]/255; % Spline color
n = 6; % Number of interpolation points
theta = linspace(0,2*pi*(n-1)/n,n)'; % Angle vector

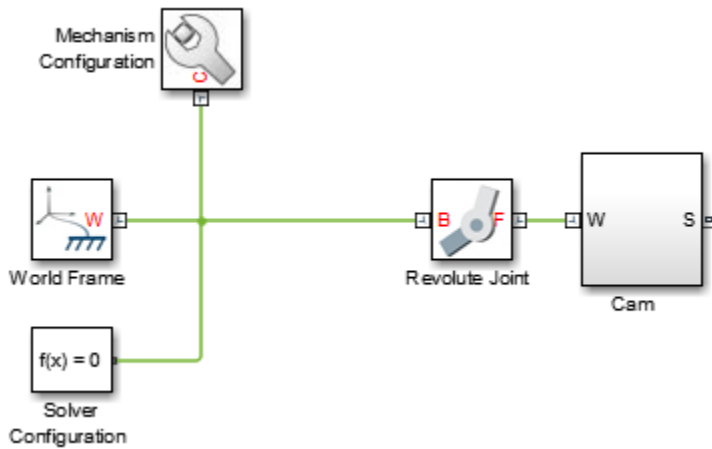
% Spline coordinates
x = r*cos(theta); % Interpolation-point x coordinates
y = r*sin(theta); % Interpolation-point y coordinates
camProfile = [x,y]; % Curve coordinate matrix
```

The spline portion of the code specifies the circular curve shown in the figure. You can view the spline curve in the visualization pane of the Spline block dialog box. The figure shows a top view of the curve.

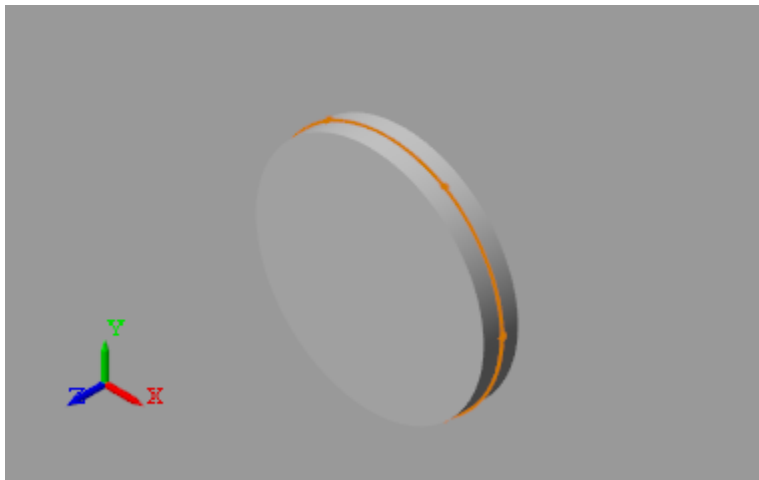




Drag a selection box around the Rigid Transform, Solid, and Spline blocks. Then, select the Create Subsystem action button. Name the new Subsystem block Cam. This block represents the cam body.



To visualize the cam body, update the diagram. You can do this from the Simulink menu bar, by selecting **Simulation > Update diagram**. Mechanics Explorer opens with a visualization of the model. To obtain the view shown, in the Mechanics Explorer toolstrip, set the **View convention** parameter to Y up (XY Front). Then, select the **Isometric View** button.



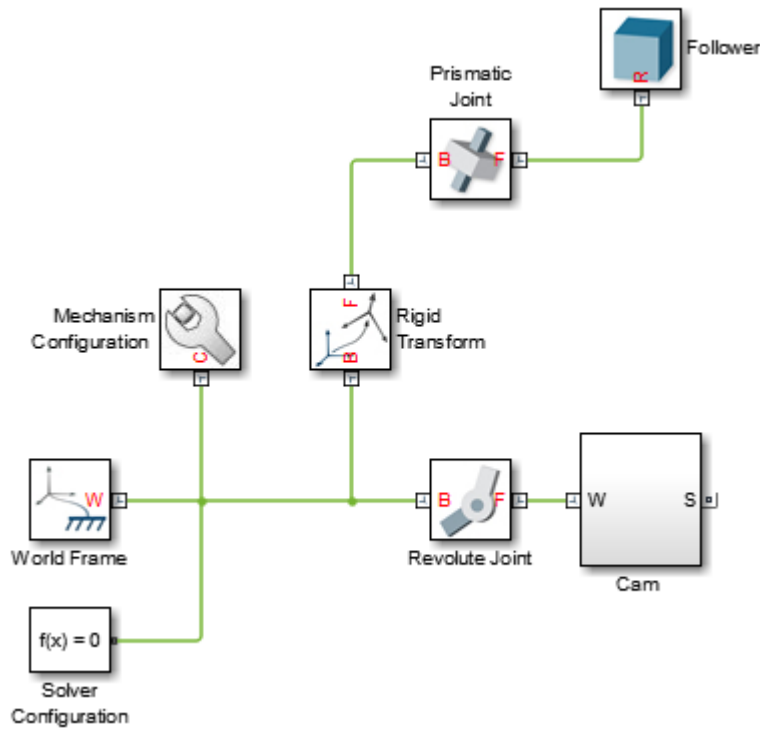
## Model Cam Follower

Add and connect blocks.

- 1 Add these blocks to the model.

Library	Block	Purpose
Joints	Prismatic Joint	Provide the cam follower one translational degree of freedom relative to the World frame.
Frames and Transforms	Rigid Transform	Specify the relative orientation of the cam follower.
Body Elements	Solid	Provide the solid properties of the cam follower, including its geometry, inertia, and color.

- 2 Connect the blocks as shown. Ensure the base frame port of the Rigid Transform block connects to the World frame line. Name the Solid block Follower.



Specify the block parameters.

- 1 In the Rigid Transform block dialog box, specify these parameters.

Parameter	Value
Rotation > Method	Standard Axis
Rotation > Axis	+X
Rotation > Angle	-90

This rotation transform ensures that the translational axis of the cam follower is in the rotation plane of the cam.

- 2 In the Solid block dialog box, specify these parameters.

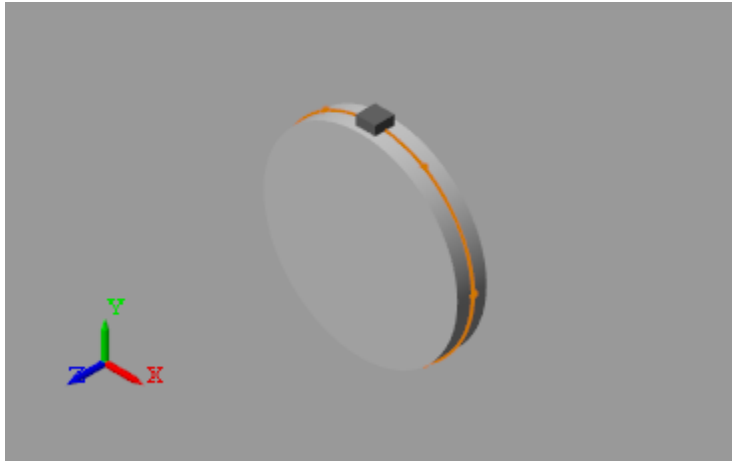
Parameter	Value
Geometry > Shape	Brick
Geometry > Dimensions	sizeFollower, units of in.
Inertia > Density	rho
Graphic > Visual Properties > Color	rgbFollower

The strings `sizeFollower`, `rho`, and `rgbFollower` are the cam follower properties specified as MATLAB variables.

- 3 In the model workspace, initialize the cam follower properties by adding this code:

```
% Follower parameters
sizeFollower = [0.2 0.2 1.5]; % Follower dimensions
rgbFollower = [0.5,0.5,0.5]; % Follower color
```


Update the diagram. Mechanics Explorer shows the updated model visualization. The cam and follower bodies overlap as the follower tip is not yet constrained to the cam periphery.

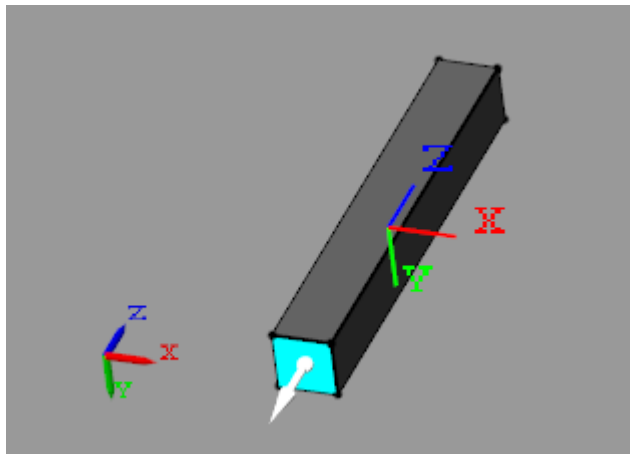


### Interactively Create Frame at Follower Tip

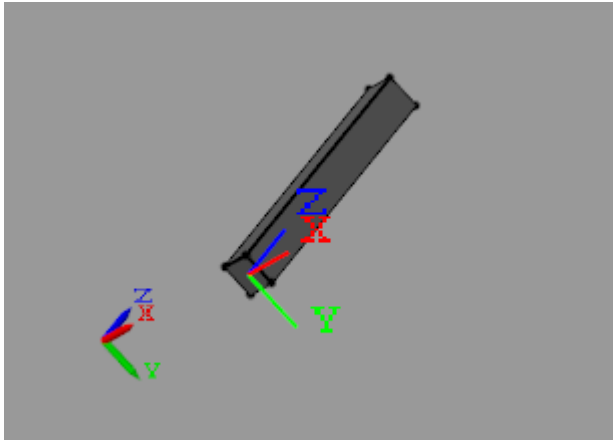
By default, the reference frame of a brick solid such as Follower is located at the center of mass. To apply the cam constraint to the bottom plane of the brick shape, you must

create a new frame. You can create new frames interactively in the Solid block dialog box using the frame-creation interface.

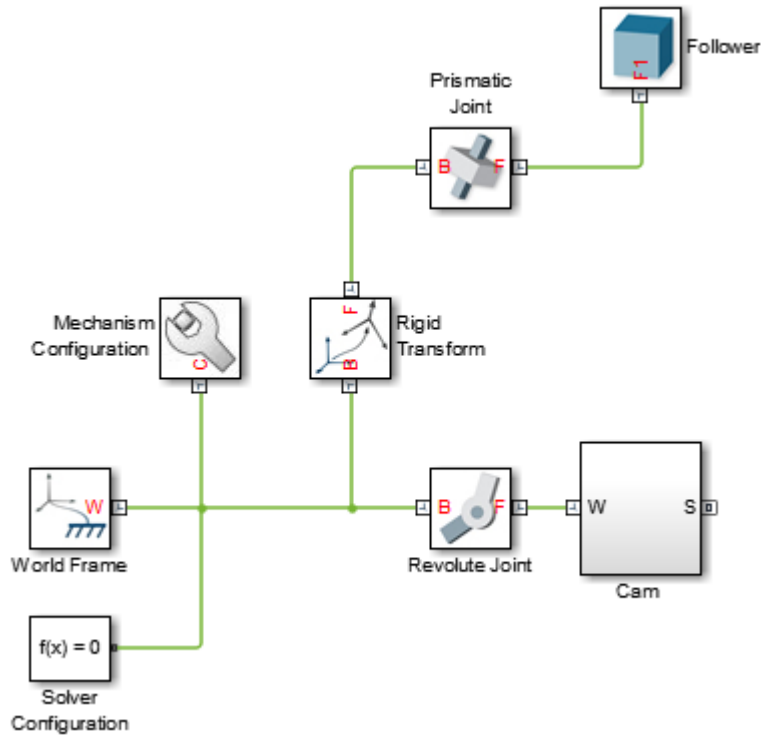
- 1 In the Solid block dialog box, expand the **Frames** area and select the **Create** button. If you make any changes to the block parameters, you must first select the Update Visualization button .
- 2 In the frame-creation interface, under **Frame Origin**, select **Based on Geometric Feature**. This option enables you to place the frame origin at the center of the selected geometric feature, be it a plane, a line, or a point. If you select a point, the frame origin coincides with that point.
- 3 In the visualization pane, rotate the solid and select its bottom plane. This plane is normal to the -z axis. The visualization pane highlights the selected plane. An arrow shows the center and normal vector of this plane.



- 4 Under **Frame Origin**, select the **Use Selected Feature** button. This button sets the center of the selected surface as the origin of the new frame. By default, the frame orientation is that of the solid reference frame.

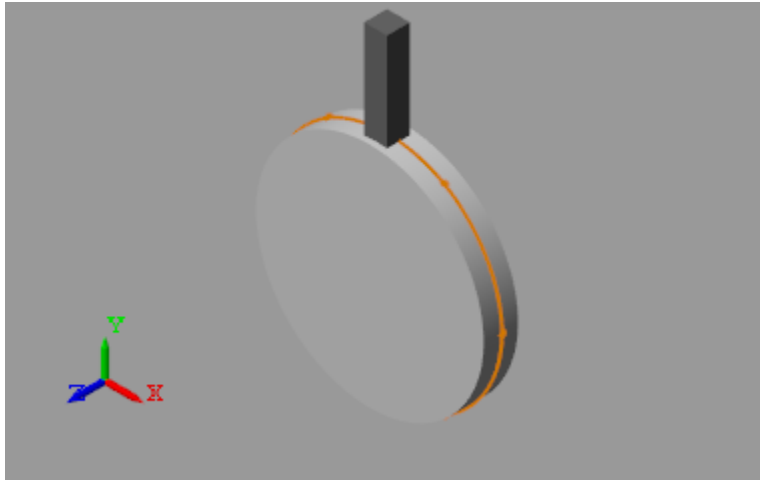


- 5 Select **Save**. The block saves the new frame definition without committing it to the model. If you close the block dialog box without first selecting **Apply** or **OK**, the block discards the new frame definition.
- 6 In the **Frames** area, clear the **Show Port R** check box. The block hides the reference frame port. You do not need this port in this model. Select **OK** to commit your changes to the model.
- 7 Rotate the Follower block and connect its new frame port, labeled F1, to the model as shown.



Update the diagram. Mechanics Explorer shows the updated model visualization. The model assembles with the cam follower in a new position, though the cam and follower bodies are still unconstrained.

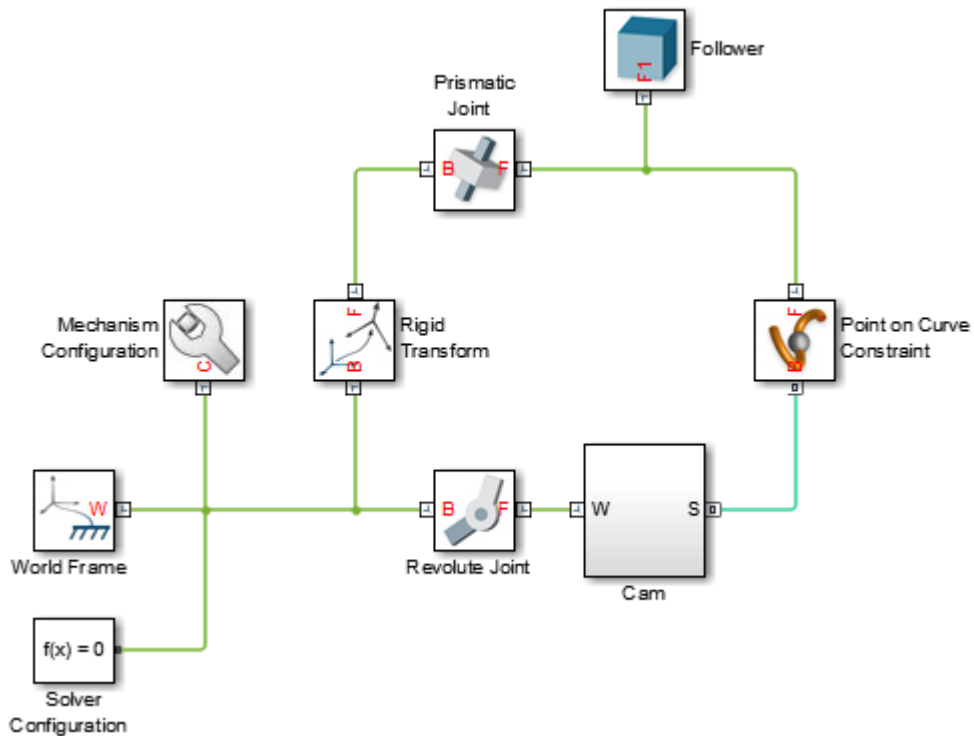




## Constrain Cam and Follower

The model now has the bodies it needs, cam and follower, each with the correct degrees of freedom. To complete the model, you need only constrain the two bodies.

- 1 From the Constraints library, drag a Point On Curve Constraint block.
- 2 Connect the block as shown. Ensure that the base geometry port connects to the geometry port of the Cam subsystem block. The base geometry port identifies the constraint curve.



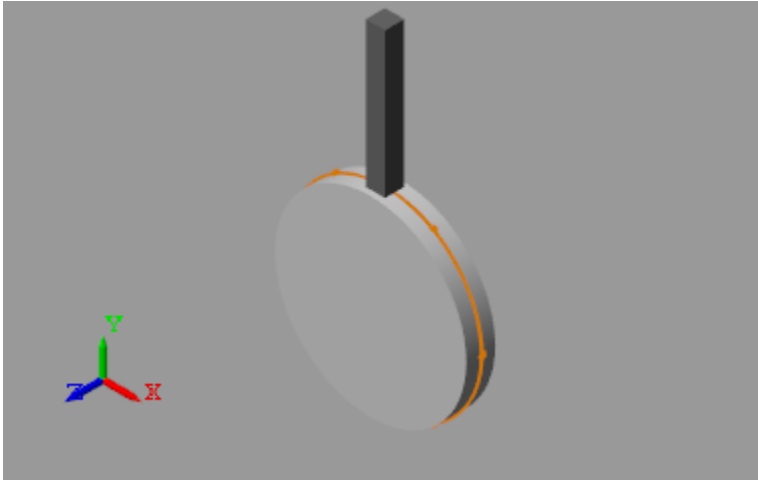
Set the cam in motion by specifying a nonzero initial velocity. Set gravity to zero to ensure uniform motion of the cam.

- 1 In the Revolute Joint block dialog box, specify these parameters.

Parameter	Setting
State Targets > Specify Velocity Target	Select check box.
State Targets > Specify Velocity Target > Value	Enter 0.5. Select units of rev/s.

- 2 In the Mechanism Configuration block, set the **Uniform Gravity** parameter to None.

Simulate the model. You can do this from the Simulink menu bar by selecting **Simulation > Run**. Mechanics Explorer shows a physics-based animation of the cam mechanism.





# Internal Mechanics, Actuation and Sensing

---

- “Modeling and Sensing System Dynamics” on page 4-2
- “Modeling Gravity” on page 4-4
- “Model Gravity in a Planetary System” on page 4-9
- “Joint Actuation” on page 4-25
- “Joint Actuation Limitations” on page 4-34
- “Actuating and Sensing with Physical Signals” on page 4-36
- “Sensing” on page 4-39
- “Force and Torque Sensing” on page 4-42
- “Motion Sensing” on page 4-46
- “Rotational Measurements” on page 4-51
- “Translational Measurements” on page 4-56
- “Measurement Frames” on page 4-64
- “Sense Motion” on page 4-67
- “Specify Joint Actuation Torque” on page 4-73
- “Analyze Motion at Various Parameter Values” on page 4-85
- “Sense Forces and Torques Acting at Joints” on page 4-91
- “Sense Constraint Forces” on page 4-99
- “Specify Joint Motion Profile” on page 4-105
- “Prescribe Joint Motion in Planar Manipulator Model” on page 4-110

## Modeling and Sensing System Dynamics

### In this section...

“Provide Joint Actuation Inputs” on page 4-2

“Specify Joint Internal Mechanics” on page 4-2

“Model Body Interactions and External Loads” on page 4-3

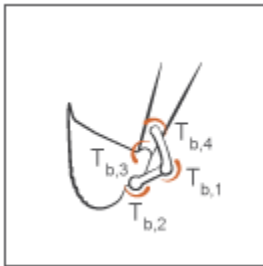
“Sense Dynamical Variables” on page 4-3

### Provide Joint Actuation Inputs



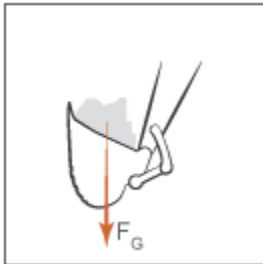
Identify the joints to actuate and the actuation type to use. Then, model the actuation inputs as time-varying physical signals and connect them to the various joints. See “Specify Joint Actuation Torque” on page 4-73 for an example.

### Specify Joint Internal Mechanics



Model damping and spring behavior at joints. Specify joint damping coefficients to model energy dissipation and joint spring constants to model energy storage.

## Model Body Interactions and External Loads



Identify the forces and torques acting at or between bodies not connected by joints. Model these forces and torques explicitly using Forces and Torques blocks. See “Model Gravity in a Planetary System” on page 4-9 for an example.

## Sense Dynamical Variables



Identify the forces, torques, and motion variables to sense. You can sense these variables at joints through Joint blocks. You can also sense motion variables using the Transform Sensor block. See “Sense Motion” on page 4-67 for an example.

## Modeling Gravity

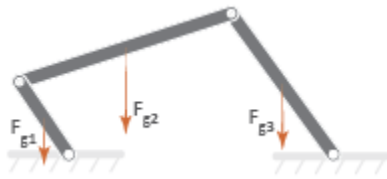
### In this section...

- “Gravity Models” on page 4-4
- “Gravitational Force Magnitude” on page 4-5
- “Force Position and Direction” on page 4-6
- “Gravitational Torques” on page 4-7

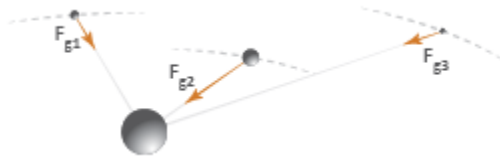
### Gravity Models

Gravity influences motion in many natural and engineered systems. These range in scale from the very large, such as the planets orbiting the sun, to the relatively small, such as the shock absorbers damping gravity-driven oscillations in a car. In Simscape Multibody, you can add gravity to systems like these using three gravity models:

- Uniform gravity, as experienced by most earthbound systems. The force on each body due to uniform gravity depends only on its mass. This force is the same everywhere in space for a given body, though it can vary in time. You model uniform gravity using the **Mechanism Configuration** block.



- Gravitational field, as experienced by the planets in the solar system. The force on each body due to a gravitational field depends not only on its mass but also on its inverse square distance to the field origin. You model a gravitational field using the **Gravitational Field** block.





- Inverse-square law force pair, similar in nature to a gravitational field, but acting exclusively between one pair of bodies. You model an inverse-square law force pair using the Inverse Square Law Force block. You must specify the body masses and force constants explicitly.



## Gravitational Force Magnitude

The force of gravity is an inverse-square law force—that is, one that decays with the square distance from the field origin to the target body. The magnitude of this force,  $F_g$ , follow from Newton’s law of universal gravitation which, for two bodies of mass  $M$  and  $m$  a distance  $R$  apart, states

$$F_g = -G \frac{Mm}{R^2}$$

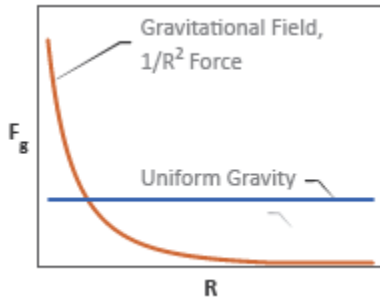
with  $G$  being the gravitational constant. This is the force that you model when you represent gravity through Gravitational Field or Inverse Square Law Force blocks. If the distance between source and target masses is constant, the gravitational force reduces to a simpler form,

$$F_g = -mg$$

with  $g$  being the nominal gravitational acceleration. Near the surface of the Earth, at a distance equal to Earth’s radius from the gravitational field origin, the nominal acceleration equals

$$g = \frac{GM}{R^2} \approx 9.80665 \text{ m/s}^2$$

This is the gravitational force that you model when you represent gravity through the Mechanism Configuration block. The figure shows how the magnitude of the gravitational force ( $F_g$ ) varies with distance ( $R$ ) for a given body under uniform gravity, a gravitational field, and an inverse-square law force pair.



### Force Position and Direction

In a physical system, the force due to a gravitational field acts at a body’s center of mass—automatically computed during simulation—along the imaginary line connecting the field origin to the center of mass. These are also the application point and direction of gravity that the Gravitational Field block provides. See “Simscape Multibody Bodies” on page 2-4 for more information on how Simscape Multibody defines a body subsystem.

Far from the field origin, the field origin-center of mass line remains approximately constant at small-to-moderate displacements, and the force of gravity behaves as if its direction were fixed. This is the approximation used in the Mechanism Configuration block. Gravity still acts at each body’s center of mass, but its direction is now fixed along the gravity vector that you specify.

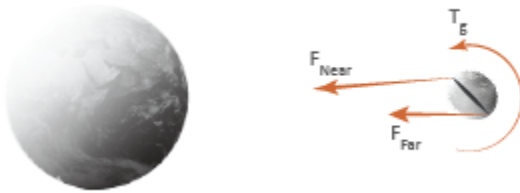
If you want to model the effects of gravity on a point other than a body’s center of mass, you can add a frame at the desired location and apply a gravitational force directly at that frame. You model the force using the Inverse Square Law Force block. This force points along the imaginary line between the two body frames that the Inverse Square Law Force block connects.

The table summarizes the application point and direction of gravity provided by the different blocks.

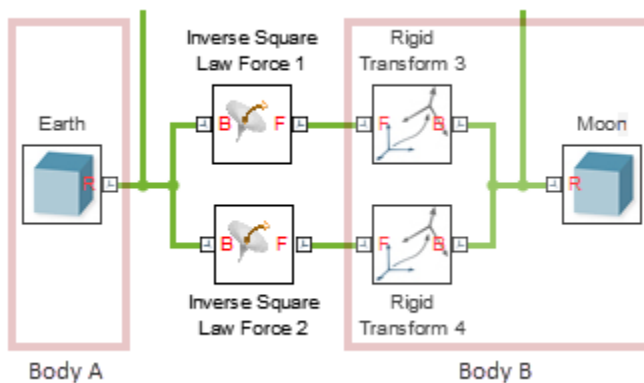
Block	Position	Direction
Mechanism Configuration	Center of Mass	Specified gravity vector
Gravitational Field	Center of Mass	Field origin-center of mass line
Inverse Square Law Force	Connection frames	Base-follower frame line

## Gravitational Torques

A gravitational torque can arise in a large body immersed in a nonuniform gravitational field. The lemon-shaped moon, with its near end perpetually facing Earth, is one example. Being placed at different distances from Earth, the near and far elongated ends experience dissimilar gravitational forces, resulting in a net gravitational torque if the line between the two ends ever falls out of alignment with the center of the Earth.

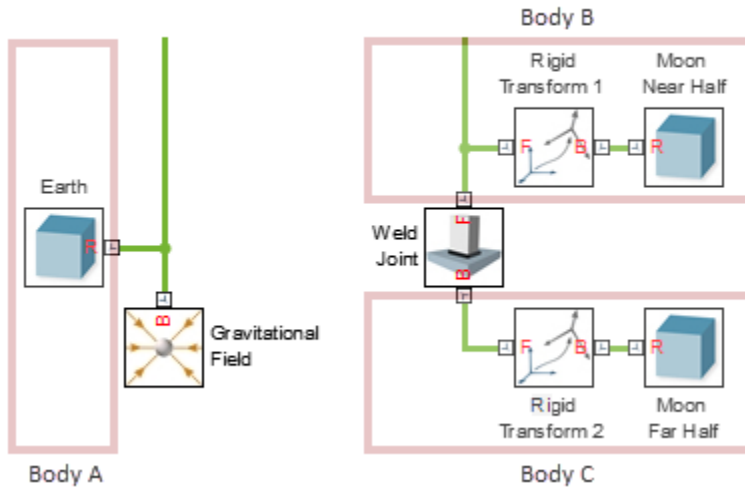


You can model such torques in Simscape Multibody by modeling the different gravitational forces acting on a body. You do this using the Inverse Square Law Force or Gravitational Field block. If you use the Inverse Square Law Force block, you must create additional frames in each body whose response to gravitational torque you want to model. You must then apply a gravitational force to each frame explicitly. The figure shows an example.



**Torque on the moon due to dissimilar gravitational forces at the elongated ends**

If you use the Gravitational Field block, you must split each body into discrete sections and connect them through Weld Joint blocks. The Gravitational Field block automatically applies a force at the center of mass of each section, approximating the compound effect of the different gravitational forces on the body—which in this case is treated as a rigid multibody system. The figure shows an example.



**Torque on the moon due to dissimilar gravitational forces at the elongated ends**

# Model Gravity in a Planetary System

## In this section...

“Model Overview” on page 4-9

“Step 1: Start a New Model” on page 4-10

“Step 2: Add the Solar System Bodies” on page 4-11

“Step 3: Add the Degrees of Freedom” on page 4-15

“Step 4: Add the Initial State Targets” on page 4-17

“Step 5: Add the Gravitational Fields” on page 4-21

“Step 6: Configure and Run the Simulation” on page 4-23

“Open an Example Model” on page 4-24

## Model Overview

This tutorial shows how to simulate the gravity-driven orbits of the major solar system bodies. The model treats the sun and planets as perfect spheres each with three translational degrees of freedom. Planet spin is ignored. Gravitational fields generate the forces that keep the planets in orbit.



Solid blocks represent the solar system bodies and provide their geometries, inertias, and colors. **Cartesian Joint** blocks define the bodies’ degrees of freedom relative to the world frame, located at the solar system barycenter. **Gravitational Field** blocks add the long-range forces responsible for bending the initial planet trajectories into closed elliptical orbits.

The **Cartesian Joint** blocks provide the initial states—positions and velocities—of the sun and planets relative to the world frame. The initial states correspond to the solar system

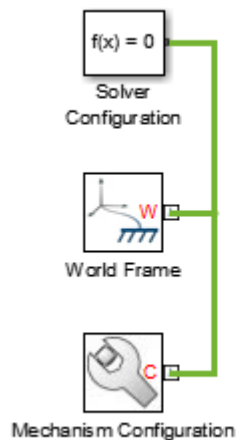
configuration on Jun 20<sup>th</sup>, 2016. They are sourced from ephemeris databases maintained by the Jet Propulsion Laboratory (JPL).

You can query the databases through the JPL Horizons system using a web or telnet interface. Aerospace Toolbox users can alternatively obtain the ephemeris data at the MATLAB command prompt using the `planetEphemeris` function after installing the Aerospace Ephemeris Data support package.

### Step 1: Start a New Model

Open the Simscape Multibody model template and remove all unnecessary blocks. Modify the gravity settings so that you can add gravitational fields to the model. The result provides a starting point for the solar system model.

- 1 At the MATLAB command prompt, enter `smnew`. MATLAB opens a model template with commonly used blocks and suitable solver settings for Simscape Multibody models.
- 2 Cut all but the Mechanism Configuration, Solver Configuration, and World Frame blocks. These three blocks provide the model with gravity settings, solver settings, and a global inertia reference frame.



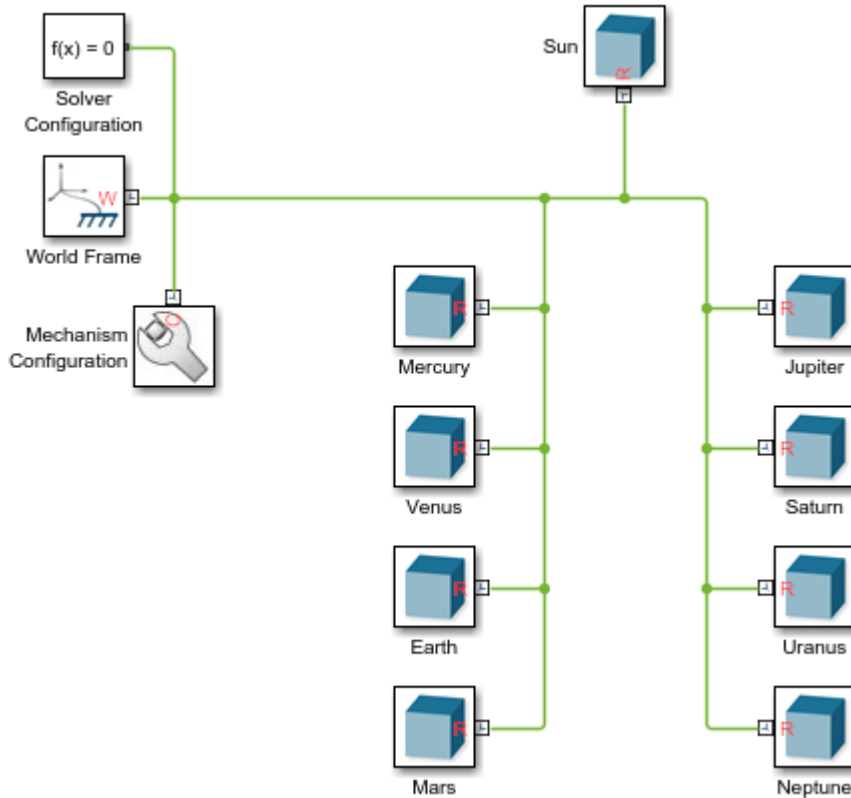
- 3 In the Mechanism Configuration block dialog box, set **Uniform Gravity** to **None**. This setting enables you to model gravity as an inverse-square law force using Gravitational Field blocks instead.

## Step 2: Add the Solar System Bodies

Represent the solar system bodies using Solid blocks. Specify the geometry and inertia parameters in terms of MATLAB variables and initialize these variables in the model workspace using Model Explorer. The variables are data structures named after the solar system bodies using proper-noun capitalization.

### Connect and Configure the Solid Blocks

- 1 Add to the model nine Solid blocks from the Body Elements library. The blocks represent the sun and eight known planets.
- 2 Connect and name the blocks as shown in the figure. The branched frame connection line between the world frame and the planets makes them rigidly connected and coincident in space. You later change this condition using Cartesian Joint blocks.



- 3 In the Solid block dialog boxes, set the **Geometry > Shape** parameter to **Sphere** and the **Inertia > Based on** parameter to **MASS**. The inertia parameter setting enables you to specify the solid mass directly so that you can scale the planet shapes without affecting the model dynamics.
- 4 Specify the following Solid block parameters in terms of MATLAB data structure fields. Enter the field names in the format *Structure.Field*, where *Structure* is the title-case name of the solar system body and *Field* is the string shown in the table—e.g., Sun.R or Earth.RGB.

Block Parameter	Field String
Geometry > Radius	R



Block Parameter	Field String
Inertia > Mass	M
Graphic > Visual Properties > Color	RGB

You later define the new structure fields in the model workspace using Model Explorer.

### Add the Solid Property Initialization Code

- 1 In the Simulink menu bar, select **Tools > Model Explorer**.
- 2 In the Model Hierarchy pane of Model Explorer, expand the node for your model and select **Model Workspace**. The Model Hierarchy pane is on the left side.
- 3 In the Model Workspace pane of Model Explorer, set **Data Source** to **MATLAB Code**. The Model Workspace pane is on the right side.
- 4 In the **MATLAB Code** field, add the initialization code for the sun and planet solid properties. The code is organized into sections named after the solar system bodies. You later add the initial position and velocity data to these sections.

```
% All values are in SI units.
% RGB color vectors are on a normalized 0-1 scale.
% Body dimensions are scaled for visualization purposes.
% Scaling has no impact on model dynamics.

% Scaling
SunScaling = 0.5e2;
TerrestrialPlanetScaling = 1.2e3;
GasGiantScaling = 2.5e2;

% Sun
Sun.M = 1.99e30;
Sun.R = 6.96e8*SunScaling;
Sun.RGB = [1 0.5 0];

% Mercury
Mercury.M =3.30e23;
Mercury.R = 2.44e6*TerrestrialPlanetScaling;
Mercury.RGB = [0.5 0.5 0.5];

% Venus
Venus.M = 4.87e24;
Venus.R = 6.05e6*TerrestrialPlanetScaling;
```

```
Venus.RGB = [1 0.9 0];

% Earth
Earth.M = 5.97e24;
Earth.R = 6.05e6*TerrestrialPlanetScaling;
Earth.RGB = [0.3 0.6 0.8];

% Mars
Mars.M = 6.42e23;
Mars.R = 3.39e6*TerrestrialPlanetScaling;
Mars.RGB = [0.6 0.2 0.4];

% Jupiter
Jupiter.M = 1.90e27;
Jupiter.R = 6.99e7*GasGiantScaling;
Jupiter.RGB = [0.6 0 0.3];

% Saturn
Saturn.M = 5.68e26;
Saturn.R = 5.82e7*GasGiantScaling;
Saturn.RGB = [1 1 0];

% Uranus
Uranus.M = 8.68e25;
Uranus.R = 2.54e7*GasGiantScaling;
Uranus.RGB = [0.3 0.8 0.8];

% Neptune
Neptune.M = 1.02e26;
Neptune.R = 2.46e7*GasGiantScaling;
Neptune.RGB = [0.1 0.7 0.8];
```

### 5 Click **Reinitialize from Source**.

The Solid blocks now have all the numerical data they need to render the planet shapes and colors. Try opening a Solid block dialog box and verify that a sphere now appears in the solid visualization pane.

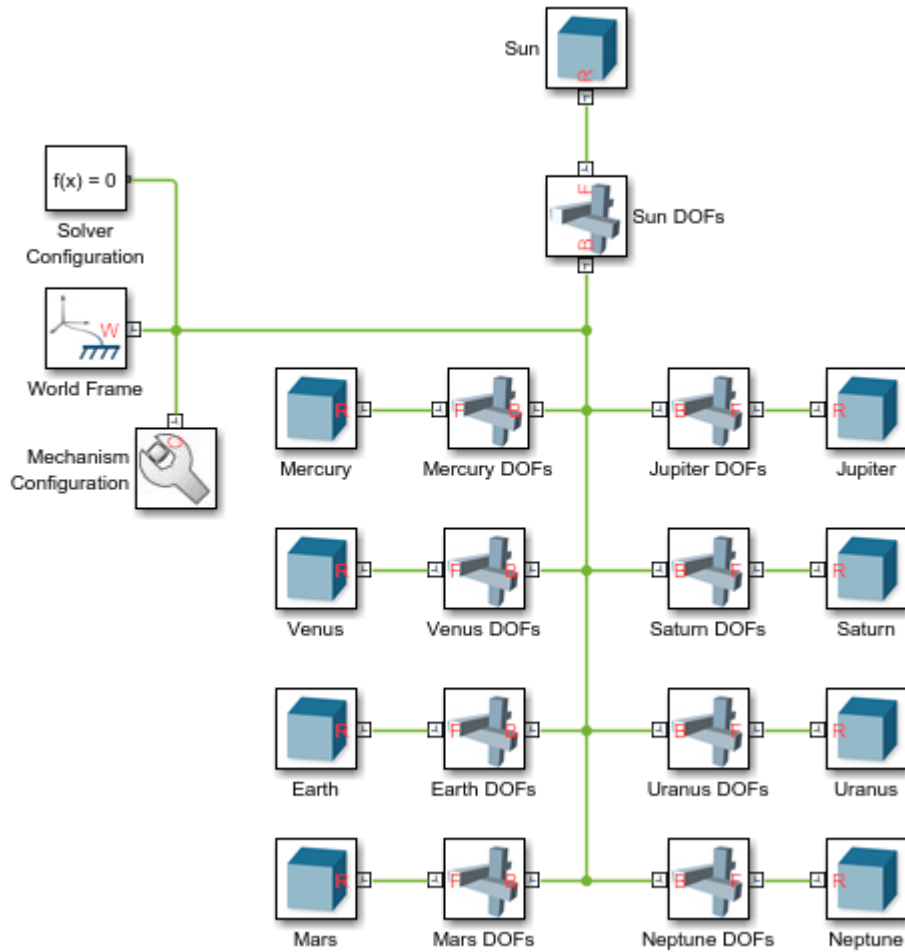


### Earth Solid Visualization

## Step 3: Add the Degrees of Freedom

Add three translational degrees of freedom between the solar system barycenter and each solar system body using Cartesian Joint blocks. You later use these blocks to specify the initial positions and velocities of the solar system bodies.

- 1 Add to the model nine Cartesian Joint blocks from the Joints library. The blocks provide the translational degrees of freedom of the sun and eight known planets.
- 2 Connect and name the blocks as shown in the figure. If you place a block on an existing connection line, Simscape Multibody software automatically connects the block to that line. Flip and rotate the joint blocks to ensure that Solid blocks connect only to follower (F) frame ports.



The sun and planets are no longer rigidly connected. They can now translate relative to each other. They are, however, still coincident in space. To place them at different initial positions and give them initial velocities, you must specify the joint state targets.

## Step 4: Add the Initial State Targets

Specify the sun and planet initial states in terms of MATLAB variables using the Cartesian Joint blocks in your model. Then, initialize the new MATLAB variables in the model workspace using Model Explorer. You define the MATLAB variables as new fields in the existing data structures.

### Configure the Cartesian Joint Blocks

- 1 In the Cartesian Joint block dialog boxes, check the **State Targets > Specify Position Target** and **State Targets > Specify Velocity Target** checkboxes for the X, Y, and Z prismatic joint primitives. These settings enable you to specify the desired initial states of the sun and planets.
- 2 Specify the Cartesian Joint state target values for the X, Y, and Z prismatic joint primitives in terms of MATLAB structure fields. Enter the field names in the format *Structure.Field*, where *Structure* is the title-case name of the solar system body and *Field* is the string shown in the table—e.g., Sun.Px or Earth.Vz.

Joint Primitive Axis	State Target	Field String
X	Position	Px
	Velocity	Vx
Y	Position	Py
	Velocity	Vy
Z	Position	Pz
	Velocity	Vz

You later define the new structure fields in the model workspace using Model Explorer.

### Add the State Target Initialization Code

- 1 In the Simulink menu bar, select **Tools > Model Explorer**.
- 2 In the Model Hierarchy pane of Mechanics Explorer, expand the node for your model and select **Model Workspace**. The Model Hierarchy pane is on the left side.
- 3 In the Model Workspace pane of Model Explorer, set **Data Source** to **MATLAB Code**. The Model Workspace pane is on the right side.
- 4 In the **MATLAB Code** field, add the initialization code for the joint state targets. The new code, shown in blue, consists of the position and velocity components

obtained from the JPL ephemeris databases. You can copy just the new code or replace your entire model workspace code with that shown.

