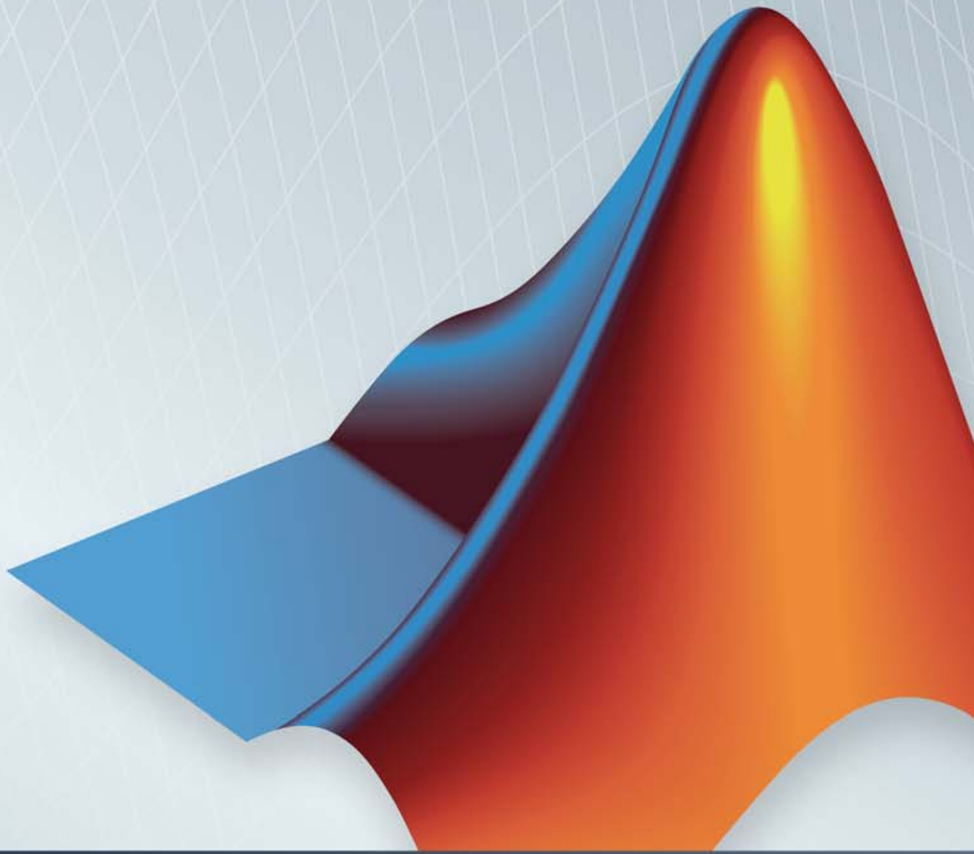
SimMechanics™

User's Guide

R2013**b**



MATLAB®&SIMULINK®

MathWorks®

How to Contact MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support
`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

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

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

SimMechanics™ User's Guide

© COPYRIGHT 2002–2013 by The MathWorks, Inc.

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

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

Trademarks

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

Patents

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

Revision History

March 2012	Online only	New for Version 4.0 (Release R2012a)
September 2012	Online only	Revised for Version 4.1 (Release R2012b)
March 2013	Online only	Revised for Version 4.2 (Release R2013a)
September 2013	Online only	Revised for Version 4.3 (Release R2013b)

Contents

Multibody Modeling

1 Spatial Relationships

2 Rigid Bodies

3 Multibody Systems

4 Internal Mechanics, Actuation and Sensing

Simulation and Analysis

5 Simulation

6 Visualization and Animation

CAD Import

7 About CAD Import

8 Code Generation

Multibody Modeling

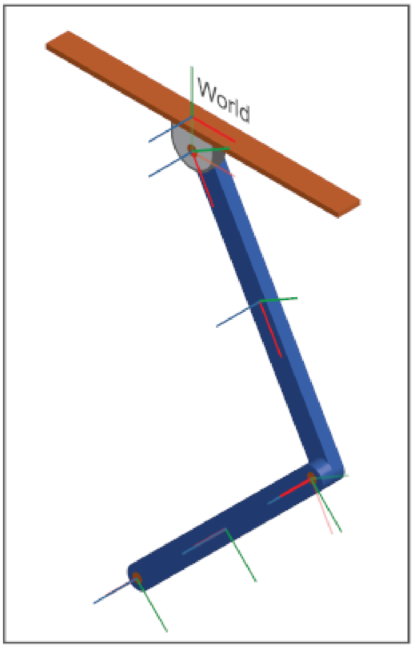
1

Spatial Relationships

Working with Frames

In this section...

Frames form the foundation of multibody modeling. These constructs encode the relative position and orientation of one rigid body with respect to another. In SimMechanics™, every rigid body contains at least one frame.

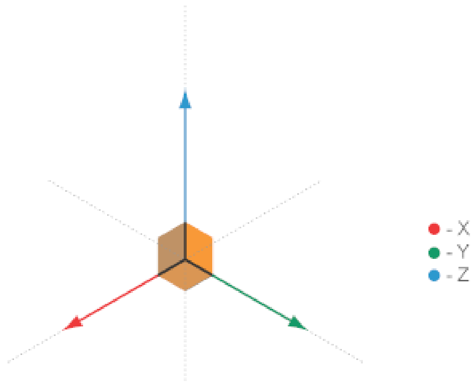


Consider a double pendulum with two links. Each link has a set of physical properties that affect its dynamic behavior and appearance—geometry, inertia, and color. Yet, none of these properties contain information about the spatial arrangement of the links. To position and orient one link with respect to another, you need frames.

You relate two rigid bodies in space by connecting two frames together. In the double pendulum, you connect the end frame of one link to the end frame of another link using a joint. In turn, each link contains a local reference frame against which you define the two end frames. You can make two frames coincident, translate them, or rotate them with respect to each other.

Frames

Frames have one origin and three axes. The origin defines the local zero coordinate of the frame. This is the point with respect to which you measure translation — the distance between two frames. The axes define the directions in which the components of a 3–D vector are resolved. For example, if you measure the translation vector between two frame origins, you can resolve the vector components along the axes of the base frame. For more information, see “Measurement Frames” on page 4-40.



Frame Types

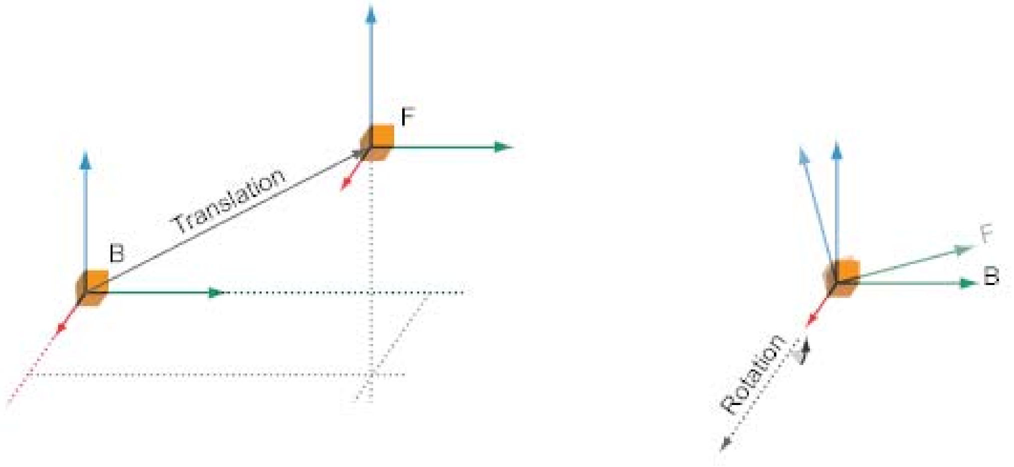
A multibody model generally contains two frame types: global and local. The global frame represents the world. It is inertial and defines absolute rest in a model. In SimMechanics, you represent the global frame with the World Frame block. This block is available in the Frames and Transforms library.

The World frame is uniquely defined in every model. You can add multiple World Frame blocks to a model, but they all represent the same frame.

A local frame represents a position and orientation in a rigid body. It can move with respect to the World frame, but not with respect to the rigid body itself. Because it can move with respect to the World frame, a local frame is generally non-inertial. To add a local frame to a rigid body, you use the Rigid Transform block. You can add multiple local frames to a rigid body—to define the position and orientation of joints, to apply an external force or torque, or to sense motion. For more information, see "Frame Transformations" on page 1-19.

Frame Transforms

To separate two frames in space, you apply a frame transformation between them. In SimMechanics, two frame transformations are possible: rotation and translation. Rotation changes the relative orientation of two frames. Translation changes their relative position.



Rigid transformations fix spatial relationships for all time. When you rigidly connect two frames, they move as a single unit. They cannot move with respect to each other. In SimMechanics, you apply a rigid transformation with the Rigid Transform block.

Note Frame transformations are important in multibody models. The Rigid Transform block is among the most commonly used in SimMechanics.

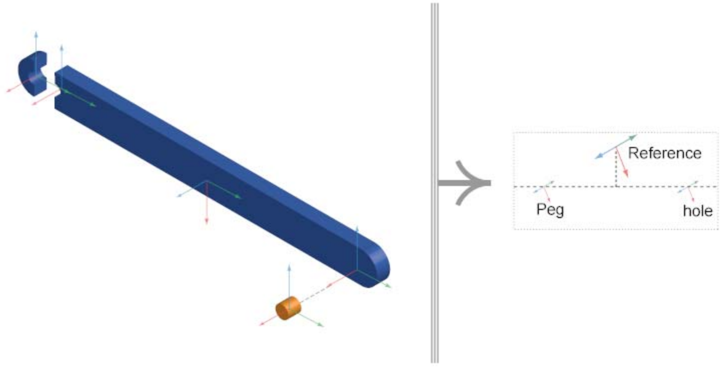
You can also relate to frames with a time-varying transformation. In this case, the rotation, translation, or both, can vary as a function of time. One example is the connection between two links in a double-pendulum. Two frames, one on each link, connect with a joint that allows their spatial relationship to vary with time.

To add a time-varying transformation, you use joint blocks. These blocks *allow* frame transformations to vary with time. However, unlike the Rigid Transform block, you cannot directly specify the time-dependence of the frame transformation. This dependence follows directly from the dynamics of the model.

Frame Networks

A single rigid body may have multiple frames. For example, a simple binary link — a link with two joints — generally has one reference frame near the geometry center and two frames at the joint locations. More complex rigid bodies may have yet more frames. In fact, SimMechanics imposes no limit on the number of frames a rigid body can have. You can add as many frames as your application requires.

The set of frames that belong to a rigid body form a *frame network*. Like other networks, it is often convenient to organize frames hierarchically. You can, for example, organize the frames of a binary link such that its two joint frames are defined with respect to the geometry center frame. In this simple example, the frame network contains two hierarchical levels: a top level containing the geometry center frame, and a lower level containing two joint frames. More complex rigid bodies generally have more hierarchical levels.



The top hierarchical level contains the parent frame. Lower hierarchical levels contain children frames. Children frames can in turn contain their own children frames. All frames in a frame network depend, directly or indirectly, on the parent frame. This is because the frame transformations used to define the children frames ultimately reference the parent frame.

Concepts

- “Frame Transformations” on page 1-19
- “Representing Frames” on page 1-7
- “Sensing Spatial Relationships” on page 4-21

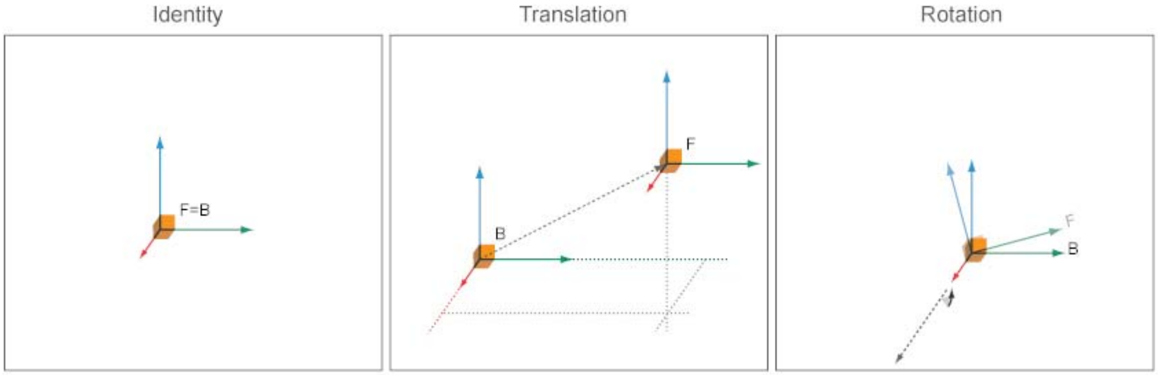
Representing Frames

In this section...

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.

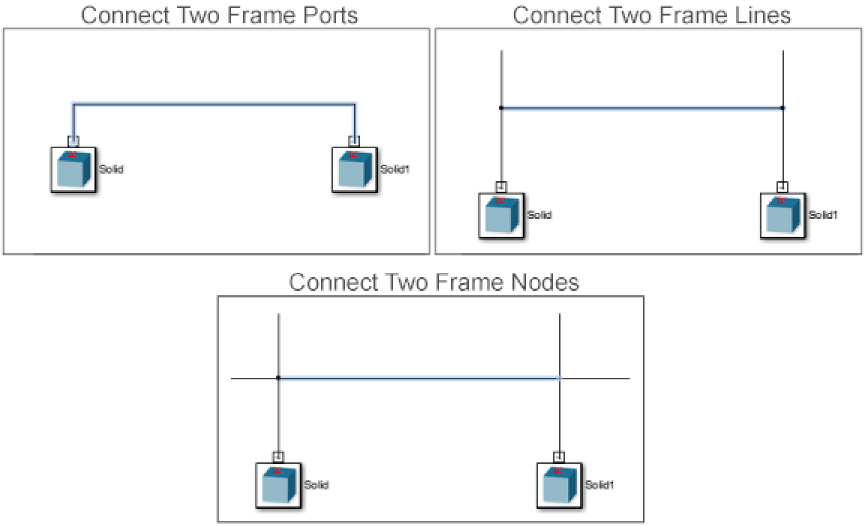
The figure illustrates these spatial relationships. Letters B and F represent the two frames between which you apply a spatial relationship.



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.

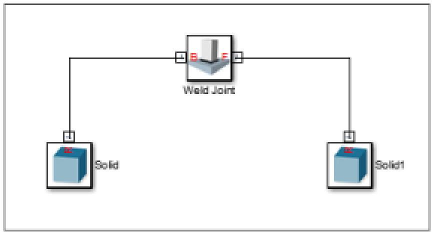
Identity Relationships

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. By connecting two frame entities to the base and follower frame ports of this block, you make them coincident for all time. Use the Weld Joint block to rigidly connect two frames that belong to different rigid bodies. In the figure, a Weld Joint block makes two solid reference frames coincident in space.

Connect Frames with Weld Joint Block



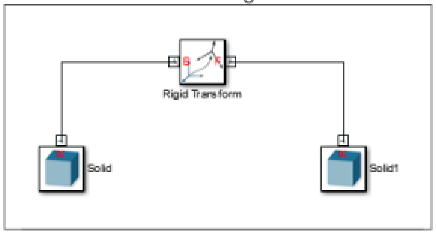
Note If you apply an identity relationship with the Weld Joint block, check that a Solid or Inertia block rigidly connects to the joint frames. Failure to do so results in a degenerate mass error during simulation.

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.

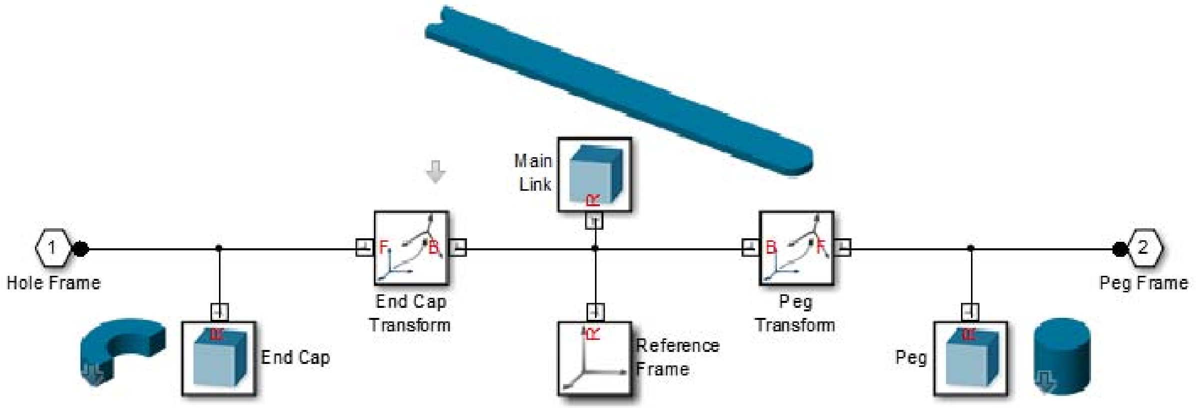
You can apply an offset distance between two frame origins, a rotation angle between the frame axes, or both. Two frames that you connect using a Rigid Transform block behave as a single entity. If you specify neither translation or rotation, the Rigid Transform block represents the identity relationship. 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

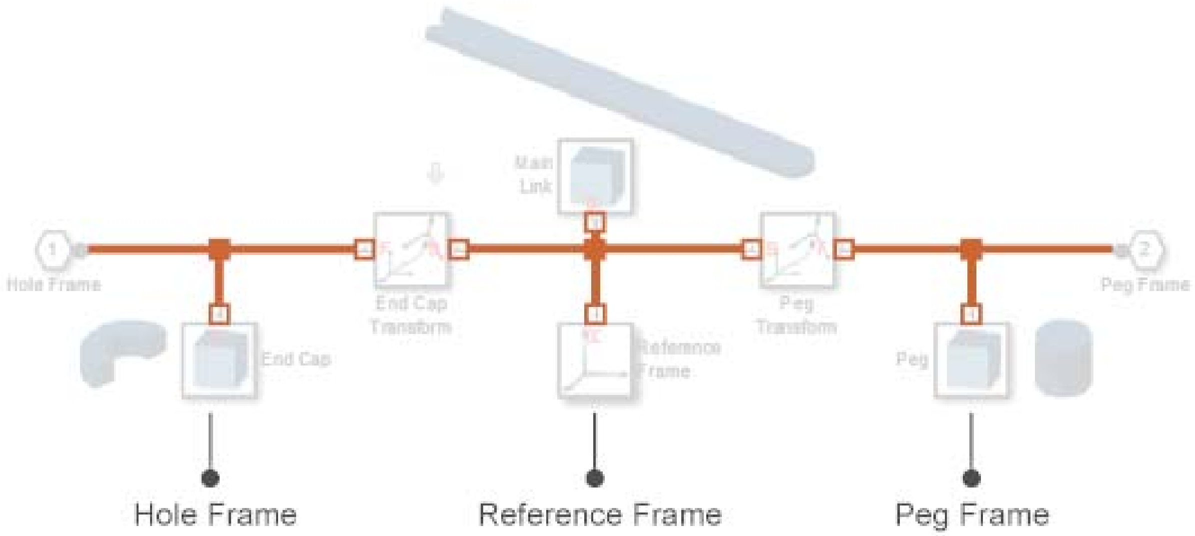


Interpreting a Frame Network

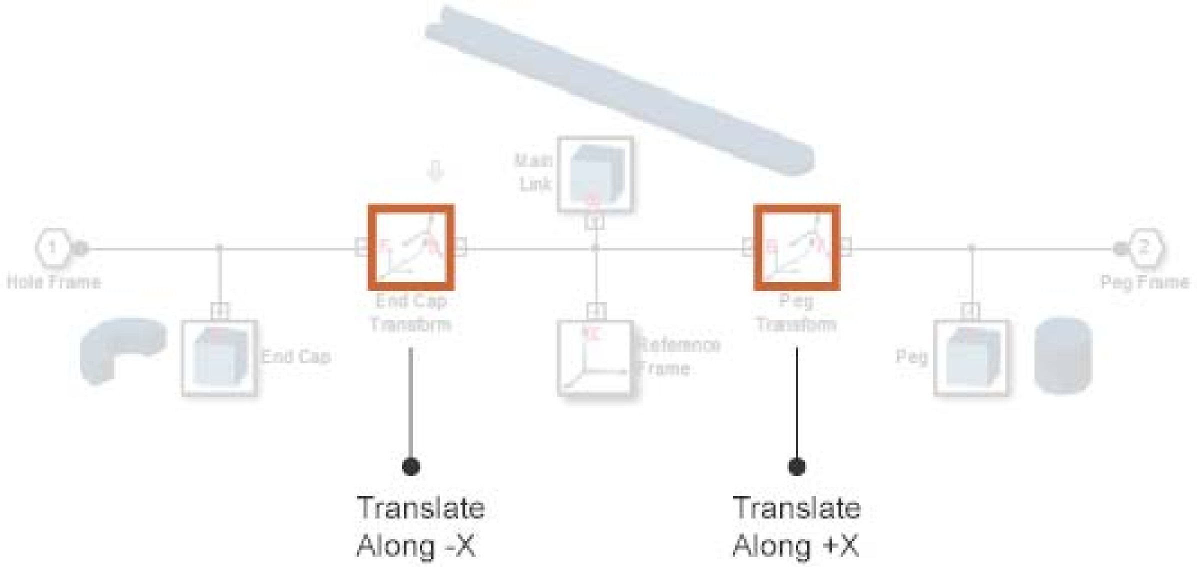
As an example, consider the frame network of a binary link. SimMechanics provides a model of this rigid body. To open it, at the MATLAB® command line enter `sm_compound_body`. Double-click subsystem Compound Body to view the underlying block diagram. The figure shows this block diagram.



To represent the binary link, the Compound Body subsystem contains three solids. These 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 1-8 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.



Related Examples

- “Represent Box Frame Tree” on page 1-44
- “Represent Binary Link Frame Tree” on page 1-36

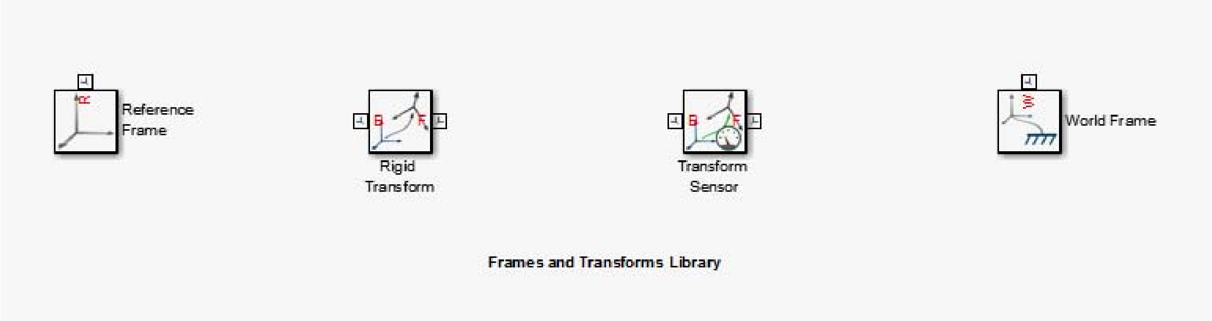
Concepts

- “Working with Frames” on page 1-2
- “Frame Transformations” on page 1-19
- “World and Reference Frames” on page 1-13
- “Find and Fix Frame Issues” on page 1-67

World and Reference Frames

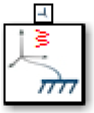
In this section...

Two preset frames are available in SimMechanics: 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.

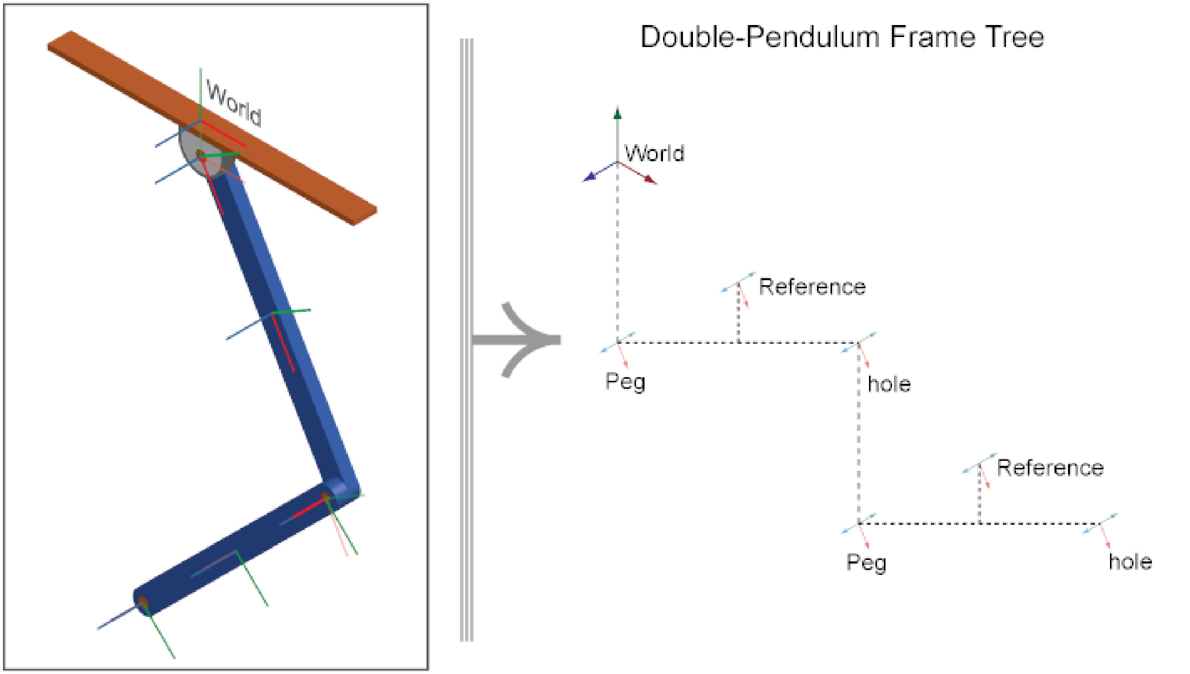


World Frame

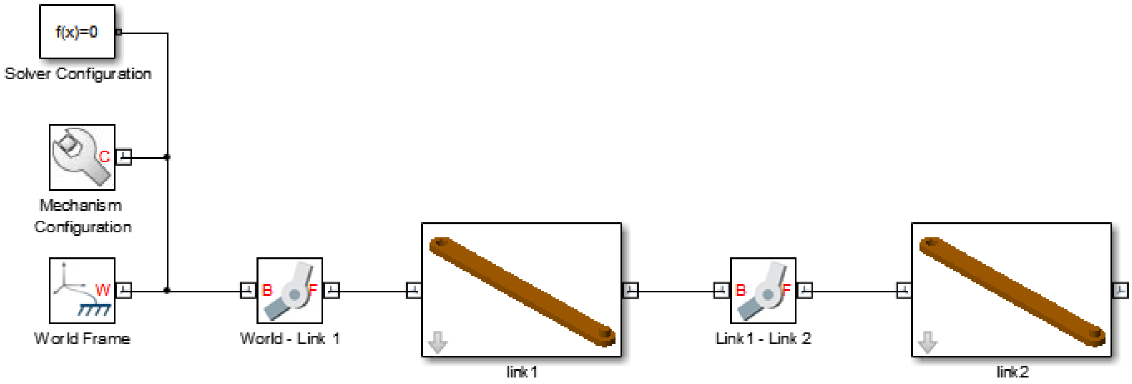
The World frame represents the external environment of a mechanical system. It is always at absolute rest, and therefore experiences zero acceleration. As a consequence, centripetal and other pseudo-forces are not present in the world frame, and it is said to be inertial. Rigidly connecting any frame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

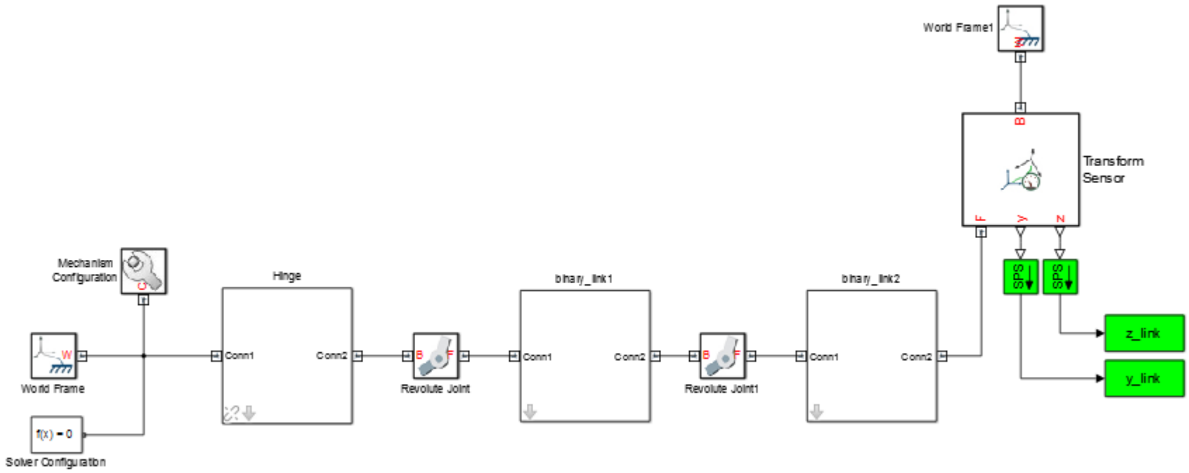


The double-pendulum block diagram is based on this frame tree. The World Frame block identifies the root of the frame tree. A Revolute Joint block applies the variable transform that relates the World frame to the binary link peg frame. A second Revolute Joint block applies a similar variable transform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



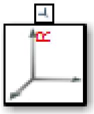
The World frame is present in every model. However, the World Frame block is strictly optional. If you do not add this block to a model, SimMechanics assigns one of the existing frames as the World frame. This implicit World frame connects to the rest of the model via an implicit 6-DOF joint, which in 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.

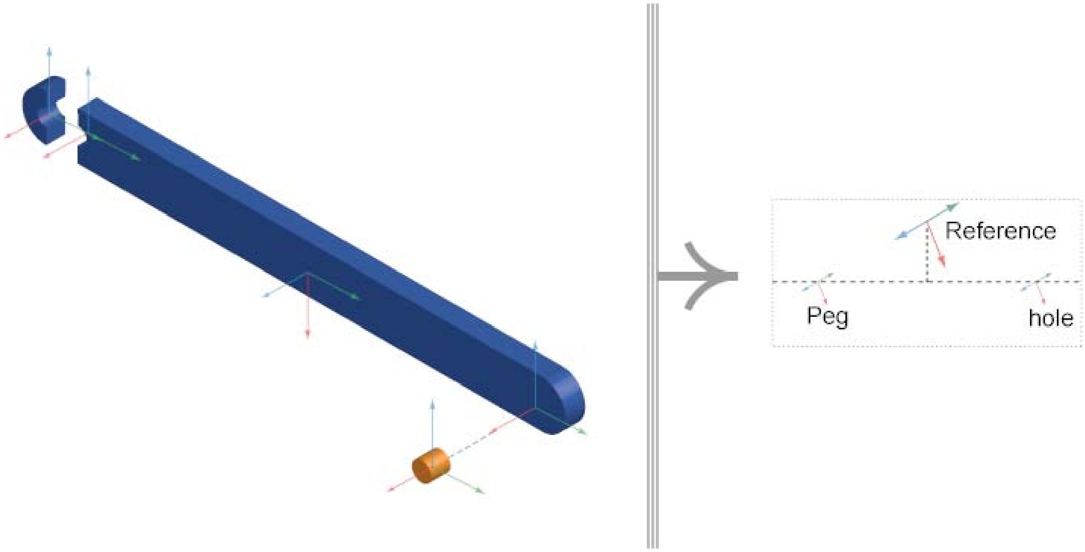


Reference Frame

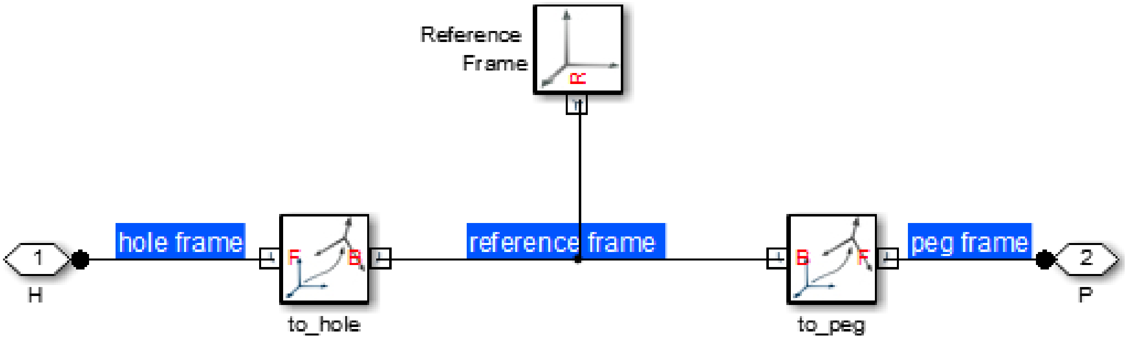
The Reference frame represents the root of a rigid body or multibody subsystem. Within a subsystem, it denotes the frame against which all remaining frames are defined. To add a Reference frame, use the Reference Frame block. Use this block to mark the top level of a subsystem frame tree.



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, SimMechanics assigns one of the existing frames as the Reference frame.

Concepts

- “Working with Frames” on page 1-2
- “Frame Transformations” on page 1-19
- “Representing Frames” on page 1-7

Frame Transformations

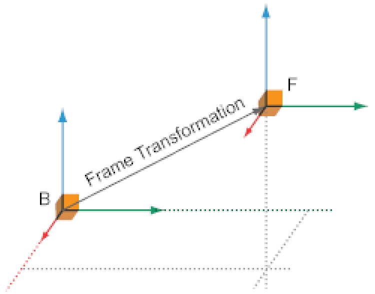
In this section...

To place a solid in space, with a given position and orientation, you use frames. By connecting the solid reference frame to another frame, you resolve its position and orientation within the model. For example, connecting the solid reference frame directly to the World frame causes their origins and axes to coincide. However, if the model does not yet contain the desired frame, you must first *add* it.

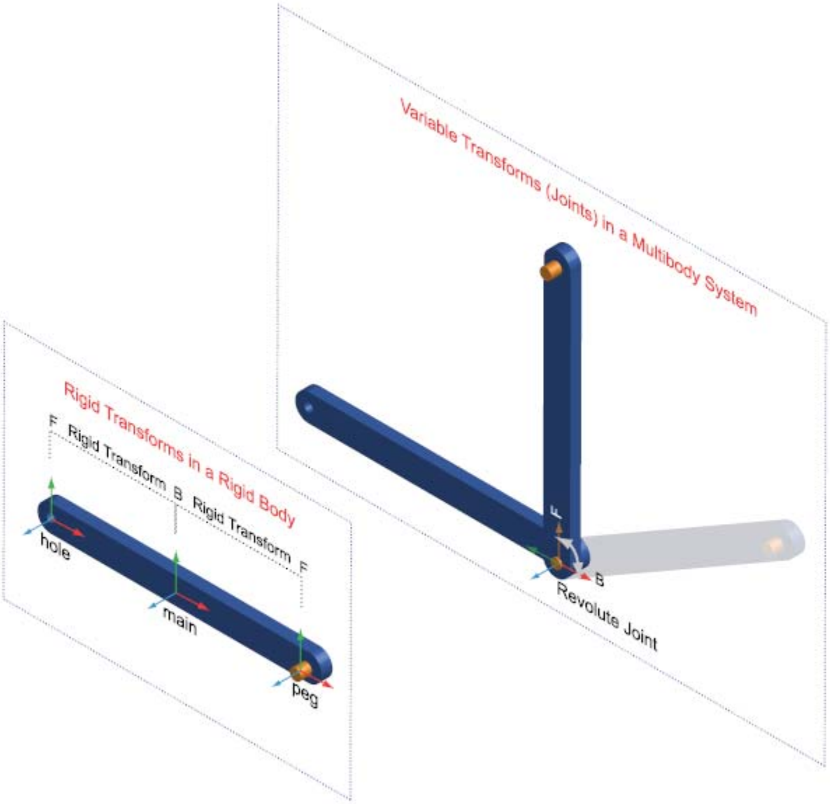
Adding a frame is the act of defining its position and orientation. Because these properties are relative, you must always define a frame with respect to another frame. Every model starts with one of two frame blocks you can use as reference: World Frame or Reference Frame. As a model grows, so does the number of frames that you can use as a reference.

Rigid and Time-Varying Transformations

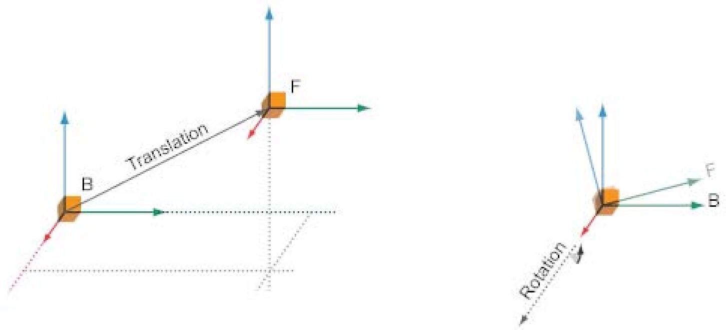
The spatial relationship between the two frames, the existing and the new, is called a frame transformation. When the transformation is fixed for all time, it is *rigid*. Two frames related by a rigid transformation can move with respect to the world, but never with respect to each other. In SimMechanics, you add a new frame by applying a rigid transformation to an existing frame. The block you use for this task is the Rigid Transform block.



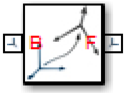
Frame transformations can also vary with time. In this case, the two frames that the transformation applies to can move with respect to each other. In SimMechanics, joint blocks provide the degrees of freedom that allow motion between two frames. Depending on the joint block, frames can move along or about an axis. For example, the Revolute Joint block allows two frames to rotate with respect to each other about a common +Z axis. Likewise, the Prismatic Joint block allows two frames to rotate with respect to each other along a common +Z axis. For more information about joints, see "Modeling Joints" on page 3-8.


Variable Transforms (Joints) in a Multibody System
F
B
Revolute Joint
Rigid Transforms in a Rigid Body
F
Rigid Transform
B
Rigid Transform
F
hole
main
peg

You can apply two rigid transformations: rotation and translation. Rotation changes the orientation of the follower frame with respect to the base frame. Translation changes the position of the follower frame with respect to the base frame. A third, implicit, transformation is available: identity. This transformation is marked by the absence of both frame rotation and translation, making base and follower frames coincident in space.