```
% All values are in SI units.  
% RGB color vectors are on a normalized 0-1 scale.  
% Body dimensions are scaled for visualization purposes.  
% Scaling has no impact on model dynamics.
```

```
% Scaling  
SunScaling = 0.5e2;  
TerrestrialPlanetScaling = 1.2e3;  
GasGiantScaling = 2.5e2;
```

```
% Sun  
Sun.M = 1.99e30;  
Sun.R = 6.96e8*SunScaling;  
Sun.RGB = [1 0.5 0];  
Sun.Px = 5.5850e+08;  
Sun.Py = 5.5850e+08;  
Sun.Pz = 5.5850e+08;  
Sun.Vx = -1.4663;  
Sun.Vy = 11.1238;  
Sun.Vz = 4.8370;
```

```
% Mercury  
Mercury.M = 3.30e23;  
Mercury.R = 2.44e6*TerrestrialPlanetScaling;  
Mercury.RGB = [0.5 0.5 0.5];  
Mercury.Px = 5.1979e+10;  
Mercury.Py = 7.6928e+09;  
Mercury.Pz = -1.2845e+09;  
Mercury.Vx = -1.5205e+04;  
Mercury.Vy = 4.4189e+04;  
Mercury.Vz = 2.5180e+04;
```

```
% Venus  
Venus.M = 4.87e24;  
Venus.R = 6.05e6*TerrestrialPlanetScaling;  
Venus.RGB = [1 0.9 0];  
Venus.Px = -1.5041e+10;  
Venus.Py = 9.7080e+10;  
Venus.Pz = 4.4635e+10;  
Venus.Vx = -3.4770e+04;  
Venus.Vy = -5.5933e+03;
```

```
Venus.Vz = -316.8994;

% Earth
Earth.M = 5.97e24;
Earth.R = 6.05e6*TerrestrialPlanetScaling;
Earth.RGB = [0.3 0.6 0.8];
Earth.Px = -1.1506e+09;
Earth.Py = -1.3910e+11;
Earth.Pz = -6.0330e+10;
Earth.Vx = 2.9288e+04;
Earth.Vy = -398.5759;
Earth.Vz = -172.5873;

% Mars
Mars.M = 6.42e23;
Mars.R = 3.39e6*TerrestrialPlanetScaling;
Mars.RGB = [0.6 0.2 0.4];
Mars.Px = -4.8883e+10;
Mars.Py = -1.9686e+11;
Mars.Pz = -8.8994e+10;
Mars.Vx = 2.4533e+04;
Mars.Vy = -2.7622e+03;
Mars.Vz = -1.9295e+03;

% Jupiter
Jupiter.M = 1.90e27;
Jupiter.R = 6.99e7*GasGiantScaling;
Jupiter.RGB = [0.6 0 0.3];
Jupiter.Px = -8.1142e+11;
Jupiter.Py = 4.5462e+10;
Jupiter.Pz = 3.9229e+10;
Jupiter.Vx = -1.0724e+03;
Jupiter.Vy = -1.1422e+04;
Jupiter.Vz = -4.8696e+03;

% Saturn
Saturn.M = 5.68e26;
Saturn.R = 5.82e7*GasGiantScaling;
Saturn.RGB = [1 1 0];
Saturn.Px = -4.2780e+11;
Saturn.Py = -1.3353e+12;
Saturn.Pz = -5.3311e+11;
Saturn.Vx = 8.7288e+03;
Saturn.Vy = -2.4369e+03;
```

```
Saturn.Vz = -1.3824e+03;

% Uranus
Uranus.M = 8.68e25;
Uranus.R = 2.54e7*GasGiantScaling;
Uranus.RGB = [0.3 0.8 0.8];
Uranus.Px = 2.7878e+12;
Uranus.Py = 9.9509e+11;
Uranus.Pz = 3.9639e+08;
Uranus.Vx = -2.4913e+03;
Uranus.Vy = 5.5197e+03;
Uranus.Vz = 2.4527e+03;