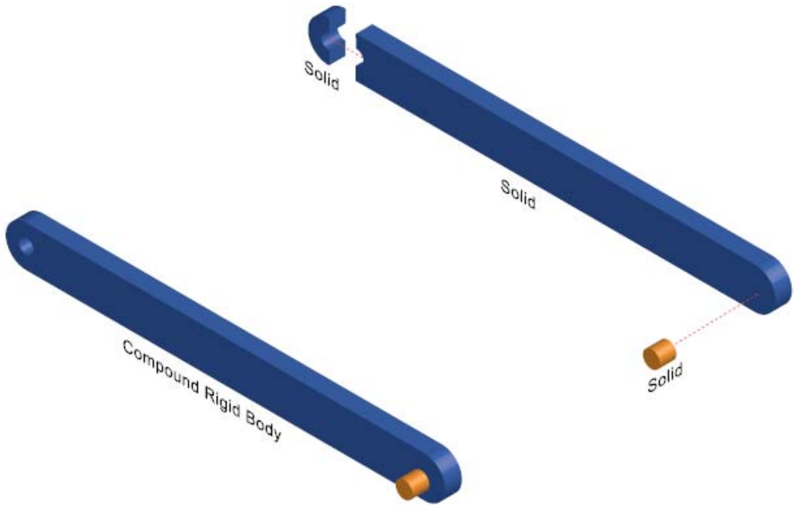


Every rigid transformation involves two frames: a base and a follower. The base frame is a reference, the starting point against which you define the new frame. Any frame can act as the base frame. When you apply a rigid transformation, you do so directly *to* the base frame. The follower frame is the new frame — the transformed version of the base frame. The Rigid Transform block identifies base and follower frames with frame ports B and F, respectively.



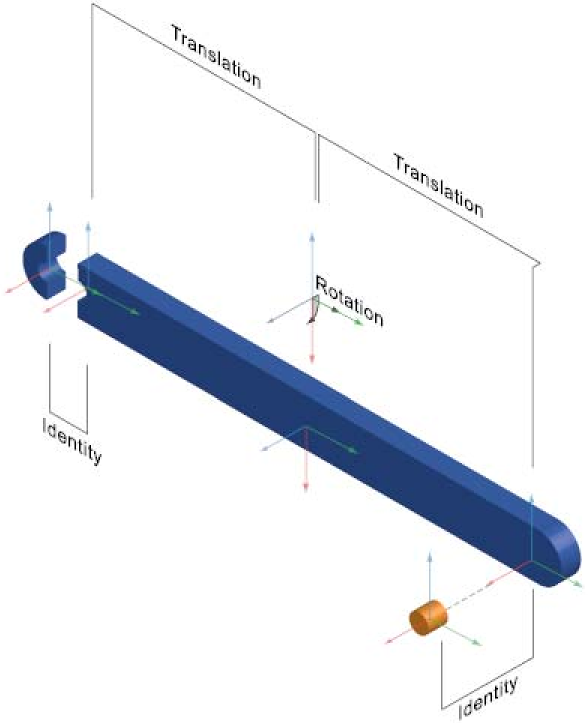
Rigid Transformation Example

As an example, consider a binary link. You can model this rigid body with three elementary solids: main body, peg, and hole sections. This type of rigid body is known as *compound*. Each solid has a local reference frame, which is fixed with respect to the solid, but which can move with respect to the world. The figure shows the binary link compound rigid body and the three solids that comprise it.

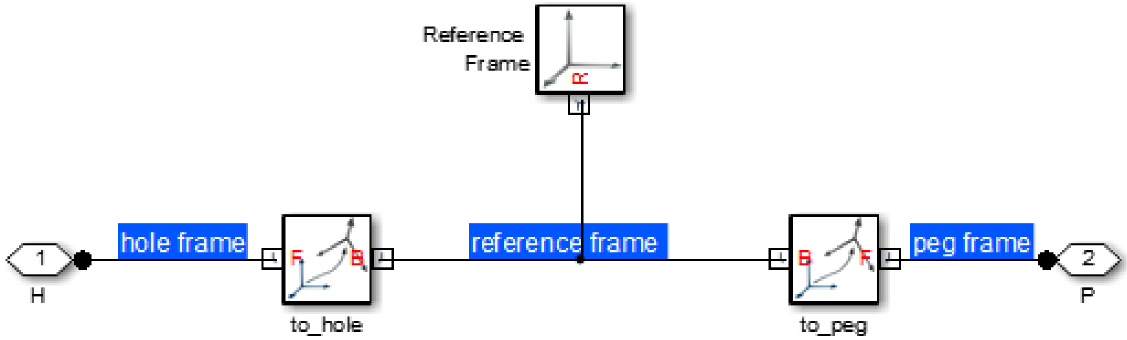


When modeling the binary link, the goal is to place the peg at one end of the link, and the hole section at the other end. The proper approach is to apply a rigid transformation between the main peg and peg reference frames, and main body and hole section reference frames. The transformations specify the separation distance and rotation angle, if any, between each pair of frames. Because the transformations are rigid, they constrain the solids to move as a single unit — a *rigid body*. The rigidly connected solids can move together with respect to the World frame, but never with respect to each other.

The figure shows the set of transformations used to model the binary link. These include translation, rotation, and identity. No Rigid Transform block is required to apply an identity transformation. See “Representing Frames” on page 1-7.



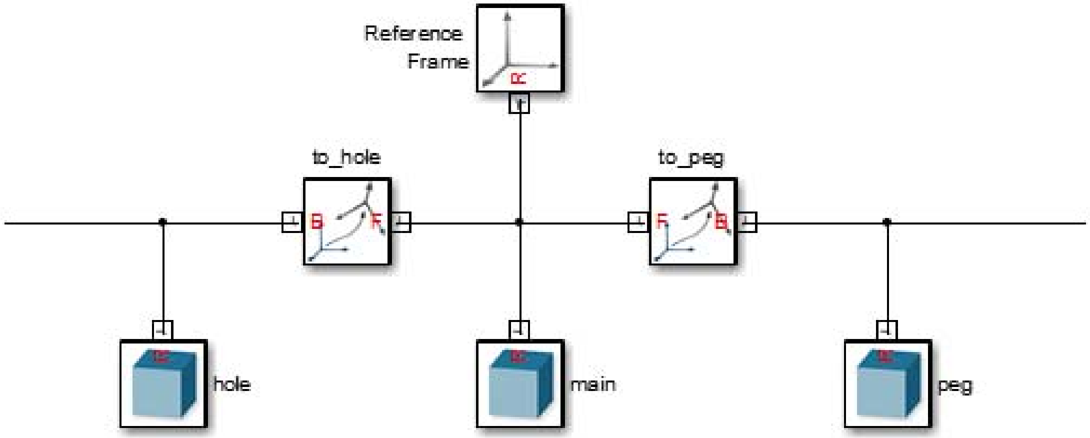
The block diagram, shown in the following figure, reflects the structure of the binary link. Three Solid blocks represent the main body, peg, and hole sections. Their R ports identify the respective reference frames. Two Rigid Transform blocks, named to_hole and to_peg apply the rigid transformations that relate the solid pairs main–hole and main–peg.



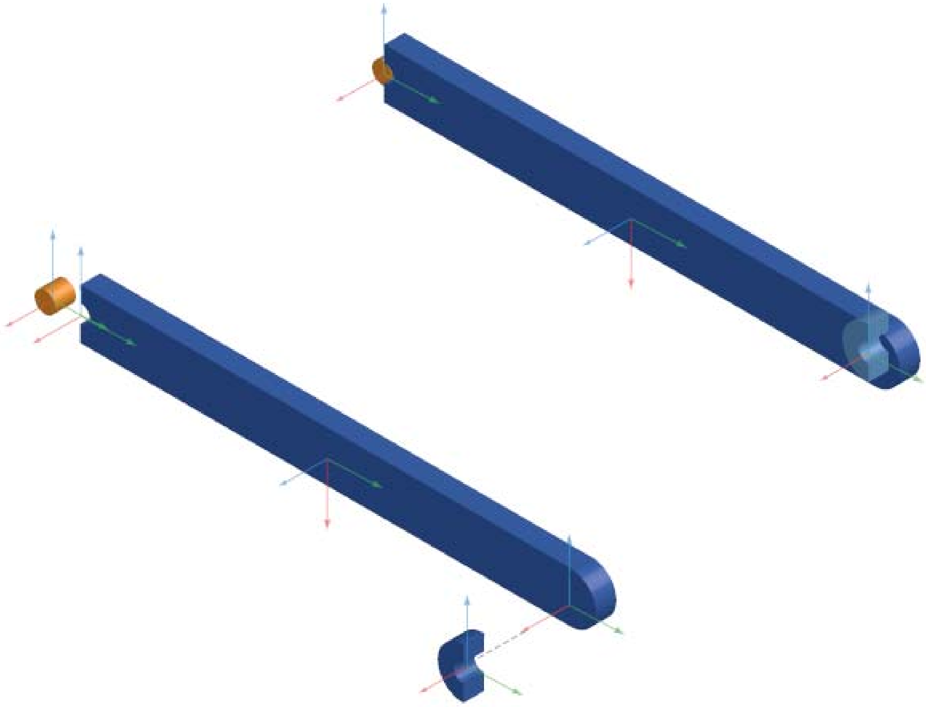
Reversing Rigid Transformations

Rigid transformations describe the operation that takes the base frame into coincidence with the follower frame. In this sense, the transformation *acts on* the base frame. Switching base and follower port frames causes the transformation to act on a different frame, changing the relationship between the two frames. The result is a follower frame with different position and orientation and, as a consequence, a different rigid body subsystem.

Consider the binary link system. In the original configuration, rigid transformations translate the peg to the right of the main body and the hole to the left. To accomplish this, the main body frame connects to the base port frame of the corresponding Rigid Transform blocks, while the hole and peg frames connect to the follower port frames. When you switch base and follower frame ports, the transformations instead translate the main body to the right of the peg and to the left of the hole.



While in the first case the peg translated to the right of the main body, in the second case the peg translated to the left. The same principle applies to the hole. The figure shows the effect of switching base and follower frames in both Rigid Transform blocks of the binary link block diagram.



Concepts

- "Rotation Methods" on page 1-28
- "Translation Methods" on page 1-32
- "Frame Transformations" on page 1-19
- "Represent Binary Link Frame Tree" on page 1-36

Rotation Methods

In this section...

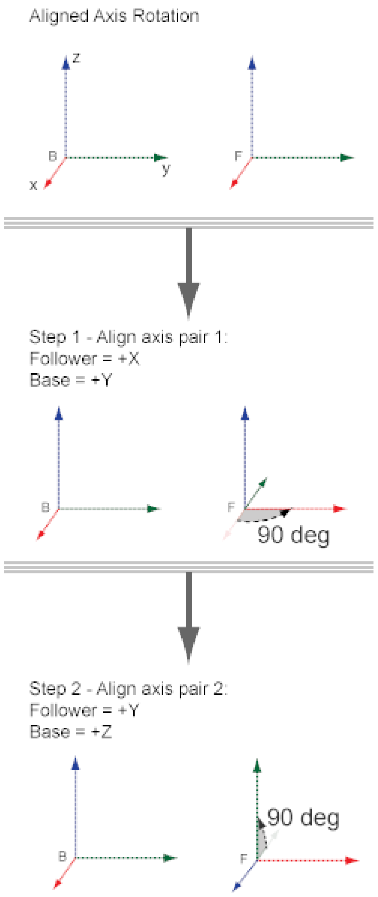
You can specify frame rotation using different methods. These include aligned axes, standard axis, and arbitrary axis. The different methods are available through the Rigid Transform block. The choice of method depends on the model. Select the method that is most convenient for the application.

Specifying Rotation

Rotation is a relative quantity. The rotation of one frame is meaningful only with respect to another frame. As such, the Rigid Transform block requires two frames to specify a transformation: base and follower. The transformation operates on the base frame. For example, a translation along the +Z axis places the follower frame along the +Z axis from the base frame. Reversing frame ports is allowed, but the transformation is reversed: the base frame is now placed along the +Z axis from the follower frame.

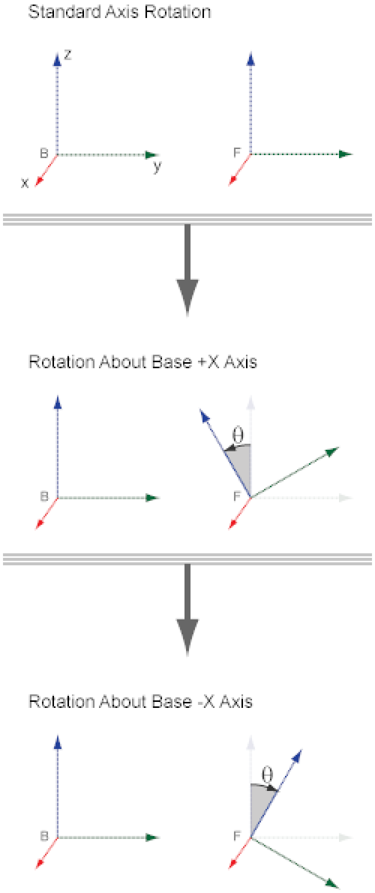
Aligned Axes

Rotate two frames with respect to each other by aligning any two axes of one with any two axes of the other. The figure illustrates the aligned axes method.



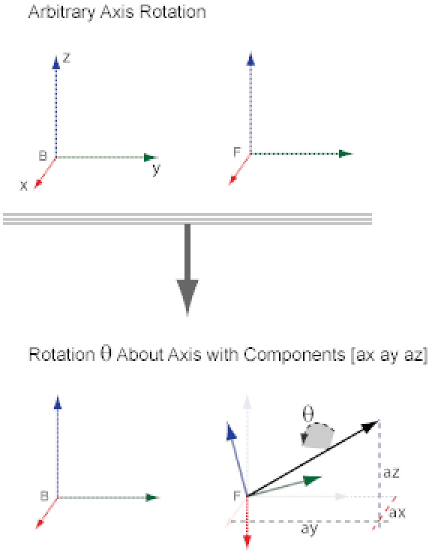
Standard Axis

Rotate frames with respect to each other about one of the three base frame axes: X, Y, or Z.



Arbitrary Axis

Rotate two frames with respect to each other about an arbitrary axis resolved in the base frame.



Concepts

- “Translation Methods” on page 1-32
- “Rotation Measurements” on page 4-26
- “Translation Measurements” on page 4-31
- “Represent Binary Link Frame Tree” on page 1-36

Translation Methods

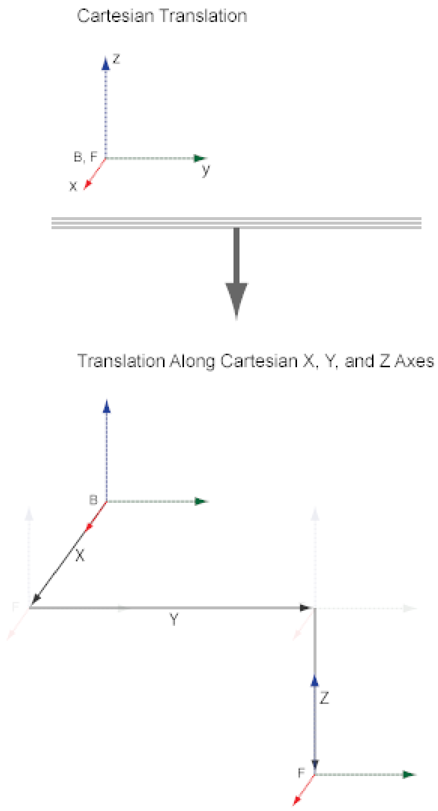
You can specify frame translation using different methods. These include Cartesian, standard axis, and cylindrical. The different methods are available through the Rigid Transform block. The choice of method depends on the model. Select the method that is most convenient for the application.

Specifying Translation

Translation is a relative quantity. The translation of one frame is meaningful only with respect to another frame. As such, the Rigid Transform block requires two frames to specify a translation: base and follower. The transformation operates on the base frame. For example, a translation along the +Z axis places the follower frame along the +Z axis from the base frame. Reversing frame ports is allowed, but the transformation is reversed: the base frame is now placed along the +Z axis from the follower frame.

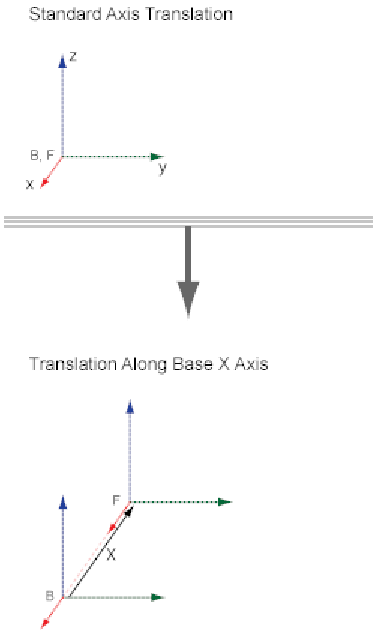
Cartesian

Translate follower frame along arbitrary Cartesian vector resolved in the base frame.



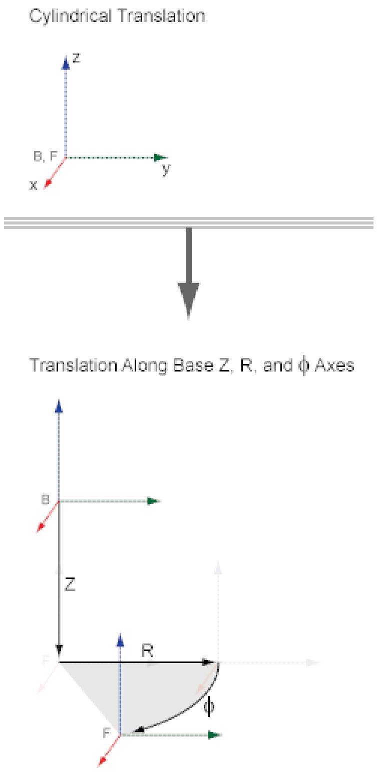
Standard Axis

Translate follower frame along one of the three axes of the base frame.



Cylindrical

Translate follower frame along cylindrical axes resolved in the base frame.



Concepts

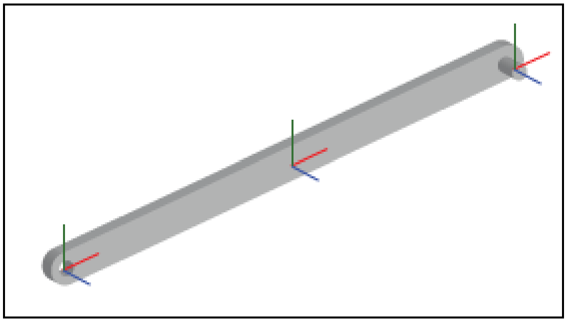
- “Translation Measurements” on page 4-31
- “Rotation Methods” on page 1-28
- “Rotation Measurements” on page 4-26

Represent Binary Link Frame Tree

In this section...

Model Overview

In this example, you model the frame tree for a binary link subsystem. This frame tree contains one Reference frame and two end frames for the joints. The Reference frame identifies the main body of the link. It is labeled **main**. The end frames identify the peg and hole sections of the link. They are labeled, **peg** and **hole**, respectively.