% Neptune
Neptune.M = 1.02e26;
Neptune.R = 2.46e7*GasGiantScaling;
Neptune.RGB = [0.1 0.7 0.8];
Neptune.Px = 4.2097e+12;
Neptune.Py = -1.3834e+12;
Neptune.Pz = -6.7105e+11;
Neptune.Vx = 1.8271e+03;
Neptune.Vy = 4.7731e+03;
Neptune.Vz = 1.9082e+03;
```

### 5 Click **Reinitialize from Source**.

The model now has the numerical data it needs to assemble the planets in the position coordinates obtained from the JPL databases. However, a model simulation at this point would show the planets moving in straight-line trajectories. To obtain elliptical orbits, you must complete the model by adding the sun and planet gravitational fields.

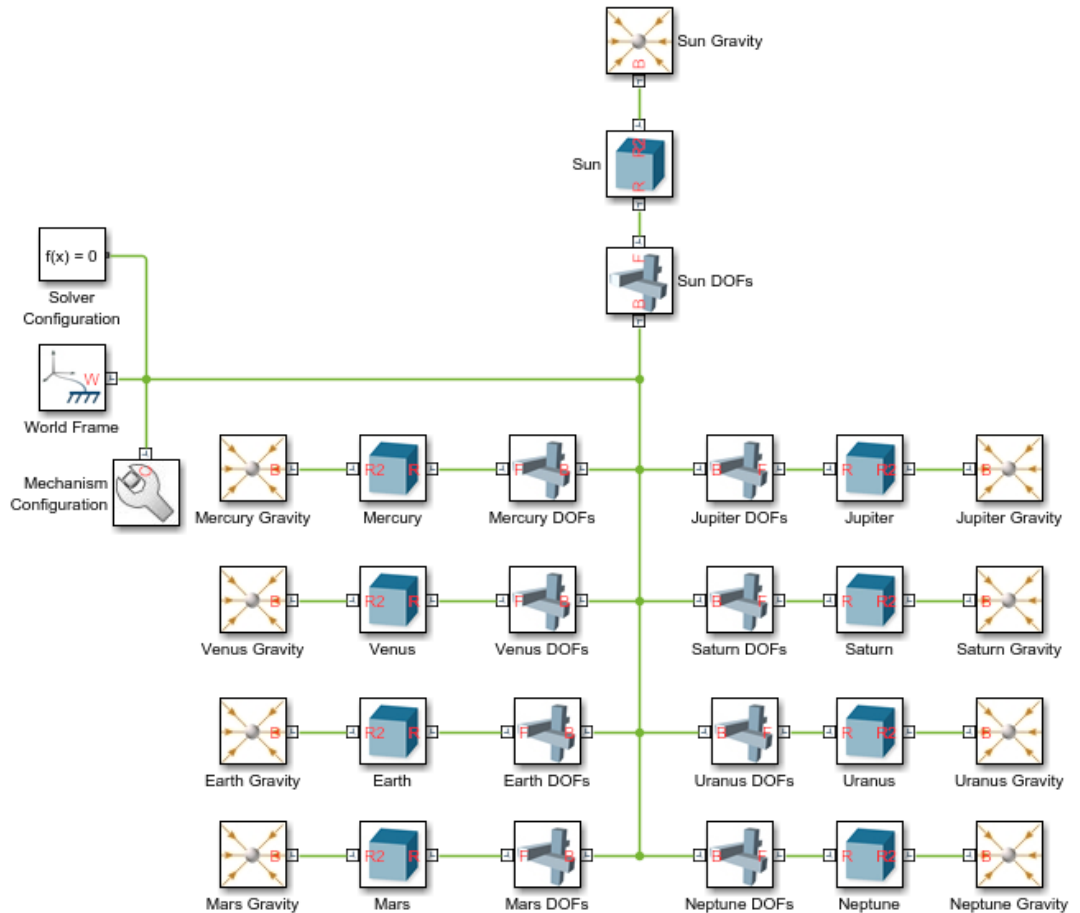




## Step 5: Add the Gravitational Fields

Model the gravitational pull of each solar system body using the Gravitational Field block. This block automatically computes the gravitational pull of a body on all other bodies using Newton's law of universal gravitation.

- 1 In each Solid block dialog box, expand the **Frames** area and click the **Create** button.
- 2 Set the **Frame Name** parameter to R2 and click the **Save** button. The new frame is an exact copy of the reference frame but has a separate frame port. You can use these ports to connect the gravitational field blocks while avoiding crossed connection lines.
- 3 Add to the model nine Gravitational Field blocks from the Forces and Torques library. The blocks provide the gravitational forces that each solar system body exerts on all other bodies.
- 4 Connect and name the blocks as shown in the figure. Ensure that the blocks connect directly to the Solid blocks. Such a connection ensures that the fields are centered on the solid spheres and rigidly connected to them.



- 5 In the Gravitational Field blocks, specify the **Mass** parameter as MATLAB structure field names. Enter the field names in the format *Structure.Field*, where *Structure* is the title-case name of the solar system body and *Field* is the string M—e.g., Sun.M or Earth.M. These fields have been previously defined in the model workspace.

## Step 6: Configure and Run the Simulation

Configure the Simulink solver settings to capture ten earth revolutions in a single simulation. Then, simulate the model and view the resulting solar system animation. Configure the animation settings to play the ten-year animation in the period of a few seconds.

### Configure the Solver Settings

- 1 In the Simulink menu bar, select **Simulation > Model Configuration Parameters**.
- 2 Set the **Stop time** parameter to  $10 \times 365 \times 24 \times 60 \times 60$ . This number, equal to ten years in seconds, allows you to simulate a full ten earth revolutions from Jun 20<sup>th</sup>, 2016 through Jun 20<sup>th</sup>, 2026.
- 3 Set the **Max step size** parameter to  $24 \times 60 \times 60$ . This number, equal to one day in seconds, is small enough to provide smooth animation results. Increase this number if you prefer faster simulation results.

### Update and Simulate the Model

Update the block diagram, for example, by selecting **Simulation > Update Diagram**. Mechanics Explorer opens with a static 3-D display of the model in its initial state. Check that the sun and planets appear in the visualization pane and that their relative dimensions and positions are reasonable.

Run the simulation, for example, by selecting **Simulation > Run Simulation**. Mechanics Explorer plays an animation of the solar system. Note that at the default base playback speed, the planets appear static. You must increase this speed in the Mechanics Explorer animation settings.

### Configure the Animation Settings

- 1 In Mechanics Explorer, select **Tools > Animation Settings**.
- 2 In **Base(1X) Playback Speed**, enter 3153600. This speed corresponds to one earth revolution every ten seconds.
- 3 Pause and play the animation to apply the new base playback speed. The figure shows the animation results at the new speed.



### Open an Example Model

You can open a complete solar system model by entering `smdoc_solar_system_wfield_b` at the MATLAB command prompt.

## Joint Actuation

### In this section...

“Actuation Modes” on page 4-25

“Motion Input” on page 4-28

“Input Handling” on page 4-30

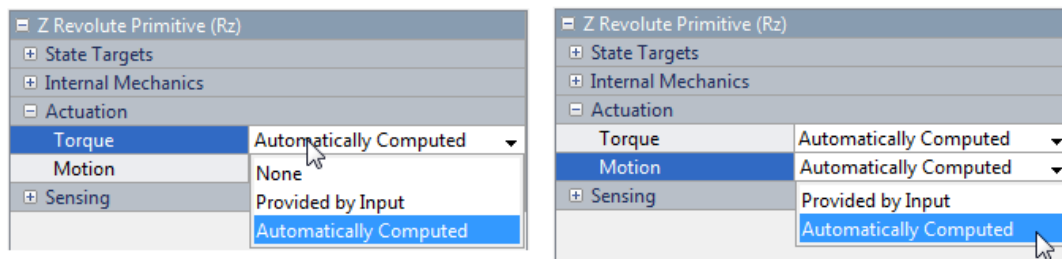
“Assembly and Simulation” on page 4-31

“Specifying Motion Input Derivatives” on page 4-32

### Actuation Modes

Joint blocks provide two actuation parameters. These parameters, **Force/Torque** and **Motion**, govern how the joint behaves during simulation. Depending on the parameter settings you select, a joint block can accept either actuation parameter as input or automatically compute its value during simulation.

An additional setting (**None**) allows you to set actuation force/torque directly to zero. The joint primitive is free to move during simulation, but it has no actuator input. Motion is due indirectly to forces and torques acting elsewhere in the model, or directly to velocity state targets.



Like all joint block parameters, you select the actuation parameter settings for each joint primitive separately. Different joint primitives in the same block need not share the same actuation settings. Using a **Pin Slot Joint** block, for example, you can provide motion input and have actuation torque automatically computed for the **Z Revolute**

**Primitive (Rz)**, while having motion automatically computed with no actuation force for the **X Prismatic Primitive (Px)**.

[-] X Prismatic Primitive (Px)	
+ State Targets	
+ Internal Mechanics	
[-] Actuation	
Force	None ▼
Motion	Automatically Computed ▼
+ Sensing	
[-] Z Revolute Primitive (Rz)	
+ State Targets	
+ Internal Mechanics	
[-] Actuation	
Torque	Automatically Computed ▼
Motion	Provided by Input ▼
+ Sensing	

By combining different **Force/Torque** and **Motion** actuation settings, you can achieve different joint actuation modes. Forward dynamics and inverse dynamics modes are two common examples. You actuate a joint primitive in forward dynamics mode by providing actuation force/torque as input while having motion automatically computed. Conversely, you actuate a joint primitive in inverse dynamics mode by providing motion as input while having actuation force/torque automatically computed.

Other joint actuation modes, including fully computed and fully specified modes, are possible. The table summarizes the different actuation modes that you can obtain by manipulating the actuation parameter settings.

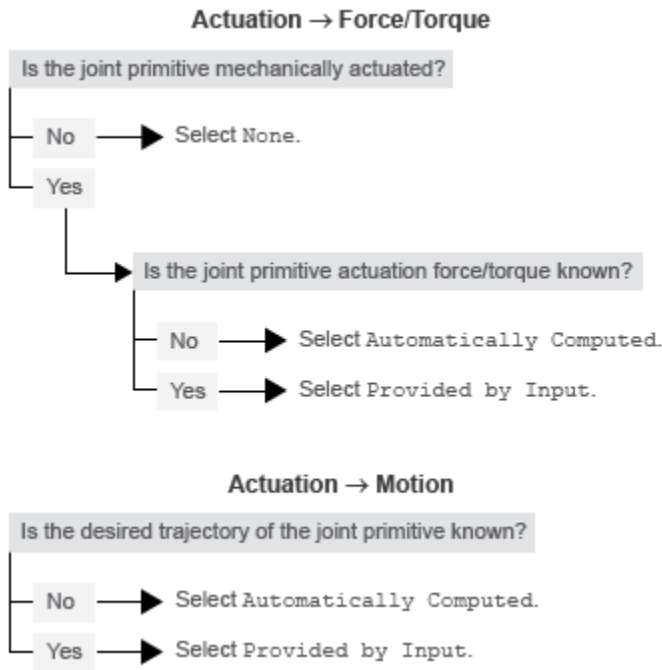
		Actuator Motion	
		Provided by Input	Automatically Computed
Actuator Force/Torque	None	Unactuated Motion	Passive
	Provided by Input	Fully Specified	Forward Dynamics
	Automatically Computed	Inverse Dynamics	Fully Computed

### Joint Actuation Modes

More generally, thinking of joint actuation in terms of the specified or calculated quantities—i.e., force/torque and motion—provides a more practical modeling approach. You may not always know the appropriate mode for a joint but, having planned the model beforehand, you should always know the answers to two questions:

- Is the joint primitive mechanically actuated?
- Is the desired trajectory of the joint primitive known?

By selecting the joint actuation settings based on the answers to these questions, you can ensure that each joint is properly set for your application. The figure shows the proper settings depending on your answers.



### Selecting Joint Primitive Actuation Settings

## Motion Input

The motion input of a joint primitive is a timeseries object specifying that primitive's trajectory. For a prismatic primitive, that trajectory is the position coordinate along the primitive axis, given as a function of time. The coordinate provides the position of the follower frame origin with respect to the base frame origin. The primitive axis is resolved in the base frame.

For a revolute primitive, the trajectory is the angle about the primitive axis, given as a function of time. This angle provides the rotation of the follower frame with respect to the base frame about the primitive axis. The axis is resolved in the base frame.

Spherical joint primitives provide no motion actuation options. You can specify actuation torque for these primitives, but you cannot prescribe their trajectories. Those trajectories are always automatically computed from the model dynamics during simulation.

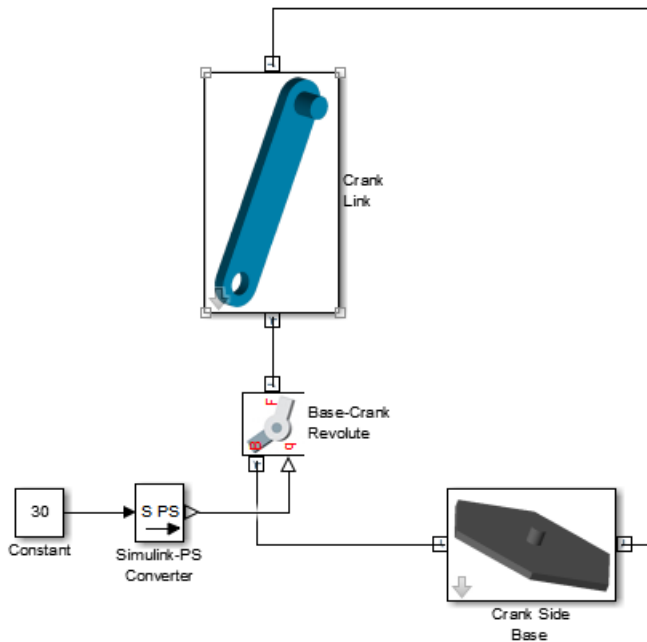


### Zero Motion Prescription

Unlike **Actuation > Force/Torque**, the **Actuation > Motion** parameter provides no zero input option, corresponding to a fixed joint primitive during simulation. You can, however, prescribe zero motion the same way you prescribe all other types of motion: using Simscape and Simulink blocks.

In Simscape Multibody, motion input signals are position-centric. You specify the joint primitive position and, if filtered to the second-order, the **Simulink-PS Converter** block smooths the signal while providing its two time-derivatives automatically. This behavior makes zero motion prescription straightforward: just provide a constant signal to the motion actuation input port of the joint primitive and simulate.

The figure shows an example of zero-motion prescription. A Simulink Constant block provides a constant position value. A Simulink-PS Converter block converts this Simulink signal into a Simscape signal compatible with the motion actuation input port of the Base-Crank Revolute Joint block. Assuming that assembly and simulation are successful, this joint will maintain a fixed angle of 30 degrees, corresponding to the value set in the Simulink Constant block and the units set in the Simulink-PS Converter block.



## Input Handling

When prescribing a joint primitive trajectory, it is practical to specify a single input, the position, and filter that input using a Simulink-PS Converter block. This filter, which must of second-order, automatically provides the two time derivatives of the motion input. Because it also smooths the input signal, the filter can help prevent simulation issues due to sudden changes or discontinuities, such as those present when using a Simulink Step block.

Filtering smooths the input signal over a time scale of the order of the input filtering time constant. The larger the time constant, the greater the signal smoothing, and the more distorted the signal tends to become. The smaller the time constant, the closer the filtered signal is to the input signal, but also the greater the model stiffness—and, hence, the slower the simulation.

As a guideline, the input filtering time constant should be only as small as the smallest relevant time scale in a model. By default, its value is 0.001 s. While appropriate for

many models, this value is often too small for Simscape Multibody models. For faster simulation, start with a value of 0.01 s. Decrease this value for greater accuracy.

If you know the two time derivatives of the motion input signal, you can specify them directly. This approach is most convenient for simple trajectories with simple derivatives. You must, however, ensure that the two derivative signals are compatible with the position signal. If they are not, even when simulation proceeds, results may be inaccurate.

## Assembly and Simulation

Simscape Multibody joints with motion inputs start simulation (**Ctrl+T**) at the initial position dictated by the input signal. This initial position may differ from the assembled state, which is governed by an assembly algorithm optimized to meet the joint state targets, if any. Even in the absence of joint state targets, the assembled state may differ from that at simulation time zero.

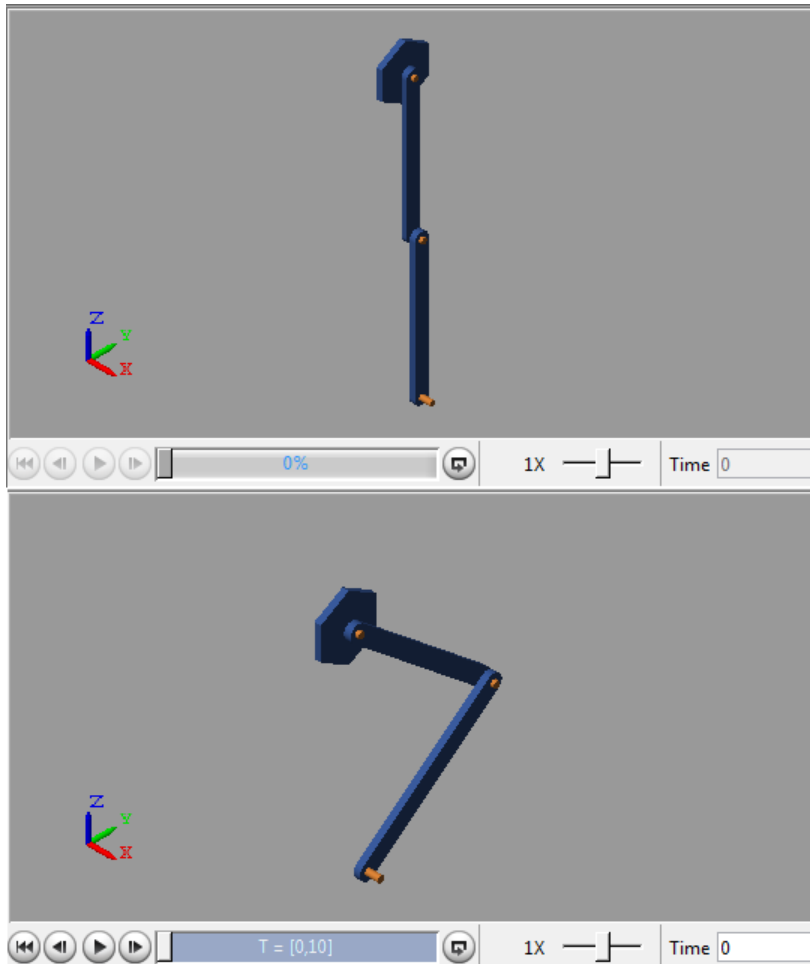
---

**Note:** You obtain the assembled state each time you update the block diagram, e.g., by pressing **Ctrl+D**. You obtain the initial simulation state each time you run the simulation, e.g., by pressing **Ctrl+T**, and pausing at time zero.

---

Due to the discrepancy between the two states, Model Report provides accurate initial state data only for models lacking motion inputs. For models possessing motion inputs, that data is accurate only when the initial position prescribed by the motion input signal exactly matches the initial position prescribed in the joint state targets.

Similarly, Mechanics Explorer displays the initial joint states accurately only for models lacking motion inputs. As it transitions from the assembled state to the initial simulation state, Mechanics Explorer may show a sudden jump if a model contains motion inputs that are incompatible with the joint state targets. You can eliminate the sudden change by making the initial position prescribed by joint motion inputs equal to the initial position prescribed by the joint state targets.



### Specifying Motion Input Derivatives

If filtering the input signal using the Simulink-PS Converter block, you need only to provide the position signal. The block automatically computes the derivatives. You must, however, select second-order filtering in the block dialog box:

- 1 Open the dialog box of the Simulink-PS Converter block and click **Input Handling**.
- 2 In **Filtering and derivatives**, select **Filter input**.

- 3** In **Input filtering order**, select **Second-order filtering**.
- 4** In **Input filtering time constant (in seconds)**, enter the characteristic time over which filter smooths the signal. A good starting value is **0.01** seconds.

If providing the input derivatives directly, you must first compute those derivatives. Then, using the Simulink-PS Converter block, you can provide them to the target joint block. To specify the input derivatives directly:

- 1** Open the Simulink-PS Converter block receiving the input signal and click the **Input Handling** tab.
- 2** In **Filtering and derivatives**, select **Provide input derivative(s)**.
- 3** To specify both derivatives, in **Input derivatives**, select **Provide first and second derivatives**.

The block displays two additional physical signal ports, one for each derivative.

## Related Examples

- “Prescribe Joint Motion in Planar Manipulator Model” on page 4-110
- “Specify Joint Motion Profile” on page 4-105
- “Specifying Motion Input Derivatives” on page 4-32

# Joint Actuation Limitations

In this section...
“Closed Loop Restriction” on page 4-34
“Motion Actuation Not Available in Spherical Primitives” on page 4-34
“Redundant Actuation Mode Not Supported” on page 4-34
“Model Report and Mechanics Explorer Restrictions” on page 4-35
“Motion-Controlled DOF Restriction” on page 4-35

## Closed Loop Restriction

Each closed kinematic loop must contain at least one joint block without motion inputs or computed actuation force/torque. This condition applies even if one of the joints acts as a virtual joint, e.g., the bushing joint in the “Prescribe Joint Motion in Planar Manipulator Model” on page 4-110 example. The joint without motion inputs or automatically computed actuation forces/torques can still accept actuation forces/torques from input.

In models not meeting this condition, you can replace a rigid connection line between two Solid blocks with a Weld Joint block. Since the Weld Joint block represents a rigid connection, this approach leaves the model dynamics unchanged. The advantage of this approach lies in its ability to satisfy the Simscape Multibody closed-loop requirement without altering model dynamics.

## Motion Actuation Not Available in Spherical Primitives

Spherical joint primitives provide no motion actuation parameters. You can prescribe the actuation torque acting on the spherical primitive, but not its desired trajectory. For models requiring motion prescription for three concurrent rotational degrees of freedom, use joint blocks with three revolute primitives instead. These blocks include **Gimbal Joint**, **Bearing Joint**, and **Bushing Joint**.

## Redundant Actuation Mode Not Supported

Redundant actuation, in which the end effector trajectory of a high-degree-of-freedom linkage is prescribed, is not allowed. Such linkages possess more degrees of freedom than are necessary to uniquely position the end effector and, as such, have no single solution.

Models that have more degrees of freedom with automatically computed actuation forces/torques than with prescribed motion inputs cause simulation errors.

## **Model Report and Mechanics Explorer Restrictions**

In models with motion input, the assembled state achieved by updating the block diagram (**Ctrl+D**) does not generally match the initial simulation state at time zero (**Ctrl+T**). This discrepancy is visible in Mechanics Explorer, where it can cause a sudden state change at time zero when simulating a model after updating it. It is also reflected in Model Report, whose initial state data does not generally apply to the simulation time zero when a model has motion inputs.

## **Motion-Controlled DOF Restriction**

The number of degrees of freedom with prescribed trajectories must equal the number of degrees of freedom with automatically computed force or torque. In models not meeting this condition, simulation fails with an error.

## **Related Examples**

- “Prescribe Joint Motion in Planar Manipulator Model” on page 4-110
- “Specify Joint Motion Profile” on page 4-105
- “Specifying Motion Input Derivatives” on page 4-32

## **More About**

- “Joint Actuation” on page 4-25

## Actuating and Sensing with Physical Signals

### In this section...

“Exposing Physical Signal Ports” on page 4-36

“Converting Actuation Inputs” on page 4-36

“Obtaining Sensing Signals” on page 4-37

Some Simscape Multibody blocks provide physical signal ports for actuation input or sensing output. These ports accept or output only Simscape physical signals. If you wish to connect these ports to Simulink blocks, you must use the Simscape converter blocks. The table summarizes the converter blocks that Simscape provides. You can find both blocks in the Simscape Utilities library.

Block	Summary
PS-Simulink Converter	Convert Simscape physical signal into Simulink signal
Simulink-PS Converter	Convert Simulink signal into Simscape physical signal

### Exposing Physical Signal Ports

In Simscape Multibody, most physical signal ports are hidden by default. To expose them, you must select an actuation input or sensing output from the block dialog box. Blocks that provide physical signal ports include certain Forces and Torques blocks as well as Joint blocks. Each port has a unique label that identifies the actuation/sensing parameter. For the ports that a block provides, see the reference page for that block.

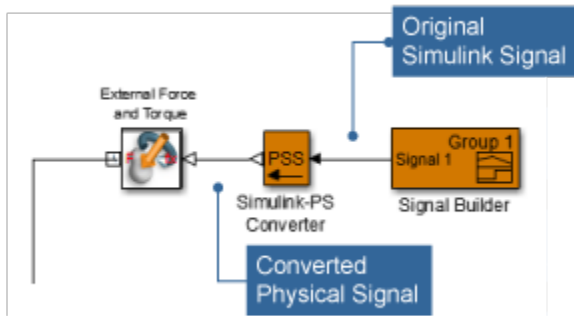
### Converting Actuation Inputs

To provide an actuation signal based on Simulink blocks, you use the **Simulink-PS Converter** block:

- 1 Specify the desired actuation signal using Simulink blocks.
- 2 Connect the Simulink signal to the input port of a Simulink-PS Converter block.
- 3 Connect the output port of the Simulink-PS Converter block to the input port of the Simscape Multibody block that you want to provide the actuation signal to.



In the figure, the connection line that connects to the input port of the Simulink-PS Converter block represents the original Simulink signal. The connection line that connects to the output port of the same block represents the converted physical signal. This is the signal that you must connect to the actuation ports in Simscape Multibody blocks.

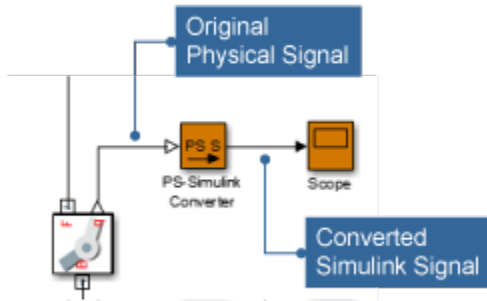


## Obtaining Sensing Signals

To connect the sensing signal of a Simscape Multibody block to a Simulink block, you use the PS-Simulink Converter block:

- 1 Connect the Simscape Multibody sensing port to the input port of a PS-Simulink Converter block.
- 2 Connect the output port of the PS-Simulink Converter block to the Simulink block of your choice.

The figure shows how you can connect a Simscape Multibody sensing signal to a Simulink Scope block.



# Sensing

## In this section...

“Sensing Overview” on page 4-39

“Variables You Can Sense” on page 4-40

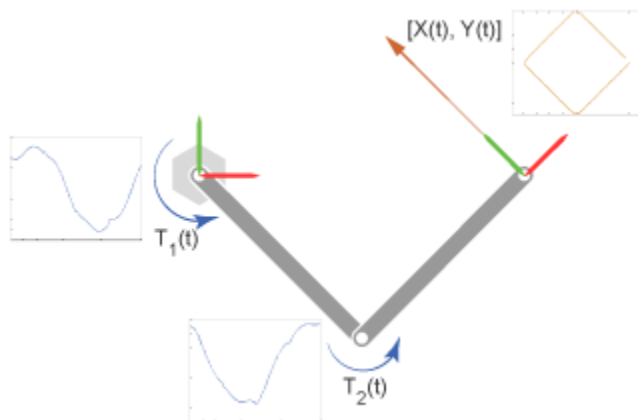
“Blocks with Sensing Capability” on page 4-40

“Sensing Output Format” on page 4-40

## Sensing Overview

Sensing enables you to perform analytical tasks on a model. For example, you can perform inverse dynamic analysis on a robotic manipulator model. By prescribing the end-effector trajectory and sensing the joint actuation forces and torques, you can obtain the time-varying profile of each joint actuation input.

The variables you prescribe, the model inputs, and those you sense, the model outputs, determine which types of analysis you can perform. By changing the model inputs and outputs, you can perform numerous other analysis types. For example, to perform forward kinematic analysis on the robotic manipulator model, you can prescribe the manipulator joint trajectories and sense the resulting end-effector trajectory.



### Variables You Can Sense

To support various analytical tasks, Simscape Multibody software provides a wide range of variables that you can sense. Each variable belongs to either of two categories:

- Motion variables — Linear and angular position, velocity, and acceleration. Linear variables are available in different coordinate systems, including Cartesian, spherical, and cylindrical. Angular variables are available in different formats, including quaternion, axis-angle, and transform matrix.
- Force and torque variables — Actuation, constraint, and total forces and torques acting at a joint, as well as certain forces and torques acting outside of a joint.

### Blocks with Sensing Capability

The entire sensing capability spans multiple Simscape Multibody blocks. Two types of blocks provide motion sensing:

- Joint blocks — Motion sensing between the base and follower port frames of a joint block. Variables that you can sense are organized by joint primitive (prismatic, revolute, or spherical).
- Transform Sensor block — Motion sensing between any two frames in a model. This block provides the most comprehensive motion sensing capability in Simscape Multibody.

Two types of blocks provide force and torque sensing:

- Joint blocks — Actuation, constraint, and total force and torque sensing between the base and follower port frames. Actuation force and torque sensing is arranged by joint primitive.
- Constraint blocks — Constraint force and torque between the base and follower port frames.
- Certain Forces and Torques blocks — Total force the block exerts between the base and follower port frames. Only certain Forces and Torques blocks provide this type of sensing. Blocks that do include **Spring and Damper Force** and **Inverse Square Law Force**.

### Sensing Output Format

Each sensing output is in a physical signal format. You can convert physical signals into Simulink signals using Simscape converter blocks, e.g., for plotting purposes using

the **Scope** block. For information on how to use physical signals in Simscape Multibody models, see “Actuating and Sensing with Physical Signals” on page 4-36.

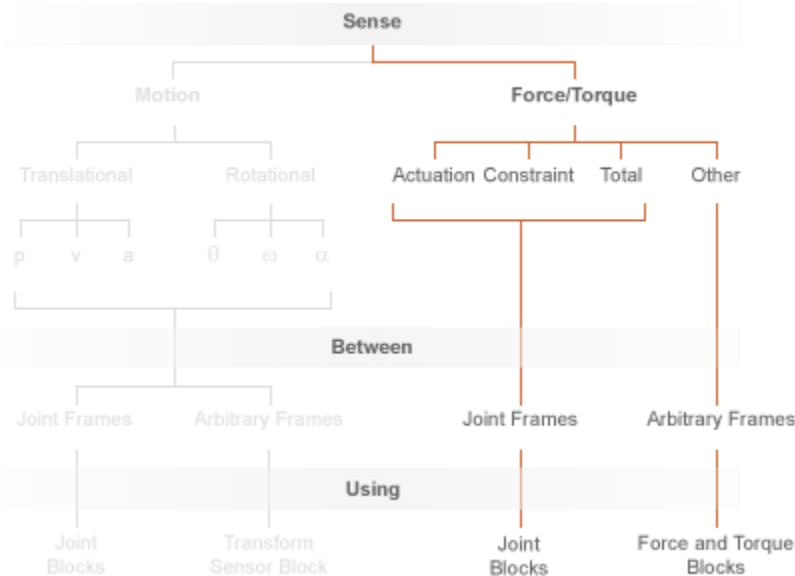
## Force and Torque Sensing

In this section...
“Blocks with Force and Torque Sensing” on page 4-42
“Joint Forces and Torques You can Sense” on page 4-43
“Force and Torque Measurement Direction” on page 4-45

### Blocks with Force and Torque Sensing

Blocks with force and torque sensing appear in two Simscape Multibody libraries:

- **Forces and Torques** — Sense the magnitude of certain forces not explicitly provided by input. Blocks with force sensing include **Inverse Square Law Force** and **Spring and Damper Force**. Each block can sense only the magnitude of its own force.
- **Joints** — Sense various forces and torques acting directly at a joint. All joint blocks provide force and torque sensing. However, the specific force and torque types that you can sense vary from joint to joint. Force and torque sensing is available strictly between the rigid bodies the joint connects.



**Force and Torque Sensing in Simscape Multibody**

**Joint Forces and Torques You can Sense**

Forces and torques that you can sense at a joint fall into two categories:

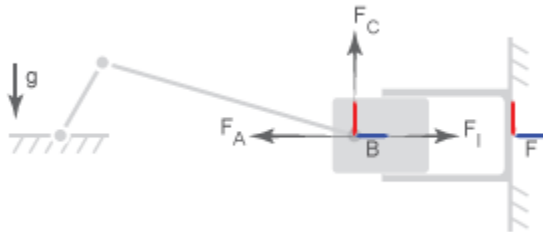
- Joint primitive forces and torques. Each such force or torque is individually computed for a given joint primitive. Joint actuator forces and torques belong to this category.
- Composite forces and torques. Each such force or torque is computed in aggregate for an entire joint. Constraint and total forces and torques belong to this category.

The table summarizes the different joint forces and torques.

Force/Torque Type	Acts On	Measures
Actuator	Individual joint primitives	Force or torque driving an individual joint primitive. The sensed force or torque can be provided by input

Force/Torque Type	Acts On	Measures
Constraint	Entire joints	or it can be automatically computed based on joint motion inputs in a model. Aggregate constraint force or torque opposing motion normal to the joint degrees of freedom. By definition, these forces and torques act orthogonally to the joint primitive axes.
Total	Entire joints	Net sum of all forces or torques acting between the joint port frames. These include actuator, internal, and constraint forces and torques.

The figure shows a basic example of these forces acting on a crank-slider piston.



In the figure:

- $F_A$  is the actuator force, which drives the piston toward the crank link.
- $F_I$  is the internal spring and damper force, which resists motion of the piston with respect to the chamber.
- $F_C$  is the constraint force, which opposes the effect of gravity on the piston, preventing it from falling.



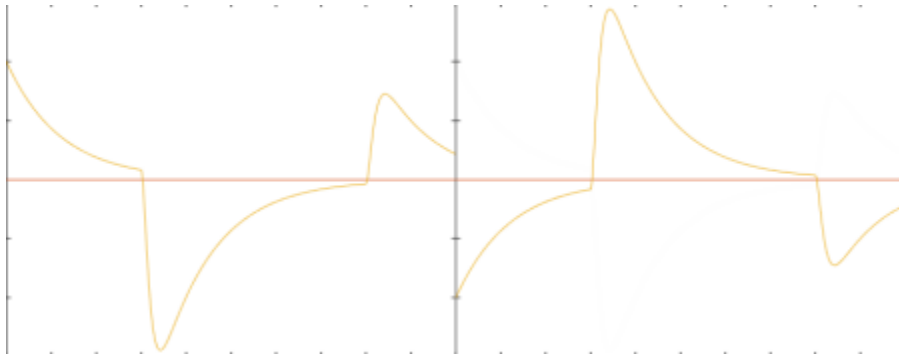
The total force equals the net sum of  $F_A$ ,  $F_L$ , and  $F_C$ .

## Force and Torque Measurement Direction

In accordance with Newton's third law of motion, a force or torque acting between two joint port frames accompanies an equal and opposite force or torque. If the base port frame of a Prismatic Joint block exerts a force on the follower port frame, then the follower port frame exerts an equal force on the base frame. When sensing composite forces and torques in joint blocks, you can specify which of the two to sense:

- Follower on base — Sense the force or torque that the follower port frame exerts on the base port frame.
- Base on follower — Sense the force or torque that the base port frame exerts on the follower port frame.

The figure shows the effect of reversing the measurement direction. Reversing this direction changes the measurement sign.



## Motion Sensing

<b>In this section...</b>
“Sensing Spatial Relationships Between Joint Frames” on page 4-46
“Sensing Spatial Relationships Between Arbitrary Frames” on page 4-48

In Simscape Multibody, you can sense the spatial relationship between two frames using two types of blocks:

- **Transform Sensor** — Sense the spatial relationship between any two frames in a model. Parameters that you can sense with this block include position, velocity, and acceleration of the linear and angular types. This block provides the most extensive motion sensing capability in the Simscape Multibody libraries.
- **Joint blocks** — Sense the spatial relationship between the base and follower frames of a Joint block. Parameters that you can sense with a Joint block include the position and its first two time derivatives (velocity and acceleration) for each joint primitive.

These blocks output a physical signal for each measurement that you specify. You can use the sensing output of these blocks for analysis or as input to a control system in a model.

### Sensing Spatial Relationships Between Joint Frames

To sense the spatial relationship between the base and follower frames of a Joint block, you can use the Joint block itself. For each joint primitive, the dialog box provides a **Sensing** menu with basic parameters that you can measure. These parameters include the position, velocity, and acceleration of the follower frame with respect to the base frame. If the sensing menu of the dialog box does not provide the parameters that you wish to sense, use the Transform Sensor block instead. See “Sensing Spatial Relationships Between Arbitrary Frames” on page 4-48.

The sensing capability of a joint block is limited to the base and follower frames of that joint block. Every measurement provides the value of a parameter for the joint follower frame with respect to the joint base frame. If sensing the spatial relationship with a spherical joint primitive, you can also select the frame to resolve the measurement in. To sense the spatial relationship between any other two frames, use the Transform Sensor block instead.

If the joint primitive is of the revolute or spherical type, the parameters correspond to the rotation angle, angular velocity, and angular acceleration, respectively. If the joint

primitive is of the prismatic type, the parameters correspond to the offset distance, linear velocity, and linear acceleration, respectively.

Regardless of joint primitive type, each parameter that you select applies only to the joint primitive it belongs to. For example, selecting **Position** in the **Z Revolute Primitive (Rz) > Sensing** menu exposes a physical signal port that outputs the rotation angle of the follower frame with respect to the base frame *about the base frame Z axis*.

The table lists the port label for each parameter that you can sense using a joint block. The first column of the table identifies the parameters that you can select. The remaining three columns identify the port labels for the three joint primitive menus that the dialog box can contain: **Spherical**, **Revolute**, and **Prismatic**.

---

**Note:** For parameter descriptions, see the reference pages for **Spherical Joint**, **Revolute Joint**, and **Prismatic Joint** blocks.

---

Parameter	Spherical	Revolute	Prismatic
Position	Q	q	p
Velocity	w	w	v
Velocity (X/Y/Z)	wx/wy/wz	N/A	N/A
Acceleration	b	b	a
Acceleration (X/Y/Z)	bx/by/bz	N/A	N/A

A joint block can contain multiple revolute and prismatic joint primitives. For blocks with multiple primitives of the same type, the port labels include an extra letter identifying the joint primitive axis. For example, the **Position** port label for the Z prismatic primitive of a Cartesian Joint block is pz.

### Select Joint Parameters To Sense

To select the spatial relationship parameters that you wish to sense:

- 1 Open the dialog box for the joint block to sense the spatial relationship across.
- 2 In the **Sensing** menu of the block dialog box, select the parameters to sense.

The block exposes one physical signal port for each parameter that you select. The label of each port identifies the parameter that port outputs.

## Sensing Spatial Relationships Between Arbitrary Frames

To sense the spatial relationship between two arbitrary frames in a model, you use the Transform Sensor block. The dialog box of this block provides a set of menus that you can use to select the parameters to sense. These parameters include position, velocity, and acceleration of the linear and angular types.

Every measurement provides the value of a parameter for the follower frame with respect to the base frame, resolved in the measurement frame that you choose. You can connect the base and follower frame ports of the Transform Sensor block to any two frames in a model. To measure a parameter for a different frame, connect the follower frame port to the frame line or port that identifies that frame. Likewise, to measure a parameter for the same frame but with respect to a different frame, connect the base frame port to the frame line or port that identifies that frame. Finally, to resolve a measurement in a different frame, select a different measurement frame in the block dialog box. For more information about measurement frames, see “Measurement Frames” on page 4-64. For more information about frame lines and ports, see “Representing Frames” on page 1-11.

Selecting a parameter from the block dialog box exposes the corresponding physical signal port in the block. Use this port to output the measurement for that parameter. To identify the port associated with each parameter, each port uses a unique label.

The table lists the port labels for each angular parameter that you can sense. The first column of the table identifies the parameters that you can select. The remaining three columns identify the port labels for the three angular parameter menus in the dialog box: **Rotation**, **Angular Velocity**, and **Angular Acceleration**. Certain parameters belong to one menu but not to others. N/A identifies the parameters that do not belong to a given menu—e.g. Angle, which is absent from the Angular Velocity.

---

**Note:** For parameter descriptions, see the Transform Sensor reference page.

---

Parameter	Rotation	Angular Velocity	Angular Acceleration
Angle	q	N/A	N/A
Axis	axs	N/A	N/A
Quaternion	Q	Qd	Qdd
Transform	R	Rd	Rdd

Parameter	Rotation	Angular Velocity	Angular Acceleration
Omega X/Omega Y/ Omega Z	N/A	wx/wy/wz	N/A
Alpha X/Alpha Y/ Alpha Z	N/A	N/A	bx/by/bz

The table lists the port labels for each linear parameter that you can sense. As in the previous table, the first column identifies the parameters that you can select. The remaining three columns identify the port labels for the three linear parameter menus in the dialog box: **Translation**, **Velocity**, and **Acceleration**.

Parameter	Rotation Port	Angular Velocity Port	Angular Acceleration Port
X/Y/Z	x/y/z	vx/vy/vz	ax/ay/az
Radius	rad	vrad	arad
Azimuth	azm	vazm	aazm
Distance	dst	vdst	adst
Inclination	inc	vinc	ainc

### Select Transform Sensor Parameters To Sense

To select the spatial relationship parameters that you wish to sense:

- 1 Open the Transform Sensor dialog box.
- 2 Expand the menu for the parameter group that parameter belongs to.

E.g. **Rotation** for parameter **Angle**.

- 3 Select the check box for that parameter.

The block exposes one physical signal port for each parameter that you select. The label of each port identifies the parameter that port outputs.

### Related Examples

- “Sense Motion” on page 4-67
- “Specify Joint Actuation Torque” on page 4-73

### **More About**

- “Rotational Measurements” on page 4-51
- “Translational Measurements” on page 4-56
- “Measurement Frames” on page 4-64

# Rotational Measurements

**In this section...**

“Rotation Sensing Overview” on page 4-51

“Measuring Rotation” on page 4-51

“Axis-Angle Measurements” on page 4-51

“Quaternion Measurements” on page 4-52

“Transform Measurements” on page 4-54

## Rotation Sensing Overview

You can measure frame rotation in different formats. These include axis-angle, quaternion, and transform. The different formats are available through the **Transform Sensor** block and, to a limited extent, in joint blocks <sup>1</sup>. The choice of measurement format depends on the model. Select the format that is most convenient for the application.

## Measuring Rotation

Rotation is a relative quantity. The rotation of one frame is meaningful only with respect to another frame. As such, blocks with rotation sensing capability require two frames to make a measurement: measured and reference frames. In these blocks, the follower frame port identifies the measured frame; the base frame port identifies the reference frame of the measurement.

Simscape Multibody defines the rotation formats according to standard conventions. In some cases, more than one convention exists. This is the case, for example, of the quaternion. To properly interpret rotation measurements, review the definitions of the rotation formats.

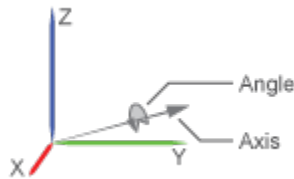
## Axis-Angle Measurements

Axis-angle is one of the simpler rotation measurement formats. This format uses two parameters to completely describe a rotation: axis vector and angle. The usefulness of the axis-angle format follows directly from Euler’s rotation theorem. According to the

---

1. Weld Joint is an exception

theorem, any 3-D rotation or rotation sequence can be described as a pure rotation about a single fixed axis.



To measure frame rotation in axis-angle format, use the Transform Sensor block. The block dialog box contains separate **Axis** and **Angle** parameters that you can select to expose the corresponding physical signal (PS) ports (labeled axis and q, respectively). Because the axis-angle parameters are listed separately, you can choose to measure the axis, the angle, or both.

Rotation	
Angle	<input type="checkbox"/>
Axis	<input type="checkbox"/>
Quaternion	<input type="checkbox"/>
Transform	<input type="checkbox"/>

The axis output is a 3-D unit vector in the form  $[a_x, a_y, a_z]$ . This unit vector encodes the rotation direction according to the right-hand rule. For example, a frame spinning in a counterclockwise direction about the +X axis has rotation axis  $[1\ 0\ 0]$ . A frame spinning in a clockwise direction about the same axis has rotation axis  $[-1\ 0\ 0]$ .

The angle output is a scalar number in the range  $0-\pi$ . This number encodes the extent of rotation about the measured axis. By default, the angle is measured in radians. You can change the angle units in the **PS-Simulink Converter** block used to interface with Simulink blocks.

## Quaternion Measurements

The quaternion is a rotation representation based on hypercomplex numbers. This representation uses a 4-vector containing one scalar ( $S$ ) and three vector components ( $V_x$ ,



$V_y, V_z$ ). The scalar component encodes the rotation angle. The vector components encode the rotation axis.

A key advantage of quaternions is the singularity-free parameter space. Mathematical singularities, present in Euler angle sequences, result in the loss of rotational degrees of freedom. This phenomenon is known as gimbal lock. In Simscape Multibody, gimbal lock causes numerical errors that lead to simulation failure. The absence of singularities means that quaternions are more robust for simulation purposes.

To measure frame rotation in quaternion format, use:

- Transform Sensor block, if measuring rotation between two general frames. The **Rotation** menu of the dialog box contains a **Quaternion** option that you can select to expose the corresponding physical signal port (labeled Q).

Rotation	
Angle	<input type="checkbox"/>
Axis	<input type="checkbox"/>
Quaternion	<input type="checkbox"/>
Transform	<input type="checkbox"/>

- Joint block possessing spherical primitive, if measuring 3–D rotation between the two joint frames. The **Sensing** menu of the dialog box contains a **Position** option that you can select to expose the corresponding physical signal port (also labeled Q). For more information, see **Spherical Joint** block reference page.

Sensing	
Position	<input type="checkbox"/>
Velocity (X)	<input type="checkbox"/>
Velocity (Y)	<input type="checkbox"/>
Velocity (Z)	<input type="checkbox"/>
Velocity	<input type="checkbox"/>
Acceleration (X)	<input type="checkbox"/>
Acceleration (Y)	<input type="checkbox"/>
Acceleration (Z)	<input type="checkbox"/>
Acceleration	<input type="checkbox"/>

The quaternion output is a 4-element row vector  $Q = (S \ V)$ , where:

$$S = \cos\left(\frac{\theta}{2}\right)$$

and

$$\mathbf{V} = [V_x V_y V_z] \sin\left(\frac{\theta}{2}\right)$$

$\theta$  is the rotation angle. The angle can take any value between 0– $\pi$ .  $[V_x, V_y, V_z]$  is the rotation axis. Axis components can take any value between 0–1.

### Transform Measurements

The rotation transform is a 3×3 matrix that encodes frame rotation. In terms of base frame axes  $[x, y, z]_B$ , the follower frame axes  $[x, y, z]_F$  are:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_B = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_F$$

Each matrix column contains the coordinates of a follower frame axis resolved in the base frame. For example, the first column contains the coordinates of the follower frame X-axis, as resolved in the base frame. Similarly, the second and third columns contain the coordinates of the Y and Z-axes, respectively. Operating on a vector with the rotation matrix transforms the vector coordinates from the follower frame to the base frame.

You can sense frame rotation in terms of a rotation matrix using the Transform Sensor block. The dialog box for this block contains a **Transform** option that when selected exposes a physical signal port labeled R. Use this port to output the rotation matrix signal, for example, for processing and analysis in a Simulink subsystem—after converting the output physical signal to a Simulink signal through the PS-Simulink Converter block.

Rotation	
Angle	<input type="checkbox"/>
Axis	<input type="checkbox"/>
Quaternion	<input type="checkbox"/>
Transform	<input type="checkbox"/>

## **Related Examples**

- “Sense Motion” on page 4-67
- “Specify Joint Actuation Torque” on page 4-73

## **More About**

- “Motion Sensing” on page 4-46
- “Translational Measurements” on page 4-56
- “Measurement Frames” on page 4-64

## Translational Measurements

In this section...
“Translation Sensing Overview” on page 4-56
“Measuring Translation” on page 4-56
“Cartesian Measurements” on page 4-57
“Cylindrical Measurements” on page 4-59
“Spherical Measurements” on page 4-61

### Translation Sensing Overview

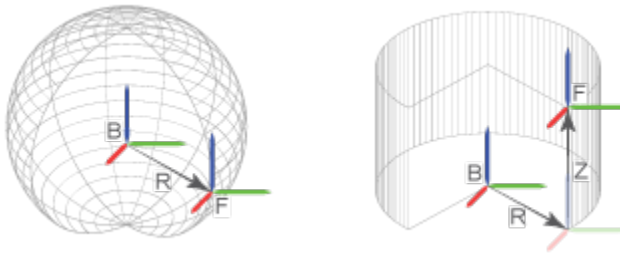
You can measure frame translation in different coordinate systems. These include Cartesian, cylindrical, and spherical systems. The different coordinate systems are available through the **Transform Sensor** block and, to a limited extent, through the **Joint** blocks. The choice of coordinate system depends on the model. Select the coordinate system that is most convenient for your application.

### Measuring Translation

Translation is a relative quantity. The translation of one frame is meaningful only with respect to another frame. As such, blocks with translation sensing capability require two frames to make a measurement: measured and reference frames. In these blocks, the follower frame port identifies the measured frame; the base frame port identifies the reference frame of the measurement.

Some measurements are common to multiple coordinate systems. One example is the **Z**-coordinate, which exists in both Cartesian and cylindrical systems. In the **Transform Sensor** dialog box, coordinates that make up more than one coordinate system appear only once. Selecting **Z** outputs translation along the **Z**-axis in both Cartesian and cylindrical coordinate systems.

Other measurements are different but share the same name. For example, radius is a coordinate in both spherical and cylindrical systems. The spherical radius is different from the cylindrical radius: the former is the distance between two frame origins; the latter is the distance between one frame origin and a frame **Z**-axis.

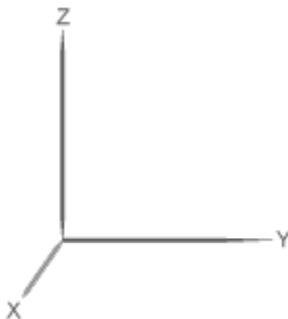


To differentiate between the two radial coordinates, Simscape Multibody uses the following convention:

- Radius — Cylindrical radial coordinate
- Distance — Spherical radial coordinate

## Cartesian Measurements

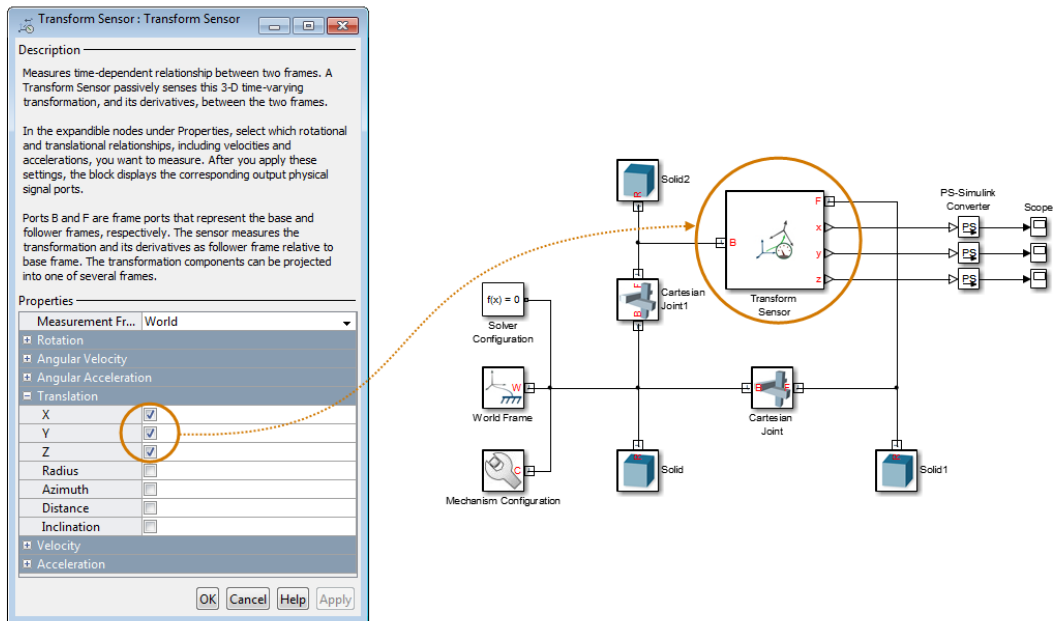
The Cartesian coordinate system uses three linear coordinates—X, Y, and Z—corresponding to three mutually orthogonal axes. Cartesian translation measurements have units of distance, with meter being the default. You can use the **PS-Simulink Converter** block to select a different physical unit when interfacing with Simulink blocks.



### Transform Sensor

You can select any of the Cartesian axes in the Transform Sensor for translation sensing. This is true even if translation is constrained along any of the Cartesian axes. Selecting the Cartesian axes exposes physical signal ports x, y, and z, respectively.

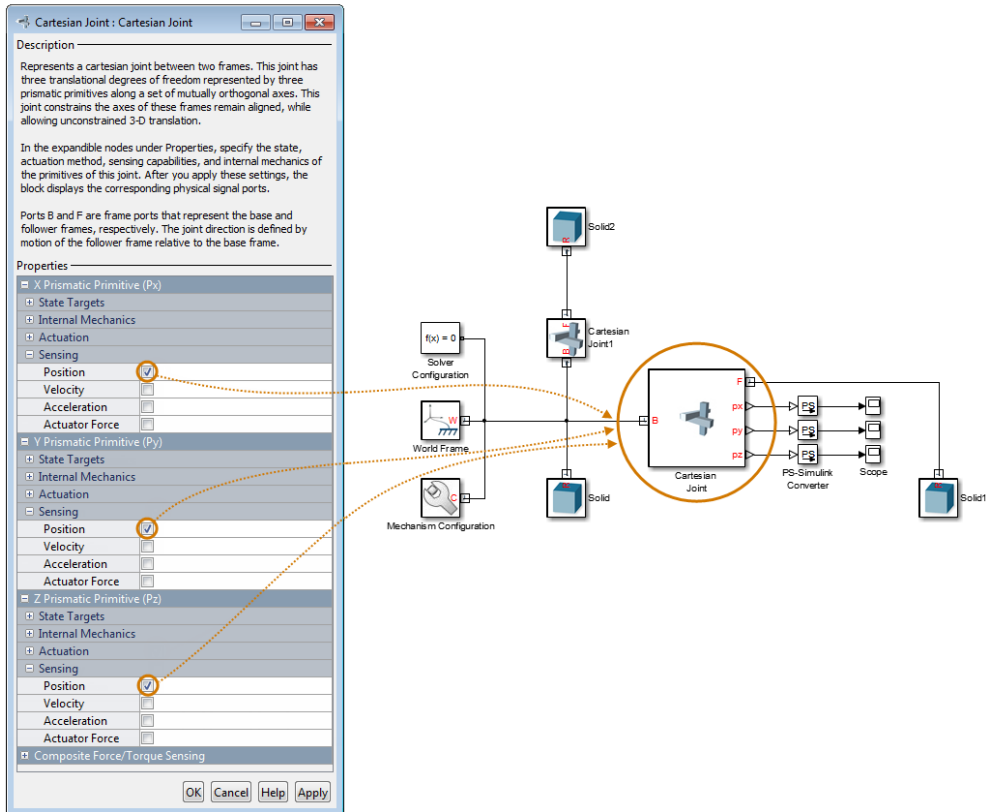
The figure shows a simple model using a Transform Sensor block to measure frame translation along all three Cartesian axes. The measurement gives the relative translation of the follower port frame with respect to the base port frame. These frames are, respectively, the Solid1 and Solid2 reference port frames. For more information, see “Representing Frames” on page 1-11.



### Joints

With joint blocks, you can sense translation along each prismatic primitive axis. Selecting a sensing parameter from a prismatic primitive menu exposes the corresponding physical signal port. For example, if you select **Position** from the **Z Prismatic Primitive (Pz)** of a Cartesian Joint block, the block exposes physical signal port z.

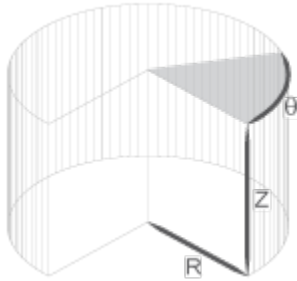
The figure shows a simple model using a **Cartesian Joint** block to sense frame translation along the three Cartesian axes. The measurement gives the relative translation of the follower port frame with respect to the base port frame. These frames coincide with the Solid1 and Solid reference port frames.



## Cylindrical Measurements

The cylindrical coordinate system uses one angular and two linear coordinates. The linear coordinates are the cylinder radius,  $R$ , and length,  $Z$ . The angular coordinate is the azimuth,  $\phi$ , about the length axis. Linear coordinates have units of distance, with meter being the default. The angular coordinate has units of angle, with radian being the

default. You can use the PS-Simulink Converter block to select a different physical unit when interfacing with Simulink blocks.



### Transform Sensor

Only the Transform Sensor block can sense frame translation in cylindrical coordinates. In the dialog box of this block, you can select one or more cylindrical coordinates to measure. The cylindrical coordinates are named **Z**, **Radius**, and **Azimuth**. Selecting the cylindrical coordinates exposes physical signal ports *z*, *rad*, and *azm*, respectively.

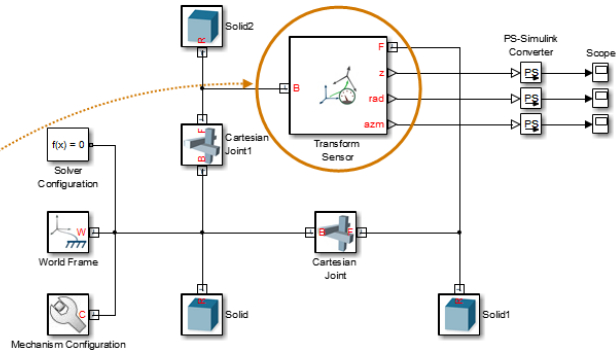
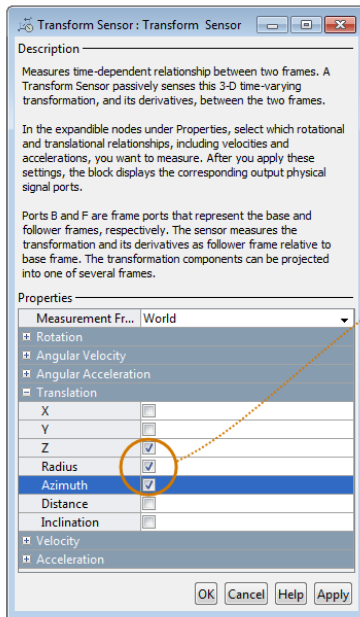
---

**Note:** **Z** belongs to both Cartesian and cylindrical systems.

---

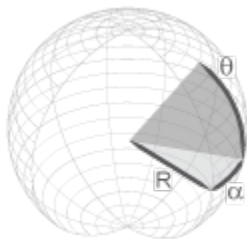
The figure shows a simple model using a Transform Sensor block to measure frame translation along all three cylindrical axes. The measurement gives the relative translation of the follower port frame with respect to the base port frame. These frames are, respectively, the Solid1 and Solid2 reference port frames.





## Spherical Measurements

The spherical coordinate system uses two angular coordinates and one linear coordinate. The linear coordinate is the spherical radius,  $R$ . The angular coordinates are the azimuth,  $\phi$ , and inclination,  $\theta$ . The linear coordinate has units of distance, with meter being the default. The angular coordinates have units of angle, with radian being the default. You can use the PS-Simulink Converter block to select a different physical unit when interfacing with Simulink blocks.



### Transform Sensor

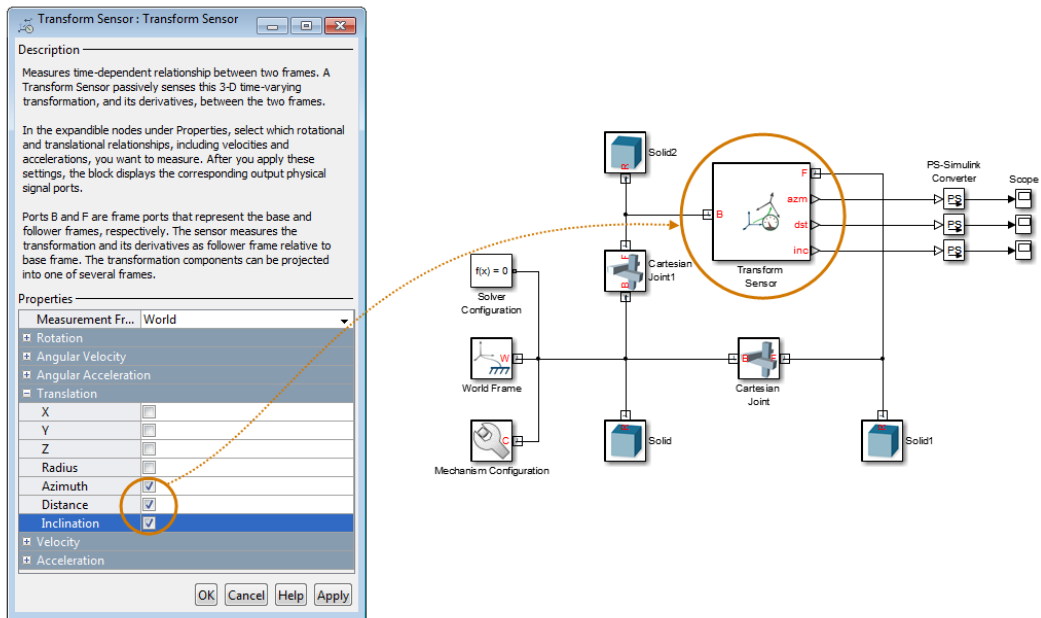
Only the Transform Sensor block can sense frame translation in spherical coordinates. In the dialog box of this block, you can select one or more spherical coordinates to measure. The spherical coordinates are named **Azimuth**, **Distance**, and **Inclination**. Selecting the spherical coordinates exposes physical signal ports *azm*, *dst*, and *inc*, respectively.

---

**Note:** **Azimuth** belongs to both cylindrical and spherical systems. **Distance** is the spherical radius.

---

The figure shows a simple model using a Transform Sensor block to measure frame translation along all three spherical axes. The measurement gives the relative translation of the follower port frame with respect to the base port frame. These frames are, respectively, the Solid1 and Solid2 reference port frames.



## **Related Examples**

- “Sense Motion” on page 4-67
- “Specify Joint Actuation Torque” on page 4-73

## **More About**

- “Motion Sensing” on page 4-46
- “Rotational Measurements” on page 4-51
- “Measurement Frames” on page 4-64

## Measurement Frames

### In this section...

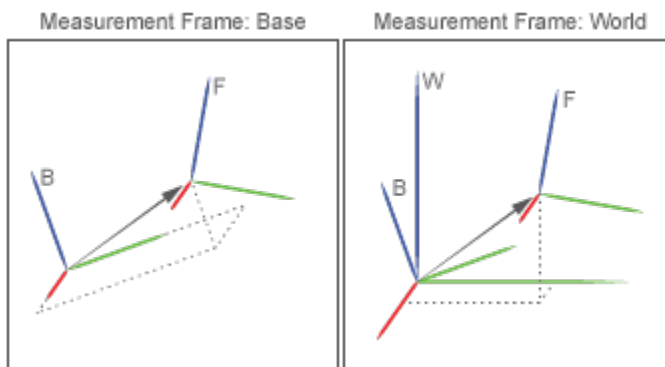
“Measurement Frame Purpose” on page 4-64

“Measurement Frame Types” on page 4-65

You can sense the spatial relationship between two frames. When you do so, Simscape Multibody resolves the measurement in a measurement frame. For most joint blocks, the measurement frame is the base frame. However, if you use either **Transform Sensor** or a joint block with a spherical primitive, you can select a different measurement frame. Measurement frames that you can select include **Base**, **Follower**, and **World**. The **Transform Sensor** block adds the choice between rotating and non-rotating versions of the base and follower frames.

### Measurement Frame Purpose

The measurement frame defines the axes that Simscape Multibody uses to resolve a measurement. The measurement still describes the relationship between base and follower frames. However, the X, Y, and Z components of that measurement refer to the axes of the measurement frame. Simscape Multibody takes the measurement and projects it onto the axes of the measurement frame that you select. The figure illustrates the measurement frame concept.



The arrow connecting base and follower frame origins is the translation vector. If you select the base frame as the measurement frame, Simscape Multibody resolves that

translation vector along the axes of the base frame. If you select the World frame as the measurement frame, Simscape Multibody instead resolves the translation vector along the axes of the World frame. The translation vector remains the same, but the frame Simscape Multibody expresses that measurement in changes.

Note that you can select the measurement frame only with certain blocks. Among joint blocks, only those with a spherical primitive offer a selection of measurement frames. All other joint blocks resolve their measurements in the base frame. The Transform Sensor block offers the most extensive selection of measurement frames.

## Measurement Frame Types

Simscape Multibody offers five different measurement frames. These include World as well as rotating and non-rotating versions of the base and follower frames. The table describes these measurement frames.

Measurement Frame	Description
World	Inertial frame at absolute rest. World is the ultimate reference frame in a model. The <code>World Frame</code> block identifies this frame in a model.
Base	Frame that connects to the B port of the sensing block. Unless you rigidly connect it to World, Base is generally non-inertial.
Follower	Frame that connects to the F port of the sensing block. Unless you rigidly connect it to World, Follower is generally non-inertial.
Non-Rotating Base/Follower	<p>Non-rotating versions of the Base and follower frames.</p> <p>A non-rotating frame is a virtual frame which, at every point in time, Simscape Multibody holds coincident with the rotating frame, but which has zero angular velocity with respect to the World frame.</p> <p>Measurements that can differ between rotating and non-rotating frames are the linear velocity and linear acceleration.</p>

### **Related Examples**

- “Sense Motion” on page 4-67
- “Specify Joint Actuation Torque” on page 4-73

### **More About**

- “Motion Sensing” on page 4-46
- “Rotational Measurements” on page 4-51
- “Translational Measurements” on page 4-56

## Sense Motion

### In this section...

“Model Overview” on page 4-67

“Modeling Approach” on page 4-68

“Build Model” on page 4-68

“Guide Model Assembly” on page 4-70

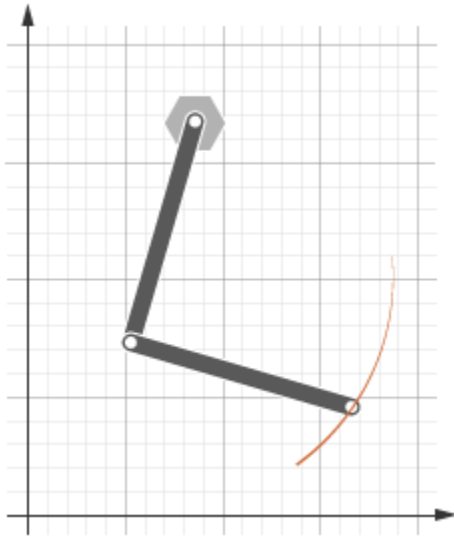
“Simulate Model” on page 4-70

“Save Model” on page 4-72

### Model Overview

The Transform Sensor block provides the broadest motion-sensing capability in Simscape Multibody models. Using this block, you can sense motion variables between any two frames in a model. These variables can include translational and rotational position, velocity, and acceleration.

In this example, you use a Transform Sensor block to sense the lower link translational position with respect to the World frame. You output the position coordinates directly to the model workspace, and then plot these coordinates using MATLAB commands. By varying the joint state targets, you can analyze the lower-link motion under quasi-periodic and chaotic conditions.



### Modeling Approach

In this example, you rely on gravity to cause the double pendulum to move. You displace the links from equilibrium and then let gravity act on them. To displace the links at time zero, you use the **State Targets** section of the Revolute Joint block dialog box. You can specify position or velocity. When you are ready, you simulate the model to analyze its motion.

To sense motion, you use the Transform Sensor block. First, you connect the base and follower frame ports to the World Frame and lower link subsystem blocks. By connecting the ports to these blocks, you can sense motion in the lower link with respect to the World frame. Then, you select the translation parameters to sense. By selecting **Y** and **Z**, you can sense translation along the Y and Z axes, respectively. You can plot these coordinates with respect to each other and analyze the motion that they reveal.

### Build Model

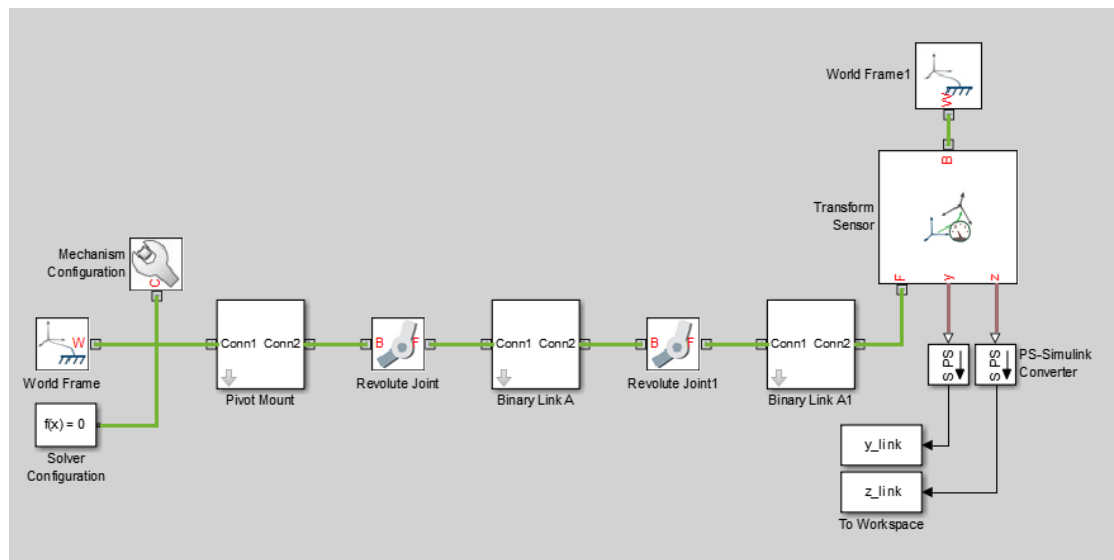
- 1 At the MATLAB command prompt, enter `smdoc_double_pendulum`. A double pendulum model opens up. For instructions on how to create this model, see “Model an Open-Loop Kinematic Chain” on page 3-16.



- 2 Drag these blocks into the model to sense motion.

Library	Block	Quantity
<b>Simscape &gt; Multibody &gt; Frames and Transforms</b>	Transform Sensor	1
<b>Simscape &gt; Multibody &gt; Frames and Transforms</b>	World Frame	1
<b>Simscape &gt; Utilities</b>	PS-Simulink Converter	2
<b>Simulink &gt; Sinks</b>	To Workspace	2

- 3 In the Transform Sensor block dialog box, select **Translation** > **Y** and **Translation** > **Z**. The block exposes two physical signal output ports, labeled y and z.
- 4 In the PS-Simulink Converter blocks, specify units of cm.
- 5 In the To Workspace blocks, enter the variable names y\_link and z\_link.
- 6 Connect the blocks to the model as shown in the figure. You must connect the base frame port of the Transform Sensor block to the World Frame block. The new blocks are shaded gray.



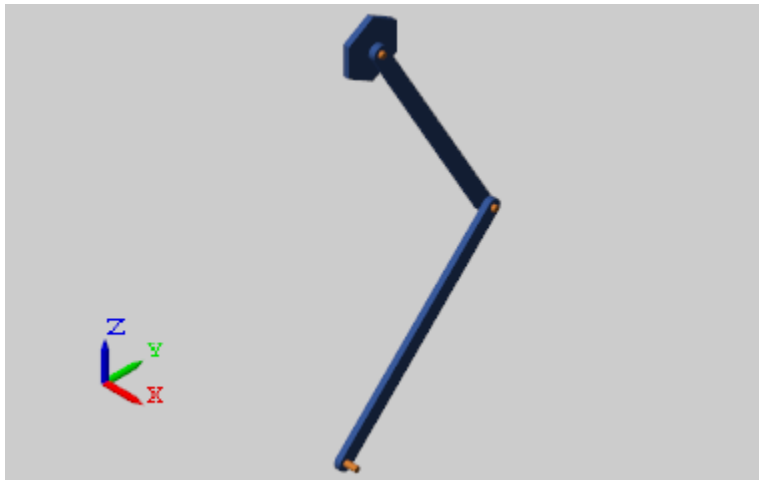
### Guide Model Assembly

Specify the initial state of each joint. Later, you can modify this state to explore different motion types. For the first iteration, rotate only the top link by a small angle.

- 1 In the Revolute Joint block dialog box, select **State Targets > Specify Position Target**.
- 2 Set **Value** to 10 deg.
- 3 In the Revolute Joint1 block dialog box, check that **State Targets > Specify Position Target** is cleared.

### Simulate Model

Run the simulation. Mechanics Explorer plays a physics-based animation of the double pendulum assembly.

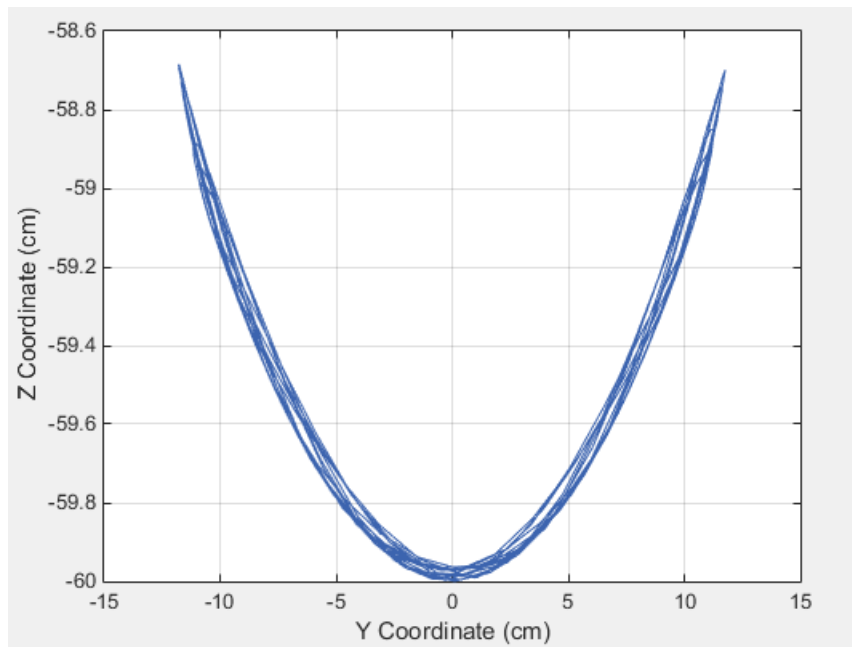


You can now plot the position coordinates of the lower link. To do this, at the MATLAB command line, enter:

```
figure;  
plot(y_link.data, z_link.data, 'color', [60 100 175]/255);  
xlabel('Y Coordinate (cm)');  
ylabel('Z Coordinate (cm)');
```