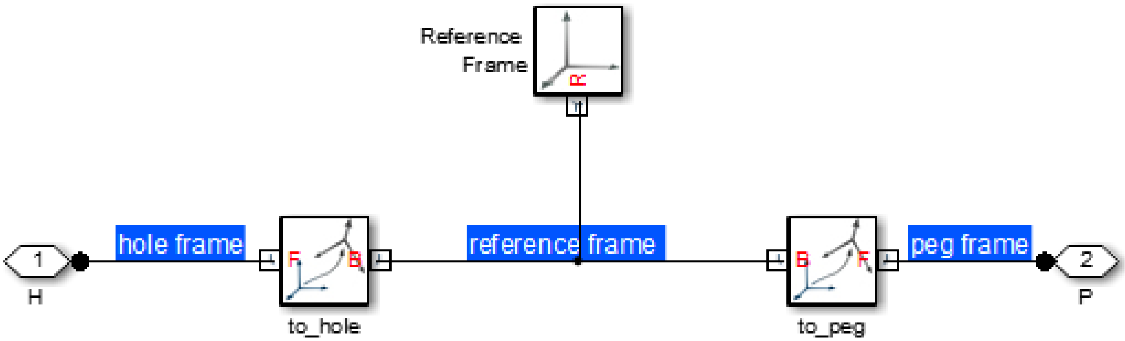
Modeling Approach

Modeling a complete binary link is a twofold task in which you specify:

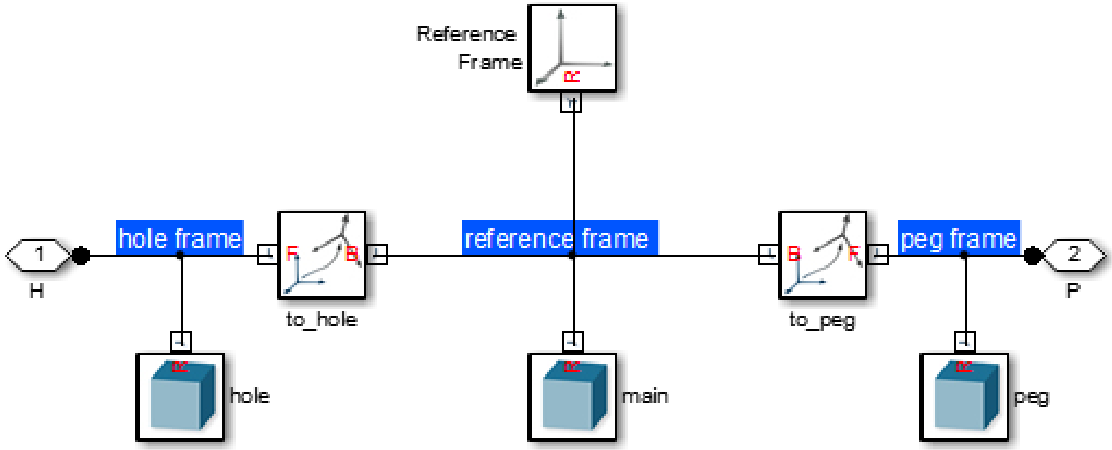
1 Frames

2 Solids

This example guides you through step 1 — modeling the frame tree for a binary link. The figure shows the resulting block diagram for this step.



Once you have completed this example, you can complete step 2 — adding to the frame tree the solids that comprise the binary link. The figure shows the final block diagram for the binary link subsystem.



Dimensions and Transforms

You can promote subsystem reusability by parameterizing link dimensions in terms of MATLAB variables. In this example, you initialize the variables in a subsystem mask. You can then specify their numerical values in the subsystem dialog box. Refer to the table for the dimensions needed to model the binary link frame tree.

Dimension	MATLAB Variable
Length	`L_Link`
Width	`W_Link`
Thickness	`T_Link`

A Reference Frame block identifies the location of the Reference frame. Two Rigid Transform blocks define the position of the end frames with respect to the Reference frame. In this example, you label these blocks **to_peg** and **to_hole**. Refer to the table for the associated transforms.

End Frame	Rigid Transform Block	Translation [X Y Z]
Peg	to_peg	[L_Link/2 0 3/2*T_Link]
Hole	to_hole	[-L_Link/2 0 0]

Build Model

Having defined the transforms required to add the end frames, you can now build the model:

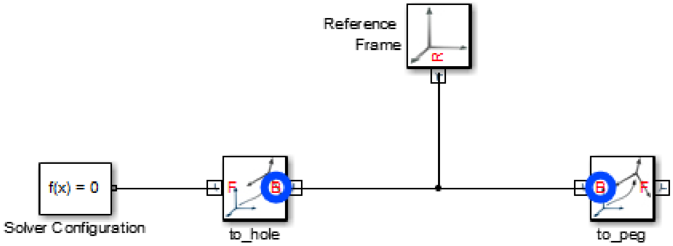
1 Start a new model.

2 Drag the following blocks to the model.

Block	Quantity	Library
Reference Frame	1	Frames and Transforms
Rigid Transform	2	Frames and Transforms
Solver Configuration	1	Simscape™ Utilities

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

Note Pay close attention to Rigid Transform port orientation. Both base (B) port frames should connect to the Reference Frame port. This ensures the rigid transform applies *to* the Reference frame, and not the end frames.



4 In the Rigid Transform block dialog boxes, expand **Translation**.

5 In **Method**, select `Cartesian`.

6 In **Offset**, enter the parameters in the table and press **OK**.

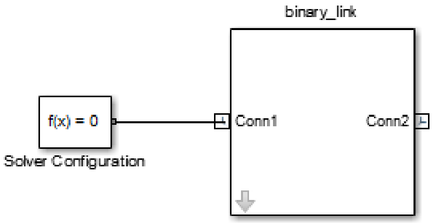
Rigid Transform Block	Offset
to_peg	`[L_Link/2 0 3/2*T_Link]`
to_hole	`[-L_Link/2 0 0]`

Generate Binary Link Subsystem

To initialize the MATLAB dimension variables used to specify the frame transforms, convert the binary link block diagram into a subsystem and use the subsystem mask:

1 Select the Reference Frame and Rigid Transform blocks.

2 Press **Ctrl+G** to enclose the blocks in a subsystem.



3 Click the Subsystem block and press **Ctrl+M** to create a subsystem mask.

4 In the **Parameters & Dialog** tab, add the following text boxes to a **Parameters** group and click **OK**.

Prompt	Name
`Length`	`L_Link`
`Width`	`W_Link`
`Thickness`	`T_Link`

5 In the subsystem dialog box, specify these parameters.

Parameter	Value
Length	`0.2`
Width	`0.02`
Thickness	`0.008`

Note Parameter values use the default length unit of meter (m).

Visualize Model

Update the model to visualize the frame tree in Mechanics Explorer.

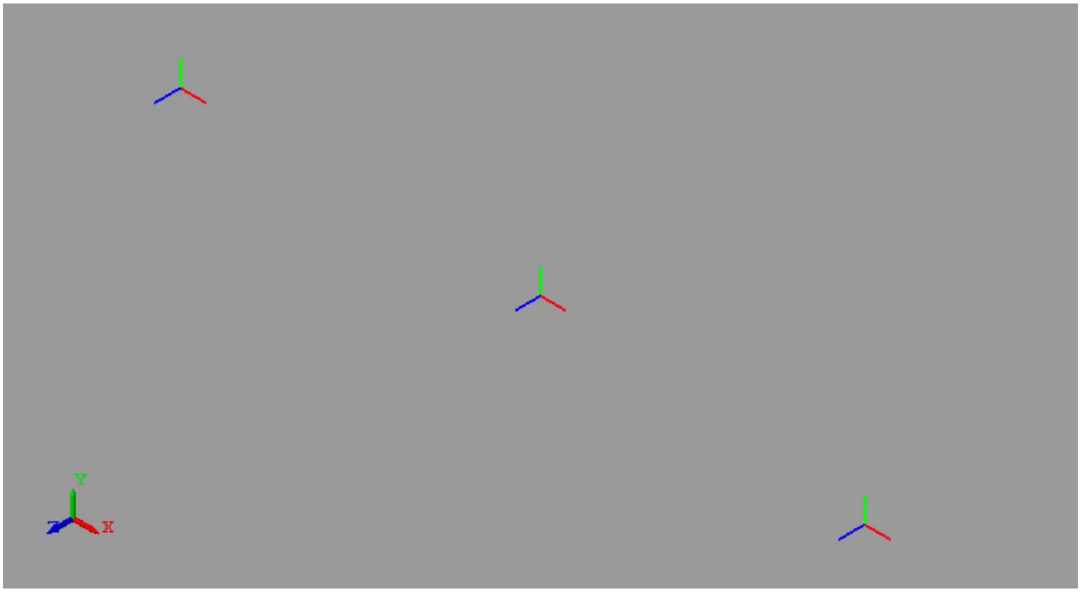
1 Press **Ctrl+D** to update the diagram.

2 In the Mechanics Explorer toolstrip, click the frame button ⊥.

The three frames appear in the visualization pane of Mechanics Explorer. The default view convention is **Z Up (XY Top)**, which differs from the Y Up convention used in the example schematics. To change the Mechanics Explorer view convention to **Y Up (XY Front)**:

- In the Mechanics Explorer toolstrip, select **Y Up (XY Front)** from the **View Convention** drop-down list.

Compare the resulting frame tree to the example schematics to confirm the validity of the transforms specified.



Save Custom Block

So that you can use it in later examples, save the binary_link subsystem as a custom library block. If you have not done so, create a new library to save the block in:

1 In the Simulink® menu bar, click **File > New > Library**.

2 Drag the binary_link subsystem block to the new library.

3. Save the library with a convenient name (e.g. linkage_elements) in an accessible folder.

Related Examples

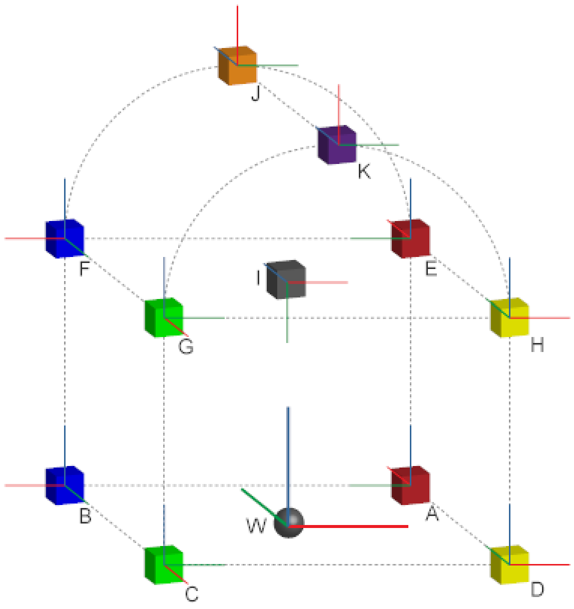
- “Model Binary Link” on page 2-57

Represent Box Frame Tree

In this section...

Model Overview

In SimMechanics, you can rigidly connect multiple Solid blocks to represent a complex rigid body. To position and orient different solids with respect to each other, you create a frame network that you can connect the solids to. The frame network contains Rigid Transform blocks that specify the spatial relationships between the different frames. In this example, you represent the frame tree for a box shape.



The example highlights the Rigid Transform block as the basic tool that you use to specify spatial relationships between frames and the solids that connect to them. The complete frame network is complex. It highlights nearly every type of rigid transformation that you can apply between two frames.

The modeling process in this example contains four stages:

1. Add World Frame (W).

   This is the ultimate reference frame against which you define all other frames.

2. Add the frames of the box bottom plane (frames A-D in the figure).

   You define these frames directly with respect to the World frame.

3. Add the frames of the box top plane (frames E-I in the figure).

   You define these frames directly with respect to the box bottom plane frames.

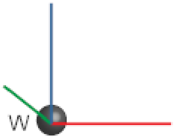
4 Add the frames of the box arch (frames K and J in the figure).

You define these frames directly with respect to the center frame of the box top plane.

This example is based on model `sm_frame_tree`, which accompanies your SimMechanics installation. To open this model, at the MATLAB command line, enter `sm_frame_tree`.

Start Model

Start a new model. Then, add a global reference frame that you can use to define other frames.



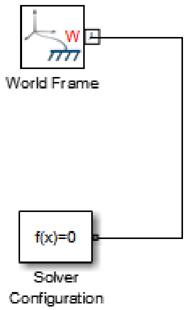
Use the World Frame block to represent the World frame:

1 Start a new model.

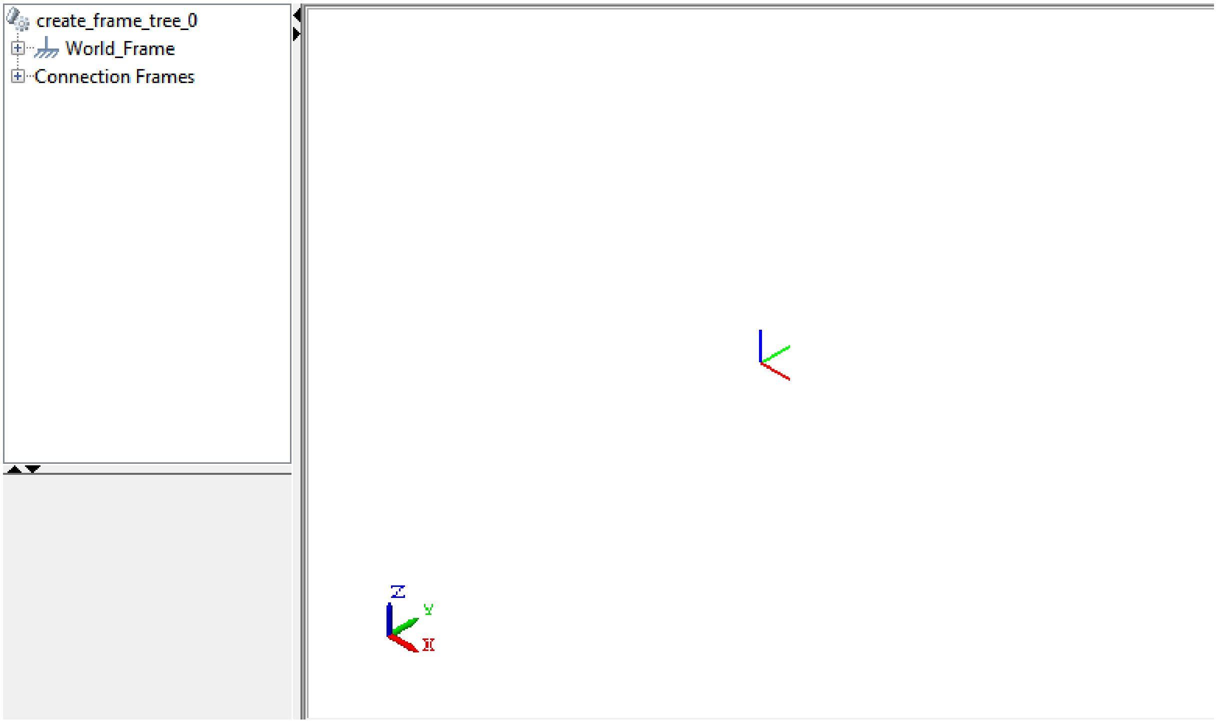
2 Drag the following blocks into the model.

Library	Block	Quantity
Frames and Transforms	World Frame	1
Simscape Utilities	Solver Configuration	1

3 Connect the blocks as they appear in the figure.



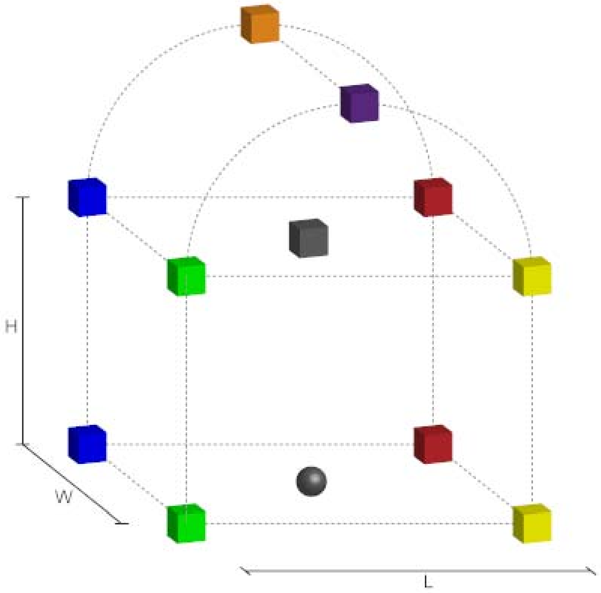
To visualize the World frame that you just added, on the Simulink menu bar, select **Simulation > Update Diagram**. Mechanics Explorer opens with a static 3-D display of your model. To view the position and orientation of this frame, on the Mechanics Explorer tool bar, toggle the frame visibility icon. Rotate, pan, and zoom to explore.



Initialize Model Workspace Parameters

To specify the distance offsets between frames, you use Rigid Transform blocks. In this example, you specify the distance offsets in terms of MATLAB variables that you initialize in the model workspace. The table lists these variables.

Dimension	Variable
Length	L
Width	W
Height	H



To initialize the MATLAB variables:

1 On the Simulink menu bar, click **Tools > Model Explorer**.

2 On the Model Hierarchy pane, double-click the name of your model (e.g. **frame_tree**).

3 Click **Model Workspace**.

4 On the **Model Workspace** pane, in the **Data Source** drop-down list, select `MATLAB Code`.

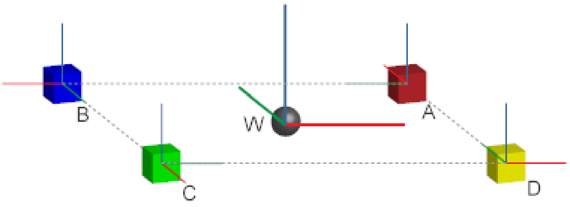
5 In the **MATLAB Code** section that appears, enter the following code:

```
% Size of Cube
L = 12;
W = 10;
H = 8;
```

6 Click **Reinitialize from Source**.

Add Bottom Plane Frames

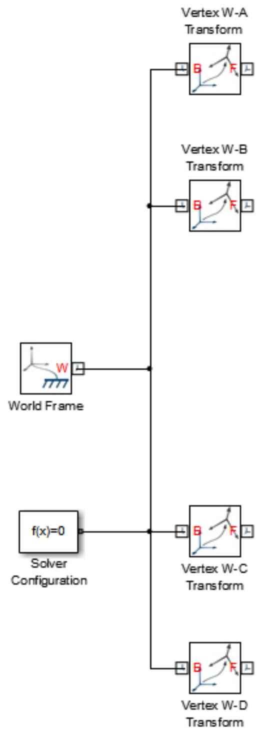
The World frame is the ultimate reference frame in a model. Now that you added the World frame to your model, you can define other frames with respect to it. You do this using the Rigid Transform block.



To define the four corner frames of the bottom box plane:

1. From the Frames and Transforms library, drag four Rigid Transform blocks to the model.

2. Connect and name the blocks as they appear in the figure.



3 Double-click the Vertex W-A Transform block and, in the dialog box, specify the parameters that the table provides.

Parameter Section	Parameter	Value
Rotation	**Method**	Select `Standard Axis`
	Axis	Select `+Z`
	Angle	Enter `90` (`deg`)
Translation	**Method**	Select `Cartesian`
	Offset	Enter `[L/2 W/2 0]` (`cm`)

4 Double-click the Vertex W-B Transform block and, in the dialog box, specify the parameters that the table provides.

Parameter Section	Parameter	Value
Rotation	**Method**	Select `Aligned Axis`
	Pair 1 > Follower/Base	Select `+X/-X`
	Pair 2 > Follower/Base	Select `+Y/-Y`
Translation	**Method**	Select `Cartesian`
	Offset	Enter `[-L/2 W/2 0]` `(cm)`

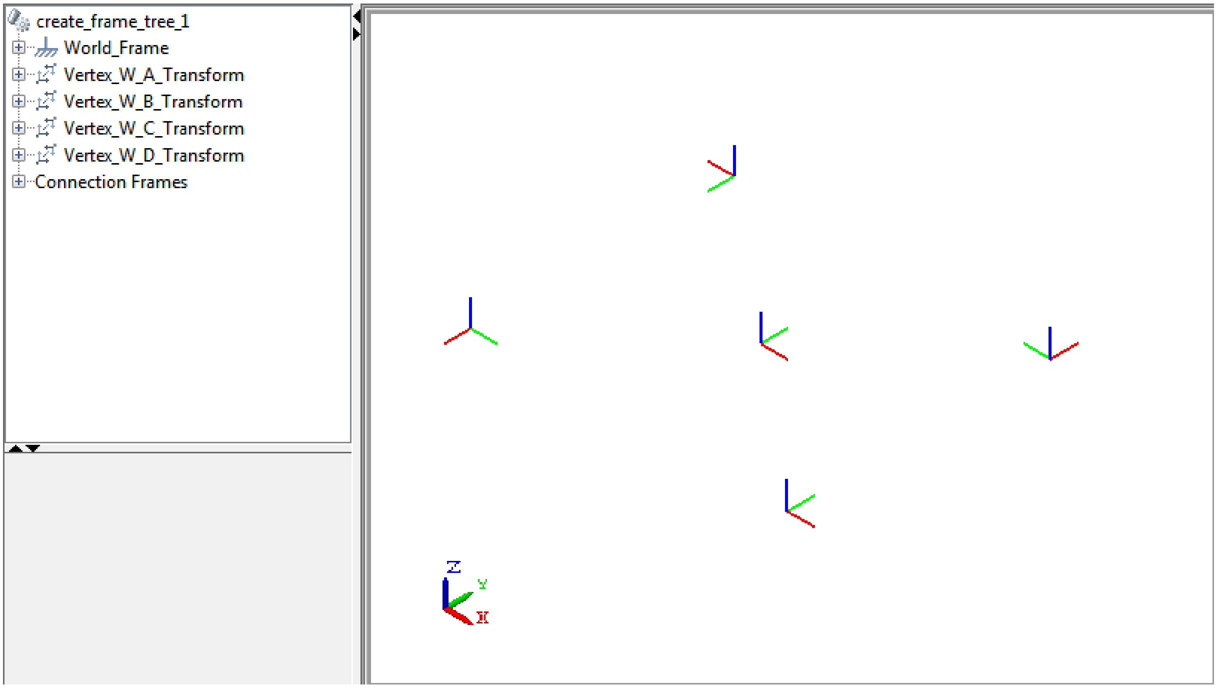
5 Double-click the Vertex W-C Transform block and, in the dialog box, specify the parameters that the table provides.

Parameter Section	Parameter	Value
Rotation	**Method**	Select `Standard Axis`
	Axis	Select `+Z`
	Angle	Enter `270` `(deg)`
Translation	**Method**	Select `Cartesian`
	Offset	Enter `[-L/2 -W/2 0]` `(cm)`

6 Double-click the Vertex W-D Transform block and, in the dialog box, specify the parameters that the table provides.

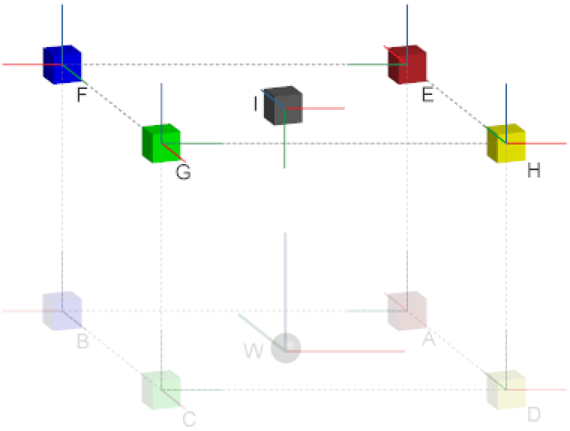
Parameter Section	Parameter	Value
Rotation	**Method**	Select `None`
Translation	**Method**	Select `Cartesian`
	Offset	Enter `[L/2 -W/2 0]` `(cm)`

To visualize the frames that you just added, on the Simulink menu bar, select **Simulation > Update Diagram**. Mechanics Explorer opens with a static 3-D display of your model. To view the position and orientation of each frame, on the Mechanics Explorer tool bar, toggle the frame visibility icon .



Add Top Plane Frames

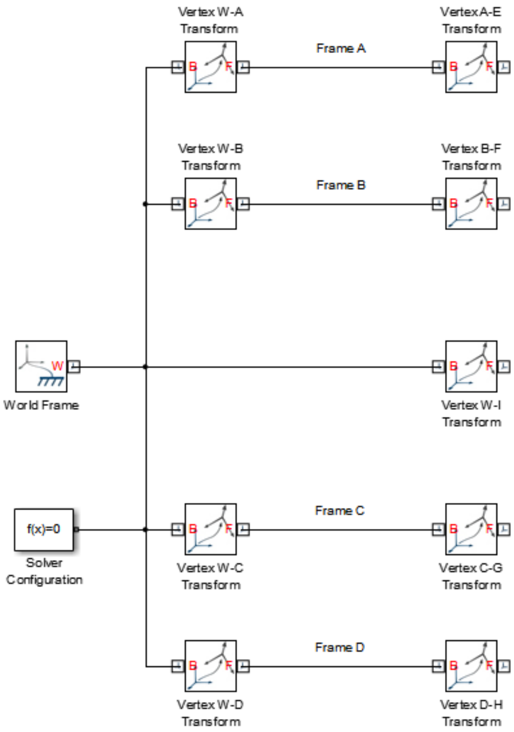
You can now define the top plane frames with respect to the bottom plane frames.



To add the top plane frames:

1 From the Frames and Transforms library, drag five Rigid Transform blocks.

2 Connect and name the blocks as they appear in the figure.



3 Double-click the following blocks:

- Vertex A-E Transform
- Vertex B-F Transform
- Vertex C-G Transform
- Vertex D-H Transform

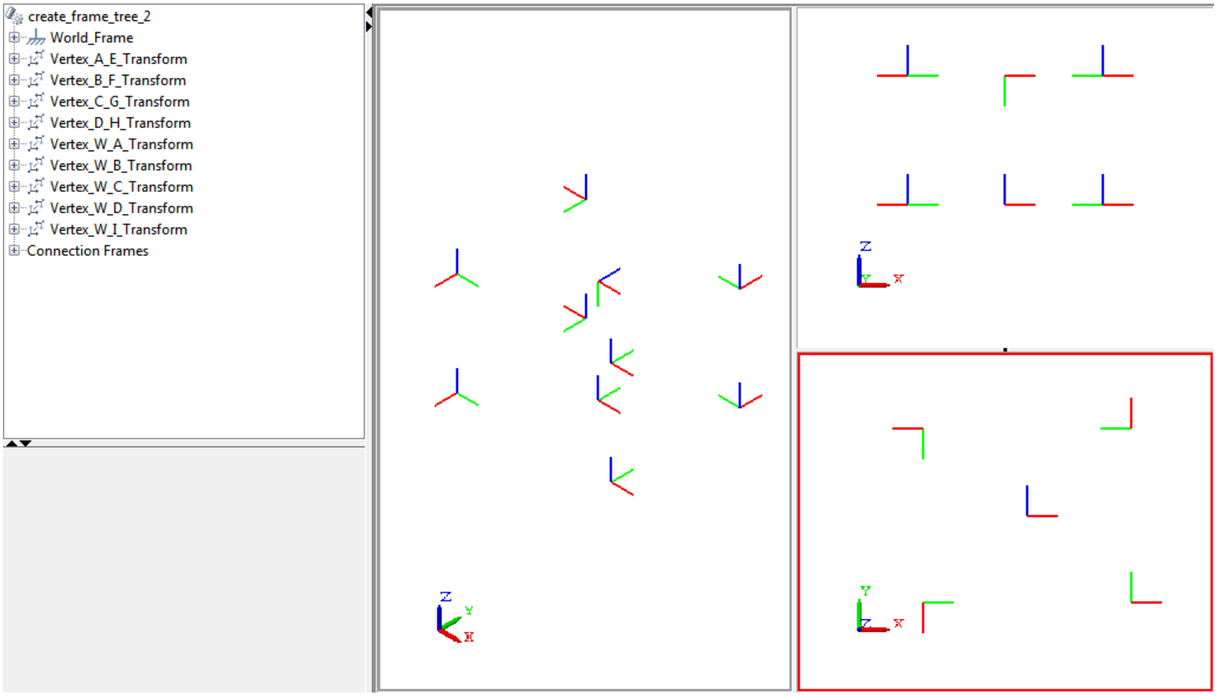
4 In each block dialog box, specify the following parameters.

Parameter Section	Parameter	Value
Rotation	**Method**	Select `None`
Translation	**Method**	Select `Standard Axis`
	Axis	`+Z`
	Offset	Enter `H` (`cm`)

5 Double-click the Vertex W-I Transform block and, in the dialog box, specify the following parameters.

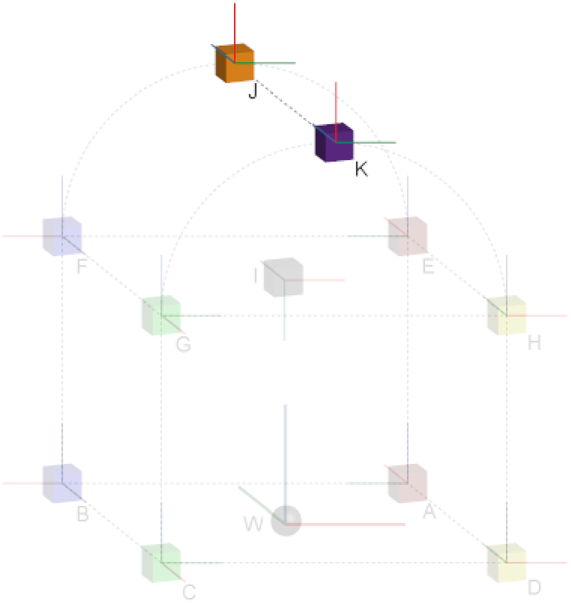
Parameter Section	Parameter	Value
Rotation	**Method**	Select `Aligned Axes`
	Pair 1 > Follower/Base	Select `+Y/-Z`
	Pair 2 > Follower/Base	Select `+Z/+Y`
Translation	**Method**	Select `Standard Axis`
	Axis	`+Z`
	Offset	Enter `H` (`cm`)

To visualize the frames that you just added, on the Simulink menu bar, select **Simulation > Update Diagram**. Mechanics Explorer opens with a static 3-D display of your model. To view the position and orientation of each frame, on the Mechanics Explorer tool bar, check that the frame visibility icon is toggled on.



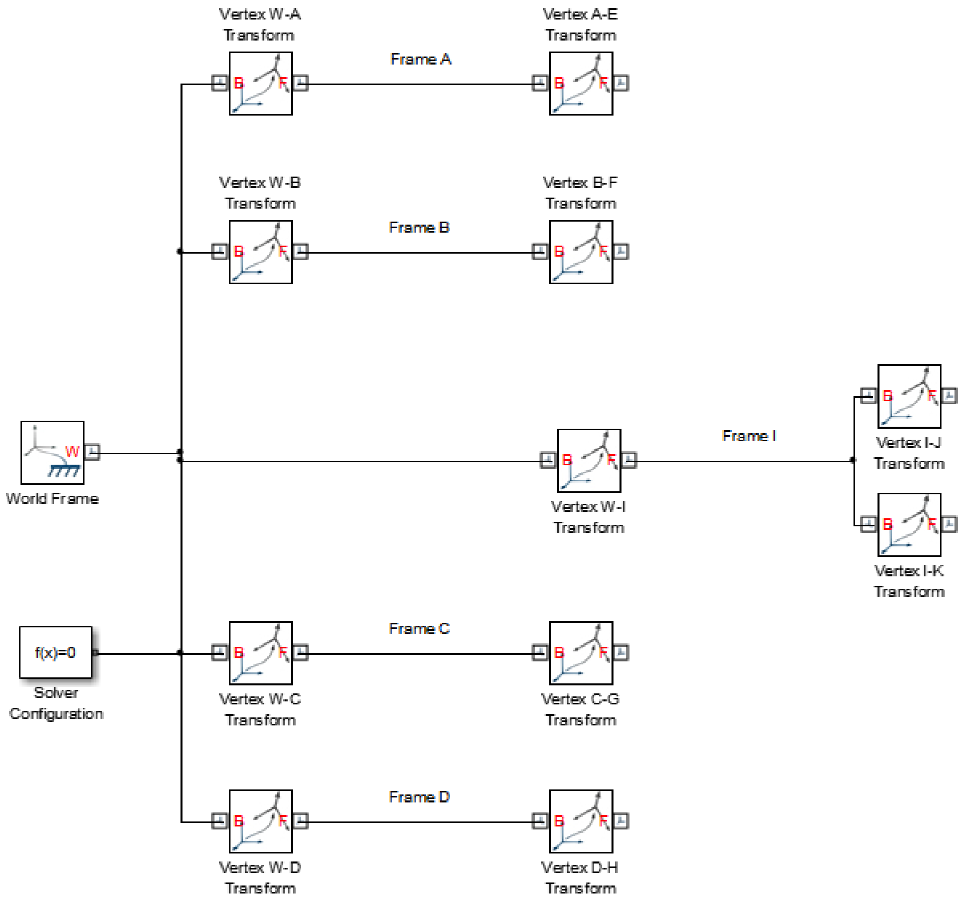
Add Arch Frames

Finally, add the two arch frames. As before, use the Rigid Transform block to define these frames. Define them with respect to the center frame of the top plane (frame I).



To define the arch frames:

1. From the Frames and Transforms library, drag two Rigid Transform blocks.
2. Connect and name the blocks as they appear in the figure.



3 Double-click the Vertex I-J Transform block and, in the dialog box, specify the parameters that the table provides.

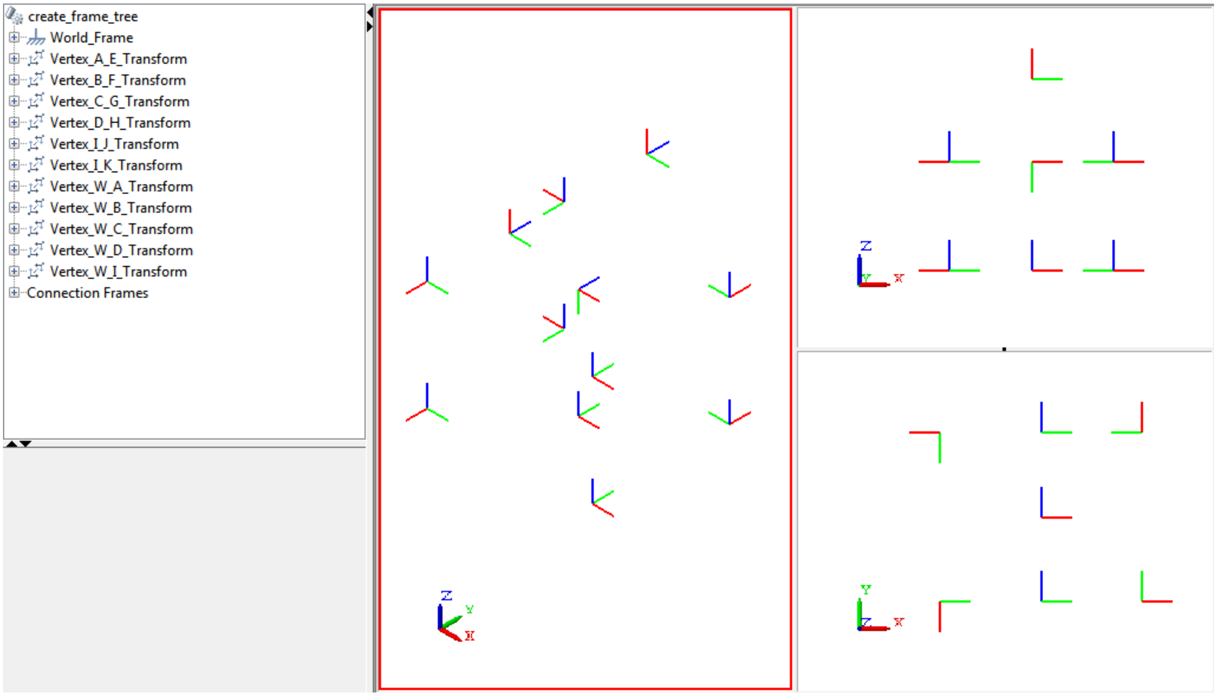
Parameter Section	Parameter	Value
Rotation	**Method**	Select `Standard Axis`
	Axis	Select `+Z`
	Angle	Enter `-90 (deg)`

Parameter Section	Parameter	Value
Translation	**Method**	Select `Cylindrical`
	Radius	Enter `L/2` (`cm`)
	Theta	Enter `-90` (`deg`)
	Z Offset	Enter `W/2` (`cm`)

4 Double-click the Vertex I-K Transform block and, in the dialog box, specify the parameters that the table provides.

Parameter Section	Parameter	Value
Rotation	**Method**	Select `Standard Axis`
	Axis	Select `+Z`
	Angle	Enter `-90` (`deg`)
Translation	**Method**	Select `Cylindrical`
	Radius	Enter `L/2` (`cm`)
	Theta	Enter `-90` (`deg`)
	Z Offset	Enter `-W/2` (`cm`)

To visualize the frames that you just added, on the Simulink menu bar, select **Simulation > Update Diagram**. Mechanics Explorer opens with a static 3-D display of your model. To view the position and orientation of each frame, on the Mechanics Explorer tool bar, check that the frame visibility icon is toggled on.



Save Model

Save the model as `frame_tree` in a convenient folder. In a subsequent example, you use Graphic blocks to represent each frame with a graphic icon. See "Visualize Box Frame Tree" on page 1-62

Related Examples

- "Visualize Box Frame Tree" on page 1-62
- "Represent Binary Link Frame Tree" on page 1-36

Concepts

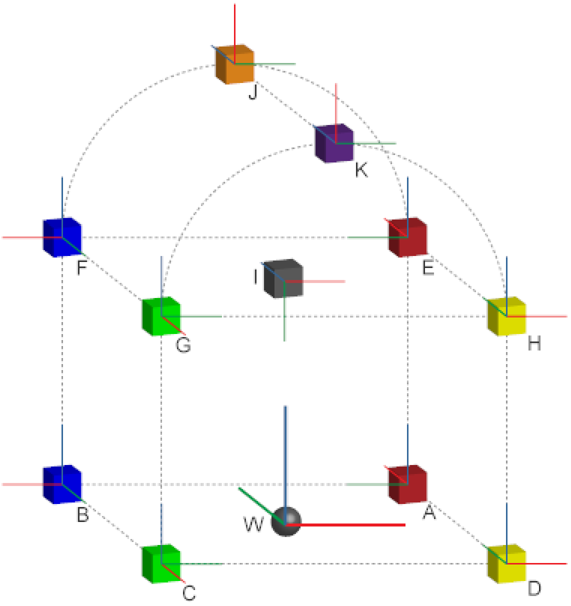
- "Representing Frames" on page 1-7
- "Frame Transformations" on page 1-19
- "World and Reference Frames" on page 1-13
- "Translation Methods" on page 1-32

Visualize Box Frame Tree

In this section...
“Model Overview” on page 1-62
“Build Model” on page 1-63
“Visualize Model” on page 1-65

Model Overview

To visualize a frame or frame network, you can use the Graphic block. By connecting this block to a frame, you add a graphic icon to that frame. The graphic icon has zero inertia and it does not affect model dynamics during simulation. In this example, you use Graphic blocks to add graphic icons to the box frame tree that you modeled in a previous example. See “Represent Box Frame Tree” on page 1-44.