```
grid on;
```

The figure shows the plot that opens. This plot shows that the lower link path varies only slightly with each oscillation. This behavior is characteristic of quasi-periodic systems.



### Simulate Chaotic Motion

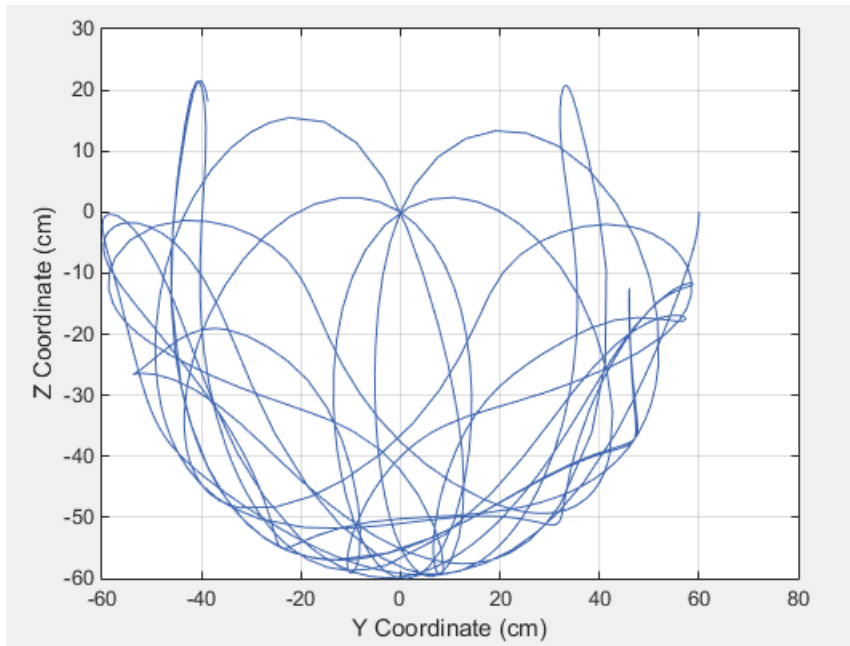
By adjusting the revolute joint state targets, you can simulate the model under chaotic conditions. One way to obtain chaotic motion is to rotate the top revolute joint by a large angle. To do this, in the Revolute Joint dialog box, change **State Targets > Position > Value** to **90** and click **OK**.

Simulate the model with the new joint state target. To plot the position coordinates of the lower pendulum link with respect to the world frame, at the MATLAB command prompt, enter this code:

```
figure;
plot(y_link.data, z_link.data, 'color', [60 100 175]/255);
xlabel('Y Coordinate (cm)');
ylabel('Z Coordinate (cm)');
```

grid on;

The figure shows the plot that opens.



### Save Model

Save the model in a convenient folder under the name `double_pendulum_sensing`. You reuse this model in a subsequent tutorial, “Prescribe Joint Motion in Planar Manipulator Model” on page 4-110.

## Specify Joint Actuation Torque

### In this section...

“Model Overview” on page 4-73

“Four-Bar Linkages” on page 4-74

“Modeling Approach” on page 4-76

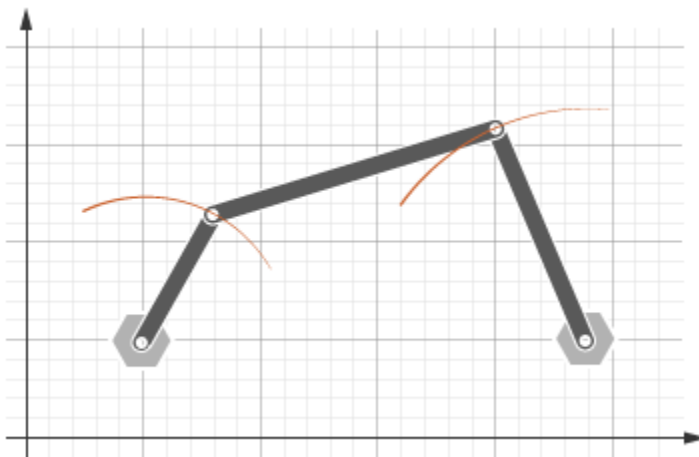
“Build Model” on page 4-77

“Simulate Model” on page 4-80

### Model Overview

In Simscape Multibody, you actuate a joint directly using the joint block. Depending on the application, the joint actuation inputs can include force/torque or motion variables. In this example, you prescribe the actuation torque for a revolute joint in a four-bar linkage model.

Transform Sensor blocks add motion sensing to the model. You can plot the sensed variables and use the plots for kinematic analysis. In this example, you plot the coupler curves of three four-bar linkage types: crank-rocker, double-crank, and double-rocker.



## Four-Bar Linkages

The four-bar linkage contains four links that interconnect with four revolute joints to form a planar closed loop. This linkage converts the motion of an input link into the motion of an output link. Depending on the relative lengths of the four links, a four-bar linkage can convert rotation into rotation, rotation into oscillation, or oscillation into oscillation.

### Links

Links go by different names according to their functions in the four-bar linkage. For example, coupler links transmit motion between crank and rocker links. The table summarizes the different link types that you may find in a four-bar linkage.

Link	Motion
Crank	Revolves with respect to the ground link
Rocker	Oscillates with respect to the ground link
Coupler	Transmits motion between crank and rocker links
Ground	Rigidly connects the four-bar linkage to the world or another subsystem

It is common for links to have complex shapes. This is especially true of the ground link, which may be simply the fixture holding the two pivot mounts that connect to the crank or rocker links. You can identify links with complex shapes as the rigid span between two adjacent revolute joints. In example “Model a Closed-Loop Kinematic Chain” on page 3-21, the rigid span between the two pivot mounts represents the ground link.

### Linkages

The type of motion conversion that a four-bar linkage provides depends on the types of links that it contains. For example, a four-bar linkage that contains two crank links converts rotation at the input link into rotation at the output link. This type of linkage is known as a double-crank linkage. Other link combinations provide different types of motion conversion. The table describes the different types of four-bar linkages that you can model.

Linkage	Input-Output Motion
Crank-rocker	Continuous rotation-oscillation (and vice-versa)

Linkage	Input-Output Motion
Double-Crank	Continuous rotation-continuous rotation
Double-rocker	Oscillation-oscillation

### Grashof Condition

The Grashof theorem provides the basic condition that the four-bar linkage must satisfy so that at least one link completes a full revolution. According to this theorem, a four-bar linkage contains one or more crank links if the combined length of the shortest and longest links does not exceed the combined length of the two remaining links.

Mathematically, the Grashof condition is:

$$s+l \leq p+q$$

where:

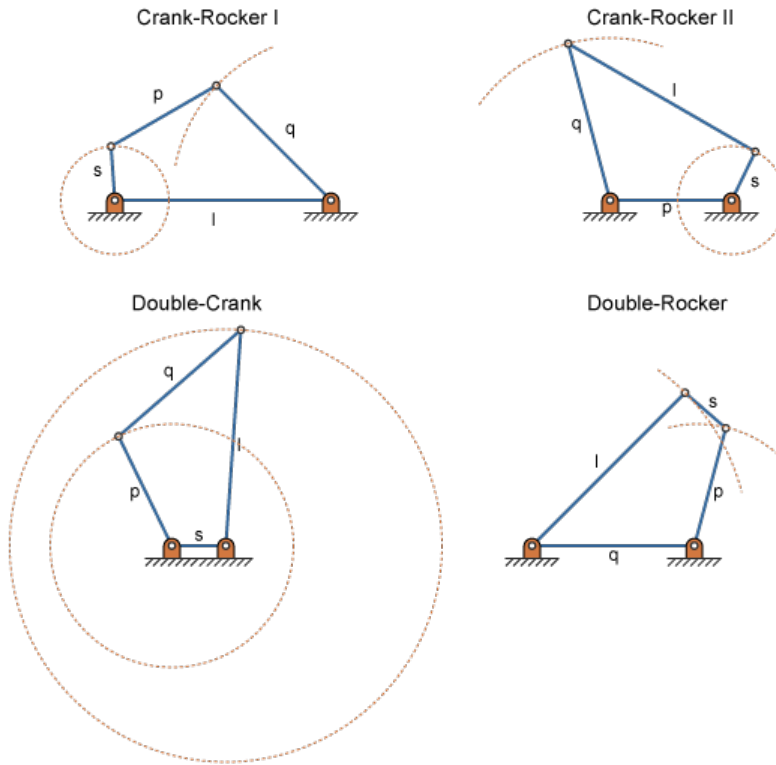
- s is the shortest link
- l is the longest link
- p and q are the two remaining links

### Grashof Linkages

A Grashof linkage can be of three different types:

- Crank-rocker
- Double-crank
- Double-rocker

By changing the ground link, you can change the Grashof linkage type. For example, by assigning the crank link of a crank-rocker linkage as the ground link, you obtain a double-crank linkage. The figure shows the four linkages that you obtain by changing the ground link.



## Modeling Approach

In this example, you perform two tasks. First you add a torque actuation input to the model. Then, you sense the motion of the crank and rocker links with respect to the World frame. The actuation input is a torque that you apply to the joint connecting the base to the crank link. Because you apply the torque at the joint, you can add this torque directly through the joint block. The block that you add the actuation input to is called Base-Crank Revolute Joint.

You add the actuation input to the joint block through a physical signal input port. This port is hidden by default. To display it, you must select **Provided by Input** from the **Actuation > Torque** drop-down list.



You can then specify the torque value using either Simscape or Simulink blocks. If you use Simulink blocks, you must use the **Simulink-PS Converter** block. This block converts the Simulink signal into a physical signal that Simscape Multibody can use. For more information, see “Actuating and Sensing with Physical Signals” on page 4-36.

To sense crank and rocker link motion, you use the Transform Sensor block. With this block, you can sense motion between any two frames in a model. In this example, you use it to sense the [Y Z] coordinates of the crank and rocker links with respect to the World frame.

The physical signal output ports of the Transform Sensor blocks are hidden by default. To display them, you must select the appropriate motion outputs. Using the **PS-Simulink Converter**, you can convert the physical signal outputs into Simulink signals. You can then connect the resulting Simulink signals to other Simulink blocks.

In this example, you output the crank and rocker link coordinates to the workspace using Simulink To Workspace blocks. The output from these blocks provide the basis for phase plots showing the different link paths.

## Build Model

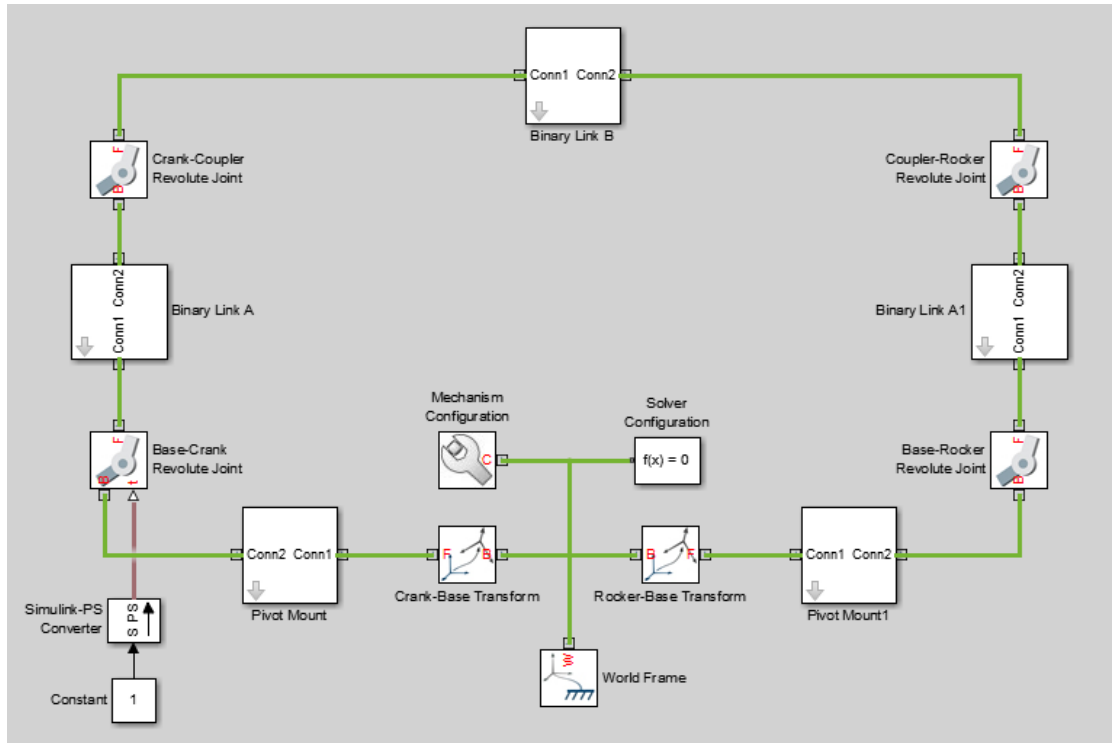
Provide the joint actuation input, specify the joint internal mechanics, and sense the position coordinates of the coupler link end frames.

### Provide Joint Actuation Input

- 1 At the MATLAB command prompt, enter `smdoc_four_bar`. A four bar model opens up. For instructions on how to create this model, see “Model a Closed-Loop Kinematic Chain” on page 3-21.
- 2 In the Base-Crank Revolute Joint block dialog box, in the **Actuation > Torque** drop-down list, select **Provided by Input**. The block exposes a physical signal input port, labeled `t`.
- 3 Drag these blocks into the model. The blocks enable you to specify the actuation torque signal.

Library	Block
Simulink > Sources	Constant
Simscape > Utilities	Simulink-PS Converter

- 4 Connect the blocks as shown in the figure. The new blocks are shaded gray.



### Specify Joint Internal Mechanics

Real joints dissipate energy due to damping. You can specify joint damping directly in the block dialog boxes. In each Revolute Joint block dialog box, under **Internal Mechanics > Damping Coefficient**, enter  $5e-4$  and press **OK**.

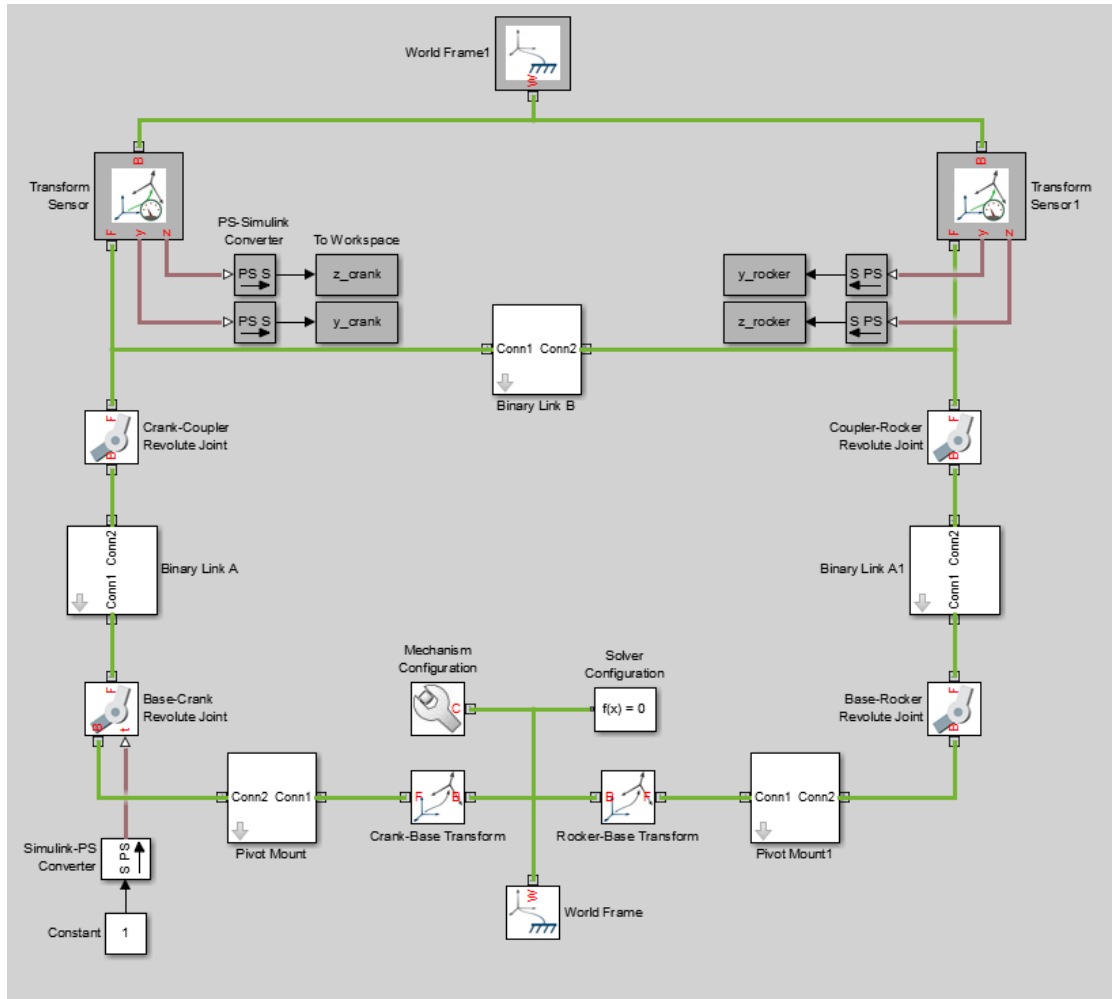
### Sense Link Position Coordinates

- 1 Add these blocks to the model. The blocks enable you to sense frame position during simulation.

Library	Block	Quantity
Simscape > Multibody > Frames and Transforms	Transform Sensor	2

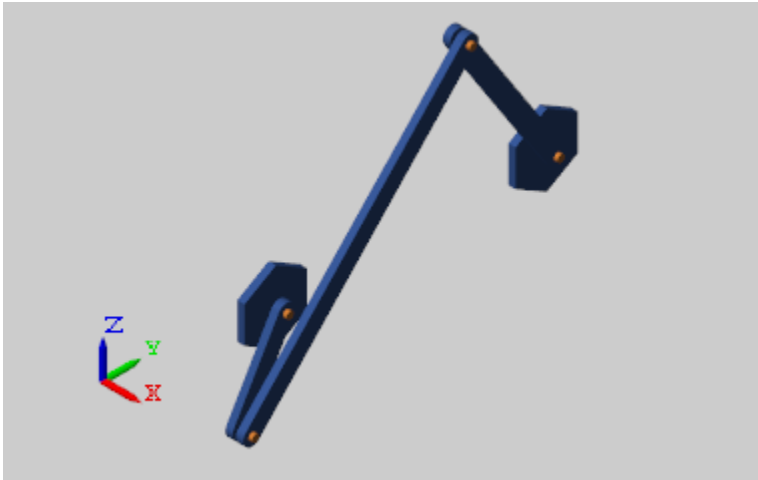
Library	Block	Quantity
Simscape > Multibody > Frames and Transforms	World Frame	1
Simscape > Utilities	PS-Simulink Converter	4
Simulink > Sinks	To Workspace	4

- 2 In the Transform Sensor block dialog boxes, select **Translation > Y** and **Translation > Z**. Resize the block as needed.
- 3 In the **Output Signal Unit** parameters of the PS-Simulink Converter block dialog boxes, enter cm.
- 4 In the **Variable Name** parameters of the To Workspace block dialog boxes, enter the variable names:
  - y\_crank
  - z\_crank
  - y\_rocker
  - z\_rocker
- 5 Connect and name the blocks as shown in the figure, rotating them as needed. Ensure that the To Workspace blocks with the z\_crank and z\_rocker variable names connect to the z frame ports of the Transform Sensor blocks. The new blocks are shaded gray.



## Simulate Model

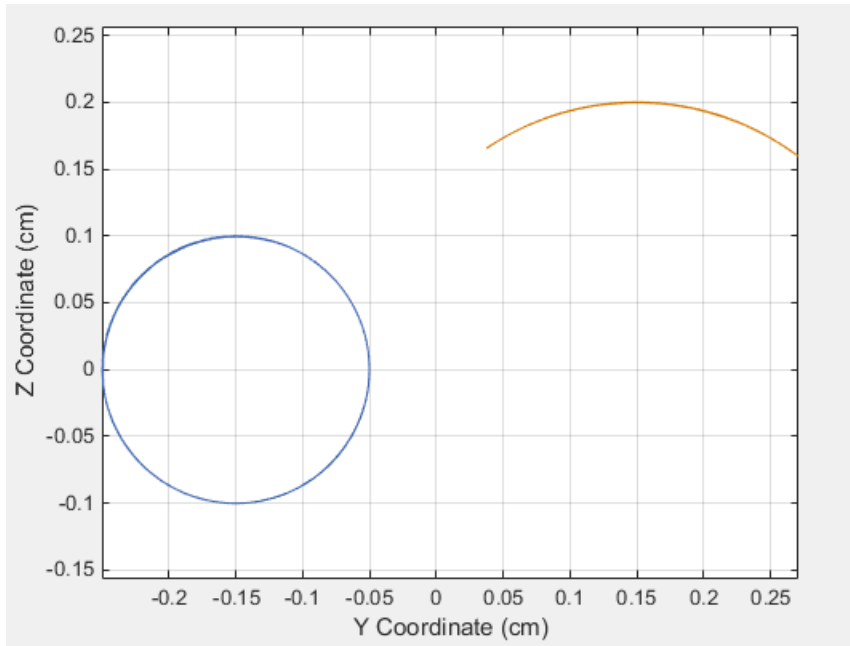
Run the simulation. You can do this in the Simulink tool bar by clicking the run button. Mechanics Explorer plays a physics-based animation of the four bar assembly.



Once the simulation ends, you can plot the position coordinates of the coupler link end frames, e.g., by entering the following code at the MATLAB command line:

```
figure;  
plot(y_crank.data, z_crank.data, 'color', [60 100 175]/255);  
hold;  
plot(y_rocker.data, z_rocker.data, 'color', [210 120 0]/255);  
xlabel('Y Coordinate (cm)');  
ylabel('Z Coordinate (cm)');  
axis equal; grid on;
```

The figure shows the plot that opens. This plot shows that the crank completes a full revolution, while the rocker completes a partial revolution, e.g., it oscillates. This behavior is characteristic of crank-rocker systems.

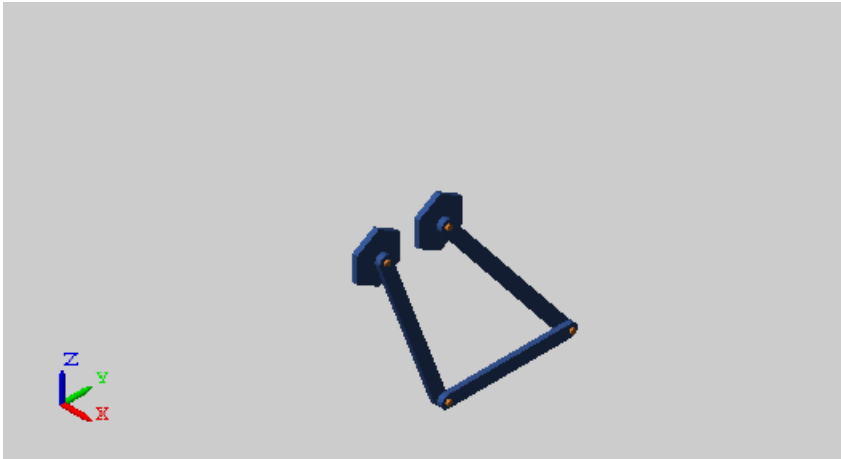


### Simulate Model in Double-Crank Mode

Try simulating the model in double-crank mode. You can change the four-bar linkage into a double-crank linkage by changing the binary link lengths according to the table.

Block	Parameter	Value
Binary Link A	Length	25
Binary Link B	Length	20
Binary Link A1	Length	30
Crank-Base Transform	Translation > Offset	5
Rocker-Base Transform	Translation > Offset	5

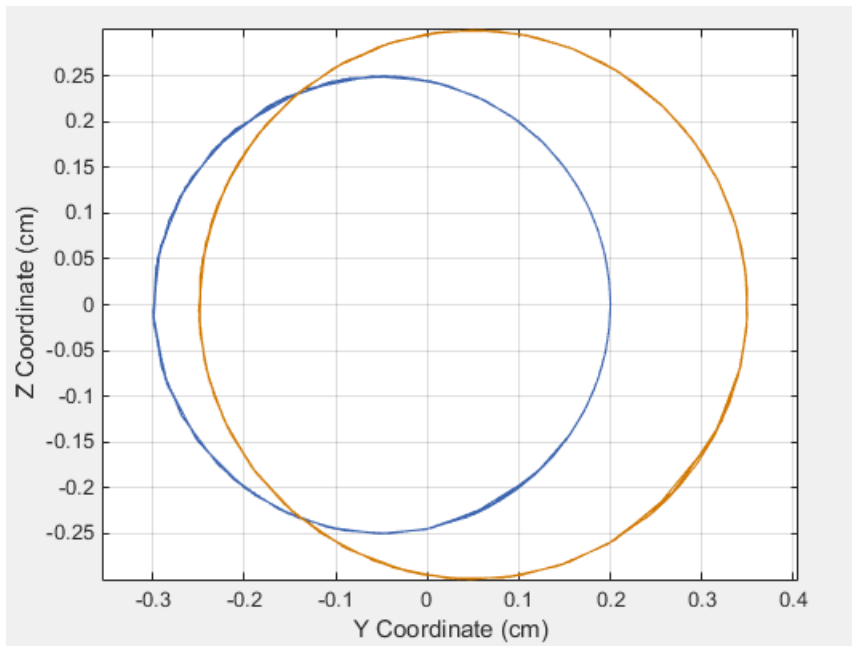
Update and simulate the model. The figure shows the updated visualization display in Mechanics Explorer.



Plot the position coordinates of the coupler link end frames. At the MATLAB command line, enter:

```
figure;  
plot(y_crank.data, z_crank.data, 'color', [60 100 175]/255);  
hold;  
plot(y_rocker.data, z_rocker.data, 'color', [210 120 0]/255);  
xlabel('Y Coordinate (cm)');  
ylabel('Z Coordinate (cm)');  
axis equal; grid on;
```

The figure shows the plot that opens. This plot shows that both links complete a full revolution. This behavior is characteristic of double-crank linkages.





## Analyze Motion at Various Parameter Values

### In this section...

“Model Overview” on page 4-85

“Build Model” on page 4-85

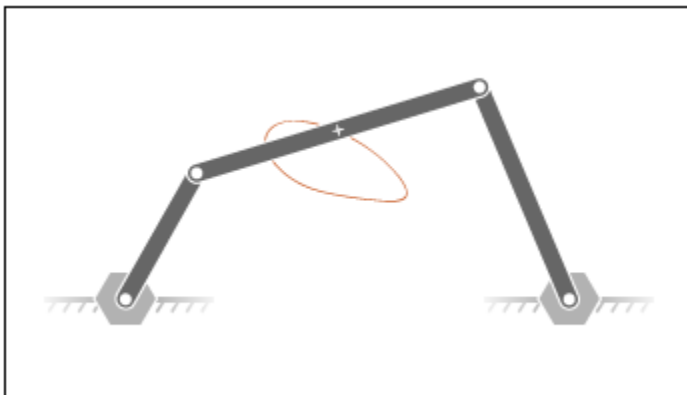
“Specify Block Parameters” on page 4-87

“Create Simulation Script” on page 4-88

“Run Simulation Script” on page 4-89

### Model Overview

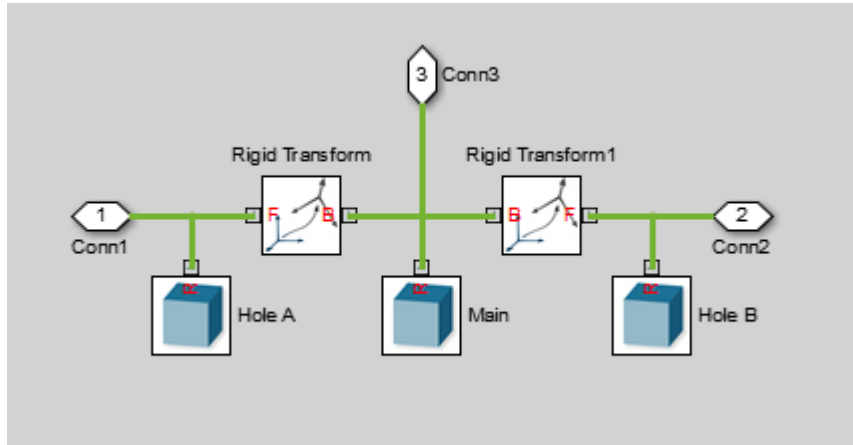
In this tutorial, you create a simple MATLAB script to simulate a four-bar model at various coupler lengths. The script uses the coupler motion coordinates, obtained using a Transform Sensor block, to plot the resulting coupler curve at each value of the coupler length. For information on how to create the four-bar model used in this tutorial, see “Model a Closed-Loop Kinematic Chain” on page 3-21.



### Build Model

- 1 At the MATLAB command prompt, enter `smdoc_four_bar`. A four-bar model opens up. For instructions on how to create this model, see “Model a Closed-Loop Kinematic Chain” on page 3-21.

- Under the mask of the Binary Link B block, connect a third Output block as shown in the figure. You can add an Output block by copying and pasting Conn1 or Conn2. The new block identifies the frame whose trajectory you plot in this tutorial.



- Add the following blocks to the model. During simulation, the Transform Sensor block computes and outputs the coupler trajectory with respect to the world frame.

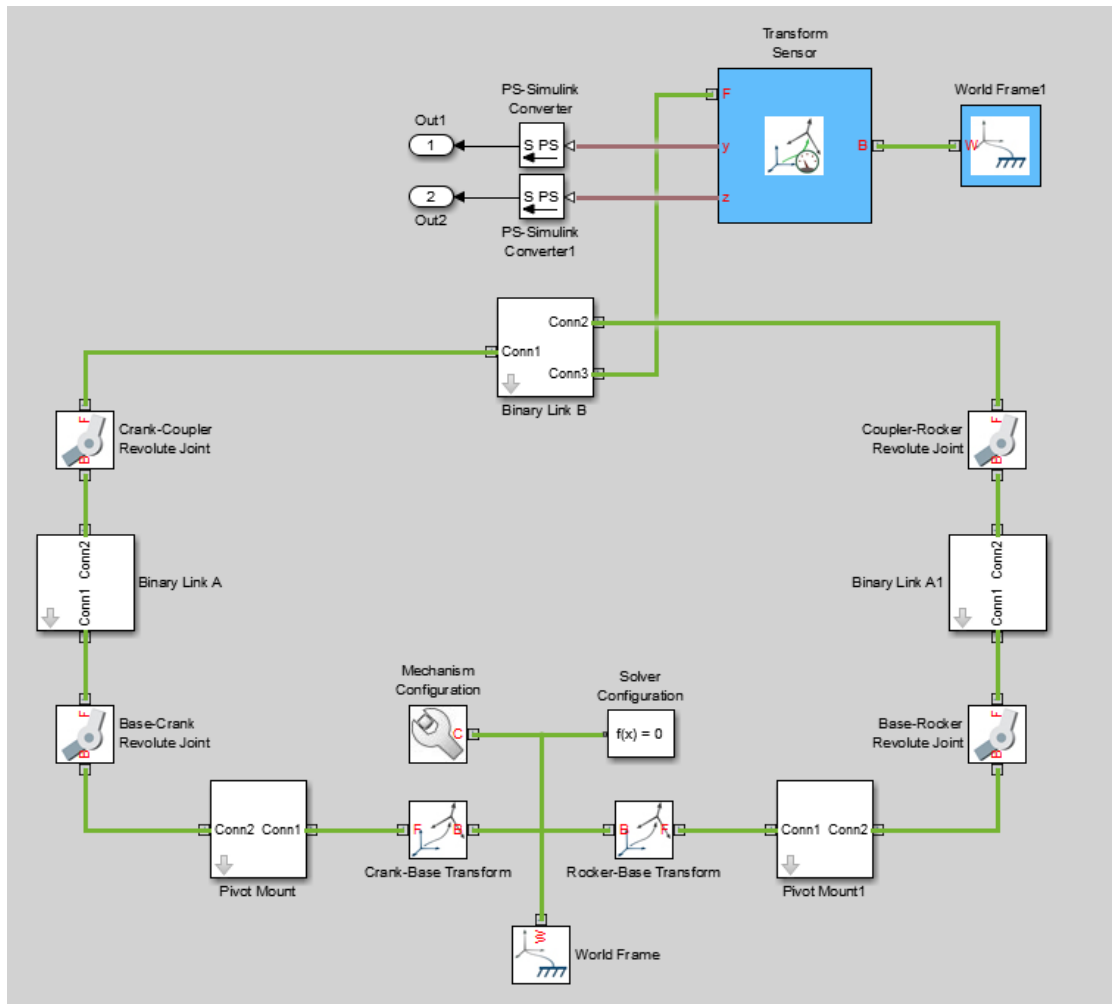
Library	Block	Quantity
Frames and Transforms	World Frame	1
Frames and Transforms	Transform Sensor	1
Simscape Utilities	PS-Simulink Converter	2
Simulink Sinks	Output	2

- In the Transform Sensor block dialog box, select these variables:

- **Translation > Y**
- **Translation > Z**

The block exposes frame ports y and z, through which it outputs the coupler trajectory coordinates.

- Connect the blocks as shown in the figure. Be sure to flip the Transform Sensor block so that its base frame port, labeled B, connects to the World Frame block.



## Specify Block Parameters

- 1 In the Mechanism Configuration block, change **Uniform Gravity** to None.
- 2 In the Base-Crank Revolute Joint block, specify the following velocity state targets. The targets provide an adequate source of motion for the purposes of this tutorial.

- Select **State Targets > Specify Velocity**.
  - In **State Targets > Specify Velocity > Value**, enter 2 rev/s.
  - Deselect **State Target > Specify Position**.
- 3 Specify the following link lengths. The coupler link length is parameterized in terms of a MATLAB variable, `LCoupler`, enabling you change its value iteratively using a simple MATLAB script.

Block	Parameter	Value
Binary Link B	Length	LCoupler
Binary Link A1	Length	25

- 4 Save the model in a convenient folder, naming it `smdoc_four_bar_msensing`.

## Create Simulation Script

Create a MATLAB script to iteratively run simulation at various coupler link lengths:

- 1 On the MATLAB toolstrip, click **New Script**.
- 2 In the script, enter the following code:

```
% Run simulation nine times, each time
% increasing coupler length by 1 cm.
% The original coupler length is 20 cm.
for i = (0:8);
    LCoupler = 20+i;

    % Simulate model at the current coupler link length (LCoupler),
    % saving the Output block data into variables y and z.
    [~, ~, y, z] = sim('smdoc_four_bar_msensing');