Build Model

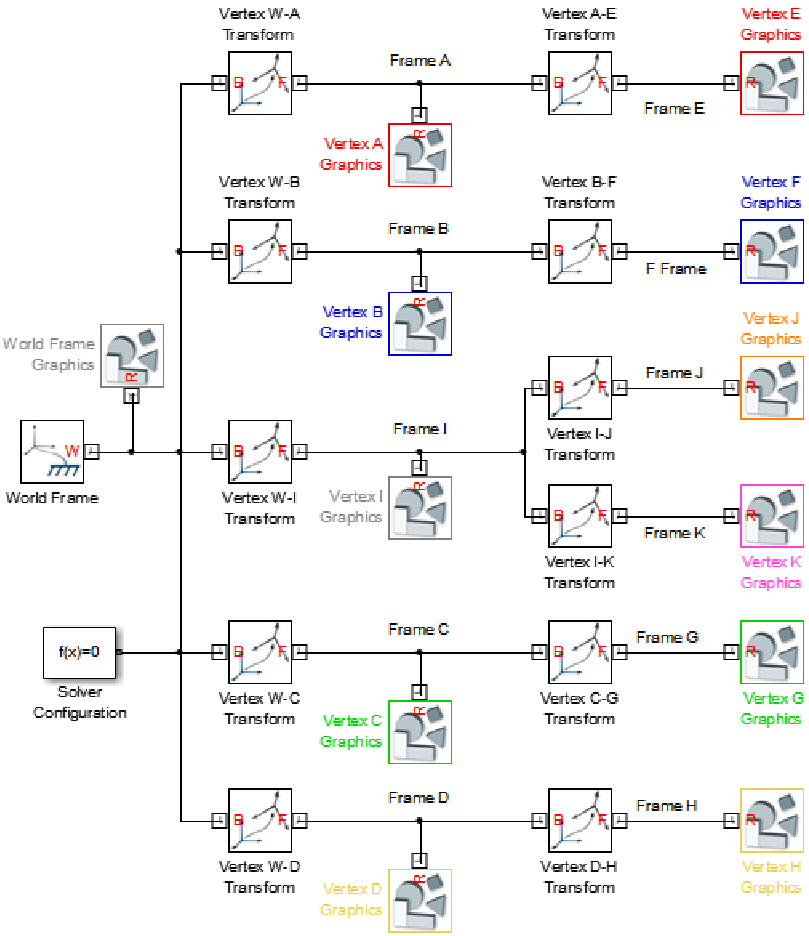
To add a graphic icon to each frame in your model:

1. Open model `frame_tree`.

   This is the model that you created in example “Represent Box Frame Tree” on page 1-44.

2. From the Body Elements library, drag 12 Graphic blocks to that model.

3. Connect and name the blocks as they appear in the figure.



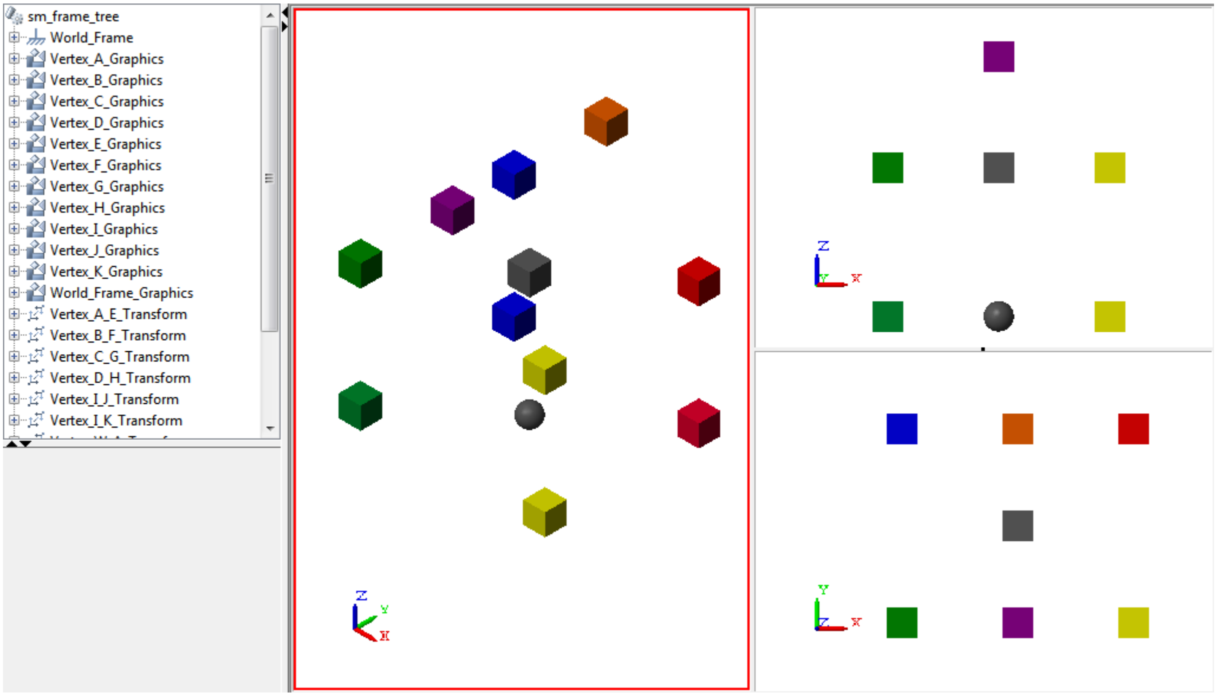
4 Double-click each Graphic block.

5 In the dialog box, specify parameters according to the following table.

Graphic Block	Color	Shape	Size
World Frame Graphics	`[0.4 0.4 0.4]`	`Sphere`	`25`
Vertex I Graphics		`Cube`	
Vertex A Graphics	`[1.0 0.0 0.0]`		
Vertex E Graphics			
Vertex B Graphics	`[0.0 0.0 1.0]`		
Vertex F Graphics			
Vertex C Graphics	`[0.0 0.6 0.2]`		
Vertex G Graphics			
Vertex D Graphics	`[1.0 1.0 0.0]`		
Vertex H Graphics			
Vertex J Graphics	`[1.0 0.4 0.0]`		
Vertex K Graphics	`[0.6 0.0 0.6]`		

Visualize Model

You can now visualize your model in Mechanics Explorer. To do this, on the Simulink menu bar, select **Simulation > Update Diagram**. Mechanics Explorer opens with a static 3-D display of your model. Rotate, pan, and zoom to explore.



Related Examples

- “Represent Box Frame Tree” on page 1-44
- “Represent Binary Link Frame Tree” on page 1-36

Concepts

- “Representing Frames” on page 1-7
- “Frame Transformations” on page 1-19
- “Rotation Methods” on page 1-28
- “Translation Methods” on page 1-32

Find and Fix Frame Issues

In this section...

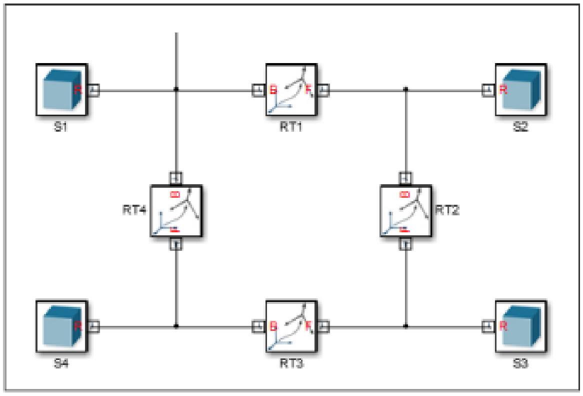
If your model contains an invalid frame connection, SimMechanics issues an error and the model does not simulate. Possible error sources include:

- 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, SimMechanics issues an error and the model does not simulate.

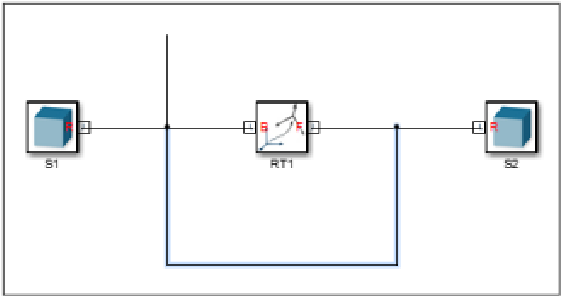
To remove the simulation error, disconnect one Rigid Transform block. This step removes the redundant constraint, and allows the model to simulate. The following figure shows a rigidity loop. The loop contains four Rigid Transform blocks directly connected to each other.



Shorted Rigid Transform Blocks

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, SimMechanics 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.



Related Examples

- “Represent Box Frame Tree” on page 1-44
- “Represent Binary Link Frame Tree” on page 1-36

Concepts

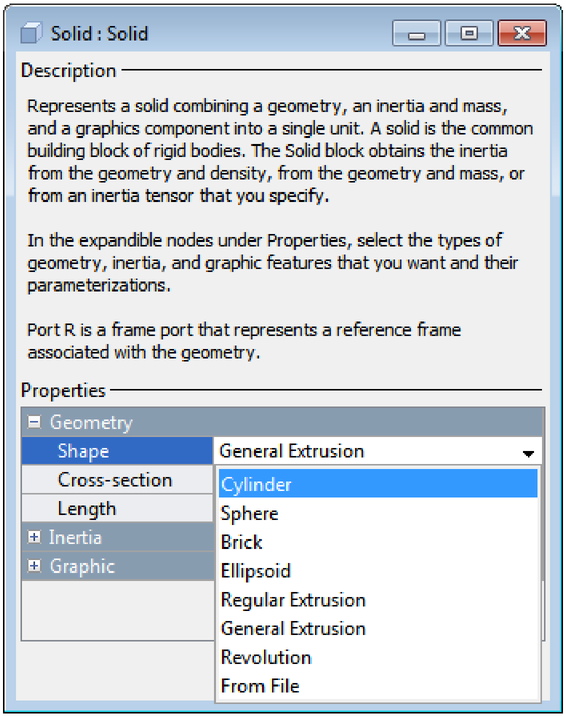
- “Representing Frames” on page 1-7
- “Frame Transformations” on page 1-19
- “Rotation Methods” on page 1-28
- “Translation Methods” on page 1-32

Rigid Bodies

Specifying Solid Geometry

In this section...
"Simple Shapes" on page 2-2
"Advanced Shapes" on page 2-4

SimMechanics provides a set of shapes that you can use to represent rigid bodies. You can specify the shapes directly in the Solid block dialog box.



Solid : Solid

Description

Represents a solid combining a geometry, an inertia and mass, and a graphics component into a single unit. A solid is the common building block of rigid bodies. The Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor that you specify.

In the expandible nodes under Properties, select the types of geometry, inertia, and graphic features that you want and their parameterizations.

Port R is a frame port that represents a reference frame associated with the geometry.

Properties

Geometry

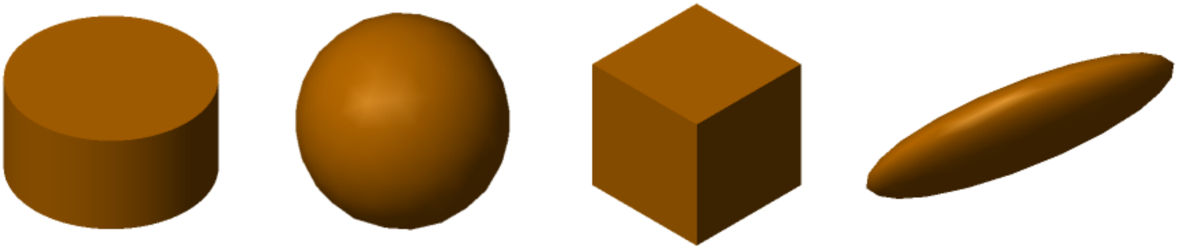
Shape: General Extrusion

Simple Shapes

Shapes range from simple to advanced. Simple shapes require a small number of dimensional parameters. The following simple shapes are available.

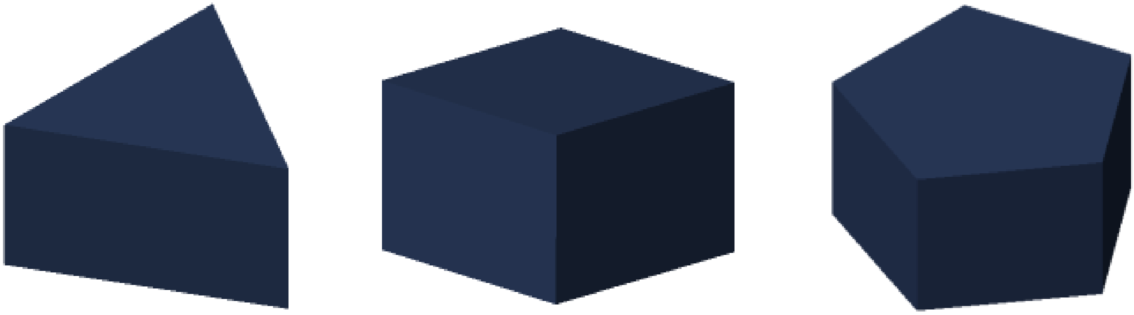
- `Cylinder` — Cylinder with custom dimensions, centroid at the solid reference frame origin, and symmetry axis along the solid reference frame z-axis.
- `Sphere` — Sphere with custom dimensions and center located at the solid reference frame origin.
- `Brick` — Brick with custom dimensions along the three Cartesian axes and centroid located at the block reference frame.
- `Ellipsoid` — Ellipsoid with custom dimensions with centroid located at the block reference frame.
- `Regular Extrusion` — Extruded solid with constant cross-section along the z-axis and centroid located at the block reference frame. The constant cross-section is a regular polygon with a custom number of sides.

Simple shapes are easier to use than advanced shapes. When modeling a rigid body, consider using a simple shape as a first approximation. After successful model assembly, you can add detail to the rigid body. The following figure shows the four simple shapes, ordered left to right: `cylinder`, `Sphere`, `Brick`, and `Ellipsoid`.



The `Regular Extrusion` shape is more versatile than other simple shapes. With this shape, you can model solids with constant cross sections. Cross-sections can have any number of sides, but all lengths and internal angles are equal.

The following figure shows a set of shapes you can model with the `Regular Extrusion` shape.



Advanced Shapes

Advanced shapes include:

- `General Extrusion` — Extruded solid with custom cross-section swept along the z-axis and centroid located at the block reference frame.
- `Revolution` — Solid of revolution with constant cross-section revolved about the z-axis and centroid located at the block reference frame.

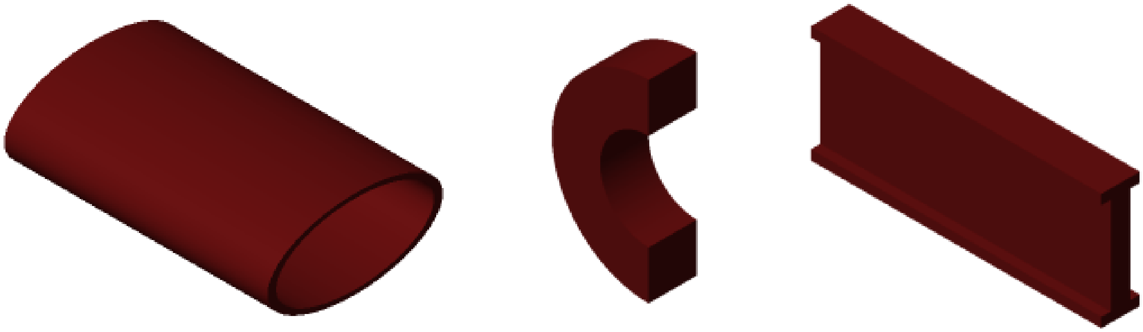
The shapes require a MATLAB cross-section matrix. To be valid, the matrix must observe a set of rules. See "Cross-Section Coordinates" on page 2-14.

General Extrusions

For extrusions with irregular cross-section, SimMechanics provides a `General Extrusion` geometry. This geometry is among the most versatile in SimMechanics. You can use it to model shapes with an increased level of detail.

This shape requires a MATLAB matrix that contains the cross-section coordinates. The matrix must follow a set of rules that are specific to the shape. See "Revolution and General Extrusion Cross-Sections" on page 2-10.

The following figure shows some shapes you can model with `General Extrusion`.



For `General Extrusion` examples, see:

- “Model I-Beam” on page 2-45
- “Model Box Beam” on page 2-51

Solids of Revolution

Solids that have a constant cross-section *about* an axis are solids of revolution. To model these solids, use the `Revolution` shape.

The `Revolution` shape requires a MATLAB matrix that contains the cross-section coordinates. The matrix must follow a set of rules specific to the `Revolution` geometry. See “Revolution and General Extrusion Cross-Sections” on page 2-10.

The following figure shows some shapes you can model with `Revolution`.

Concepts

- “Advanced Solid Shapes” on page 2-7
- “Revolution and General Extrusion Cross-Sections” on page 2-10
- “Specifying Solid Inertia” on page 2-20
- “Solid Color” on page 2-28

Advanced Solid Shapes

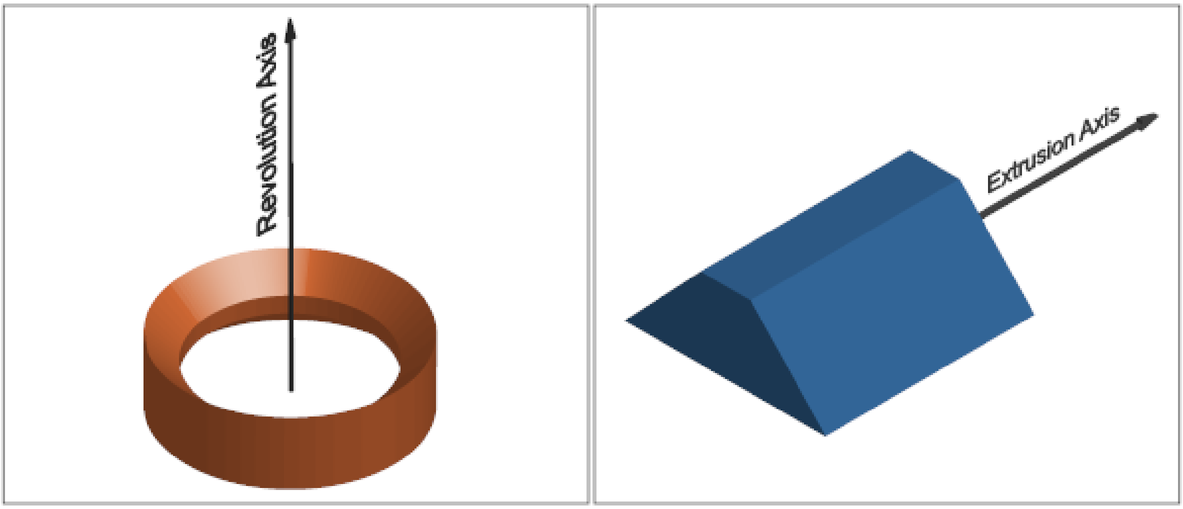
In this section...

With the Solid block, you can specify the geometry of a solid. This block provides a set of standard shapes so that you can easily specify simple shapes, e.g., `Cylinder`. For more complex shapes, the block provides two shapes: `General Extrusion` and `Revolution`.