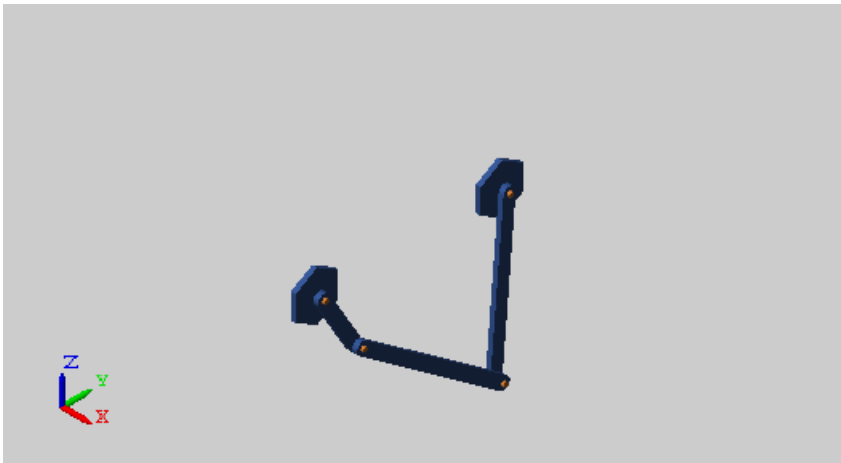
    % Plot the [y, z] coordinates of each coupler curve
    % on the x = i plane. i corresponds to the simulation run number.
    x = zeros(size(y)) + i;
    plot3(x, y, z, 'Color', [1 0.8-0.1*i 0.8-0.1*i]);
    view(30, 60); hold on;
end
```

The code runs simulation at nine different coupler link lengths. It then plots the trajectory coordinates of the coupler link center frame with respect to the world frame. Coupler link lengths range from 20 cm to 28 cm.

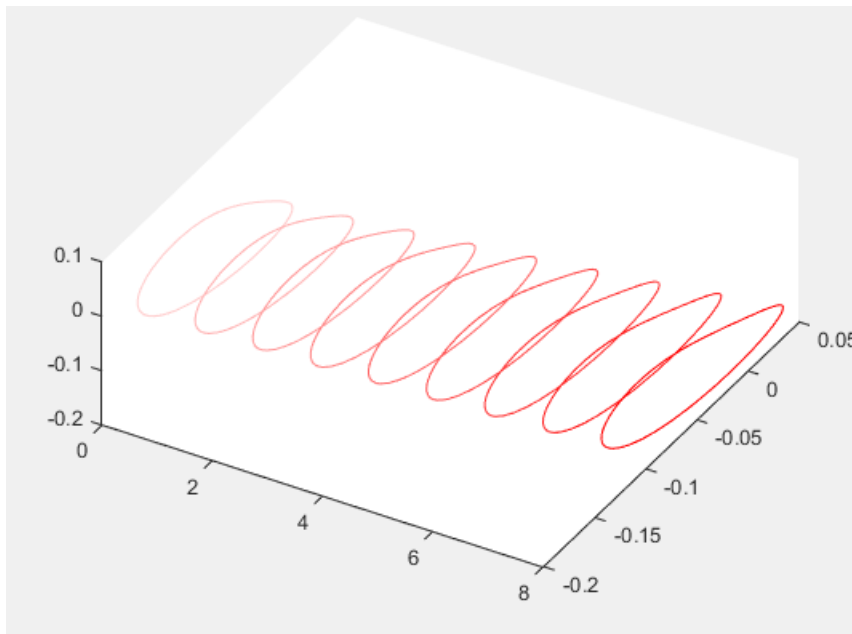
- 3 Save the script as `sim_four_bar` in the folder containing the four-bar model.

## Run Simulation Script

Run the `sim_four_bar` script. In the MATLAB Editor toolstrip, click the **Run** button or, with the editor active, press **F5**. Mechanics Explorer opens with a dynamic 3-D view of the four-bar model.



Simscape Multibody iteratively runs each simulation, adding the resulting coupler link curve to the active plot. The figure shows the final plot.



You can use the simple approach shown in this tutorial to analyze model dynamics at various parameter values. For example, you can create a MATLAB script to simulate a crank-slider model at different coupler link lengths, plotting for each simulation run the constraint force acting on the piston.

## Sense Forces and Torques Acting at Joints

### In this section...

“Overview” on page 4-91

“Open Model” on page 4-92

“Sense Actuation Torque” on page 4-92

“Sense Constraint Forces” on page 4-95

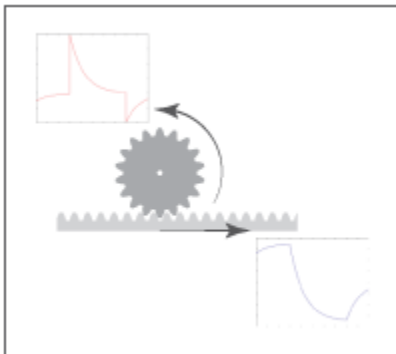
“Sense Total Forces” on page 4-97

### Overview

Simscape Multibody provides force and torque sensing in joint blocks. You can use this sensing capability to compute and output various types of forces and torques acting directly at joints. Force and torque types that you can sense include those due to:

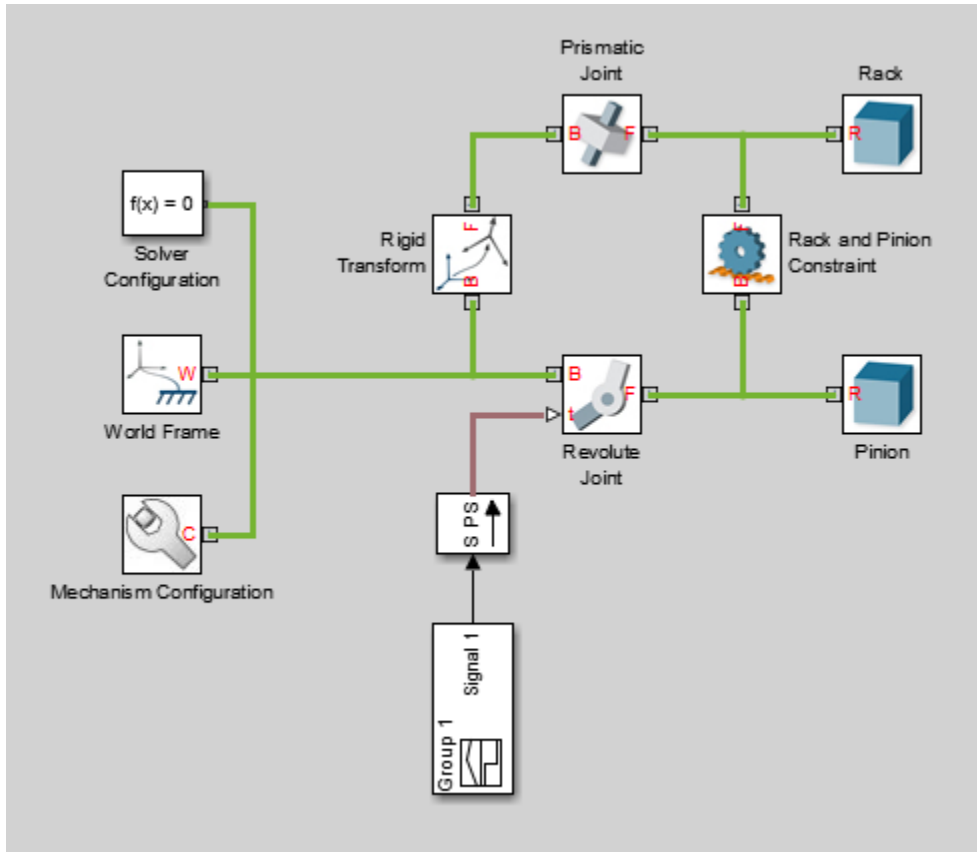
- Joint actuation inputs
- Joint constraints
- Joint actuation inputs, constraints, and internal mechanics combined

In this tutorial, you explore the different types of force and torque sensing that Simscape Multibody joint blocks provide.



## Open Model

At the MATLAB command prompt, enter `smdoc_rack_pinion_c`. Simscape Multibody opens a rack and pinion model that you can use to explore the force and torque sensing capability of joint blocks.

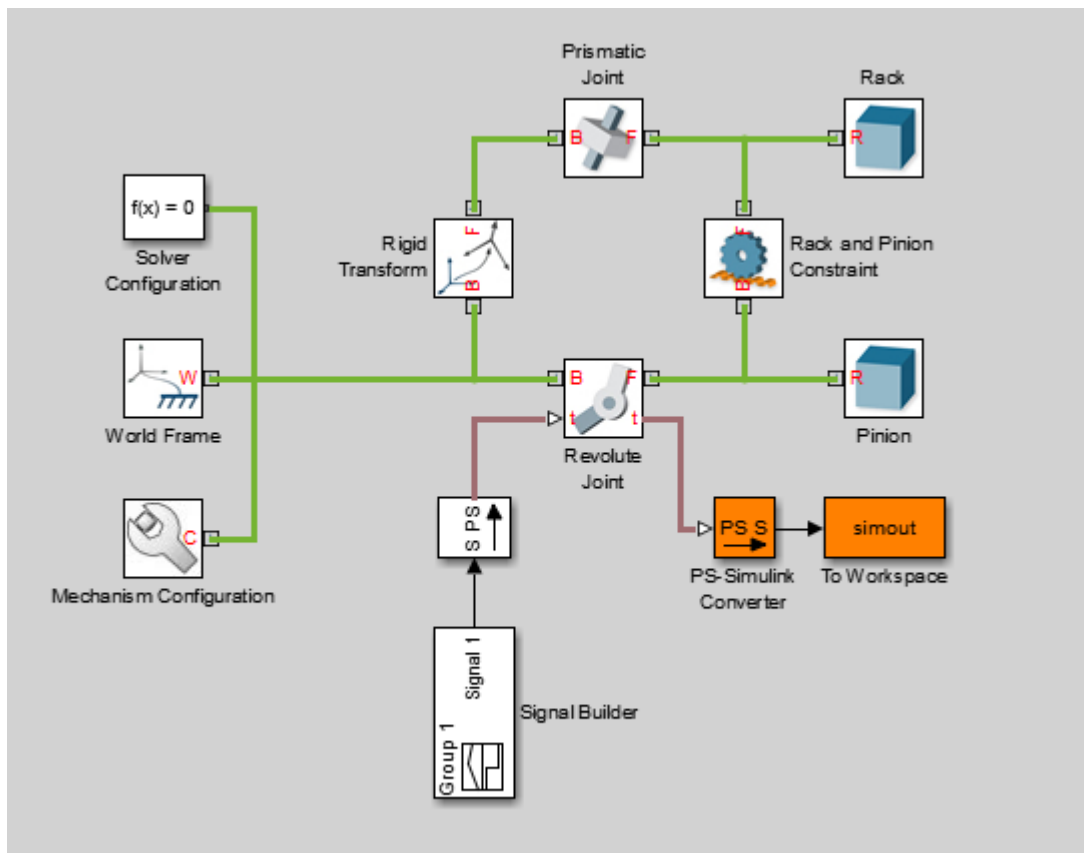


## Sense Actuation Torque

The rack and pinion model contains an actuation torque input that drives the pinion revolute joint. A Simulink-PS Converter block processes the input signal using a second-order filter, smoothing any abrupt changes or discontinuities the signal may have. To sense the actuation torque as observed at the Revolute Joint block:



- 1 In the Revolute Joint block dialog box, select **Z Revolute Primitive (Rz) > Sensing > Actuator Torque**. The block exposes a physical signal port, labeled *t*. This port outputs the 3-D vector components of the joint actuator torque in a Simscape physical signal.
- 2 Drag the following blocks into the model:
  - PS-Simulink Converter from the **Simscape > Utilities** library
  - To Workspace from the **Simulink > Sinks** library
- 3 Connect the blocks as shown in the figure.



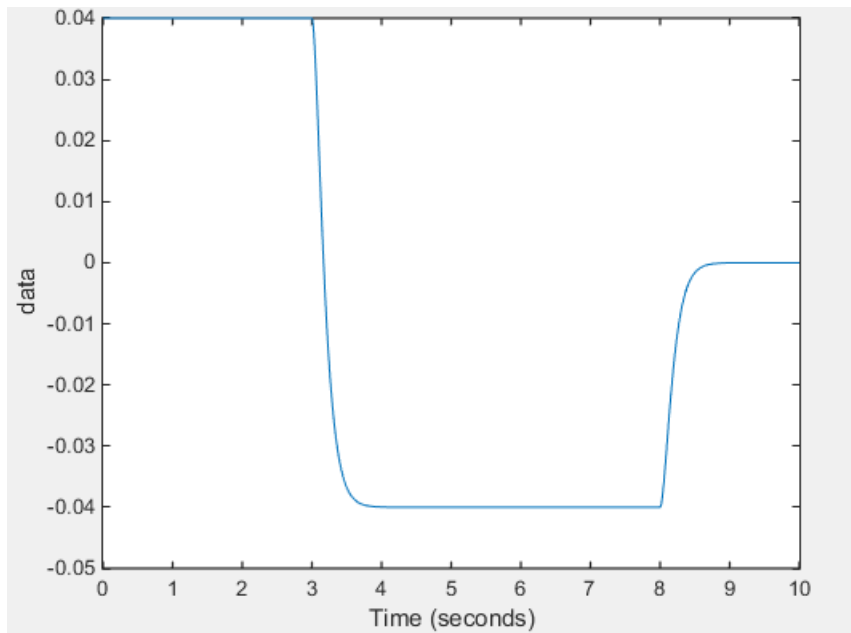
- 4 Simulate the model. The To Workspace block outputs the actuator torque signal into a time-series variable, *simout*, available in the MATLAB base workspace.

- 5 At the MATLAB command prompt, enter:

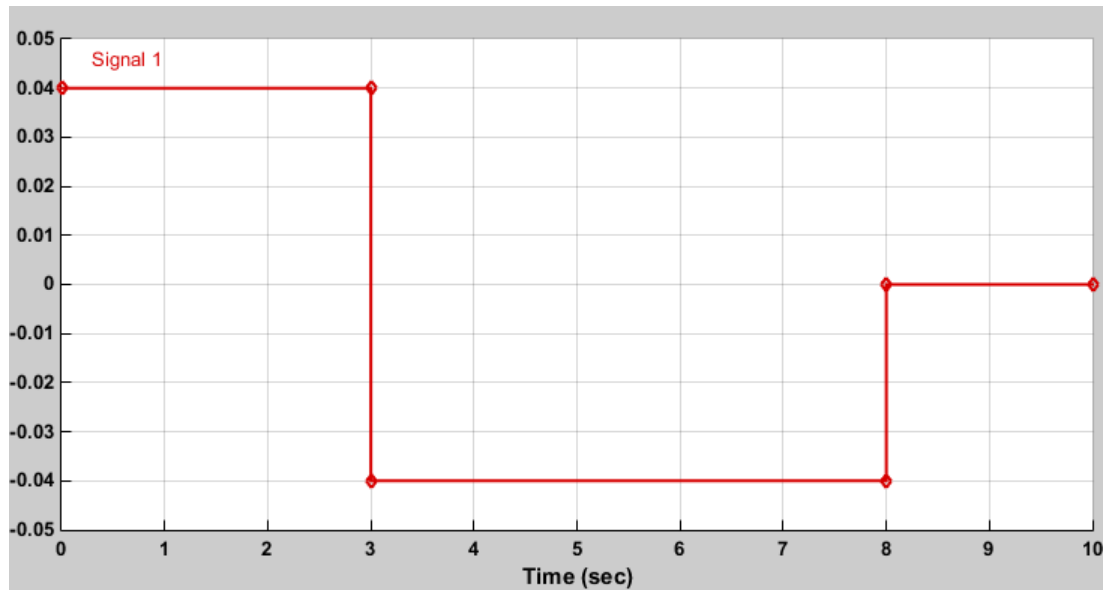
```
figure;
```

```
plot(simout);
```

MATLAB plots the vector components of the joint actuator torque. All but the Z component are zero throughout the simulation.



Compare the actuator torque plot to the original input signal in the Signal Builder block. Neglecting any signal smoothing due to the second-order filtering, the two signals are identical. The following figure shows the original input signal.



Actuator force and torque sensing enables you to analyze the required forces and torques to yield a prescribed joint trajectory. Use this feature in your model to perform inverse dynamic and other types of analysis.

## Sense Constraint Forces

Joint constraint forces, which act normal to the joint primitive axes, restrict motion to the allotted joint degrees of freedom. In the Revolute Joint block, the constraint forces resist the pull of gravity, keeping the pinion fixed with respect to the world frame. To sense the constraint forces:

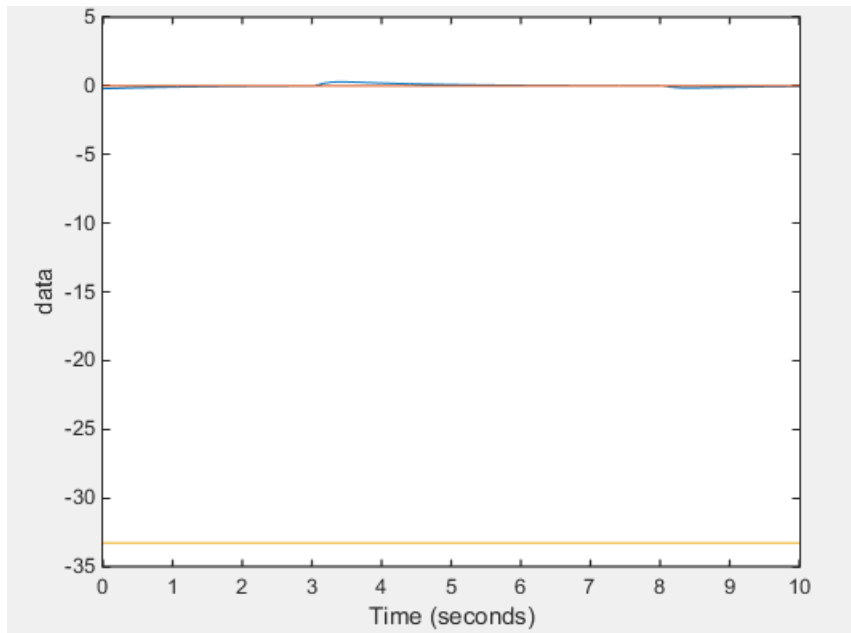
- 1 In the Mechanism Configuration block, set **Uniform Gravity** to **Constant**. This setting ensures that gravity acts on the rack and pinion system. Check that the gravity vector is  $[0 \ 0 \ -9.80665]$ .
- 2 In the Revolute Joint block dialog box, select **Composite Force/Torque Sensing** > **Constraint Force**. The block exposes the physical signal port `fc`. This port provides the vector components of the joint-wide constraint force in a Simscape physical signal. By default, this is the constraint force that the follower port frame exerts on the base port frame, resolved in the base port frame.

- 3 Deselect **Z Revolute Primitive (Rz)** > **Sensing** > **Actuator Torque**.
- 4 Check that the PS-Simulink Converter block now connects to the physical signal port `fc`.
- 5 Simulate the model. At the MATLAB command prompt, enter:

```
figure;
```

```
plot(simout);
```

MATLAB plots the constraint force components with respect to time. All but one component are zero throughout simulation. The Z component, which opposes the gravity vector, is the only component needed to hold the joint frames in place.



Constraint forces ensure that weld joint frames remain fixed with respect to each other. You can place a Weld Joint block inside a rigid body subsystem to sense the internal forces and torques acting within that body during simulation. For an example of how you can do this in a double pendulum model, see “Sense Constraint Forces” on page 4-99.

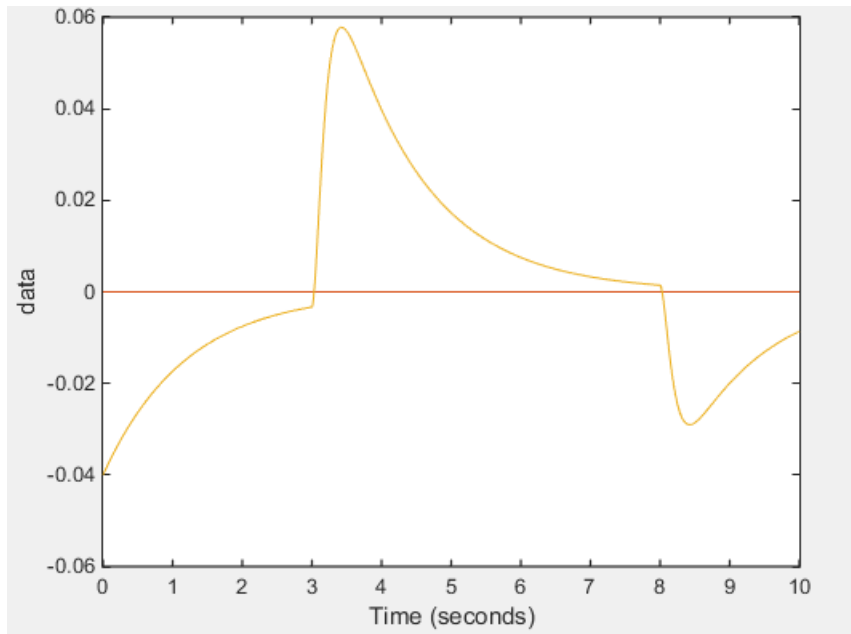
## Sense Total Forces

In addition to actuation and constraint forces and torques, joint frames can also interact by exchanging internal forces and torques. These forces and torques, which are due to spring and damper elements internal to the joint itself, enable you to account for mechanical energy dissipation and storage between the joint frames. You can sense the total composite force and torque acting at a joint, which includes contributions from actuation, constraint, and internal forces and torques. To sense the total torque acting between the port frames of the Revolute Joint block:

- 1 In the Revolute Joint block dialog box, select **Composite Force/Torque Sensing > Total Torque**. The block exposes the physical signal port `tt`. This port outputs the total torque acting between the joint frames as a Simscape physical signal.
- 2 Deselect **Composite Force/Torque Sensing > Constraint Force**.
- 3 Simulate the model.
- 4 At the MATLAB command prompt, enter:

```
figure;  
plot(simout);
```

MATLAB plots the vector components of the total torque vector as a function of time. All but one component are zero throughout simulation. The nonzero component, a torque directed about the Z axis, contains torque contributions from actuation and internal torques, but none from constraint torques.



The torque peaks correspond to the actuation torque values specified in the input signal. These peaks decay with time due to the internal damping torques specified in the Revolute Joint block dialog box. The damping torques cause the energy dissipation evident in the transient portions of the total torque plot.

To verify that the total torque excludes any contribution from constraint torques, try sensing the constraint torques directly. A plot of the constraint torques will show that they are in fact negligible.

## Sense Constraint Forces

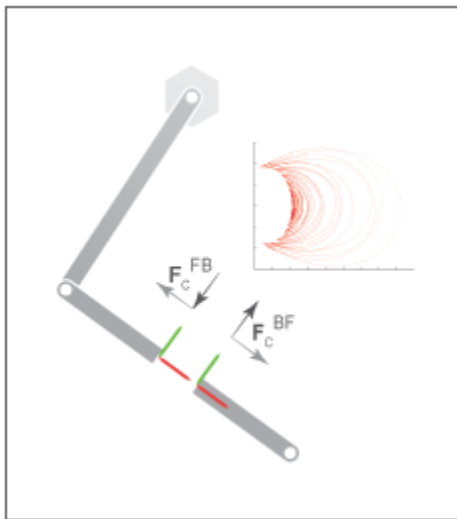
### In this section...

- “Model Overview” on page 4-99
- “Add Weld Joint Block to Model” on page 4-100
- “Add Constraint Force Sensing” on page 4-101
- “Add Damping to Joints” on page 4-102
- “Simulate Model” on page 4-102
- “Plot Constraint Forces” on page 4-103

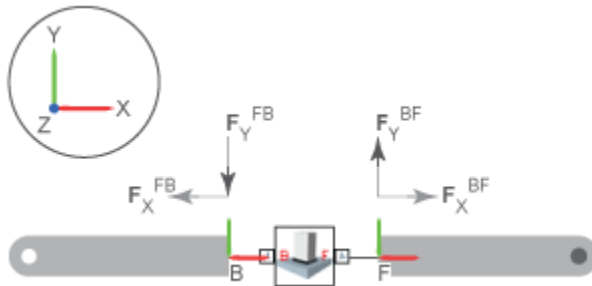
### Model Overview

Simscape Multibody provides various types of force and torque sensing. Using joint blocks, you can sense the actuation forces and torques driving individual joint primitives. You can also sense the total and constraint forces acting on an entire joint.

In this tutorial, you use a Weld Joint block to sense the time-varying internal forces that hold a rigid body together. A double-pendulum model, `smdoc_double_pendulum`, provides the starting point for the tutorial. For information on how to create this model, see “Model an Open-Loop Kinematic Chain” on page 3-16.



By connecting the Weld Joint block between solid elements in a binary link subsystem, you can sense the constraint forces acting between them. The following figure shows the constraint forces that you sense in this tutorial. The longitudinal constraint force aligns with the X axis of the weld joint frames. The transverse constraint force aligns with the Y axis. The constraint force along the Z axis is negligible and therefore ignored.



The Weld Joint block enables you to sense the constraint force that the follower frame exerts on the base frame or, alternatively, the constraint force that the base frame exerts on the follower frame. The two forces have the same magnitude but, as shown in the binary link schematic, opposite directions. In this tutorial, you sense the constraint force that the follower frame exerts on the base frame.

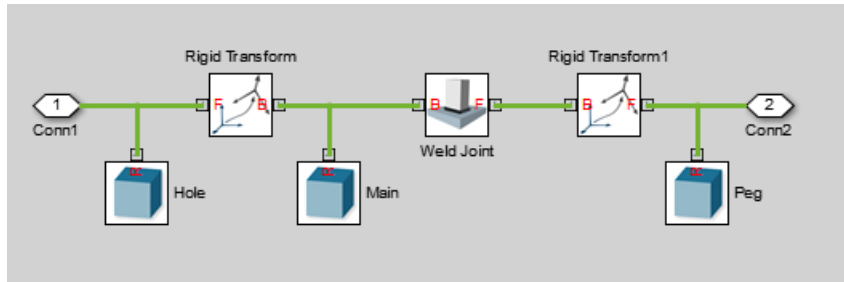
You can also select the frame to resolve the constraint force measurement in. The resolution frame can be either the base frame or the follower frame. Certain joint blocks allow their port frames to have different orientations, causing the same measurement to differ depending on your choice of resolution frame. However, because the Weld Joint block provides zero degrees of freedom, both resolution frames yield the same constraint force vector components.

## Add Weld Joint Block to Model

- 1 At the MATLAB command prompt, enter `smdoc_double_pendulum`. A double-pendulum model opens up.
- 2 Click the Look Inside Mask arrow in the Binary Link A1 subsystem block.
- 3 From the **Simscape** > **Multibody** > **Joints** library, drag a **Weld Joint** block.
- 4 Connect the Weld Joint block as shown in the figure. This block enables you to sense the constraint forces that hold the rigid body together during motion. Because it

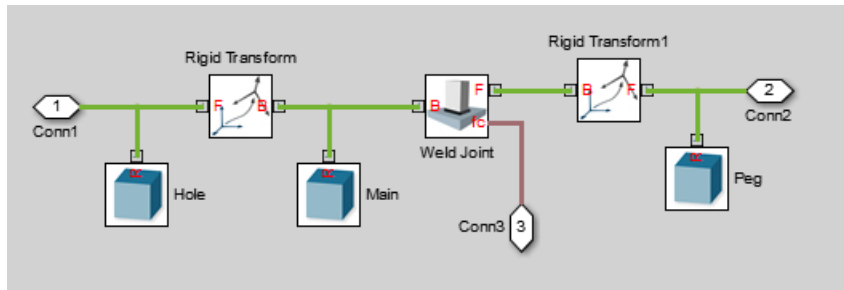


provides zero degrees of freedom between its port frames, it has no effect on model dynamics.



## Add Constraint Force Sensing

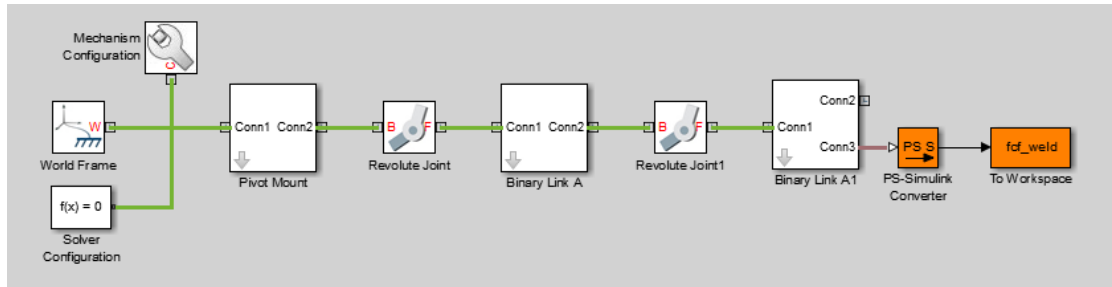
- 1 In the Weld Joint block dialog box, select **Constraint Force**. The block exposes a physical signal output port labeled fc.
- 2 Add a Simscape Output port to the subsystem block diagram. Connect the block as shown in the figure and exit the subsystem view.



- 3 Drag the following blocks into the main window of the model. These blocks enable you to output the constraint force signal into the MATLAB workspace.

Library	Block
Simscape > Utilities	PS-Simulink Converter
Simulink > Sinks	To Workspace

- 4 Connect the blocks as shown in the figure. Check that the PS-Simulink Converter block connects to the newly added Simscape port.



5 Specify these block parameters.

Block	Dialog Box Parameter	Value
PS-Simulink Converter	Output signal unit	mN
To Workspace	Variable name	fcf_weld

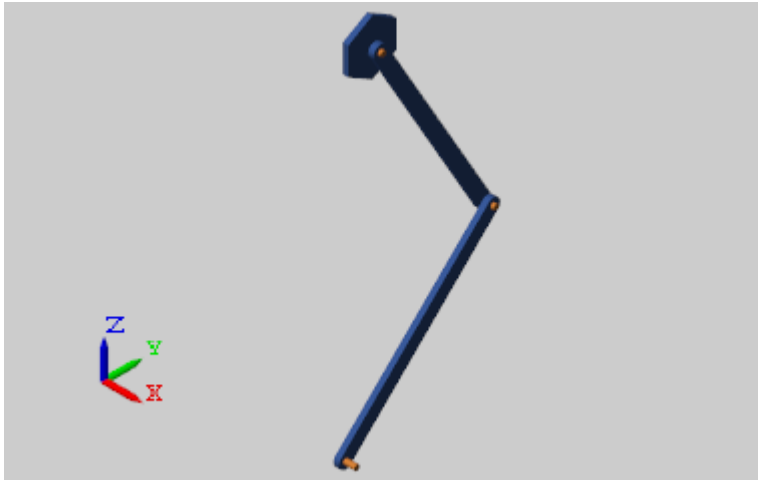
Units of mN are appropriate for this model, which contains Aluminum links roughly 30 cm × 2 cm × 0.8 cm.

## Add Damping to Joints

In each Revolute Joint block dialog box, select **Internal Mechanics > Damping Coefficient** and enter  $1e-5$ . The damping coefficient enables you to model energy dissipation during motion, so that the double-pendulum model eventually comes to rest.

## Simulate Model

- 1 In the Simulink Editor menu bar, select **Simulation > Model Configuration Parameters**.
- 2 In the Solver tab of the Configuration Parameters window, set the **Solver** parameter to **ode15s**. This is the recommended solver for physical models.
- 3 In the same tab, set the **Max step size** parameter to **0.001 s**.
- 4 Run the simulation. You can do this from the Simulink Editor menu bar, by selecting **Simulation > Run**. Mechanics Explorer opens with a dynamic view of the model. In the Mechanics Explorer menu bar, select the Isometric View button to view the double pendulum from an isometric perspective.

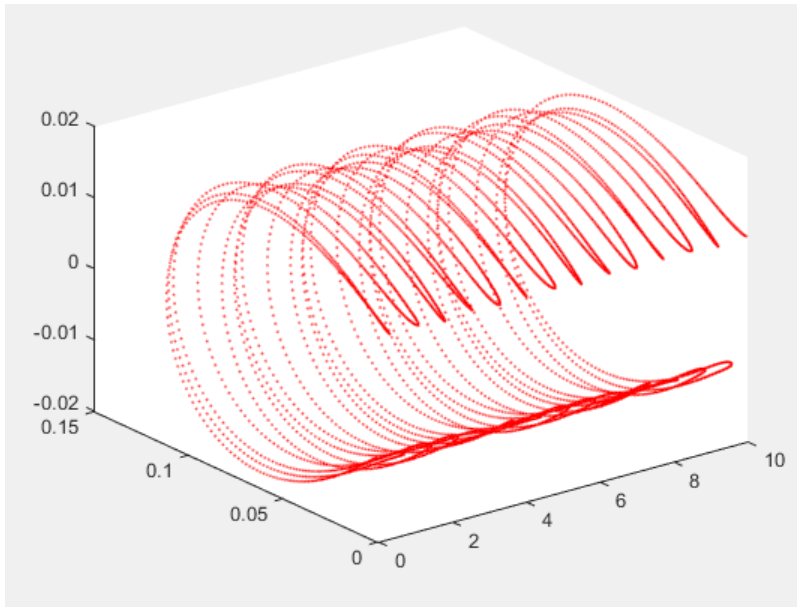


## Plot Constraint Forces

At the MATLAB command prompt, enter the following plot commands:

```
figure;  
grid on;  
xlabel('T, s');  
ylabel('F_{C,X}, mN');  
zlabel('F_{C,Y}, mN');  
plot3(gcf_weld.time, fcf_weld.data(:,1), fcf_weld.data(:,2),...  
'.', 'MarkerSize', 1, 'Color', 'r');
```

MATLAB plots the axial and transverse constraint forces with respect to time in 3-D. The figure shows the resulting plot.



## Specify Joint Motion Profile

### In this section...

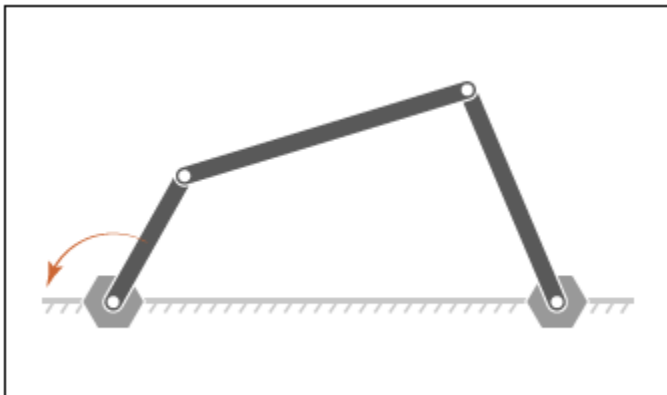
“Model Overview” on page 4-105

“Build Model” on page 4-105

“Simulate Model” on page 4-108

### Model Overview

In this tutorial, you prescribe the time-varying crank angle of a four-bar linkage using a Revolute Joint block. Then, during simulation, you sense the actuation torque at the same joint corresponding to the prescribed motion.



### Build Model

- 1 At the MATLAB command prompt, enter `smdoc_four_bar`. A four-bar model opens. This is the model you create in tutorial “Model a Closed-Loop Kinematic Chain” on page 3-21.
- 2 In the dialog box of the Base-Crank Revolute Joint block, specify the following parameters settings.

Parameter	Setting
Actuation > Torque	Automatically Computed

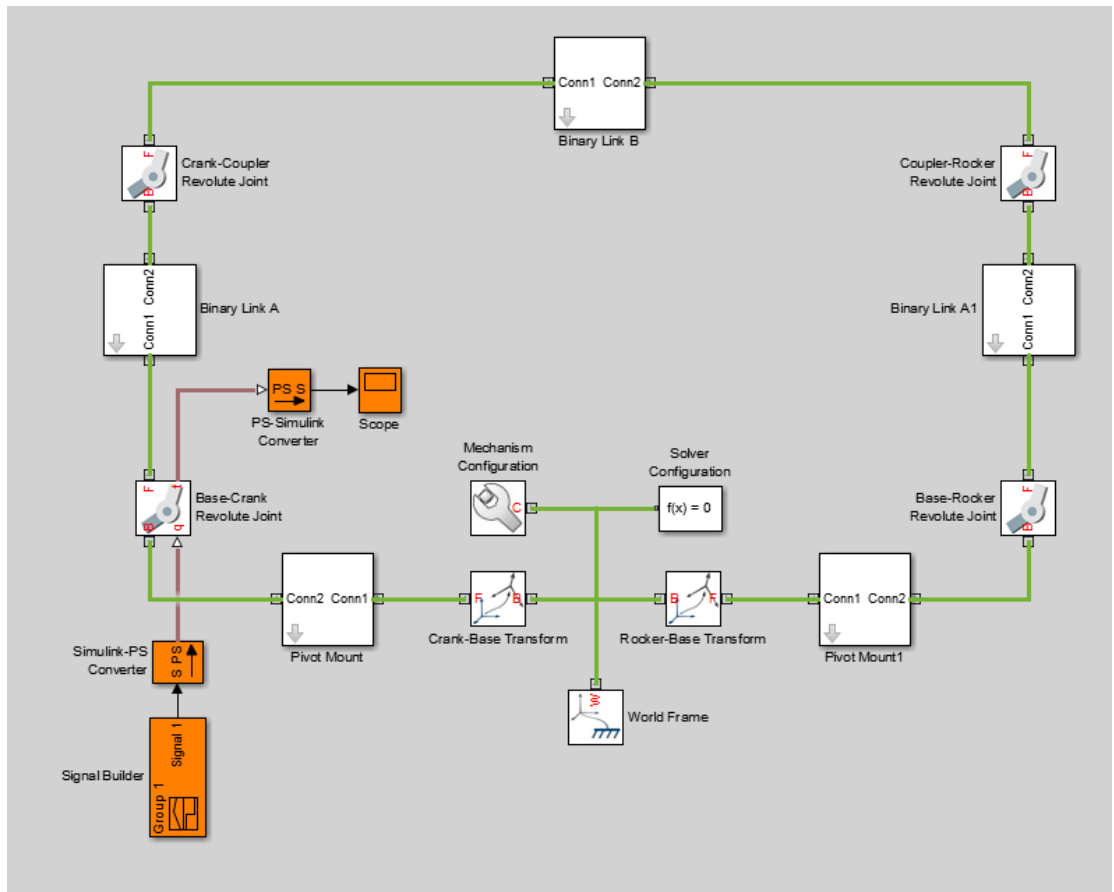
Parameter	Setting
Actuation > Motion	Provided by Input
Sensing > Actuator Torque	Selected

The joint block displays two physical signal ports. Input port  $q$  accepts the joint angular position. Output port  $t$  provides the joint actuation torque required to achieve that angular position.

- 3 In each of the four Revolute Joint block dialog boxes, set **Internal Mechanics > Damping Coefficient** to  $5e-4 \text{ N}\cdot\text{m}/(\text{deg}/\text{s})$ . During simulation, damping forces between the joint frames account for dissipative losses at the joints.
- 4 Drag the following blocks into the model. These blocks enable you to specify an actuation torque signal and plot the joint position.

Block	Library
Simulink-PS Converter	Simscape > Utilities
PS-Simulink Converter	Simscape > Utilities
Scope	Simulink > Sinks
Signal Builder	Simulink > Sources

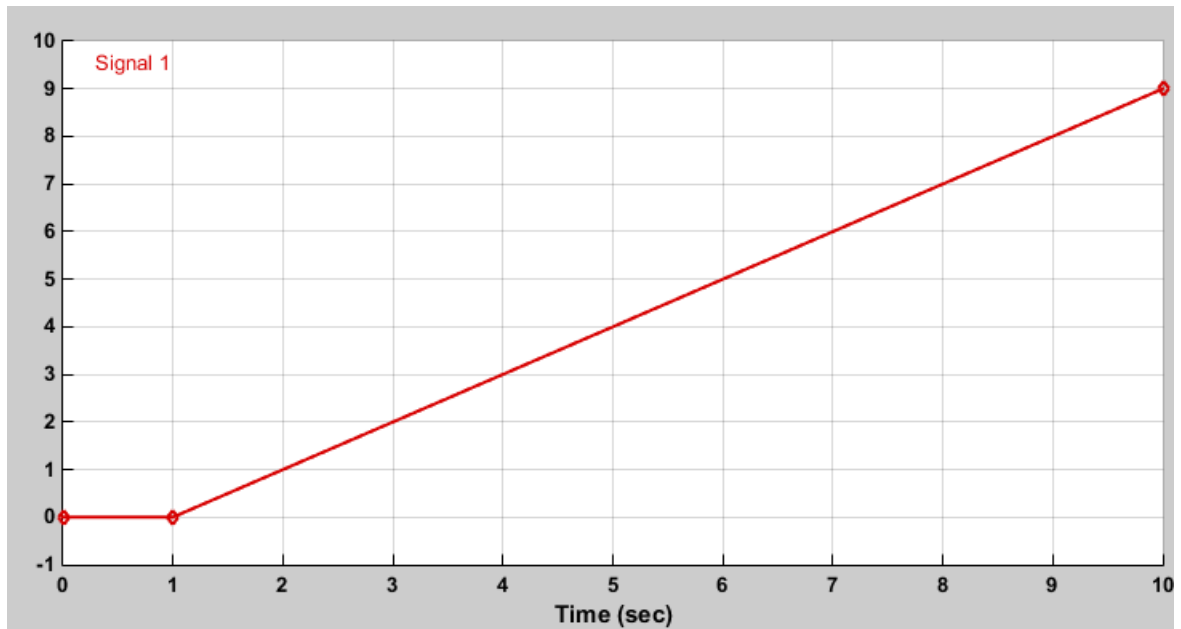
- 5 Connect the blocks as shown in the figure.



- 6 In the Input Handling tab of the Simulink-PS Converter block dialog box, specify the following block parameters.

Parameter	Value
<b>Filtering and derivatives</b>	Filter input
<b>Input filtering order</b>	Second-order filtering

- 7 In the Signal Builder window, specify the joint angular trajectory as shown in the figure.

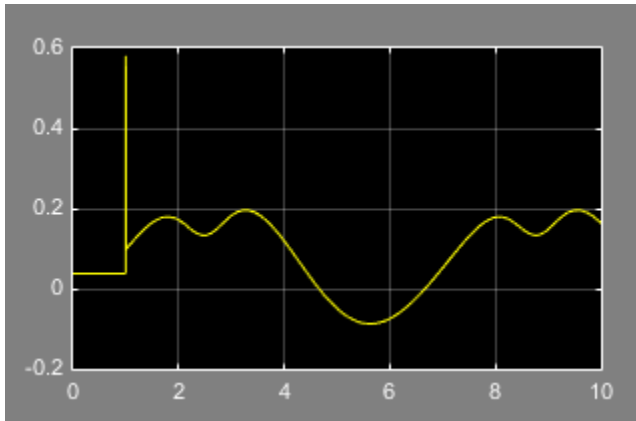


This signal corresponds to a constant angular speed of 1 rad/s from  $t = 1$ s onwards.

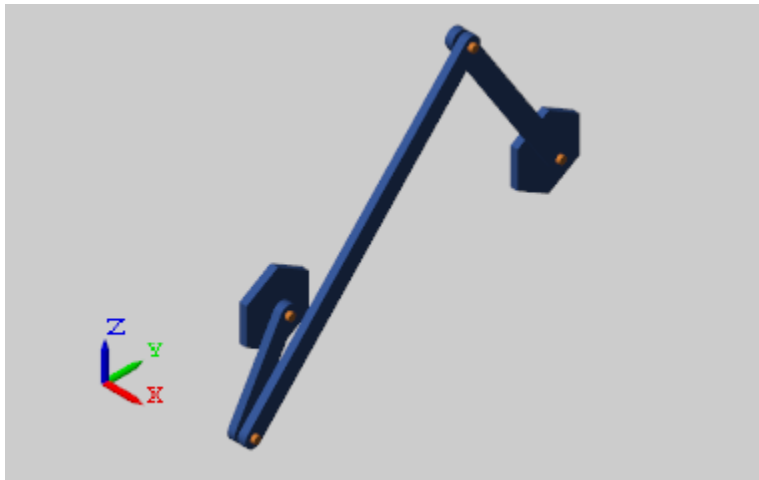
### Simulate Model

Run the simulation, e.g., by selecting **Simulation** > **Run** from the Simulink menu bar. Mechanics Explorer opens with a dynamic display of the four-bar model.





Open the Scope window. The scope plot shows the joint actuation torque with which you can achieve the motion you prescribed.



## Related Examples

- “Sense Motion” on page 4-67
- “Prescribe Joint Motion in Planar Manipulator Model” on page 4-110
- “Specifying Motion Input Derivatives” on page 4-32

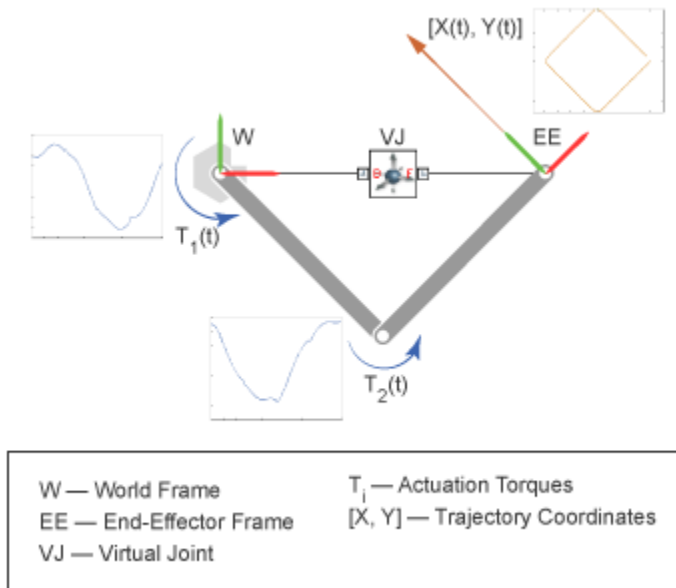
## Prescribe Joint Motion in Planar Manipulator Model

In this section...
“Model Overview” on page 4-110
“Add Virtual Joint” on page 4-111
“Prescribe Motion Inputs” on page 4-112
“Sense Joint Actuation Torques” on page 4-116
“Simulate Model” on page 4-117

### Model Overview

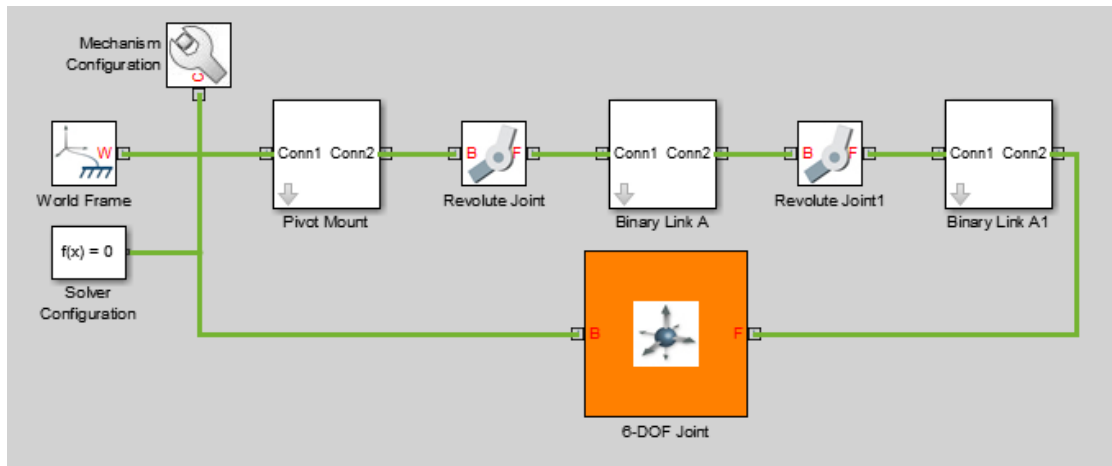
In this tutorial, you prescribe the time-varying trajectory coordinates of a planar manipulator end frame with respect to the world frame using a 6-DOF Joint block. This block provides the requisite degrees of freedom between the two frames, but it does not represent a real physical connection between them. The joint it represents is said to be virtual.

The time-varying coordinates trace a square pattern, achieved by automatically computing and applying actuation torques at the various manipulator joints. During simulation, you can output the automatically computed torques and plot them using Simulink blocks or MATLAB commands, e.g. for analysis purposes.



## Add Virtual Joint

- 1 At the MATLAB command prompt, enter `smdoc_double_pendulum`. A double pendulum model, which in this tutorial you adapt as a simple planar manipulator model, opens. For instructions on how to create this model, see “Model an Open-Loop Kinematic Chain” on page 3-16
- 2 From the **Simscape** > **Multibody** > **Joints** library, drag a 6-DOF Joint block and connect it as shown in the figure. This block represents a virtual joint, which you use to specify the manipulator end frame with respect to the world frame.



**Note:** Check that the base port frame (B) connects to the world frame. The base port frame functions as the reference frame for any joint motion input that you provide. Switching the base and follower port frames causes the block to interpret any motion input with respect to a different frame, possibly altering the manipulator end frame trajectory.

## Prescribe Motion Inputs

- 1 In the 6-DOF Joint block dialog box, specify these parameter settings.

Parameter	Select
<b>Y Prismatic Primitive (Py) &gt; Actuation &gt; Motion</b>	Provided by Input
<b>Z Prismatic Primitive (Pz) &gt; Actuation &gt; Motion</b>	Provided by Input

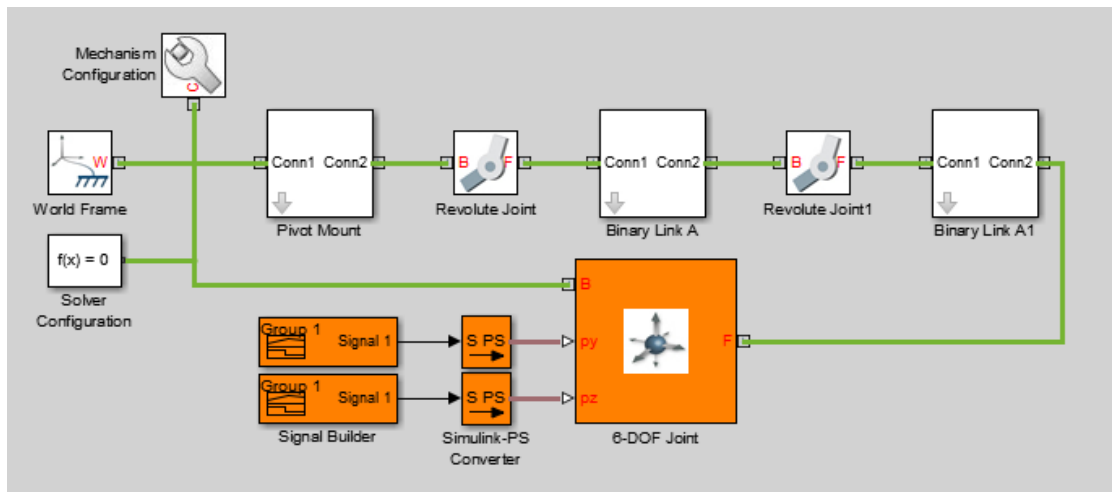
The block exposes two physical signal ports through which you can provide the joint motion inputs.

- 2 Drag these blocks into the model.

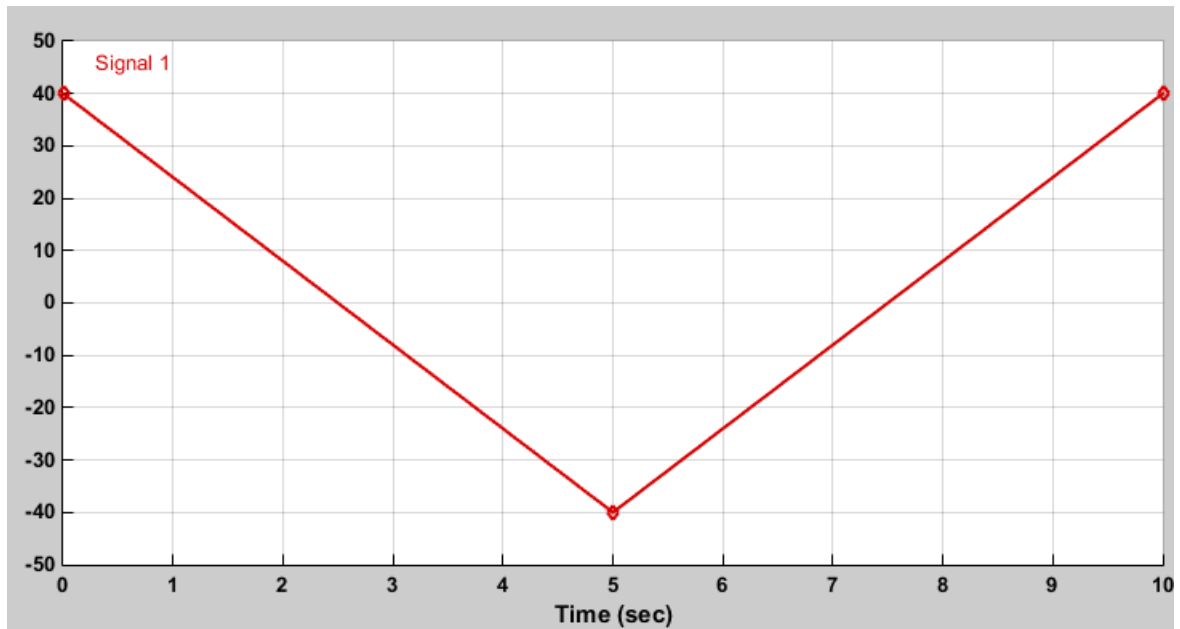
Library	Block	Quantity
Simscape > Utilities	Simulink-PS Converter	2
Simulink > Sources	Signal Builder	2

The Signal Builder blocks provide the motion inputs as Simulink signals. The Simulink-PS Converter blocks convert the Simulink signals into Simscape physical signals compatible with Simscape Multibody blocks.

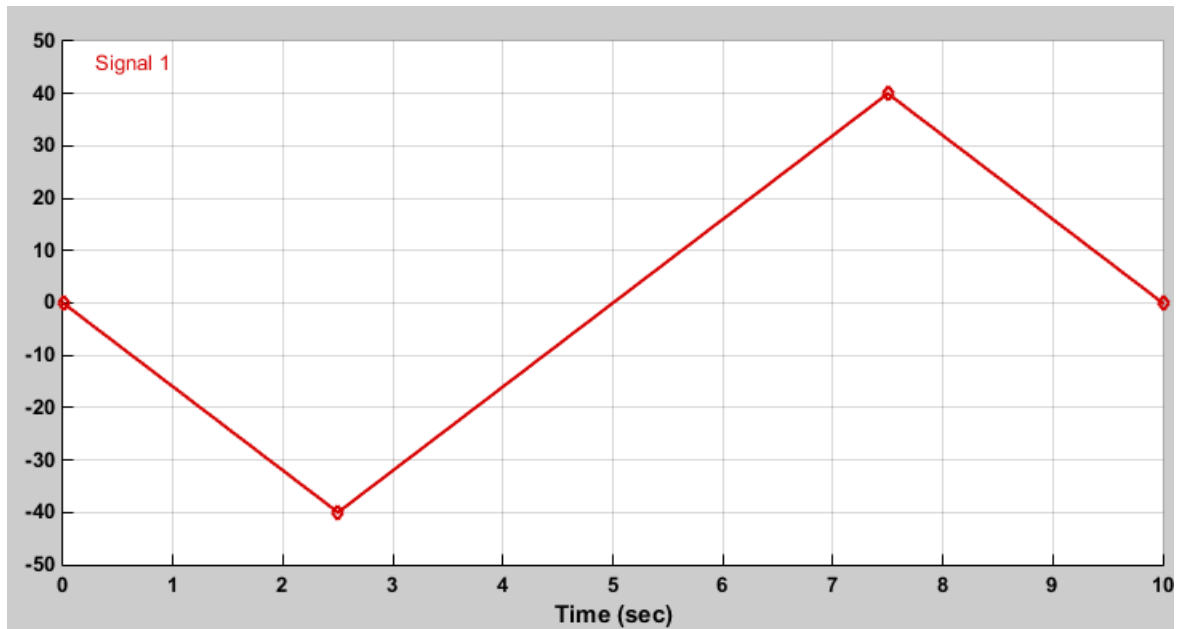
- 3 Connect the blocks as shown in the figure.



- 4 Open the dialog box of the Signal Builder block connected to port  $py$  of the 6-DOF Joint block. Specify this signal, the time-varying Y coordinate of the square trajectory the manipulator end frame is to follow.



- 5 Open the dialog box of the Signal Builder block connected to port pz of the 6-DOF Joint block. Specify this signal, the time-varying Z coordinate of the square trajectory the manipulator end frame is to follow.



- 6 In the dialog boxes of the Simulink-PS Converter blocks, specify the input signal units and filtering settings. Simscape Multibody requires that you either specify second-order filtering or provide the first two time derivatives of the trajectory coordinates.

Parameter	Value
Units > Input signal unit	cm
Input Handling > Filtering and derivatives	Filter input
Input Handling > Input filtering order	Second-order filtering
Input Handling > Input filtering time constant (in seconds)	0.1

Small filtering constants can slow simulation significantly. For most Simscape Multibody models, a value of 0.1 seconds is a good choice. In this tutorial, this value suffices.

## Sense Joint Actuation Torques

- 1 In the dialog boxes of the two Revolute Joint blocks, set the following actuation and sensing parameters.

Parameter	Setting
Actuation > Torque	Automatically Computed
Sensing > Actuation Torque	Selected

Simscape Multibody requires the number of joint primitive degrees of freedom with motion inputs to equal the number with automatically computed joint actuation forces and torques. If the model does not meet this condition, simulation fails with an error.

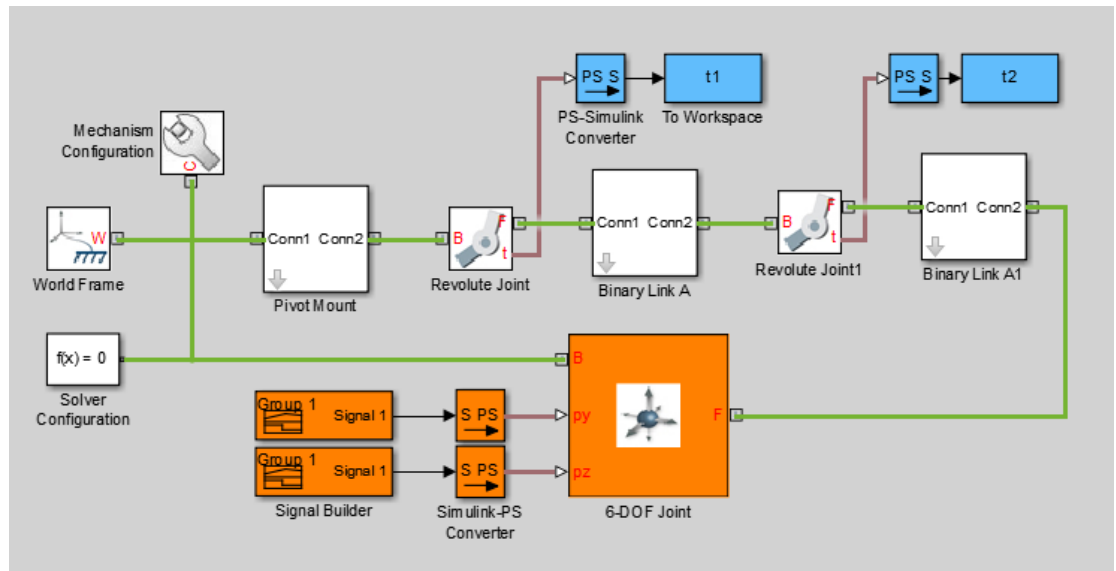
- 2 Drag these blocks into the model.

Library	Block	Quantity
Simscape > Utilities	PS-Simulink Converter	2
Simulink > Sinks	To Workspace	2

The PS-Simulink Converter blocks convert the physical signal outputs into Simulink signals compatible with other Simulink blocks.

- 3 In the two To Workspace block dialog boxes, enter the variable names `t1` and `t2`.
- 4 Connect the blocks as shown in the figure.

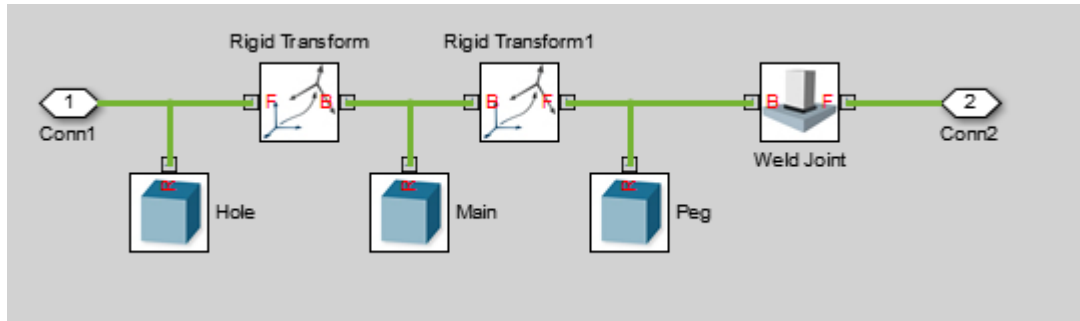




## Simulate Model

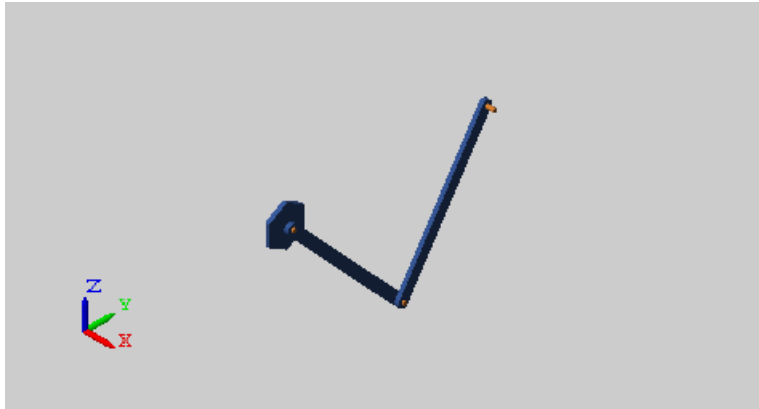
Attempt to run the simulation. You can do this in the Simulink Editor menu bar, by selecting **Simulation > Run**. Simulation fails with an error arising from the closed kinematic loop present in the model. Simscape Multibody requires this loop to contain at least one joint block without motion inputs or automatically computed actuation forces or torques.

- 1 From the **Simscape > Multibody > Joints** library, drag a Weld Joint block and connect it inside one of the Binary Link A subsystems.



Adding the Weld Joint block ensures that the now-closed-loop system contains at least one joint block without motion inputs or computed actuation torques.

Run the simulation once again. Mechanics Explorer opens with a dynamic 3-D display of the two-bar linkage.

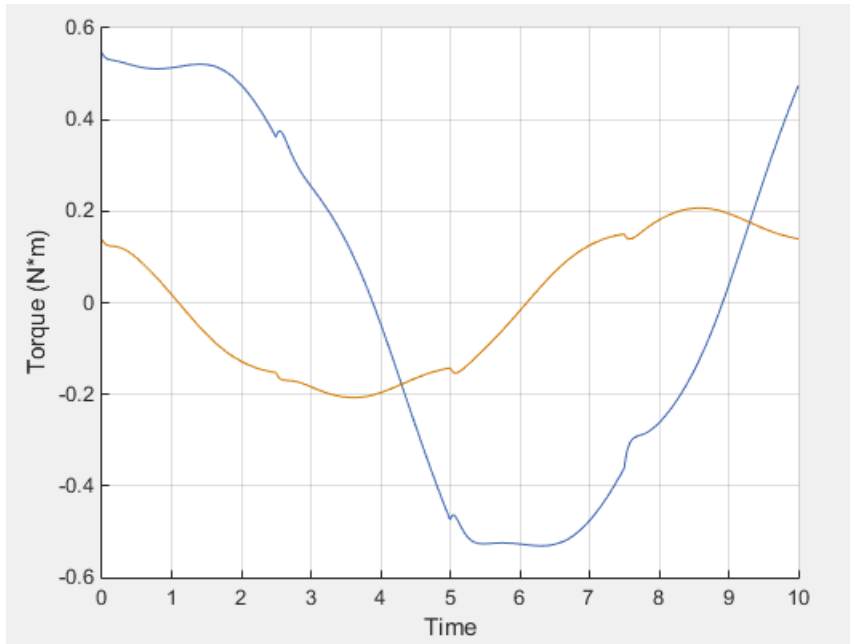


Plot the computed actuation torques acting at the two revolute joints in the linkage. At the MATLAB command line, enter this code:

```
figure;
hold on;
plot(t1.time, t1.data, 'color', [60 100 175]/255);
plot(t2.time, t2.data, 'color', [210 120 0]/255);
xlabel('Time');
ylabel('Torque (N*m)');
```

grid on;

The plot shows the time-varying actuation torques acting at the two revolute joints. These torques enable the manipulator end frame to trace the prescribed square trajectory.



## Related Examples

- “Sense Motion” on page 4-67
- “Specify Joint Motion Profile” on page 4-105
- “Specifying Motion Input Derivatives” on page 4-32

## More About

- “Joint Actuation” on page 4-25
- “Actuating and Sensing with Physical Signals” on page 4-36



# Simulation and Analysis



# Simulation

---

- “Update and Simulate a Model” on page 5-2
- “Troubleshoot Simulation Issues” on page 5-5

## Update and Simulate a Model

### In this section...

“Create or Open a Model” on page 5-2

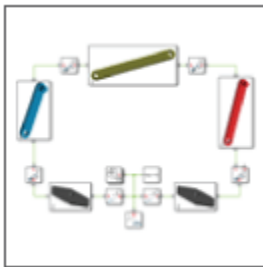
“Update the Block Diagram” on page 5-2

“Examine the Model Assembly” on page 5-3

“Configure the Solver Settings” on page 5-3

“Run Simulation and Analyze Results” on page 5-4

### Create or Open a Model



You can create a model manually or import it from a supported CAD application. For an example showing how to create a model manually, see “Model an Open-Loop Kinematic Chain” on page 3-16. For an example showing how to import a model from CAD, see “Import a Robotic Arm Model” on page 7-17.

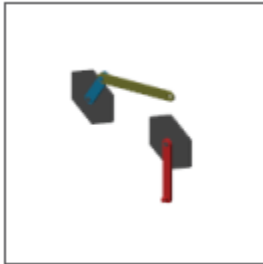
### Update the Block Diagram





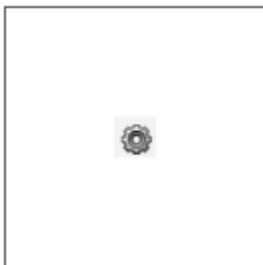
In the Simulink menu bar, select **Simulation > Update Diagram**. Mechanics Explorer opens with a static visualization of the model in its initial state.

## Examine the Model Assembly



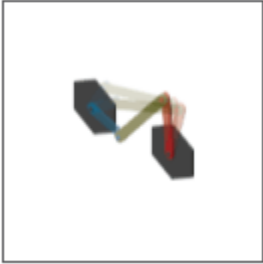
Check the model assembly in the visualization pane of Mechanics Explorer. Look for bodies placed and oriented in unexpected ways. Use the Simscape Variable Viewer or the Simscape Multibody Model Report to identify any assembly issues. For an example showing the use of Model Report, see “Model a Closed-Loop Kinematic Chain” on page 3-21.

## Configure the Solver Settings



In the Simulink menu bar, select **Simulation > Model Configuration Parameters**. Pick a solver and specify the desired step sizes. Ensure that the time steps are small enough to accurately capture the fastest meaningful changes in your model. Use care, though, as small time steps slow down simulation.

### Run Simulation and Analyze Results



In the Simulink menu bar, select **Simulation > Run**. Mechanics Explorer plays a physics-based animation of your model. Examine any data generated during simulation, for example, through Simulink Scope plots. For an example showing how to work with sensing data from a model, see “Sense Motion” on page 4-67.

## Troubleshoot Simulation Issues

### In this section...

“Models with For Each Subsystem Blocks Have Limited Visualization” on page 5-5

“Models with Model Blocks Have No Visualization” on page 5-5

“Simscape Local Solvers Do Not Work with Simscape Multibody” on page 5-5

Under certain conditions, a model that you simulate can behave in unexpected ways. Some issues that you can encounter while simulating a Simscape Multibody model include:

- Models with For Each Subsystem blocks have limited visualization
- Models with Model blocks have no visualization
- Simscape local solvers do not work for Simscape Multibody

### Models with For Each Subsystem Blocks Have Limited Visualization

Models with one or more **For Each Subsystem** blocks simulate with limited visualization. The **Mechanics Explorer** visualization utility displays the model in only one of the instances which the **For Each Subsystem** block provides. The visualization limitation does not affect model simulation—Simscape Multibody simulates the model for all instances of the block.

### Models with Model Blocks Have No Visualization

Models with **Model** blocks (known as referenced models) simulate with no visualization. During model simulation, Simscape Multibody issues a warning at the MATLAB command line. The **Mechanics Explorer** visualization utility does not open.

### Simscape Local Solvers Do Not Work with Simscape Multibody

Simscape Multibody software does not support Simscape local solvers. If you select a local solver in the **Simscape Solver Configuration** block, the solver does not apply to the Simscape Multibody portion of a model. Simscape Multibody blocks continue to use the Simulink global solver that you select in **Model Configuration Parameters** for your model.

---

**Note:** Simscape Multibody requires the Simulink global solver to be *continuous*. If the global solver is discrete, Simscape Multibody issues an error and the model does not simulate. This requirement applies to both fixed- and variable-step solvers.

---

### Related Examples

- “Configure Model for Rapid Accelerator Mode” on page 8-7

# Visualization and Animation

---

- “Enable Mechanics Explorer” on page 6-2
- “Model Animation” on page 6-3
- “Manipulate the Visualization Viewpoint” on page 6-5
- “Visualization Cameras” on page 6-10
- “Create a Dynamic Camera” on page 6-15
- “Selective Model Visualization” on page 6-19
- “Selectively Show and Hide Model Components” on page 6-25
- “Visualize Simscape Multibody Frames” on page 6-32
- “Go to a Block from Mechanics Explorer” on page 6-37
- “Create a Model Animation Video” on page 6-39
- “Multibody Visualization Issues” on page 6-41

## Enable Mechanics Explorer

The Mechanics Explorer visualization utility opens by default whenever you update or simulate a Simscape Multibody model. Each model that you update or simulate adds a new tab to Mechanics Explorer. If Mechanics Explorer does not open, ensure that model visualization is turned on:

- 1 In the Simscape Multibody menu bar, select **Simulation > Model Configuration Parameters**.
- 2 Expand the **Simscape Multibody** node and select **Explorer**.
- 3 Ensure the **Open Mechanics Explorer on model update or simulation** check box is selected.

# Model Animation

## In this section...

“Animation Playback” on page 6-3

“Looping Playback” on page 6-3

“Changing Playback Speed” on page 6-3

“Jumping to Playback Time” on page 6-4

## Animation Playback

Animation is cached during model simulation. What you see when you run a simulation is the animation playback, unless the simulation is slower than the animation caching. In that case, the animation goes no faster than the simulation can produce the cache.

Once a partial or complete animation is cached from simulation, starting the animation again plays back the cache, without running the simulation a second time. You can move backward and forward to any time in the cached animation.

The animation cache is stored until you close Mechanics Explorer. When you simulate the model, the cache is updated with new animation data. To create a permanent record of a model animation, you must create an animation video. See “Create a Model Animation Video” on page 6-39

## Looping Playback

Use the Loop button in the Mechanics Explorer playback toolbar to automatically replay an animation from the start once it reaches the end. The cached animation replays indefinitely until you click the **Stop** button. Enable looping by clicking the Loop button. Disable looping by clicking the button again.

## Changing Playback Speed

Use the playback speed slider in the Mechanics Explorer toolbar to set the animation playback speed ratio relative to real time. Set the slider to a number greater than 1 for faster playback. Set it to a number smaller than 1 for slower playback.

You can set the slider to multiples of 2 from 1/256 to 256. For slower or faster animations, adjust the base playback speed for the model. To change this parameter, from the Mechanics Explorer menu bar, select **Tools > Animation Settings**.

## **Jumping to Playback Time**

Use the playback slider in the Mechanics Explorer toolstrip to move the playback time to an arbitrary point in the animation timeline. The playback time counter shows the current playback time. Alternatively, enter the desired playback time directly in the playback time counter.



## Manipulate the Visualization Viewpoint

### In this section...

- “Model Visualization” on page 6-5
- “Select a Standard View” on page 6-5
- “Set View Convention” on page 6-6
- “Rotate, Roll, Pan, and Zoom” on page 6-7
- “Split Model View” on page 6-8

### Model Visualization

Multibody models lend themselves to 3-D visualization, a qualitative means of analysis that you can use to examine rigid body geometries, mechanical connections, and trajectories in three-dimensional space. In Simscape Multibody, you can visualize a model using Mechanics Explorer, adjusting the view point and detail level as needed. You can modify the model view by:

- Selecting a view convention.
- Selecting a standard view.
- Rotating, panning, and zooming.

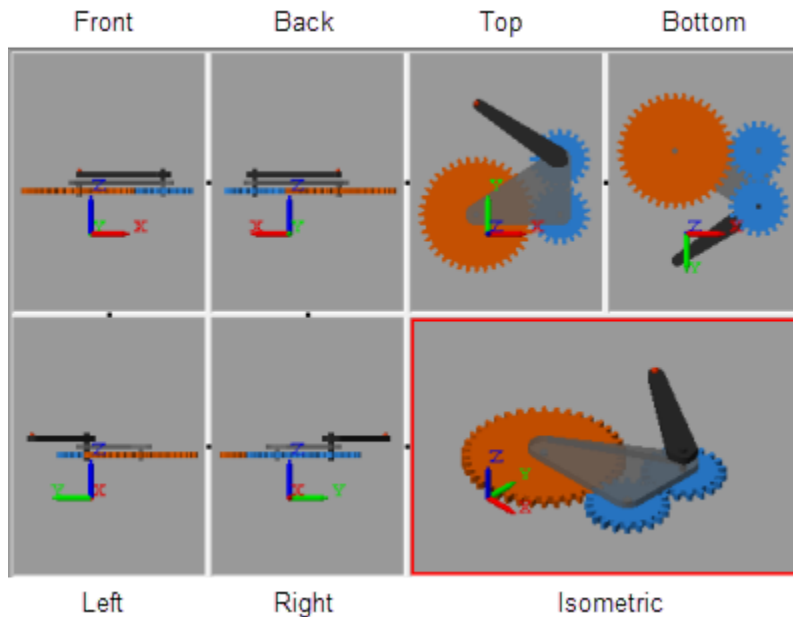
### Select a Standard View

Some view points are so widely used that they are called standard. The isometric view point, corresponding to equal  $120^\circ$  angles between any two world frame axes, is one example. In Mechanics Explorer, you can select such view points by clicking the standard view buttons.



### Standard View Buttons

The figure shows a Cardan gear model from the different view points using a Z up (XY Top) view convention.

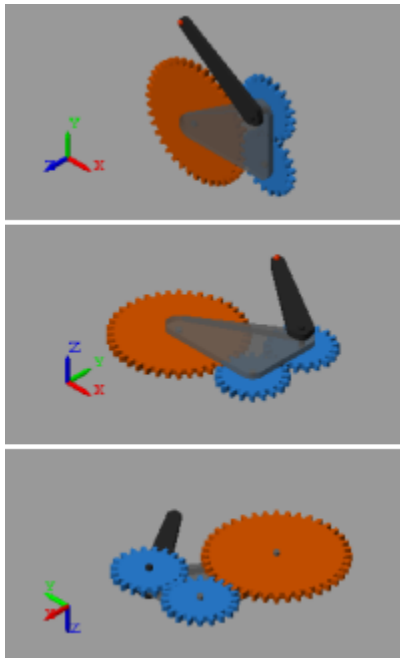


## Set View Convention

The view convention helps to determine the perspective from which you view your model. You can align three world frame axes with the vertical direction on your screen, each corresponding to a different view convention:

- Y up (XY Front)
- Z up (XY Top)
- Z down (YZ Front)

The figure shows a Cardan gear model from an isometric perspective using the three view conventions: Y up, Z up, and Z down.





To change the view convention:




- 1 In the Mechanics Explorer tool strip, set **View convention** to one of the three options.
- 2 Select a standard view button.

The new view convention takes effect the moment you select a standard view.

## Rotate, Roll, Pan, and Zoom

To view your model from an arbitrary point of view or at varying zoom levels, use the Rotate, Roll, Pan, and Zoom buttons. You can find these buttons in the Mechanics Explorer tool strip:

-  — Rotate the camera about a general 3-D axis.
-  — Roll the camera about its current aim axis.

-  — Pan the camera in the current visualization plane.
-  — Increase or decrease the camera zoom level.
-  — Change the camera zoom to show only the selected region.

You can also use keyboard-and-mouse shortcuts. The table summarizes the available shortcuts.

Button	Shortcut
Rotate	<ol style="list-style-type: none"> <li>1 Click and hold the mouse scroll wheel.</li> <li>2 Move the mouse in the direction you want to rotate the model.</li> </ol>
Pan	<ol style="list-style-type: none"> <li>1 Press and hold <b>Shift</b>.</li> <li>2 Click and hold the mouse scroll wheel.</li> <li>3 Move the mouse in the direction you want to pan the model.</li> </ol>
Zoom	<ol style="list-style-type: none"> <li>1 Press and hold <b>Ctrl</b>.</li> <li>2 Click and hold the mouse scroll wheel.</li> <li>3 Move the mouse up to zoom in, down to zoom out.</li> </ol>

## Split Model View

You can view your model from different perspectives, for example, to examine its motion in different planes. So that you can compare different model views, Mechanics Explorer enables you to split the visualization pane into tiles, each with its own view. To split the screen, you use the Mechanics Explorer toolstrip buttons shown in the figure.

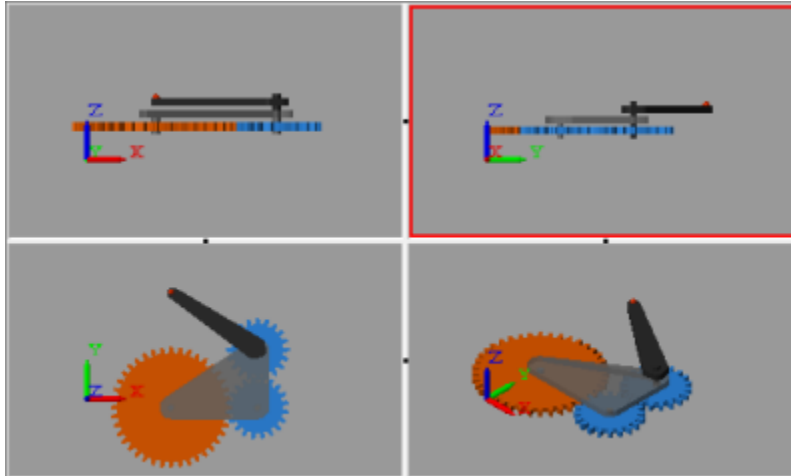


Use the buttons to:

- Split the model view into four equally sized tiles, each with a different view point (front, right, top, and isometric views).
- Merge all tiles into a single pane with the view point of the last highlighted tile.

- Split a visualization tile vertically or horizontally into two equally sized tiles.

The figure shows the Cardan gear model with a four-way visualization split.



You can merge two tiles by clicking the black dot between the tiles. To ensure that the resulting tile uses the view point of one or the other tile, select that tile first before clicking the black dot between the tiles.

## Visualization Cameras

### In this section...

“Camera Types” on page 6-10

“Global Camera” on page 6-11

“Dynamic Cameras” on page 6-11

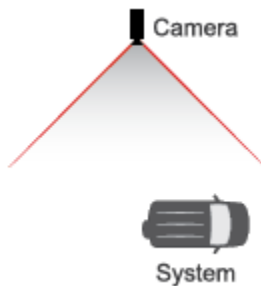
“Camera Trajectory Modes” on page 6-12

“Dynamic Camera Selection” on page 6-13

“Dynamic Camera Reuse” on page 6-14

### Camera Types

Cameras define the model viewpoints used during animation playback. Mechanics Explorer supports two camera types—global and dynamic. The global camera provides a static viewpoint that you can manipulate interactively during animation playback. Dynamic cameras provide moving viewpoints that you predefine using Camera Manager.



### Camera in a Model

The moving viewpoint of a dynamic camera enables you to more easily track the motion of a system. You can use a dynamic camera to keep a moving vehicle such as an automobile or aircraft in view during animation playback. You must define and select a dynamic camera in order to use it in a model. For an example, see “Create a Dynamic Camera” on page 6-15.

## Global Camera

The global camera is a static camera that:

- Has no planned trajectory.

You must manipulate the camera manually to change the camera viewpoint, for example, by using the Pan, Rotate, Roll, and Zoom buttons.

- Is external to the model.

You cannot position the global camera between bodies, for example, to prevent one body from obstructing another during animation playback.

- Uses an orthographic projection.

Apparent body sizes remain constant regardless of object distance to the camera. This effect, shown in the figure, is consistent with a camera located relatively far from the model.



The global camera is the default camera for all model visualization tiles—each a subdivision of the model visualization pane, when split. In the absence of custom dynamic cameras, the global camera is the only camera available in a model.

## Dynamic Cameras

Dynamic cameras are custom cameras that:

- Have planned trajectories.

Every dynamic camera follows a trajectory that you prespecify through Camera Manager. You cannot use the Pan, Rotate, Roll, or Zoom buttons during animation playback.

- Can be internal to a model.

Dynamic cameras can be inside or outside the perimeter of a model. Position a camera between bodies for a viewpoint internal to the model.

- Use a perspective projection.

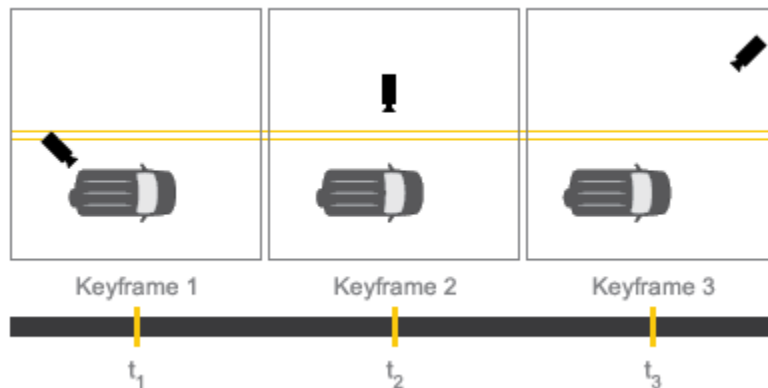
Apparent body sizes vary noticeably with object distance to the camera, creating a more realistic 3-D effect. This effect, shown in the figure, is consistent with a camera located relatively close to the model.



## Camera Trajectory Modes

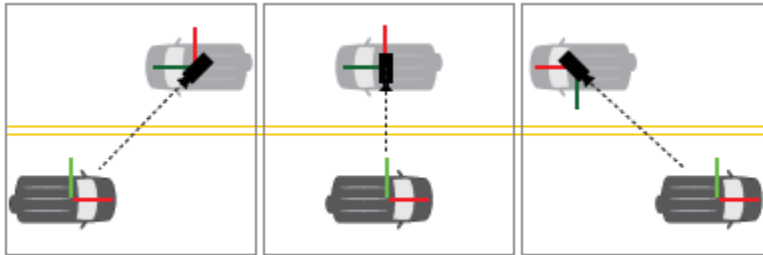
Camera Manager provides two dynamic camera modes that you use to define the camera trajectories:

- **Keyframes** — Set the camera viewpoints at various playback times. Each viewpoint constitutes a keyframe. During playback, the camera transitions between the keyframes using the smooth interpolation method of the `pchip` MATLAB function. Use this camera mode to obtain camera trajectories independent of any components in your model.



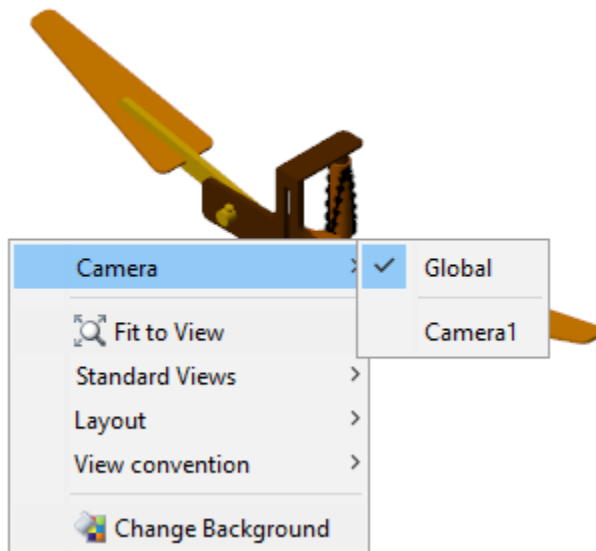
- **Tracking** — Constrain the camera position, aim, and up vector to coordinate frames in your model. During playback, the camera moves with the frames it is constrained to, translating and rotating as needed to satisfy the specified constraints. Use this camera mode to track frames and bodies during playback.





## Dynamic Camera Selection

To visualize a model using a dynamic camera, you must first select that camera. To do this, Mechanics Explorer provides the list of available cameras in the visualization pane context-sensitive menu. Right-click the visualization pane to open the menu and select **Camera** to select from the list.



## **Dynamic Camera Reuse**

Dynamic cameras exist only in the models they are defined in. The camera trajectories are based on model-specific frames or viewpoints and are not transferable to other models. You cannot move, copy, or reference a dynamic camera outside of its model. To use a camera in a different model, recreate the camera in that model.

## **Related Examples**

- “Create a Dynamic Camera” on page 6-15

## Create a Dynamic Camera

### In this section...

“Start a New Camera Definition” on page 6-15

“Define a Keyframes Camera” on page 6-16

“Define a Tracking Camera” on page 6-17

“Select a Dynamic Camera” on page 6-17

### Start a New Camera Definition

If you are new to dynamic cameras, see “Visualization Cameras” on page 6-10. To start a new dynamic camera:

- 1 Simulate the model that you want to add the camera to.

Dynamic cameras exist only in the models that you define them in.

- 2 In the Mechanics Explorer menu bar, select **Tools > Camera Manager**.

Camera Manager opens with a list of previously created dynamic cameras. The list is by empty until you create your first camera.

- 3 In Camera Manager, click the  button.

Camera Manager switches to a camera definition view that lets you select the camera mode and specify the camera motion.

- 4 In the **Camera Name** field, enter a name for your camera.

Make the camera name descriptive so that you can later identify it when selecting an active camera from the Mechanics Explorer visualization context-sensitive menu.

Complete the camera definition by selecting the camera mode and specifying the camera motion. See:

- “Define a Keyframes Camera” on page 6-16 to define the camera motion in **Keyframes** mode. Keyframes are viewpoints that you specify at various playback times and that Simscape Multibody software interpolates to obtain smooth camera trajectories.

- “Define a Tracking Camera” on page 6-17 to complete the camera definition in **Tracking** mode. Tracking constraints include position, aim, and up vector constraints that you specify relative to coordinate frames in a model.

### Define a Keyframes Camera

- 1 In Camera Manager, set the **Mode** parameter to **Keyframes**.

Camera Manager switches to a **Keyframes** view that lets you define the camera keyframes.

- 2 In the Mechanics Explorer toolstrip, set the playback time for the current keyframe.

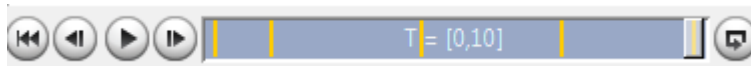
Drag the playback slider to the desired point in the animation timeline. Alternatively, enter the time directly in the playback time counter.

- 3 In the visualization pane or tile, manipulate the model viewpoint for your keyframe.

Use the Rotate, Roll, Pan, and Zoom buttons to manipulate the model viewpoint. Use the preset view buttons to obtain standard views such as front, side, or isometric.

- 4 In the Camera Manager **Keyframes** window, click the **Set** button.

Playback must be paused or stopped. Camera Manager commits the keyframe to the camera. The playback slider identifies the keyframe with a colored line marker located at the specified playback time.



- 5 Set new keyframes as in steps 2–4 until you are satisfied with the camera motion.

Simscape Multibody software transitions between keyframes using the smooth interpolation method of the `pchip` MATLAB function to yield the final camera motion.

- 6 Click the **Save** button in the camera definition and main panes of Camera Manager.

Camera Manager saves the camera and its motion to the model. The visualization context-sensitive menu adds the camera to the list of available cameras.

To edit an existing keyframe, use the **Previous** and **Next** buttons to navigate to the keyframe you want to edit. Then, repeat the procedure for adding a keyframe. Use the

colored markers in the playback slider to identify the existing keyframes in your dynamic camera.

Click the **Remove** button if you want to delete the current keyframe. Click the **Save** button in the main pane to commit your changes to the camera.

## Define a Tracking Camera

- 1 In Camera Manager, set the **Mode** parameter to **Tracking**.

Camera Manager switches to a **Tracking** view that lets you define the camera constraints—position, aim, and up vector—relative to frames in your model.

- 2 In the Camera Manager tracking window, set the camera **Position**, **Aim**, and **Up Vector** constraints:

- a In the tree view or visualization pane, select a frame to constrain the camera to.

If using the visualization pane, click a frame icon. If using the tree view pane, click a frame node. It is not enough to click the body that the frame belongs to.

- b Click the **Use Selected Frame** button to constrain the camera motion to the frame.

If you accidentally select the wrong frame, pick a new frame and click the **Use Selected Frame** button again.

- c For the **Aim** and **Up Vector** dropdown lists, select how to constrain the camera:
  - The **Position** constraint fixes the camera to the frame origin only and has no options dropdown list.
  - The **Aim** constraint provides the option to aim the camera at the frame origin or along a selected frame axis.
  - The **Up vector** constraint provides the option to align the up vector along a selected frame axis.

## Select a Dynamic Camera

The dynamic cameras that you create through Camera Manager are by default inactive during animation playback. To set a particular camera as the active camera for a visualization pane, use the visualization pane context-sensitive menu. You can perform

this task separately for each visualization pane that you have open in Mechanics Explorer:

- 1 Right-click the visualization pane or tile whose camera you want to switch.

The visualization context-sensitive menu opens up.

- 2 Select **Cameras** and, from the cameras list, select the desired camera.

The model viewpoint switches to that provided by the selected camera.

## Selective Model Visualization

### In this section...

“What Is Visualization Filtering?” on page 6-19

“Changing Component Visibility” on page 6-20

“Visualization Filtering Options” on page 6-21

“Components You Can Filter” on page 6-21

“Model Hierarchy and Tree Nodes” on page 6-22

“Filtering Hierarchical Subsystems” on page 6-22

“Updating Models with Hidden Nodes” on page 6-23

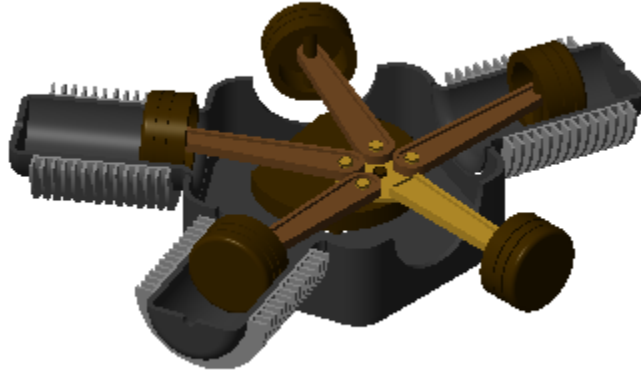
“Alternative Ways to Enhance Visibility” on page 6-24

### What Is Visualization Filtering?

A multibody model can get so complex that you cannot easily tell its components apart. Solids, bodies, and multibody subsystems often hide behind each other, hindering your efforts to examine geometry, pose, and motion on model update or during simulation.

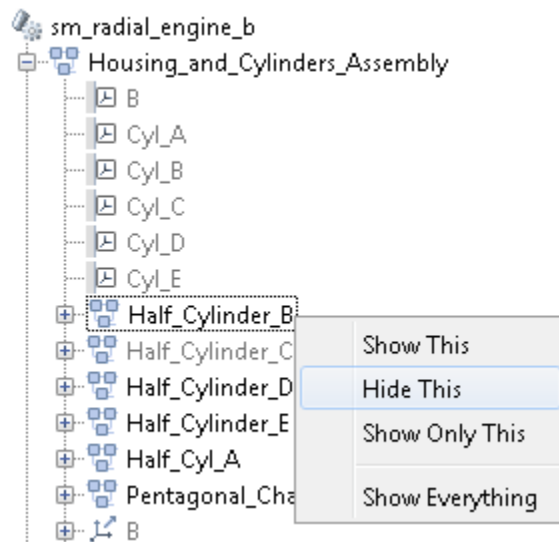
Visualization filtering is a Mechanics Explorer feature that lets you selectively show and hide parts of your model. By showing only those parts that you want to see, you can more easily discern any components placed within or behind other components—such as an engine piston traveling inside a cylinder casing.

The figure shows an example of visualization filtering. Two cylinders, one at the front and one at the rear, are hidden in the model visualization of the `sm_radial_engine` featured example. For a tutorial showing how to use visualization filtering, see “Selectively Show and Hide Model Components” on page 6-25.



## Changing Component Visibility

You can show and hide components through a context-sensitive menu accessible in the tree-view pane of Mechanics Explorer. Right-click a model-tree node to open the menu and select the desired option. The figure shows the visualization filtering menu.





## Visualization Filtering Options

The visualization filtering menu provides four options for you to select from:

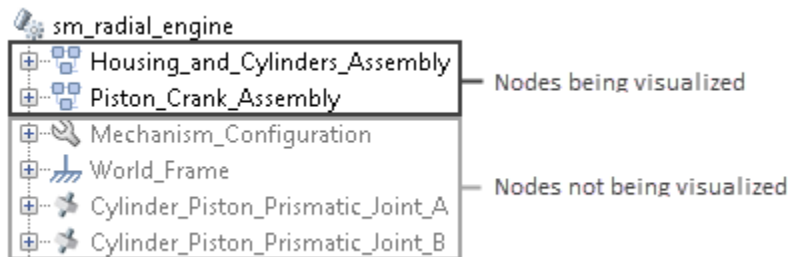
- **Show This** — Enable visualization for the selected component. This option has no effect if the component is already visible.
- **Hide This** — Disable visualization for the selected component. This option has no effect if the component is already hidden.
- **Show Only This** — Enable visualization for the selected component and disable visualization for the remainder of the model. This option has no effect if the selected component is already the only component visible.
- **Show Everything** — Enable visualization for every component in the model. This option has no effect if every component in the model is already visible.

## Components You Can Filter

You can filter the visualization of any component with solid geometry. This includes individual solids, rigid bodies, and multibody subsystems. In general, if a subsystem contains at least one Solid block, then you can switch its visualization on and off.

Frames, joints, constraints, forces, and torques have no solid geometry to visualize and therefore cannot be filtered in Mechanics Explorer. You can still open the visualization filtering context-sensitive menu by right-clicking these nodes, but only one option is active—**Show Everything**.

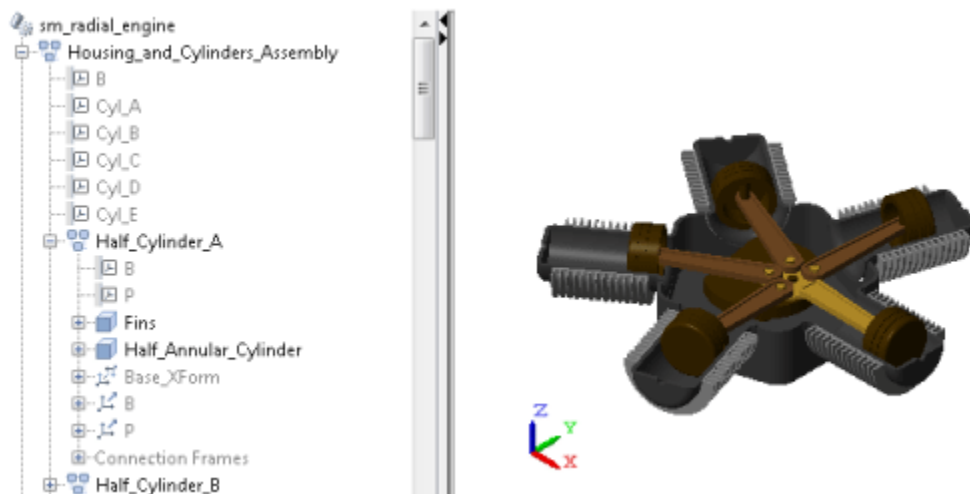
The tree-view pane identifies any node not being visualized by graying out its name. This includes nodes that can be visualized but are currently hidden and nodes that cannot be visualized at all. The figure shows an example with the grayed-out names of nodes not being visualized.



## Model Hierarchy and Tree Nodes

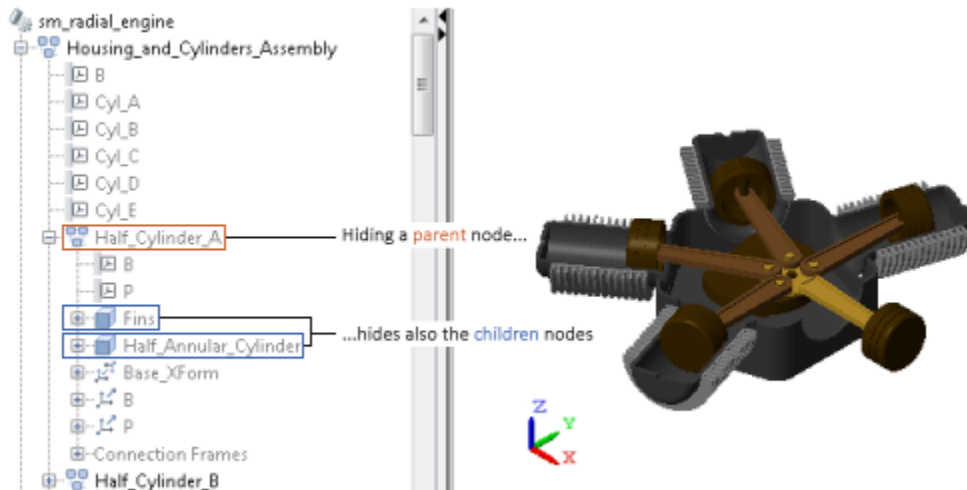
Multibody models are hierarchical in nature. They often contain multibody subsystems comprising rigid-body subsystems, each with one or more solids. The tree-view pane of Mechanics Explorer represents such a model structure through nodes arranged hierarchically. A node is a parent node if it contains other nodes, and a child node if it appears under another node. Nodes can simultaneously be children to some nodes and parents to others.

The figure shows portion of the tree-view pane of the `sm_radial_engine` featured example. The `Half_Cylinder_A` node is a child to the `Housing_and_Cylinders_Assembly` node and a parent to the `Fins` and `Half_Annular_Cylinder` nodes.



## Filtering Hierarchical Subsystems


Any changes to the visualization settings of a tree node apply equally to all children of that node, if any. Nodes higher up in the model tree are not affected. As shown in the following figure, hiding the `Half_Cylinder_A` node in the `sm_radial_engine` model causes the `Fins` and `Half_Annular_Cylinder` nodes (children nodes) to hide, but not the `Housing_and_Cylinders_Assembly` node (parent node) or the `Half_Cylinder_B` node (sibling node).



If you want to show part of a subsystem you have previously hidden, you can change the visibility settings for the children nodes that you want to show. For example, if after hiding the `Half_Cylinder_A` node, you want to show the `Fins` child node, you need only right-click the `Fins` node and select **Show This**. Such changes have no effect on the remainder of the `Half_Cylinder_A` parent node.

## Updating Models with Hidden Nodes

The following apply when you update or simulate a model with previously hidden nodes:

- If the model remains unchanged, the node visibility settings remain unchanged—that is, the hidden nodes remain hidden and the visible nodes remain visible. This happens even if you save the Mechanics Explorer configuration to the model by clicking the  icon.
- If you close Mechanics Explorer before updating the model, Mechanics Explorer reopens with all nodes visible, including any nodes you may have previously hidden.
- If you change the name of a block corresponding to a hidden node—e.g., a Solid block or a Subsystem block containing a Solid block—the hidden node and any children nodes it may have become visible.

- If you uncomment a block that corresponds to a hidden node and that you had previously commented out, the hidden node and any children nodes it may have become visible.
- If you add to a hidden Subsystem block a Solid block or another Subsystem block with a Solid block, the child node corresponding to the new block becomes visible upon model update but the visibility of the hidden parent node remains unchanged.
- If you change the parameters of a block corresponding to a hidden node, that node and its children nodes retain their original visibility settings—that is, hidden nodes remain hidden and visible nodes remain visible.

### Alternative Ways to Enhance Visibility

Visualization filtering is not the only approach you can use to enhance component visibility in a model. However, it is often the simplest. It is also the only approach that doesn't require you to modify the model in any way. Alternative approaches you can use include:

- Lowering the opacity of obstructive components—those obscuring other parts of the model—for example, making the cylinder encasing an engine piston transparent.
- Modeling obstructive components only in part—for example, treating engine cylinders as half-cylinders to preserve piston visibility during simulation.
- Omitting obstructive components altogether if they serve a purely aesthetic purpose and have no impact on model dynamics—for example, removing the cylinder subsystems from the `sm_radial_engine` featured example.
- Commenting out or through obstructive components if they serve a purely aesthetic purpose and have no impact on model dynamics—for example, removing the cylinder subsystems from the `sm_radial_engine` featured example.

## Selectively Show and Hide Model Components

### In this section...

“Visualization Filtering” on page 6-25

“Open Example Model” on page 6-26

“Update Example Model” on page 6-26

“Hide Half-Cylinder Subsystem” on page 6-27

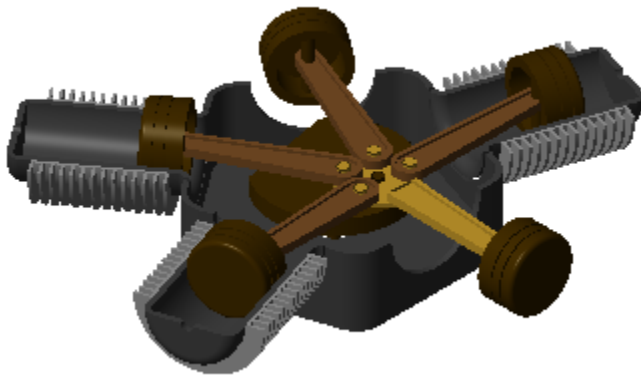
“Show Solid in Hidden Subsystem” on page 6-28

“Show Only Piston Subsystem” on page 6-29

“Show Everything” on page 6-30

### Visualization Filtering

Visualization filtering is a Mechanics Explorer feature that enables you to selectively show and hide solids, bodies, and multibody subsystems. This tutorial shows you how to use this feature to control the visualization of a Simscape Multibody model, for example, to observe a model component that might otherwise remain obstructed during simulation. For more information, see “Selective Model Visualization” on page 6-19.

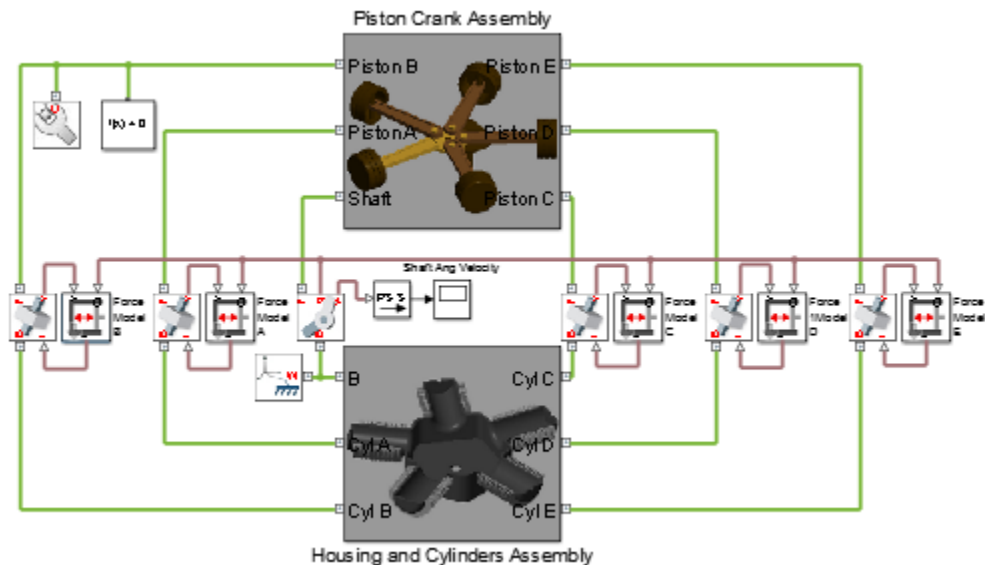


**Radial Engine Visualization with Two Cylinders Hidden**

## Open Example Model

In this tutorial, you filter the visualization of the Simscape Multibody radial engine featured example. To open this model, at the MATLAB command prompt, enter `sm_radial_engine`.

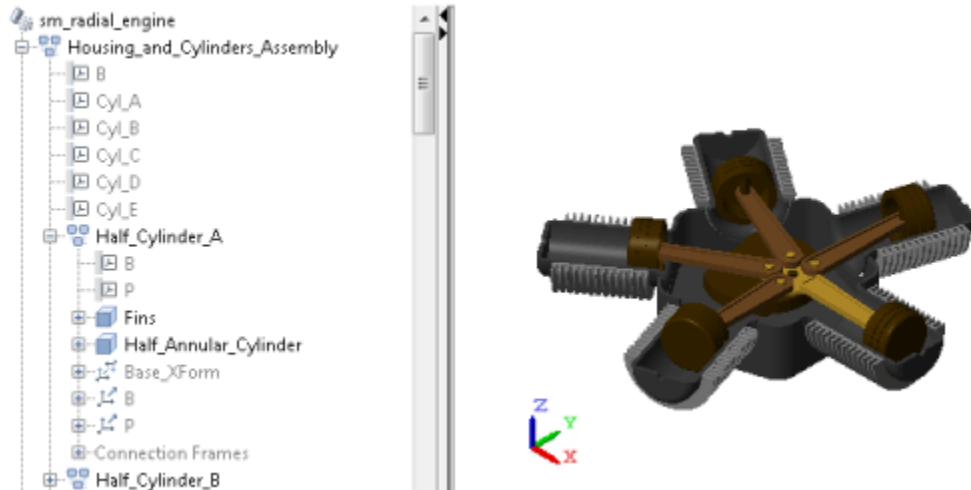
The model contains two top-level subsystems—the housing subsystem, named `Housing_and_Cylinders_Assembly`, and the piston subsystem, named `Piston_Crank_Assembly`. The housing subsystem contains five half cylinders. The piston subsystem contains five pistons that travel inside the half cylinders.



## Radial Engine Block Diagram

## Update Example Model

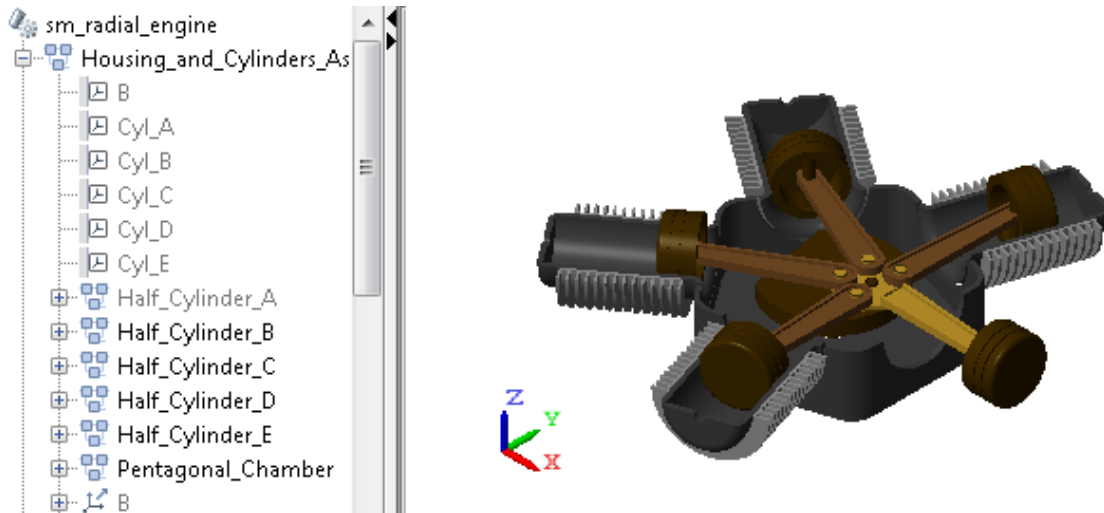
To open Mechanics Explorer, the Simscape Multibody visualization utility you must first update the example model. To do this, in the Simulink menu bar, select **Simulation > Update Diagram** (Windows shortcut **Ctrl + D**). Note the tree-view pane on the left side of Mechanics Explorer. You access the visualization filtering menu by right-clicking a node on this pane.



## Radial Engine Model Visualization

### Hide Half-Cylinder Subsystem

In the tree-view pane, expand the `Housing_and_Cylinders_Assembly` node. Right-click the `Half_Cylinder_A` node and select **Hide This**. Mechanics Explorer hides the half-cylinder subsystem and the solids it contains, corresponding to the nodes `Fins` and `Half_Annular_Cylinder`. The hidden-node names are grayed out in the tree-view pane. The figure shows the resulting model visualization.

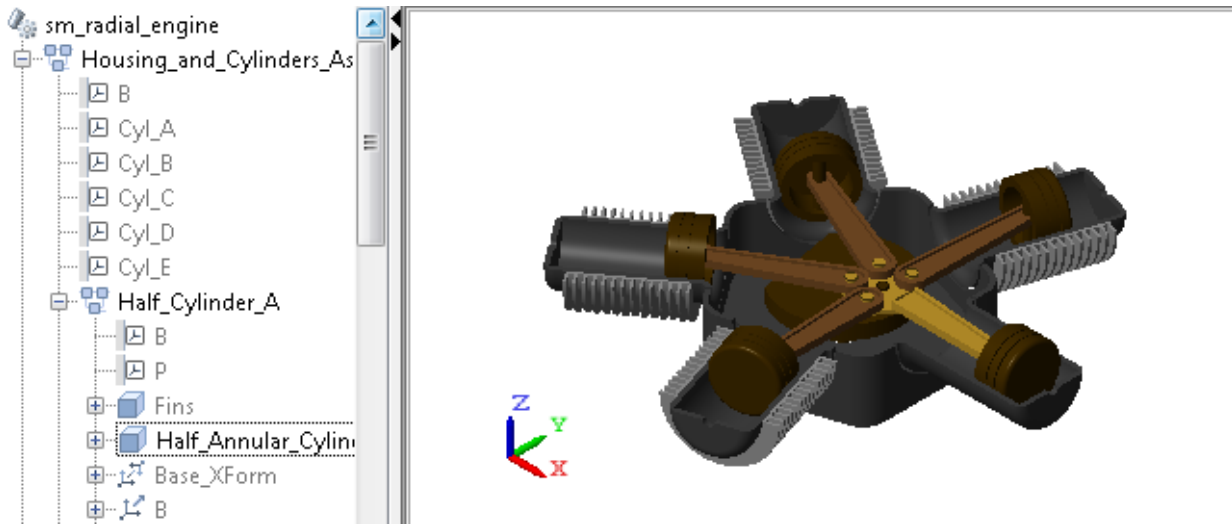


### Radial Engine with Hidden Half-Cylinder Subsystem

### Show Solid in Hidden Subsystem

In the tree-view pane, expand the `Half_Cylinder_A` node. Then, right-click the `Half_Annular_Cylinder` node and select `Show This`. The half-cylinder solid is now visible, but the remainder of its parent of its parent subsystem—in this case, just the `Fins` solid—remains hidden. The newly visible half-cylinder node name is no longer grayed out in the tree-view pane. The figure shows the resulting model visualization.

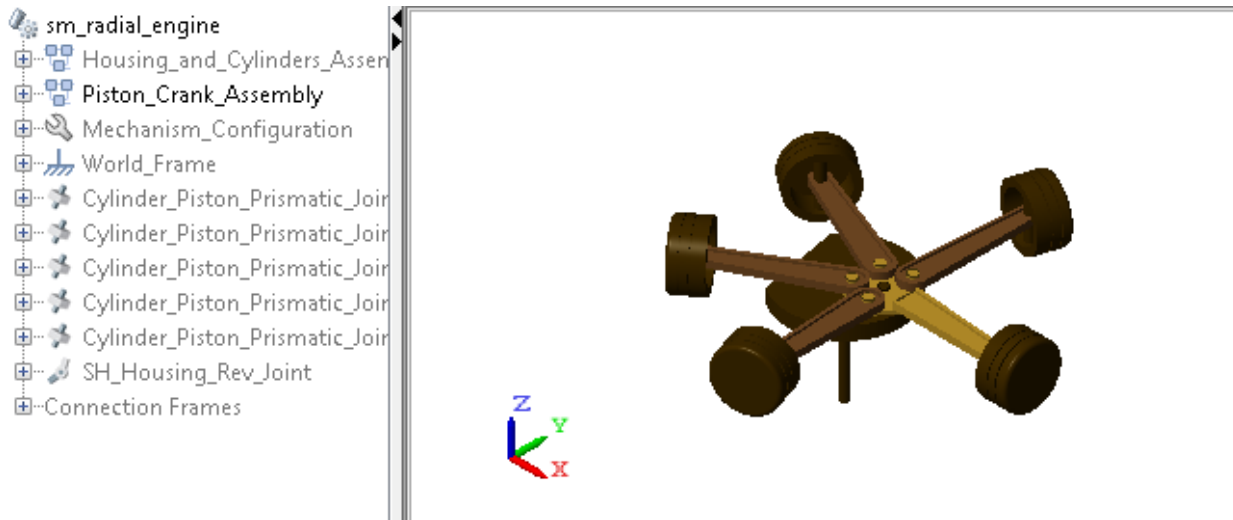




### Radial Engine with Visible Solid in Hidden Half-Cylinder Subsystem

### Show Only Piston Subsystem

In the tree-view pane, collapse the `Housing_and_Cylinders_Assembly` node. Then, right-click the `Piston_Crank_Assembly` node and select `Show Only This`. Mechanics Explorer shows the selected node and hides the remainder of the model. In the tree-view pane, the name of the selected node is the only that is not grayed out. The figure shows the resulting model visualization.

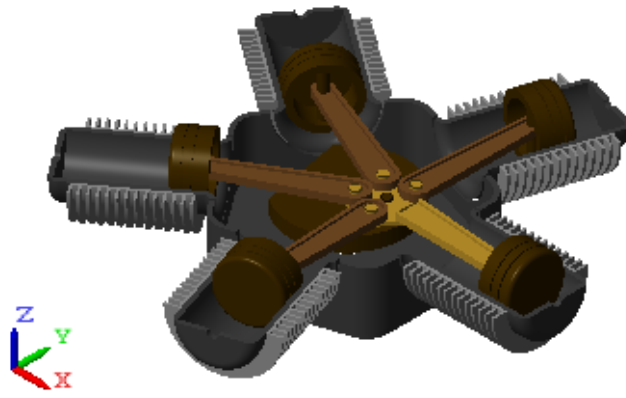


### Radial Engine with Only Piston Subsystem Visible

## Show Everything

In the tree-view pane, right-click any node and select **Show Everything**. All hidden components become visible. The corresponding nodes are no longer grayed out in the tree-view pane. The figure shows the resulting model visualization.

- sm\_radial\_engine
  - Housing\_and\_Cylinders\_Assen
  - Piston\_Crank\_Assembly
  - Mechanism\_Configuration
  - World\_Frame
  - Cylinder\_Piston\_Prismatic\_Joir
  - Cylinder\_Piston\_Prismatic\_Joir
  - Cylinder\_Piston\_Prismatic\_Joir
  - Cylinder\_Piston\_Prismatic\_Joir
  - Cylinder\_Piston\_Prismatic\_Joir
  - SH\_Housing\_Rev\_Joint
  - Connection Frames



## Visualize Simscape Multibody Frames

### In this section...

“What Are Frames?” on page 6-32

“Show All Frames” on page 6-32

“Highlight Specific Frames” on page 6-33

“Visualize Frames via Graphical Markers” on page 6-35

### What Are Frames?

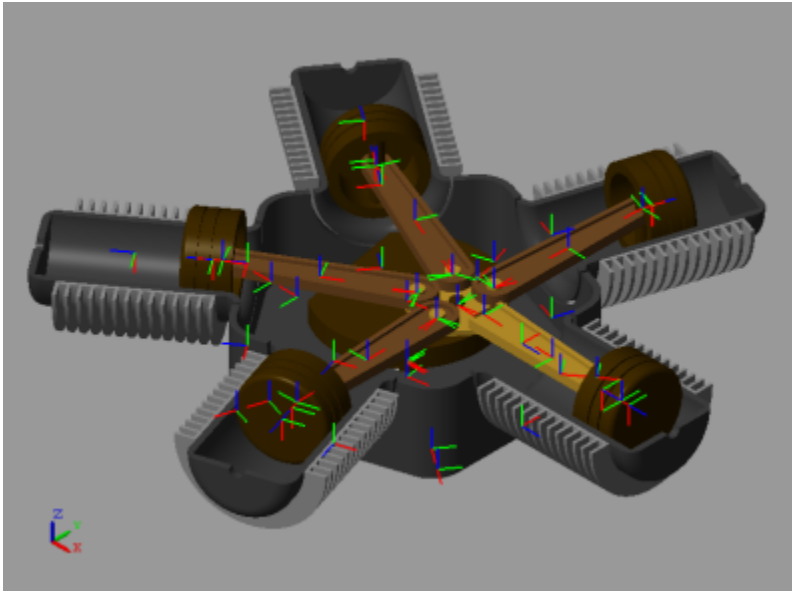
Simscape Multibody models are based on frames, abstract axis triads that contain all the position and orientation data in a model. These constructs enable you to connect solids into rigid bodies, assemble rigid bodies into mechanisms, and prescribe and sense forces, torques, and motion. Given their importance, then, it makes sense to visualize where and how you place different frames in a model.

### Show All Frames

The easiest way to view the frames in your model is to toggle their visibility on. You can do this by clicking the **Toggle Frames** icon in the Mechanics Explorer tool strip, shown in the following figure.



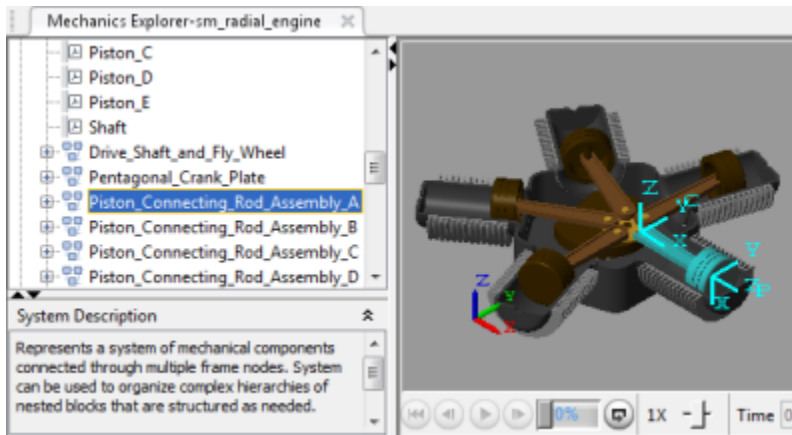
Alternatively, you can select **View > Show Frames** in the menu bar. Mechanics Explorer shows all the frames in your model, suiting this approach well for models with small numbers of frames. The figure shows a radial engine model with frame visibility toggled on.



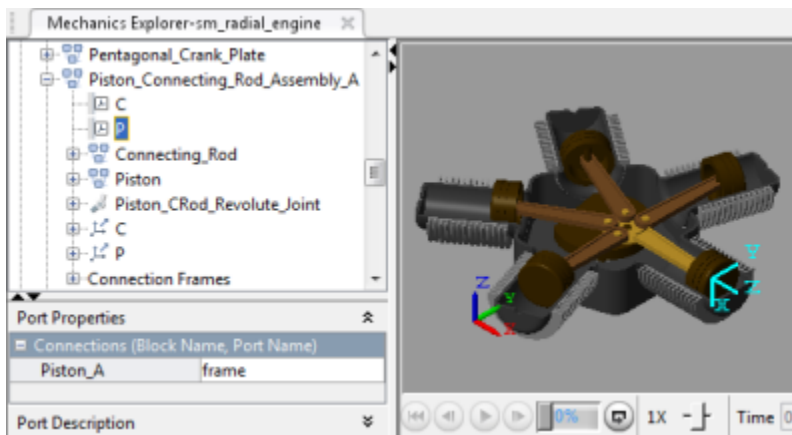
If your model has many frames, a different approach may be ideal, as toggling frame visibility may clutter the visualization pane with frames that you don't want to track.

## Highlight Specific Frames

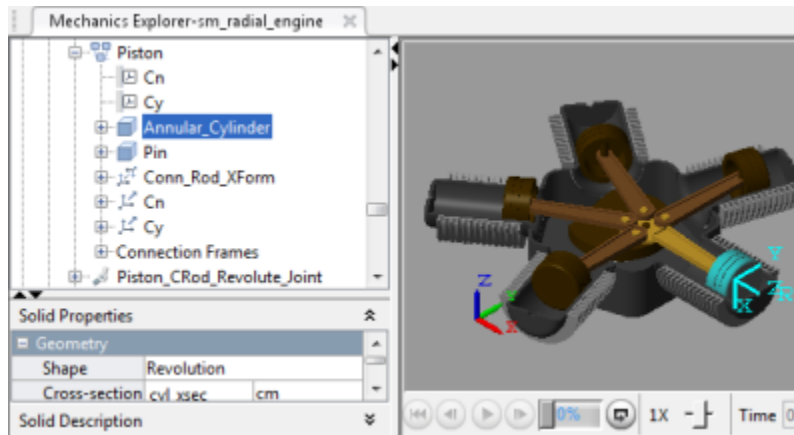
To view only the port frames of a block, including those of a subsystem block, you can select a node in the tree view pane. Mechanics Explorer highlights the port frames associated with the selected node using a turquoise color. The following figure shows an example in which one of the connecting rod assemblies in the radial engine model is highlighted in turquoise.



You can also select individual port frames, which you expose by expanding the tree nodes. For example, expanding the `Piston_Connecting_Rod_Assembly_A` node exposes the port frame `P` node, which you can then select in order to highlight that frame. The figure shows the result.

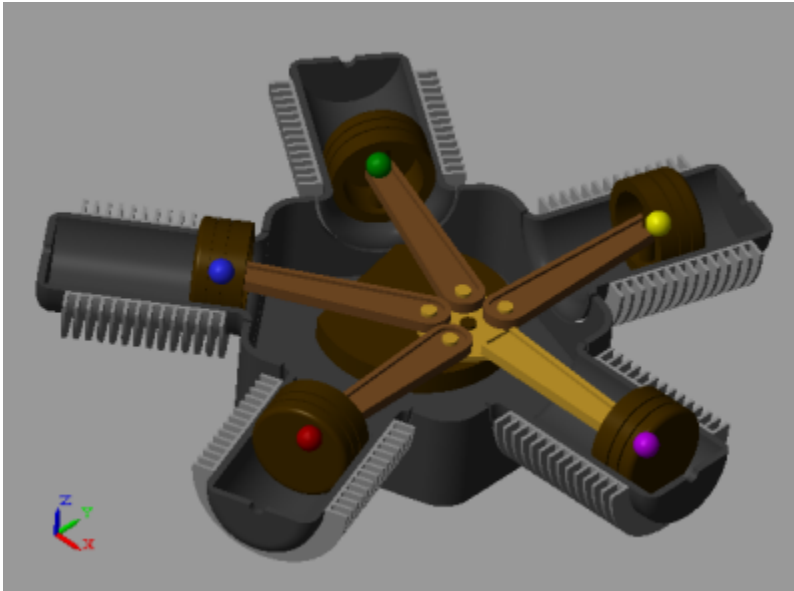


Finally, you can select individual solids directly in the visualization pane, highlighting their reference frames. The figure shows the result of selecting one of the piston solids directly. Mechanics Explorer highlights the solid and its reference frame, while the tree view pane reveals the associated `Solid` block name. This is the block that you need to change if you want to modify this particular solid.



## Visualize Frames via Graphical Markers

If a frame in your model has special significance—e.g., if its origin is the point of application for an external force—you can connect to it a graphical marker. So that you can perform this task, the Body Elements library provides a Graphic block. Simply connect the block to the frame you want to visualize and select the marker type to use—sphere, cube, or frame. The figure shows the radial engine model with a sphere marker highlighting each of the piston connection frames.





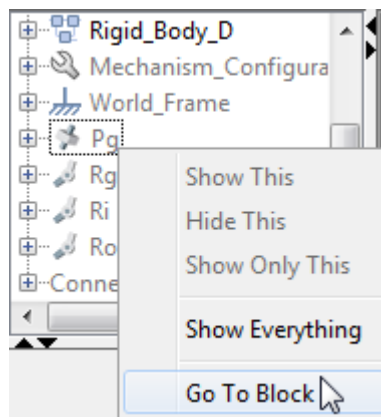
## Go to a Block from Mechanics Explorer

The first indication that something is wrong in a model is often an unexpected result in the visualization pane. Unexpected results can include disparities in solid shape and size, incorrect translation and rotation transforms between solids, and even joints and constraints that fail to assemble.

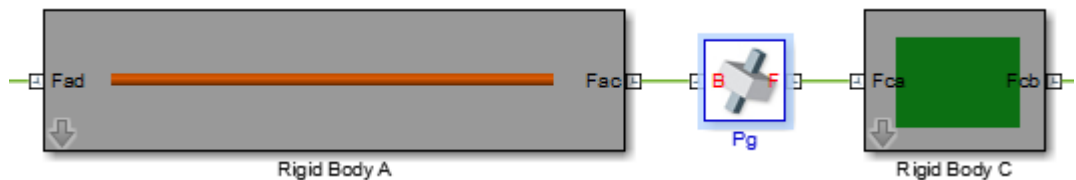
To help you troubleshoot such modeling issues, Mechanics Explorer enables you to go directly to a block associated with a node in the tree view pane. This feature helps you also to iterate on a model that is working properly, for example, if you want to replace a body subsystem with an alternative version.

To highlight a block corresponding to a Mechanics Explorer tree node:

- 1 In the tree view pane of Mechanics Explorer, right-click the node whose block you want to examine.



- 2 From the context-sensitive menu, select **Go to Block**. Simscape Multibody brings the block diagram to the front and highlights the block corresponding to the selected node.



For an example showing how to troubleshoot a model using Mechanics Explorer block highlighting, see “Troubleshoot Assembly Errors in Aiming Mechanism Model” on page 3-27.

## Create a Model Animation Video

### In this section...

- “UI and Command-Line Tools” on page 6-39
- “Before Creating a Video” on page 6-39
- “Create a Video Using Video Creator” on page 6-39
- “Create a Video Using smwritevideo” on page 6-40

### UI and Command-Line Tools

You can create a model animation video interactively, using the Video Creator tool, or programmatically, using the `smwritevideo` function. The tool and function provide equivalent ways to perform the same task. Use the tool to more intuitively configure and create a video. Use the function for your command-line workflows, e.g., to automate video capture following model simulation.

### Before Creating a Video

- Mechanics Explorer must be set to open on model update.

You can view and change the current setting in the **Simscape Multibody > Visualization** tab of the Model Configuration Parameters window.

- Only the active visualization tile in Mechanics Explorer is recorded.

A visualization tile is a subdivision of the Mechanics Explorer visualization pane that shows a specific view of the model. The active tile is demarcated by a colored bounding box.

- The video viewpoint is always that of the active tile.

To change the model viewpoint in the recorded video, you must change the viewpoint of the active tile. Use a dynamic camera for a moving point or the global camera for a static viewpoint.

### Create a Video Using Video Creator

- 1 Simulate the model to record.

Animations and the videos generated from them are based on your model simulation data.

- 2 In Mechanics Explorer, select **Tools > Video Creator**.

Video Creator relies on your model visualization and is accessible through Mechanics Explorer only.

- 3 In Video Creator, specify the desired video parameters.

Video parameters that you can modify include the video frame rate, frame size, playback speed ratio, and video file format.

- 4 Click the **Create** button.

Video Creator generates a model animation video and saves it with the specified name in the specified folder.

### Create a Video Using `smwritevideo`

- 1 Simulate the model to record.

For video-capture scripts, consider running the simulation using the `sim` command.

- 2 Programmatically define the video parameters deviating from the current Video Creator settings. E.g.,

```
fps = 60; speedRatio = 2;
```

Unspecified video parameters are set to the current values stored in the Video Creator UI.

- 3 Call the `smwritevideo` function with the video parameters as function arguments.

```
smwritevideo('robottosModel.slx','robottosVideo',...  
'PlaybackSpeedFactor',speed,'FrameRate',fps);
```

The `smwritevideo` function creates a model animation video and saves it with the specified name in the current MATLAB folder.

## Multibody Visualization Issues

### In this section...

“Mechanics Explorer Not Opening” on page 6-41

“Model Oriented Sideways” on page 6-41

“Imported CAD Parts Missing” on page 6-42

“Imported Part Colors Missing” on page 6-43

### Mechanics Explorer Not Opening

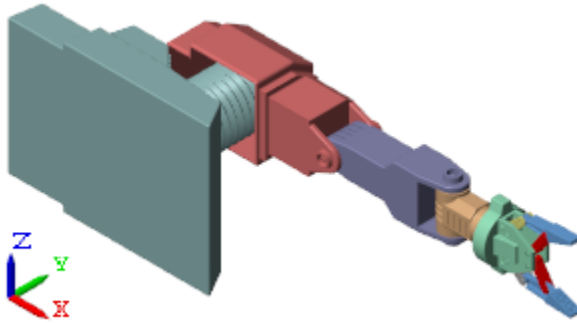
Mechanics Explorer is by default configured to open on the first model update or simulation of a Simscape Multibody session. You can, however, change your model configuration parameters to suppress model visualization and prevent Mechanics Explorer from opening. If Mechanics Explorer does not open for a model, check the visualization setting for that model:

- 1 In the Simulink menu bar, select **Simulation > Model Configuration Parameters**.
- 2 In the tree view pane of the **Model Configuration Parameters** window, select **Simscape Multibody > Explorer**.
- 3 Ensure that the **Open Mechanics Explorer on model update or simulation** checkbox is selected.

### Model Oriented Sideways

The standard view convention of Mechanics Explorer, *y*-axis up, differs from that of typical CAD applications, *z*-axis up. The different view convention may cause an imported CAD assembly model to appear sideways in Mechanics Explorer. To orient your model properly, change the **View convention** parameter to match your CAD application:

- 1 In the Mechanics Explorer toolstrip, set the **View Convention** parameter to **Y up (XY Front)**.
- 2 Click a standard-view button, e.g., for an isometric view, to refresh the model visualization and apply the new view convention.



### Imported CAD Model Visualization

## Imported CAD Parts Missing

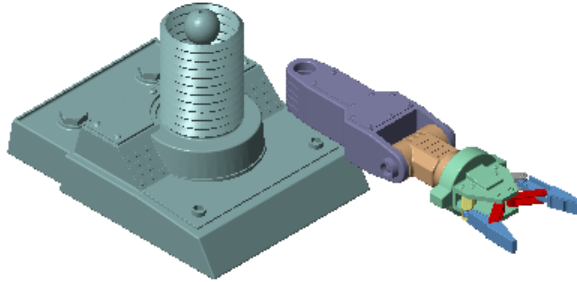
Simscape Multibody software obtains the part geometries for an imported CAD model from external part geometry files. If the files are invalid, or if the file paths specified in the Solid block dialog boxes are incorrect, Mechanics Explorer cannot render those parts. To troubleshoot this issue:

- 1 Identify the Solid blocks associated with the parts not rendered.

You can select a part by name from the tree view pane of Mechanics Explorer. If the geometry file is valid and the file path is correct, the visualization pane highlights that part.

- 2 In the **Geometry** area of the Solid block dialog boxes, ensure that the specified file paths are correct.

The file paths appear in the **File Name** parameter of the **From File** shape parameterization. If the path is correct, it is possible that the geometry file is invalid.



## Imported Part Colors Missing

Only colors specified at the part level are imported from a CAD assembly model. Colors specified for individual part features, such as surfaces and edges, or for the complete assembly as a unit, are not imported. Parts that have no color specified at the part level are by default shown in gray. You can specify the desired part color in the Solid block dialog box for that part or, prior to CAD translation, in the source CAD assembly model.





# CAD Import



# About CAD Import

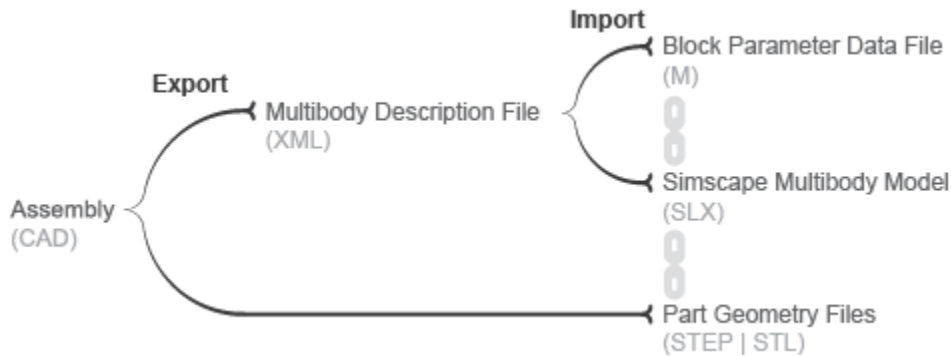
---

- “CAD Translation” on page 7-2
- “Translated Model” on page 7-3
- “Model Data File” on page 7-7
- “CAD Model Export” on page 7-8
- “CAD Export Errors” on page 7-9
- “XML File Import” on page 7-10
- “CAD Import Errors” on page 7-11
- “Data File Update” on page 7-12
- “Install Simscape Multibody Link Plug-In” on page 7-13
- “Import a CAD Assembly Model” on page 7-15
- “Import a Robotic Arm Model” on page 7-17

## CAD Translation

CAD translation is the conversion of a CAD assembly model into a Simscape Multibody model. This conversion relies on a specially formatted XML file to pass a detailed model description from your CAD application to Simscape Multibody software. The model description enables Simscape Multibody software to recreate the assembly model as a block diagram.

You translate a CAD assembly in two steps—export and import. The export step converts the CAD assembly model into an XML multibody description file and a set of STEP or STL part geometry files. The import step converts the multibody description and part geometry files into an SLX Simscape Multibody model and an M data file. The model obtains all block parameter inputs from the data file.



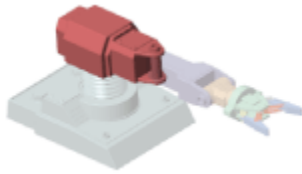
### CAD Translation

#### More About

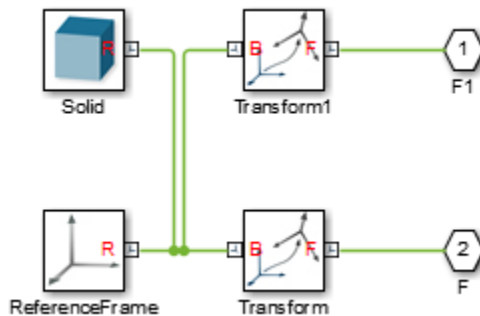
- “CAD Model Export” on page 7-8
- “XML File Import” on page 7-10
- “Data File Update” on page 7-12

## Translated Model

The translated model represents the CAD parts using Simulink subsystems based on **Solid** and **Rigid Transform** blocks. The **Solid** blocks provide the part geometries, inertias, and colors. The **Rigid Transform** blocks provides frames with the required poses for connection to other parts. Consider the upper arm of the robotic arm model shown in the figure.

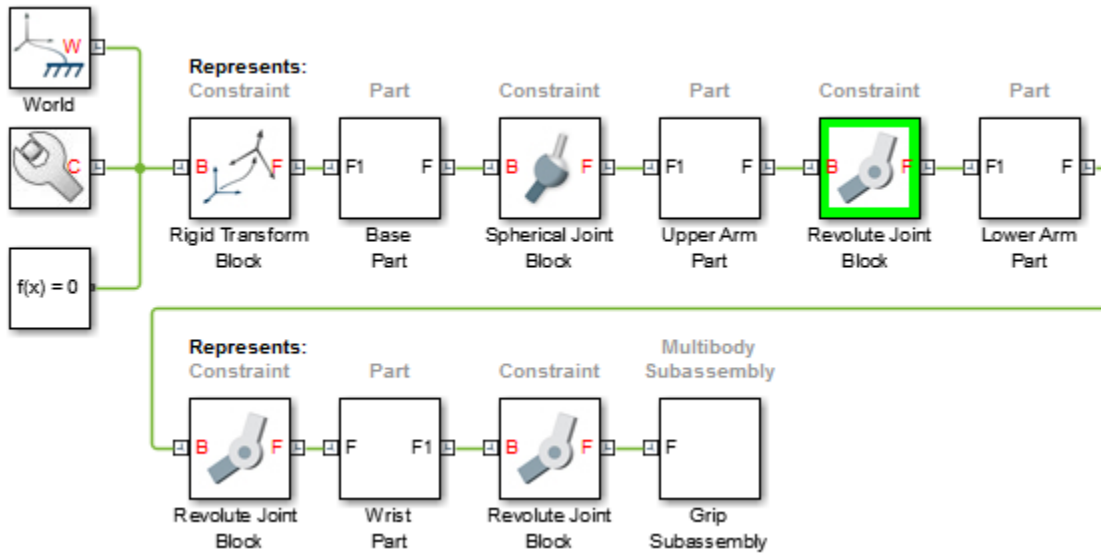


The corresponding Simulink subsystem for the upper arm part consists of one **Solid** block and two **Rigid Transform** blocks. The **Solid** block provides the solid properties of the upper arm part. The **Rigid Transform** blocks provide the frames for connection to the robotic arm base and lower arm parts.



### Simulink Subsystem Representing Upper Arm Part

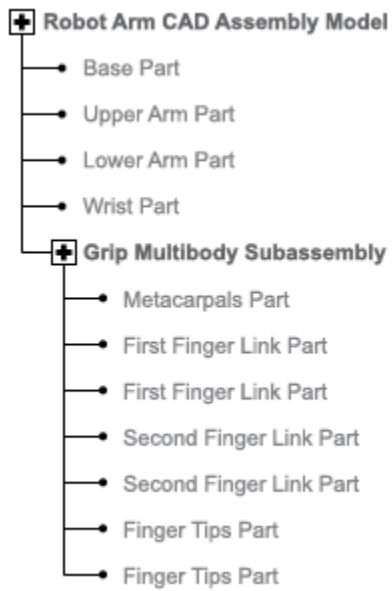
The CAD constraints between the various parts translate into joint and constraint blocks. In the robotic arm example, the constraints between the upper arm and lower arm parts translate into a **Revolute Joint** block. This block sits between the Simulink Subsystem blocks that represent the upper arm and lower arm parts.



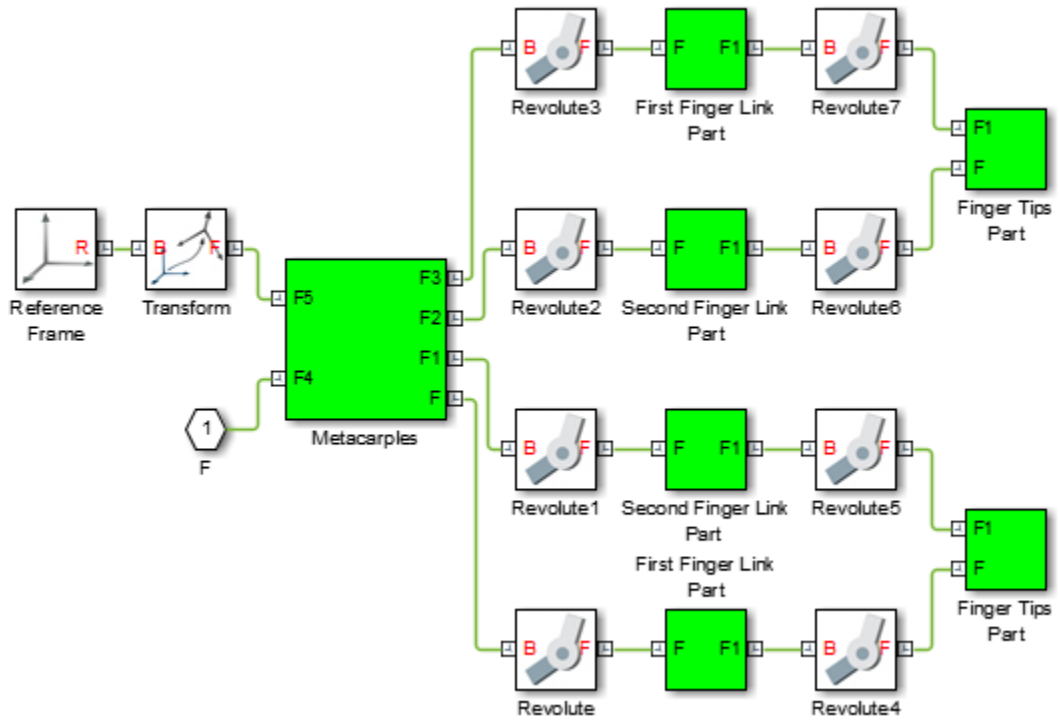
### Simulink Subsystem Representing Upper Arm Part

The translated model preserves the structural hierarchy of the CAD assembly model. If the CAD parts are nested in a multibody subassembly, then, in the translated model, the Simulink subsystems for the parts are nested in a Simulink subsystem for the multibody subassembly.

Consider the structural hierarchy of the robotic arm assembly model, shown in the figure. The model contains a grip multibody subassembly that consists of seven parts.



Such a subassembly translates into a Simulink subsystem containing seven Simulink subsystems, one for each part.



**Multibody Simulink Subsystem with Part Simulink Subsystems**

**More About**

- “CAD Translation” on page 7-2
- “Model Data File” on page 7-7
- “Data File Update” on page 7-12



## Model Data File

Blocks in the translated model are parameterized in terms of MATLAB variables defined in the data file. These variables are stored in structure arrays named after the various block types. The structure arrays are nested in a parent data structure named `smiData` or a custom string that you specify.

Consider an imported model with a data structure named `smiData`. If the model contains `Revolute Joint` blocks, the parameter data for these blocks is the structure array `smiData.RevoluteJoint`. This structure array contains a number of data fields, each corresponding to a different block parameter.

The structure array fields are named after the block parameters. For example, the position state target data for the `Revolute Joint` blocks is in a field named `Rz_Position_Target`. If the model has two `Revolute Joint` blocks, this field contains two entries—`smiData.RevoluteJoint(1).Rz_Position_Target` and `smiData.RevoluteJoint(2).Rz_Position_target`.

Each structure array index corresponds to a specific block in the imported model. The index assignments can change if you regenerate a data file from an updated XML multibody description file. The `smimport` function checks the prior data file, when specified, to ensure the index assignments remain the same. See “Data File Update” on page 7-12.

### More About

- “CAD Translation” on page 7-2
- “Translated Model” on page 7-3
- “Data File Update” on page 7-12

## CAD Model Export

You can export a CAD assembly model in two ways. The simplest is to use the Simscape Multibody Link CAD plug-in. The plug-in is compatible with three CAD applications—Autodesk Inventor<sup>®</sup>, PTC<sup>®</sup> Creo<sup>™</sup>, and SolidWorks<sup>®</sup>. The Simscape Multibody plug-in enables you to export a CAD assembly model in the following formats:

- Simscape Multibody — XML multibody description file compatible with the newer Simscape Multibody blocks. You must import the XML file using the `smimport` function.
- Simscape Multibody First Generation — XML multibody description file compatible with the older Simscape Multibody First Generation blocks. You must import the XML file using the `mech_import` function.

For information on how to export a CAD assembly model using the Simscape Multibody Link CAD plug-in, see “Export a CAD Assembly Model”.

Alternatively, if you use an unsupported CAD application, you can write a program that uses the CAD API and Simscape Multibody XML schema to generate the multibody description and part geometry files. This task requires knowledge of XML documents, XSD schema definitions, and CAD APIs.

See the schema website for the XSD schema definitions. See MATLAB Central for an example program built on the SolidWorks CAD API.

### More About

- “CAD Export Errors” on page 7-9
- “CAD Translation” on page 7-2
- “XML File Import” on page 7-10

## CAD Export Errors

If the Simscape Multibody Link plug-in cannot export a part geometry file or translate a CAD constraint set, the software issues an error message. The error message identifies the parts with missing geometry files and any unsupported constraints. You can import the generated XML multibody description file into Simscape Multibody software, but the resulting model may not accurately represent the original CAD assembly model.

### More About

- “CAD Import Errors” on page 7-11
- “CAD Model Export” on page 7-8

### XML File Import

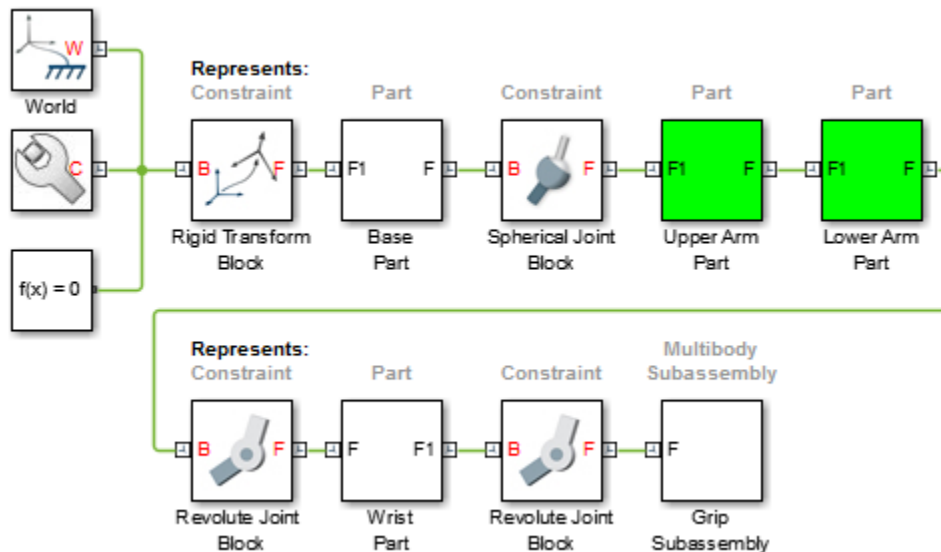
You import an XML multibody description file using the Simscape Multibody `smimport` function in its default import mode. The function parses the file and generates a Simscape Multibody model and associated data file. For step-by-step instructions on to import a CAD assembly model via its XML multibody description file, see “Import a CAD Assembly Model” on page 7-15.

#### More About

- “CAD Translation” on page 7-2
- “Data File Update” on page 7-12
- “CAD Import Errors” on page 7-11

## CAD Import Errors

If a part geometry file is invalid or missing, the corresponding part does not show in the Simscape Multibody visualization utility. If a CAD assembly model contains an unsupported constraint combination between parts, Simscape Multibody software joins the parts with a rigid connection. The rigid connection can take the form of a direct frame connection line, Rigid Transform block, or Weld Joint block.



### Rigid Connection Due to Unsupported Constraints

If Simscape Multibody software cannot translate a CAD constraint combination, it issues a warning message on the MATLAB command window identifying the affected parts and their connection frames. For example:

Warning: The set of constraints between upperarm\_1\_RIGID and forearm\_1\_RIGID could not

### More About

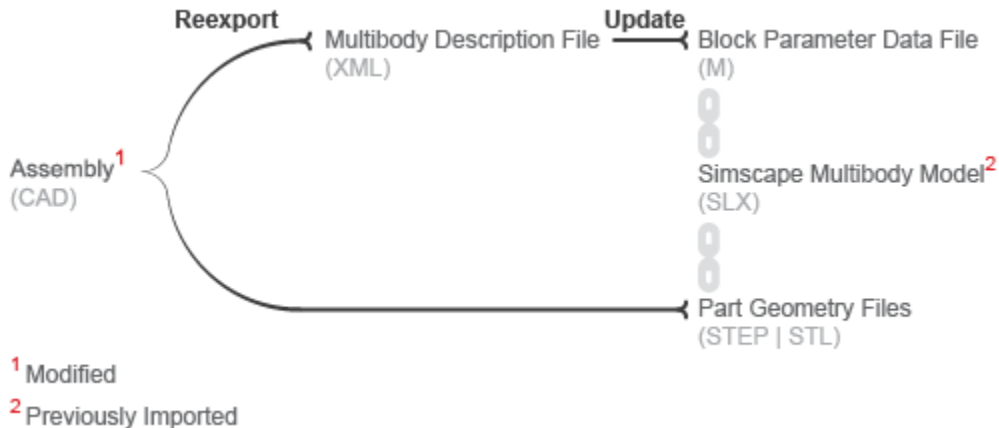
- “XML File Import” on page 7-10
- “CAD Export Errors” on page 7-9

## Data File Update

You regenerate the data file for a previously imported model by running the `smimport` function in `dataFile` mode. You specify this mode using the optional `ImportMode Name, Value` pair argument. The function uses the prior data file to keep the mapping between structure array indices and blocks consistent.

Before regenerating a data file, you must export a new XML multibody description file from the updated CAD assembly model. The `smimport` function uses the data in the new multibody description file to generate the new data file.

The function does not update the block diagram when run in `dataFile` mode. If you add or delete parts in the source CAD assembly model, you must manually add or delete the corresponding blocks in the previously imported model.



### CAD Update

### More About

- “CAD Translation” on page 7-2
- “Model Data File” on page 7-7
- “Translated Model” on page 7-3

# Install Simscape Multibody Link Plug-In

## In this section...

“Before You Begin” on page 7-13

“Step 1: Get Installation Files” on page 7-13

“Step 2: Run Installation Function” on page 7-13

“Step 3: Register MATLAB as Automation Server” on page 7-14

“Step 4: Enable Simscape Multibody Link Plug-In” on page 7-14

## Before You Begin

You must have a valid MATLAB license and one of the supported CAD applications:

- Autodesk Inventor software
- PTC Creo software
- SolidWorks software

Your MATLAB and CAD installations must have the same system architecture—e.g., Windows 64-bit.

## Step 1: Get Installation Files

- 1 Go to the Simscape Multibody Link download page.
- 2 Follow the prompts on the download page.
- 3 Save the zip archive and MATLAB file in a convenient folder.

Select the file versions matching your MATLAB release number and system architecture—e.g., release R2015b and Win64 architecture. Do not extract the zip archive.

## Step 2: Run Installation Function

- 1 Run MATLAB as administrator.
- 2 Add the saved installation files to the MATLAB path.

You can do this by entering `addpath('foldername')` at the MATLAB command prompt. Replace `foldername` with the name of the folder in which you saved the installation files—e.g., `C:\Temp`.

- 3 At the MATLAB command prompt, enter `install_addon('zipname')`.

Replace `zipname` with the name of the zip archive—e.g., `smlink.r2015b.win64.zip`.

### Step 3: Register MATLAB as Automation Server

Each time you export a CAD assembly model, the Simscape Multibody Link plug-in attempts to connect to MATLAB. For the connection to occur, you must register MATLAB as an automation server. You can do this in two ways:

- In a MATLAB session running in administrator mode — At the command prompt, enter `regmatlabserver`.
- In an MS-DOS window running in administrator mode — At the command prompt, enter `matlab -regserver`.

### Step 4: Enable Simscape Multibody Link Plug-In

Before you can export an assembly, you must enable the Simscape Multibody Link plug-in on your CAD application. To do this, see:

- “Enable Simscape Multibody Link Inventor Plug-In”
- “Enable Simscape Multibody Link Creo-Pro/E Plug-In”
- “Enable Simscape Multibody Link SolidWorks Plug-In”



# Import a CAD Assembly Model

## In this section...

“Before You Begin” on page 7-15

“Example Files” on page 7-15

“Import Model” on page 7-15

“After Import” on page 7-16

## Before You Begin

CAD import is the generation of a Simscape Multibody model from an XML multibody description file. You can generate a valid multibody description file using the downloadable Simscape Multibody Link CAD plug-in. The plug-in version must be the same as your Simscape Multibody software version.

## Example Files

Simscape Multibody software provides two sets of multibody description and part geometry files that you can use as CAD import examples. One set corresponds to a robotic arm assembly model. The other corresponds to a Stewart platform assembly model. The file sets are in different folders named

`matlabroot\toolbox\physmod\sm\smdemos\import\modelFolder`,  
where:

- *matlabroot* is the root folder of your MATLAB application, for example:  
`C:\Programs\MATLAB\`
- *modelFolder* is the name of the folder that contains the example file sets.

The robot folder provides the robotic arm example files. The `stewart_platform` folder provides the Stewart platform example files.

## Import Model

You import a model into Simscape Multibody software using the `smimport` function in its default mode. Consider the example file sets in your Simscape Multibody installation.

To recreate the CAD assembly model described by the files as Simscape Multibody block diagrams, enter:

```
smimport(multibodyDescriptionFile);
```

where *multibodyDescriptionFile* is the XML multibody description file name for the example model you want to import, specified as a string. Use `sm_robot` for the robotic arm model and `stewart_platform` for the Stewart platform model. For example, to import the robotic arm model, enter:

```
smimport('sm_robot');
```

The function generates a new Simscape Multibody block diagram and a supporting data file. The block diagram recreates the original CAD assembly model using Simscape Multibody blocks. The data file provides the numerical values of the block parameters used in the model.

### After Import

Check the imported model for unexpected rigid connections between parts. Simscape Multibody software replaces unsupported CAD constraints with rigid connections that may appear as direct frame connection lines, `Rigid Transform` blocks, or `Weld Joint` blocks.

A warning message in the MATLAB command window identifies the parts and connection frames affected by the unsupported constraints. Replace the artificial rigid connections between the parts with suitable `Joint`, `Constraint`, or `Gear` blocks from the Simscape Multibody library.

Update the block diagram to rule out model assembly errors. Run simulation to ensure the model dynamics are as expected. If you update the source CAD assembly model, you can generate an updated data file directly from a new multibody description file. For more information, see “CAD Translation” on page 7-2.

# Import a Robotic Arm Model

## In this section...

“Example Overview” on page 7-17

“Example Files” on page 7-17

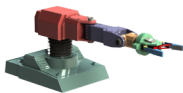
“Import the Model” on page 7-17

“Visualize the Model” on page 7-18

“Build on the Model” on page 7-20

## Example Overview

This example shows how to generate a Simscape Multibody model from a multibody description XML file using the `smimport` function. The example is based on a multibody description file named `sm_robot` and a set of part geometry files included in your Simscape Multibody installation. These files describe the robotic arm model shown in the figure.



## Example Files

The multibody description and part geometry files used in this example are located in the folder

```
matlabroot\toolbox\physmod\sm\smdemos\import\robot
```

where *matlabroot* is the root folder of your MATLAB installation, for example:

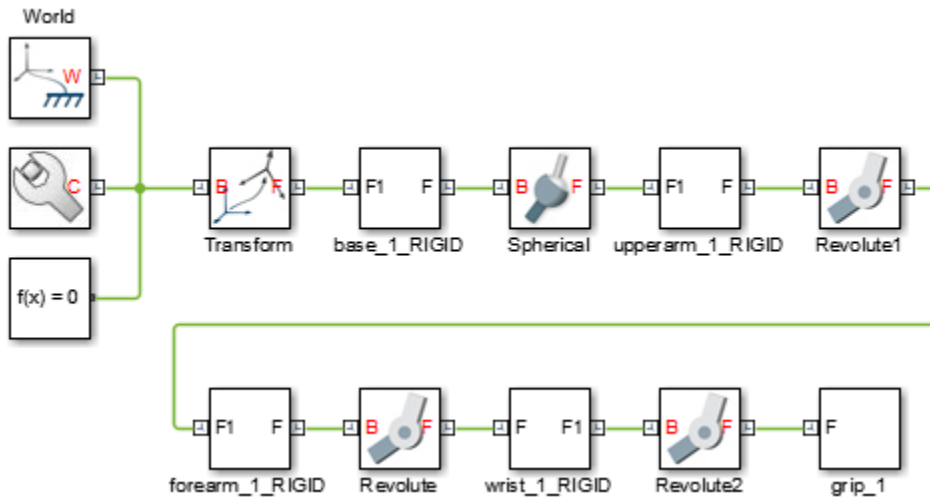
```
C:\Programs\MATLAB\
```

## Import the Model

At the MATLAB command prompt, enter the command:

```
smimport('sm_robot');
```

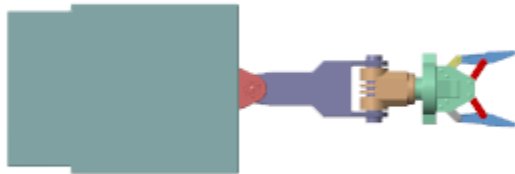
Simscape Multibody software generates the model described in the sm\_robot.xml file using the default smimport function settings.



The blocks in the generated model are parameterized in terms of MATLAB variables. The numerical values of these variables are defined in a data file that is named sm\_robot.m and stored in the same active folder as the generated model.

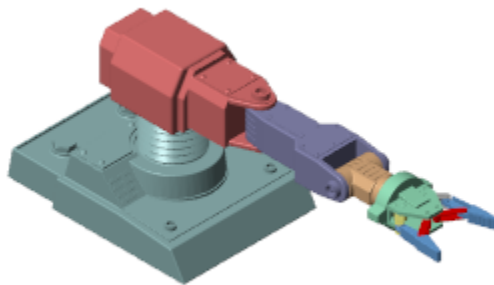
## Visualize the Model

Update the diagram to visualize the model. You can do this from the Simulink menu bar by selecting **Simulation > Update Diagram**. Mechanics Explorer opens with a static visualization of the robotic arm model in its initial configuration.



The default view convention in Mechanics Explorer differs from that in the CAD application used to create the original assembly model. Mechanics Explorer uses a Z-axis-up view convention while the CAD application uses a Y-axis-up view convention.

Change the view convention from the Mechanics Explorer toolstrip by setting the **View convention** parameter to **Y up (XY Front)**. Then, select a standard view from the **View > Standard Views** menu to apply the new view convention.



### **Build on the Model**

Try to simulate the model. Because the robotic arm lacks a control system, it simply flails under gravity. You can use Simulink blocks to create the control system needed to guide the robotic arm motion. A control system would convert motion sensing outputs into actuation inputs at the various joints. You can expose the sensing and actuation ports from the joint block dialog boxes.

### **See Also**

`smimport`

### **Related Examples**

- “Import a CAD Assembly Model” on page 7-15

### **More About**

- “CAD Translation” on page 7-2
- “XML File Import” on page 7-10
- “CAD Import Errors” on page 7-11

# Deployment





# Code Generation

---

## Code Generation Applications

### In this section...

“Code Generation Overview” on page 8-2

“Simulation Acceleration” on page 8-2

“Model Deployment” on page 8-3

### Code Generation Overview

Simscape Multibody supports code generation through Simulink Coder™. You can generate C/C++ code from a Simscape Multibody model to accelerate simulation in the Simulink environment or to deploy a model onto external hardware. Model deployment requires an active Simulink Coder license while simulation acceleration does not.



### Code Generation Applications

#### Simulation Acceleration

Simulink can generate C/C++ executable code to shorten simulation time. Two simulation modes generate code:

- Accelerator
- Rapid Accelerator

Simscape Multibody supports the two accelerator modes. You can access the simulation accelerator modes in the Simulink Editor window for your model. Click **Simulation**

> **Mode**, and select **Accelerator** or **Rapid Accelerator**. Accelerator modes do not require additional Simulink code generation products.

---

**Note:** Simulation accelerator modes do not support model visualization. When you simulate a Simscape Multibody model in **Accelerator** or **Rapid Accelerator** modes, **Mechanics Explorer** does not open with a 3-D display of your model.

---

## Model Deployment

With Simulink Coder, you can generate standalone C/C++ code for deployment outside the Simulink environment. The code replicates the source Simscape Multibody model. You can use the stand-alone code for applications that include:

- Hardware-In-Loop (HIL) testing
- Software-In-Loop (SIL) testing
- Rapid prototyping

---

**Note:** Simscape Multibody supports, but does not perform, code generation for model deployment. Code generation for model deployment requires the Simulink Coder product.

---

## Related Examples

- “Generate Code for a Multibody Model” on page 8-5
- “Configure Model for Rapid Accelerator Mode” on page 8-7

## Code Generation Setup

In this section...
“Before You Begin” on page 8-4
“Solver Selection” on page 8-4
“Target Selection” on page 8-4

### Before You Begin

Simscape Multibody software supports code generation for fast simulation in the Simulink environment or for model deployment onto external targets. If your goal is to obtain standalone C/C++ code for real-time simulation on an external target, you must have an active Simulink Coder installation.

### Solver Selection

Simscape Multibody models have continuous states and require a continuous or hybrid Simulink solver. You can change solvers from the **Solver** pane of the **Model Configuration Parameters** window. Select any solver but that marked **discrete** (no continuous states). Consider the ODE1 fixed-step solver if you need to approximate the behavior of a discrete solver.

### Target Selection

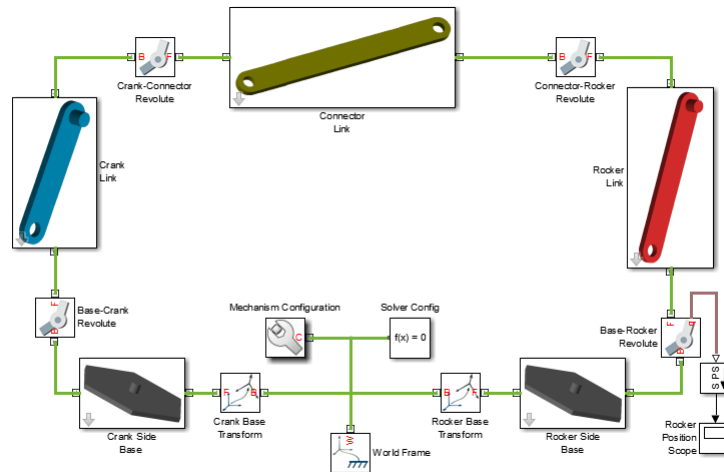
The choice of code generation target depends on the Simulink solver used. If you select a variable-step solver, you must set `rsim.tlc` as the system target file. You can specify the system target file from the **Model Configuration Parameters** window. Look for the **System target file** parameter in the **Target selection** area of the **Code Generation** pane.

## Generate Code for a Multibody Model

This example shows how to configure and generate C code for a simple Simscape Multibody model. The example is based on a four-bar model named `sm_four_bar`. The model uses a variable-step solver, `ode45` (Dorman-Prince), and therefore requires the `rsim` target to generate code.

- 1 At the MATLAB command prompt, enter `sm_four_bar`.

MATLAB software opens the four-bar example model. Save the model in a convenient folder.



- 2 In the Simulink menu, select **Simulation > Model Configuration Parameters**.

The Model Configuration Parameters window enables you to specify a code generation target and set the code generation report options for your model.

- 3 In the **Code Generation** node of the Model Configuration Parameters window, set the **System target file** parameter to `rsim.tlc`.

The `rsim.tlc` target file is compatible with Simscape Multibody models that have variable-step solvers.

- 4 In the **Code Generation > Report** node of the Model Configuration Parameters window, check the **Create code generation report** check box and click **OK**.

MATLAB software creates and opens a code generation report when you build your model.

- 5 In the Simulink menu bar, select **Code > C/C++ Code > Build Model**.

Simulink Coder software generates C code for the four-bar model. The code generation report for your model opens with a list of generated code and data files.

### Related Examples

- “Configure Model for Rapid Accelerator Mode” on page 8-7

### More About

- “Code Generation Applications” on page 8-2

## Configure Model for Rapid Accelerator Mode

### In this section...

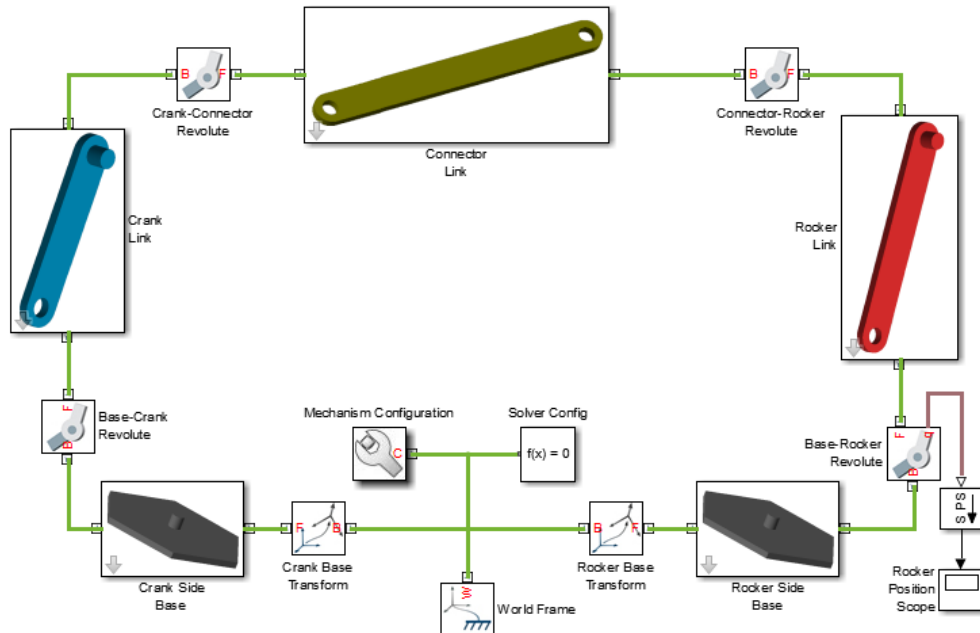
“Model Overview” on page 8-7

“Configure Model” on page 8-8

### Model Overview

You can run a Simscape Multibody model in Accelerator and Rapid Accelerator modes. When you select an accelerator mode, Simscape Multibody generates executable code that accelerates the model simulation. This example shows how to configure a four-bar model for Rapid Accelerator simulation mode. The simulation uses the default Simulink solver ode45 (Dormand-Prince).

The four-bar model is present in your Simscape Multibody installation. To open the model, at the MATLAB command line type `sm_four_bar`. A new Simulink Editor window opens with the block diagram of the four-bar model.



## Configure Model

To configure the model for Rapid Acceleration simulation mode, follow these steps:

- 1 In the Simulink Editor window for your model, select **Simulation**.
- 2 In the drop-down menu, select **Mode > Rapid Accelerator**.
- 3 Select **Simulation > Model Configuration Parameters**.
- 4 In **Code Generation**, under **System target file**, enter `rsim.tlc`.

---

**Note:** You must use the `rsim.tlc` target each time you generate code with a variable-step solver. Both Accelerator and Rapid Accelerator modes generate executable code that requires the `rsim.tlc` target to be used with variable-step solvers.

---

- 5 Expand the **Simscape Multibody** node.
- 6 Select **Explorer**.
- 7 Clear the **Open Mechanics Explorer on model update or simulation** check box.

---

**Note:** Clearing the **Open Mechanics Explorer on model update or simulation** check box disables visualization with Mechanics Explorer. Disabling visualization prevents Simscape Multibody from issuing a warning message when you simulate a model in Accelerator or Rapid Accelerator mode.

---

- 8 Press **Ctrl+T** to simulate the model.

---

**Note:** The Rapid Accelerator mode incurs an initial time cost to generate the executable code. Once the code is generated, the simulation proceeds more rapidly. Rapid Accelerator mode is suggested for large or complex Simscape Multibody models with long simulation times.

---

The Rapid Accelerator mode does not support visualization. Mechanics Explorer does not open, and you cannot view a dynamic simulation of the model. All other simulation capabilities remain functional, including graphics and scopes.

## More About

- “Code Generation Applications” on page 8-2