When to Use Extrusion and Revolution Shapes

Use `General Extrusion` and `Revolution` shapes to represent solids that are to complex for standard shapes. The choice of shape depends on the symmetry of the solid. If the solid has translational symmetry, use `General Extrusion`. If the solid has rotational symmetry, use `Revolution`.

The solid has translational symmetry if its cross-section is constant along its length axis. The solid has rotational symmetry if its cross-section is constant about its length axis. The figure shows two solids that you can represent using `Revolution` and `General Extrusion` shapes.



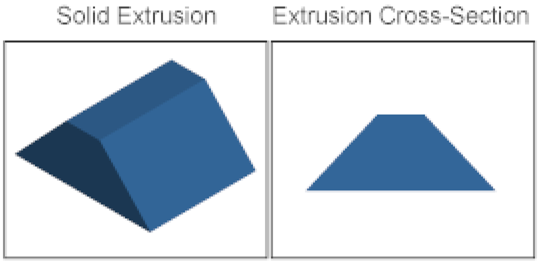
Both solids are too complex for standard shapes like `Cylinder`, `Brick`, or `Sphere`. The solid on the left possesses a constant cross-section about its length axis. You can represent it using shape `Revolution`. The solid on the right possesses a constant cross-section along its length axis. You can represent it using shape `General Extrusion`.

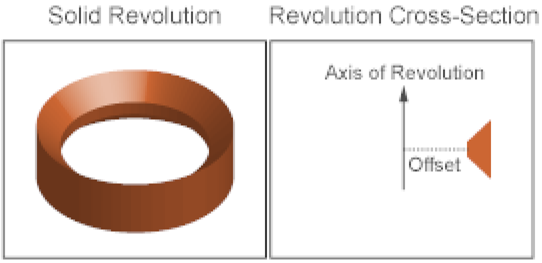
Specifying Extrusion and Revolution Shapes

To specify `General Extrusion` and `Revolution` shapes, you must provide the cross-section coordinates. Enter these coordinates as a matrix in the **Geometry > Cross-Section** parameter of the Solid block dialog box. SimMechanics connects the coordinate pairs with straight lines to generate the cross-section shape.

Note To see the **Cross-Section** parameter, you must first select `General Extrusion` or `Revolution` from the **Geometry > Shape** drop-down list.

The figure shows the cross-sections that you must specify to represent the extrusion and solid of revolution introduced in this section. The coordinate matrices for these cross-sections must follow a set of rules to be valid as input. For more information, see “Cross-Section Coordinates” on page 2-14.





Use the **Length** parameter of the `General Extrusion` shape to specify the length to extrude the cross-section along. Use the **Revolution Angle** parameter of the `Revolution` shape to specify the angle to sweep the cross-section about.

Note To see the **Revolution Angle** parameter, you must first select `Custom` from the **Geometry > Extent of Revolution** drop-down list.

Related Examples

- “Model Box Beam” on page 2-51
- “Model Dome” on page 2-40

Concepts

- “Cross-Section Coordinates” on page 2-14
- “Revolution and General Extrusion Cross-Sections” on page 2-10

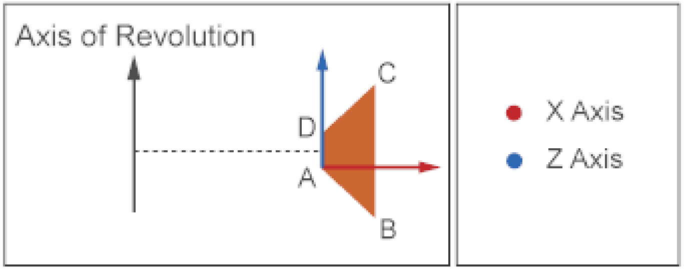
Revolution and General Extrusion Cross-Sections

In this section...

SimMechanics interprets the coordinate matrices of `Revolution` and `General Extrusion` according to a set of rules. These rules ensure consistency across all `Revolution` and `General Extrusion` shapes.

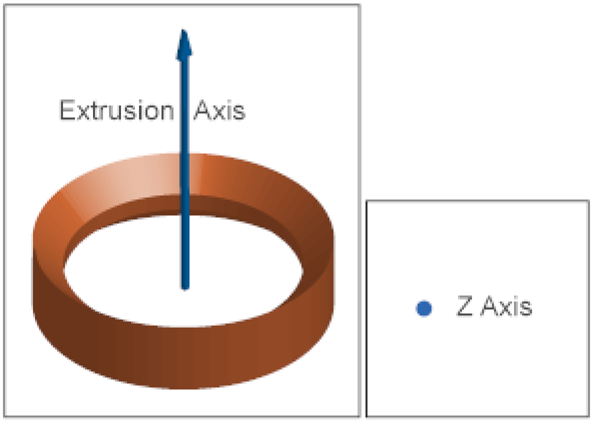
Revolution Coordinates are [x z] Pairs

SimMechanics maps the cross-section that you specify onto the XZ plane of the solid reference frame. When you enter the coordinate matrix in the **Cross-Section** parameter of the Solid block, SimMechanics treats those coordinates as [X Z] pairs, in that order.



Revolution Axis Aligns with Z-Axis

SimMechanics revolves the cross-section that you specify about the Z axis of the solid reference frame. The revolution axis runs along the thickness of the revolution.

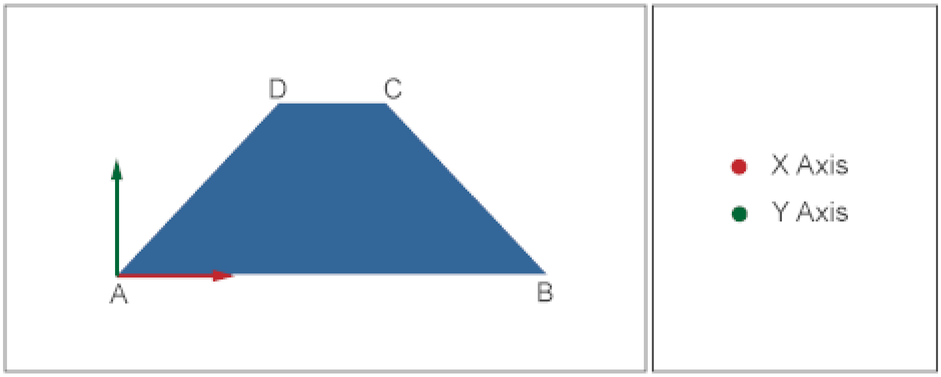


Revolution X-Coordinates Must Equal or Exceed Zero

The X coordinates of a revolution cross-section must be equal to or greater than zero. Negative X coordinates causes the cross-section to overlap during revolution. If you specify a cross-section with negative X coordinates, SimMechanics issues an error and the model does not simulate.

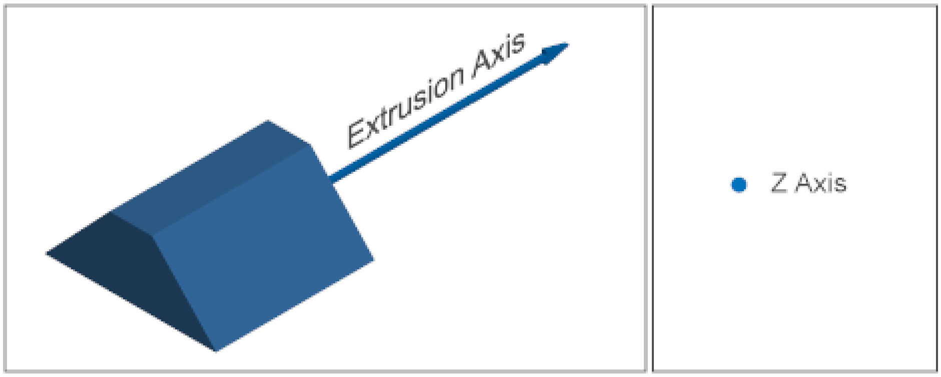
Extrusion Coordinates are [x y] Pairs

SimMechanics maps the cross-section that you specify onto the XY plane. When you enter the coordinate matrix in the **Cross-Section** parameter of the Solid block, SimMechanics treats those coordinates as [X Y] pairs, in that order.



Extrusion Axis Aligns with Z-Axis

SimMechanics extrudes the cross-section that you specify along the Z axis of the solid reference frame. The extrusion axis runs along the length of the extrusion.



Related Examples

- “Model Box Beam” on page 2-51
- “Model I-Beam” on page 2-45
- “Model Cone” on page 2-35

- “Model Dome” on page 2-40

Concepts

- “Cross-Section Coordinates” on page 2-14
- “Advanced Solid Shapes” on page 2-7

Cross-Section Coordinates

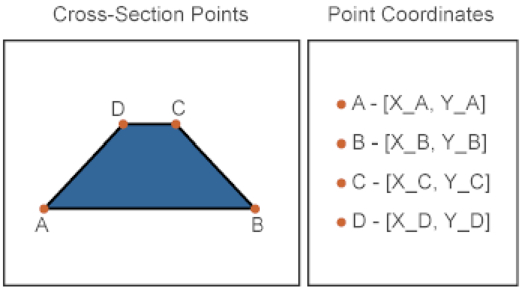
To represent a solid using the `Revolution` or `General Extrusion` shapes, you must provide the cross-section coordinates for that solid. For example, to represent a beam with a trapezoidal cross-section, you must provide the coordinates for that trapezoid. You must enter these coordinates according to a set of rules that ensure SimMechanics properly represents the cross-section shape.

Specifying Coordinates

The Solid block accepts the cross-section coordinates as an M×2 matrix. This matrix contains M rows, each with the coordinates of a cross-section point. Enter the coordinates sequentially: SimMechanics connects adjacent coordinate pairs with a straight line to represent the complete cross-section shape.

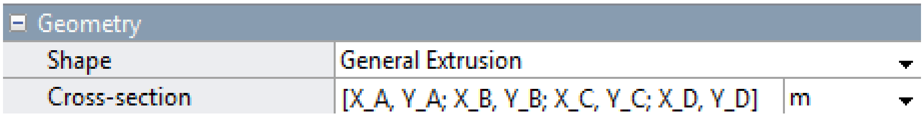
The figure shows the cross-section of a trapezoidal beam. SimMechanics connects adjacent points with straight lines, so you need to provide only four points. The figure labels these points A, B, C, and D. Specify the coordinates for these points in the order [A; B; C; and D]. Using the point coordinates in the figure, the MATLAB matrix for the trapezoid cross-section is:

```
trapezoid = [X_A, Y_A; X_B, Y_B; X_C, Y_C; X_D, Y_D]
```



SimMechanics automatically connects the first and last points of a coordinate matrix. This ensures that every cross-section path is closed. For example, in the trapezoid cross-section, SimMechanics automatically connects point D to point A. The result is a closed trapezoid path that SimMechanics can extrude.

You can enter the MATLAB matrix directly in the **Cross-Section** parameter of the Solid block. The figure shows an example. You can replace the X and Y coordinates with the numerical values directly, or you can define their values elsewhere, e.g., a subsystem mask or the model workspace.

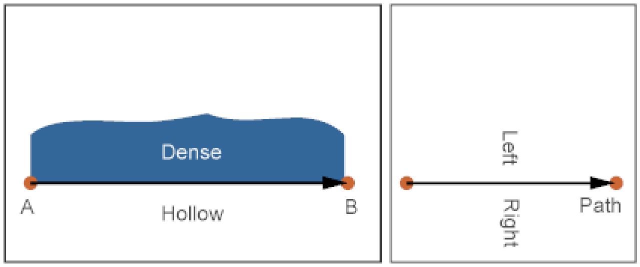


⊟ Geometry

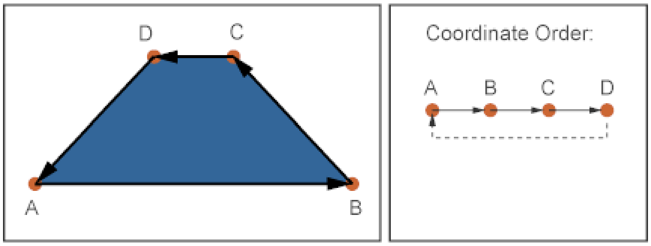
Shape	General Extrusion	
Cross-section	[X_A, Y_A; X_B, Y_B; X_C, Y_C; X_D, Y_D]	m

Coordinate Order

Any boundary path separates the dense and hollow regions of a cross-section. The dense region is to the left of the path, and the hollow region is to its right. The figure illustrates how a cross-section path divides dense and hollow regions.

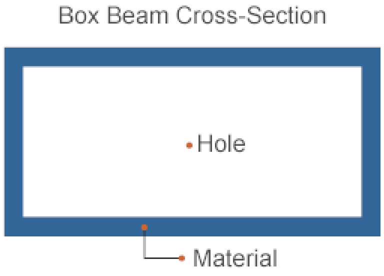


Always enter the cross-section coordinates so that the dense region is to the left of the arrow connecting one coordinate pair to the next. For example, to represent the trapezoidal cross-section in the figure, enter the coordinates in the order `[A B C D]`. This matrix specifies that the dense region is to the left of the arrows connecting A to B, B to C, C to D, and D to A.

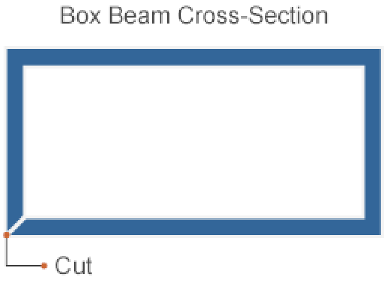


Hollow Cross-Sections

A cross-section need not be dense. You can specify a hollow cross-section. One example is the cross-section of a box beam. This cross-section has a rectangular shape with a dense area at the periphery, and a hole at the center. The figure shows that cross-section.

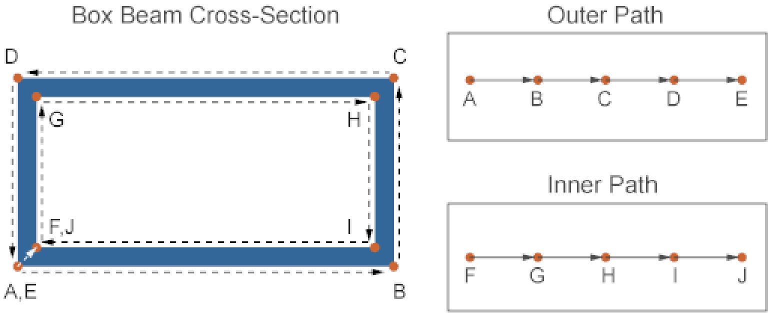


As with dense cross-sections, you specify a hollow cross-section as a single path. To do this, you must cut the cross-section across its dense region. By cutting the cross-section, you can merge the inner and outer paths into a single path. The figure shows the cut box beam cross-section.

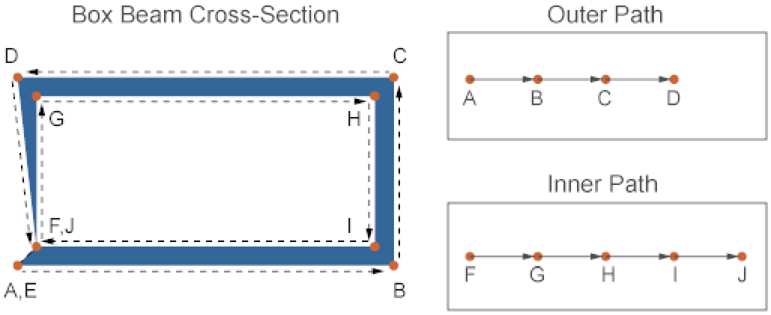


The cut connects the first and last cross-section coordinate pairs. As with dense cross-sections, you must specify the coordinate pairs so that the dense region is to the left of the path. A counterclockwise order satisfies this requirement for the outer portion of the path. A clockwise order satisfies this requirement for the inner portion of the path. Always specify all coordinates as a single path—not as two paths. You do this by connecting the inner and outer portions of the path through the cut.

The figure shows the order that you specify the cross-section coordinates in. The cut joins the last outer path point to the first inner path point. You specify the outer path in a counterclockwise order: `[A, B, C, D, E]`. You specify the inner path in a clockwise order: `[F, G, H, I, j]`. The entire coordinate matrix is `[A, B, C, D, E, F, G, H, I, J]`. SimMechanics automatically closes the path by connecting the last point that you specify (J) to the first point (A).

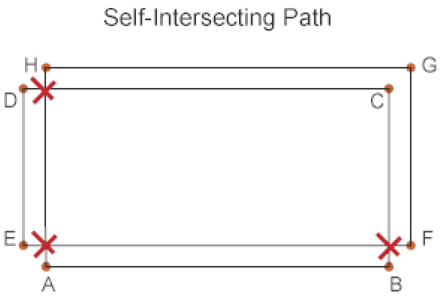


To connect the outer path to the inner path through the cut, you must repeat the first point of each path. For the outer path, you repeat point A (labeled E). For the inner path, you repeat point F (labeled J). Omitting these points distorts the cross-section that you specify. The figure shows the cross-section that results if you omit point E. As before, SimMechanics automatically closes the path by connecting the last point that you specify (J) to the first point (A).



Path Intersection

The coordinate matrix must define a path that does not self-intersect. If the path intersects itself at any point, SimMechanics issues an error and the model does not simulate. Path intersection is a common error source in hollow cross-sections. The figure shows a self-intersecting path.



Related Examples

- “Model Box Beam” on page 2-51
- “Model I-Beam” on page 2-45
- “Model Cone” on page 2-35
- “Model Dome” on page 2-40

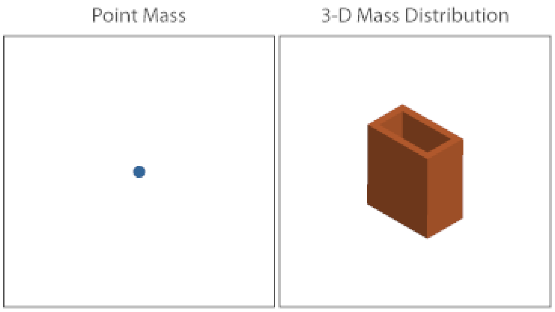
Concepts

- “Revolution and General Extrusion Cross-Sections” on page 2-10
- “Advanced Solid Shapes” on page 2-7

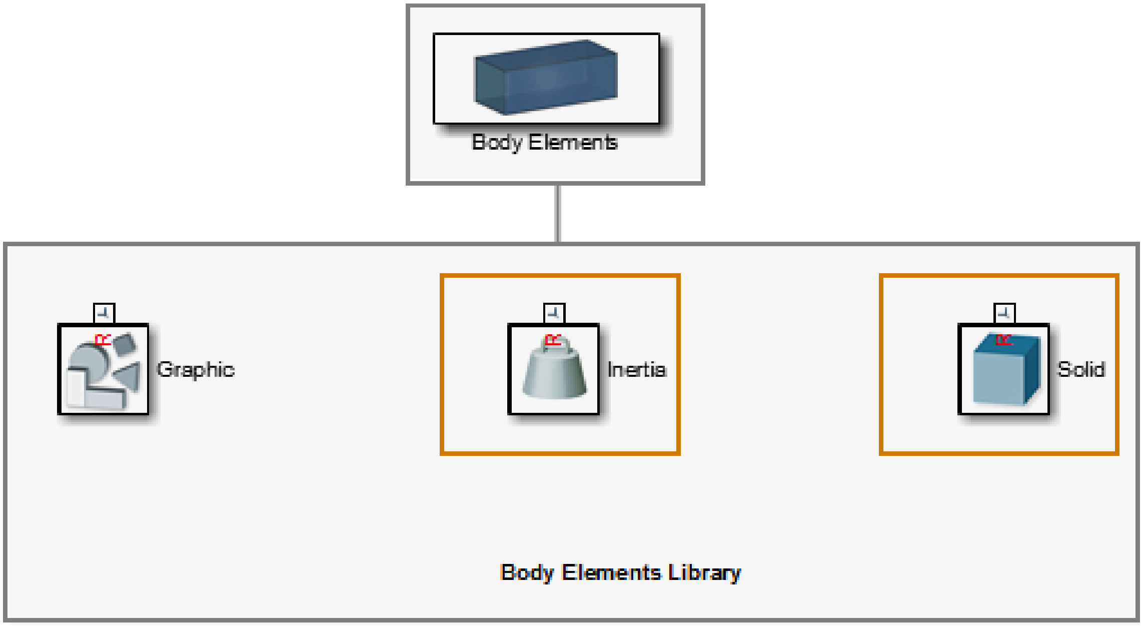
Specifying Solid Inertia

In this section...

The inertial properties of a rigid body influence its dynamic behavior. One example is the flywheel: the greater its inertia, the greater the rotational energy that it can store. In SimMechanics, you specify the inertial properties using a Solid or Inertia block. Use a Solid block to specify geometry and color in addition to inertia. Use an Inertia block to specify only inertia. Both blocks provide multiple inertia types that you can select. You can represent a solid as a point mass or as a mass distribution (3-D solid).



To use the blocks, drag them from the Body Elements library.



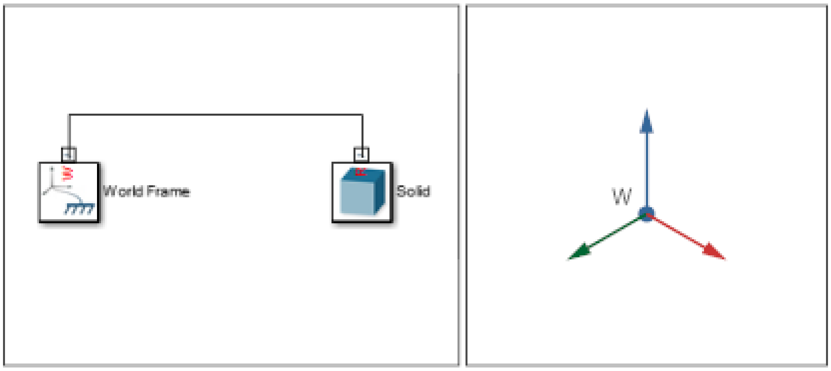
Point Mass

A point mass occupies an infinitesimally small volume. When you treat a solid as a point mass, you assume its total mass exists at its center of mass. The moments and products of inertia of a point mass are zero, and you need only specify the total mass.

Point Mass Parameters
Mass

Adding a Point Mass to a Model

To position a point mass in a model, connect the block reference frame port (R) to the frame of your choice. A frame port, line, or node represents the frame. The point mass coincides with the origin of this frame. For example, connect the Solid block R frame port to the World Frame block W frame port to represent a point mass that coincides with the World frame origin.



For more information about frames, see "Representing Frames" on page 1-7.

Specifying Point Mass Inertia

To specify the inertial parameters of a point mass:

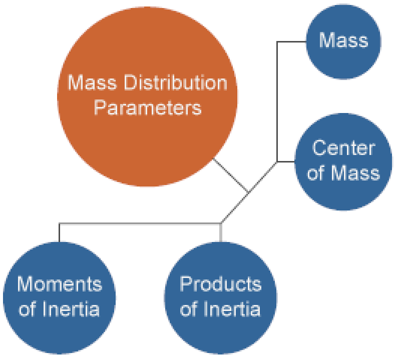
1 In the block dialog box, expand **Inertia**.

2 In **Type**, select `Point Mass`.

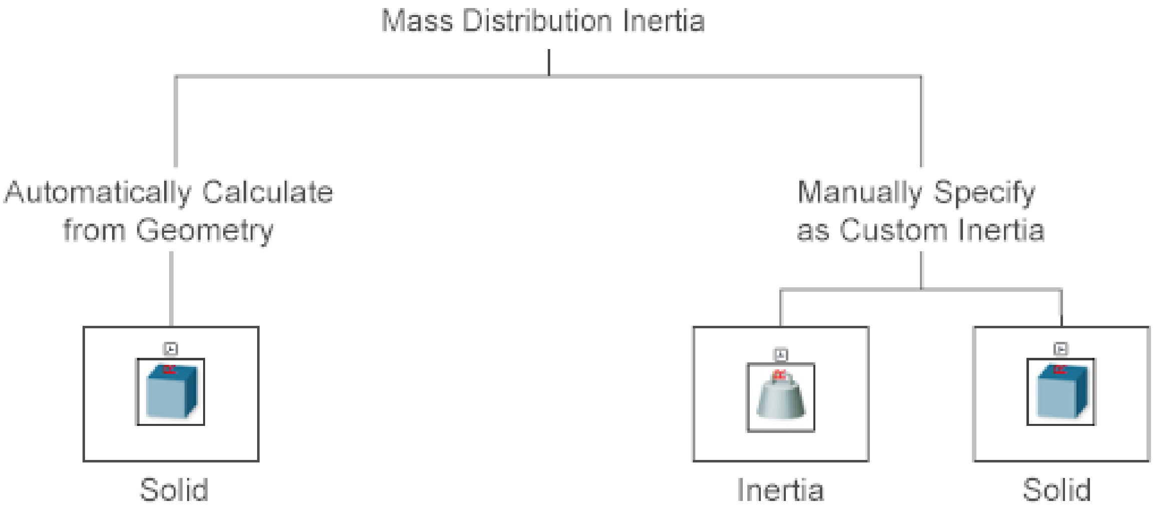
3 In **Mass**, enter the total mass of the solid.

Mass Distribution

A mass distribution occupies a measurable region of space. All rigid bodies are 3-D mass distributions. To completely describe a mass distribution, you specify the total mass, center of mass, moments of inertia, and products of inertia.

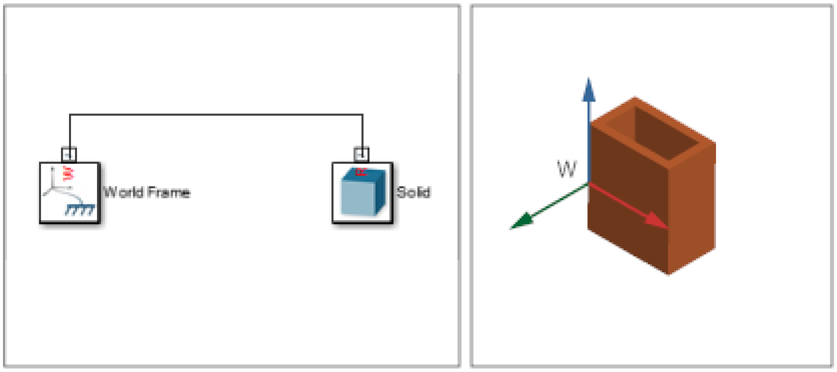


You can use two inertia types to represent a mass distribution. Select `Calculate from Geometry` to automatically calculate the center of mass, moments of inertia, and products of inertia from the solid geometry. Select `Custom` to manually specify all inertial properties. The Solid block provides both inertia types. The Inertia block provides only `Custom`.


Mass Distribution Inertia
Automatically Calculate from Geometry
Manually Specify as Custom Inertia
Solid
Inertia
Solid

Adding a Mass Distribution to a Model

To position a mass distribution, connect the block reference frame port (R) to the frame of your choice. A frame, line, or node represents the frame. The reference frame origin coincides with the origin of this frame. For example, connect the Solid block R frame port to the World Frame block W frame port to represent a 3-D mass distribution whose reference frame origin coincides with the World frame origin.



The center of mass of the solid depends on the inertia type you use. If you select `Custom`, the center of mass depends on the coordinates that you manually specify with respect to the solid reference frame. If you select `Calculate from Geometry`, the center of mass depends on the geometry that you use. For more information, see the Solid block reference page.

Automatically Calculating Inertia

To automatically calculate the center of mass, moments of inertia, and products of inertia of a mass distribution:

1. In the Solid block dialog box, expand **Inertia**.

2. In **Type**, select `Calculate from Geometry`.

3. Specify the remaining parameters as defined in the Solid block reference page.

Note To automatically calculate the inertia of a solid from its geometry, you must specify a valid SimMechanics shape. If you specify a geometry from a file, you must manually enter all inertia parameters using inertia type `Custom`.

Specifying Custom Inertia

To manually specify all inertia parameters of a mass distribution:

1. In the block dialog box, expand **Inertia**.
2. In **Type**, select **Custom**.
3. Specify the remaining parameters as defined in the Solid block reference page.

Related Examples

- "Model Binary Link" on page 2-57
- "Model Pivot Mount" on page 2-77

Inertia Tensor

In this section...

The inertia tensor is a 3×3 matrix that governs the rotational behavior of a rigid body. This matrix is symmetric: elements with reciprocal indices have the same value. That is:

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

Because the inertia tensor is symmetric, it requires only six elements. Three are the moments of inertia and three are the products of inertia. The complete inertia tensor has the form:

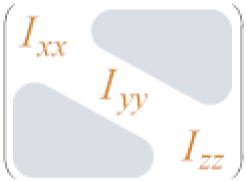
$$\begin{pmatrix} I_{xx} & I_{xy} & I_{zx} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix}$$

Specifying Inertia Tensor

You can specify the inertia tensor manually, using one of two blocks: Solid and Inertia. To do this, in the block dialog box select `Custom` from the **Inertia > Type** drop-down menu. In the new set of parameters that appears, specify the inertia tensor in terms of the moments and products of inertia.

Moments of Inertia

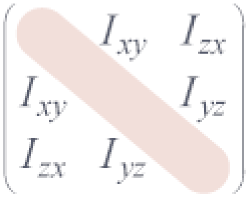
The moments of inertia are the three diagonal terms of the inertia tensor:



In the **Moments of Inertia** dialog box parameter, enter the three diagonal elements as a row vector. Enter the elements in the order [I_{xx}, I_{yy}, I_{zz}]. These are the moments of inertia of the solid with respect to a frame whose axes align with the block reference frame, and whose origin coincides with the solid center of mass.

Products of Inertia

The products of inertia are the three unique off-diagonal elements. Because the inertia tensor is symmetric, each off-diagonal element appears twice in the matrix.



In the **Products of Inertia** dialog box parameter, enter the three unique off-diagonal elements. Enter the elements in the order [I_{yz}, I_{zx}, I_{xy}]. One easy way to remember the element order is to think of the missing subscript component: x, y, and z respectively. The elements are the products of inertia of the solid with respect to a frame whose axes align with the block reference frame, and whose origin coincides with the solid center of mass.

Related Examples

- "Model Binary Link" on page 2-57
- "Model Pivot Mount" on page 2-77

Concepts

- "Specifying Solid Inertia" on page 2-20

Solid Color

In this section...
"Basic Graphic Parameters" on page 2-29
"Advanced Graphic Parameters" on page 2-31

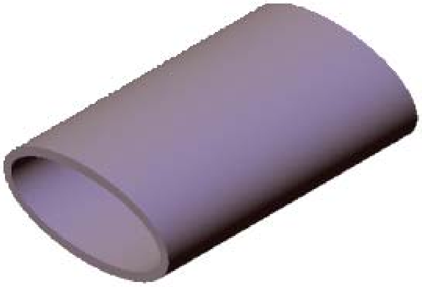
To make the most of the visualization capability of Mechanics Explorer, the Solid block provides two parameterizations that you can use to specify the graphic appearance of a solid: `Simple` and `Advanced`. The two parameterizations accept material color and opacity parameters as input. Light source parameters are fixed for all models. The table provides a comparison of the input parameters present in each graphic parameterization.

Graphic Parameter	`Simple`	`Advanced`
Diffuse Color	✓	✓
Ambient Color		✓
Specular Color		✓
Emissive Color		✓
Opacity	✓	✓
Shininess		✓

As an example, the figure shows two identical elliptical extrusions, one based on `Simple` and the other on `Advanced` graphic parameterizations. In both cases, the extrusion is completely opaque with a gray diffuse color. The advanced version adds to the solid a set of blue highlights, through the use of specular color, and a red ambient hue, through the use of ambient color.

Color Parameter	`Simple`	`Advanced`
Diffuse Color	`[0.8 0.8 0.8]`	`[0.8 0.8 0.8 1.0]`
Ambient Color	—	`[0.1 0.05 0.05 1.0]`
Specular Color	—	`[0 0 1.0 1.0]`

Solid with *Simple* color parameterization.



Solid with *Advanced* color parameterization.

The material colors — diffuse, ambient, specular, and emissive — form the core of the graphical representation of a solid in SimMechanics. You can specify the material colors in terms of “RGBA Color Vectors” on page 2-34.

Basic Graphic Parameters

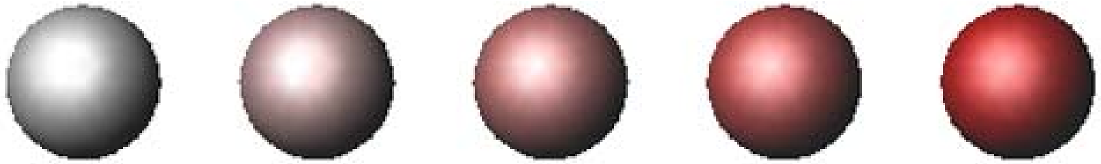
Both `Simple` and `Advanced` graphic parameterizations require you to specify the diffuse color and opacity of the solid. Together, these two parameters represent the graphical core of a SimMechanics solid. The way in which you specify the parameters differs slightly between the two parameterizations, but the meaning of each parameter remains the same.

Diffuse Color

Apparent color of a rough solid surface exposed to direct white light. Diffuse light scatters equally in all directions according to Lambert’s law, causing the intensity and color of the scattered light to appear the same from all angles. The diffuse color normally provides the dominant contribution to the color of a solid surface. In most cases, you can think of the diffuse color as the “true color” of a solid surface.

Parameterization	Parameter Name Used	Specification
`Simple`	**Color**	[R G B] vector
`Advanced`	**Diffuse Color**	[R G B A] vector

The figure shows the effect of varying the diffuse color of a solid. The array of spheres have identical graphical properties, with the exception of **Diffuse Color**. The RGBA color vector of the diffuse color progresses from [1 1 1], at the left corner, to [0.85 0.45 0], at the right corner. A gray ambient color gives the solid a darker appearance.



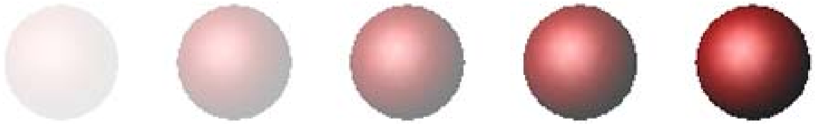
Opacity

The opacity is the degree to which a solid blocks light from passing through. A completely opaque solid blocks all light penetration through the solid. The opposite of a completely opaque solid is a transparent solid, which allows all light to pass through. You can reduce the opacity of a solid in order to improve the visibility of other solids otherwise blocked from view.

Parameterization	Parameter Name Used	Specification
`Simple`	**Opacity**	Scalar number (0–1)
`Advanced`	*A* element of **Diffuse Color** [R G B A] vector	Scalar number (0–1)

The figure shows the effect of varying the opacity of a solid. The array of spheres have identical graphical properties, with the exception of

Opacity. The opacity value progresses from 0.1, at the left corner, to 1, at the right corner. An opacity value of 0 represents a completely transparent, or invisible, solid. An opacity value of 1 represents a completely opaque solid.



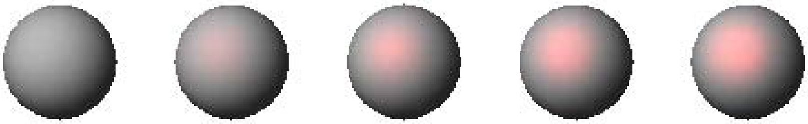
Advanced Graphic Parameters

In addition to the diffuse color and opacity, the `Advanced` parameterization provides a set of colors that enhance the 3–D graphical appearance of the solid. The additional colors include specular, ambient, and emissive colors, each of which includes an opacity (*A*) element in the [R G B A] color vector. You can omit the fourth element in the RGBA vector, in which case the color uses a maximum opacity value of `1`.

Specular Color

The specular color is the apparent color of the glossy highlights arising from a solid surface exposed to direct light. The size of the specular highlights depends on the value of the **Shininess** parameter. The intensity of the specular color is not uniform in space, and has a strong dependence on the viewing angle. Changing the specular color changes only the color of the specular highlights. For most applications, the [R G B A] vector [0.5 0.5 0.5 1] works well.

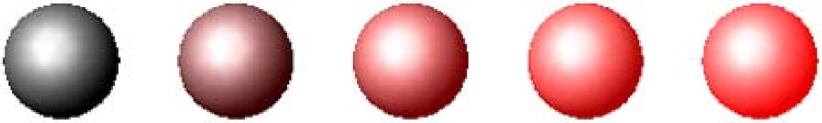
The figure shows the effect of varying the specular color of a solid. The array of spheres have identical graphical properties, with the exception of **Specular Color**. The RGBA color vector of the specular color progresses from [1 1 1 1], at the left corner, to [1 0 0 1], at the right corner. A gray ambient color gives the solid a darker appearance.



Ambient Color

The ambient color is the apparent color of a solid surface exposed only to indirect light. Changing the ambient color changes the overall color of the entire solid surface. For most applications, the RGBA vector [0.15 0.15 0.15 1] works well.

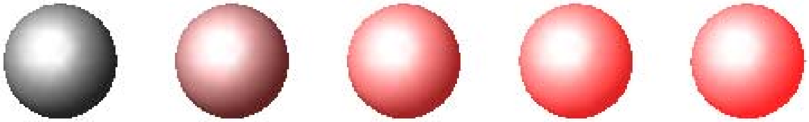
The figure shows the effect of varying the ambient color of a solid. The array of spheres have identical graphical properties, with the exception of the **Ambient Color**. The RGBA color vector of the ambient color progresses from [1 1 1 1], at the left corner, to [1 0 0 1], at the right corner. A gray ambient color gives the solid in the left corner a darker appearance.



Emissive Color

The emissive color is the apparent color of light emitted directly by the solid surface. Examples of solids with a nonzero emissive color include glowing hot metal, light displays, and the Sun. For most applications, the RGBA vector [0 0 0 1] works well.

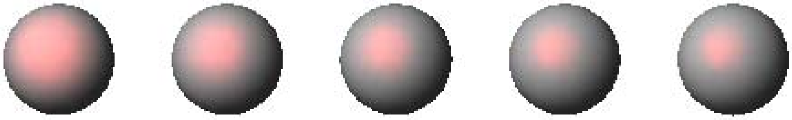
The figure shows the effect of varying the emissive color of a solid. The array of spheres have identical graphical properties, with the exception of the **Emissive Color**. The RGBA color vector of the emissive color progresses from [1 1 1 1], at the left corner, to [1 0 0 1], at the right corner. A gray ambient color gives the solid in the left corner a darker appearance. The glowing appearance of the emissive color differentiates the emissive color from ambient and diffuse colors.



Shininess

The shininess is a parameter that encodes the size and rate of decay of specular highlights on a solid surface. A small shininess value corresponds to a large specular highlight with gradual falloff in highlight intensity. On the other hand, a large shininess value corresponds to a small specular highlight with sharp falloff in highlight intensity.

The figure shows the effect of varying the shininess of a solid. The array of spheres have identical graphical properties, with the exception of **Shininess**. The shininess value progresses from 5, at the left corner, to 25, at the right corner. As the shininess value increases, the area of the specular highlight decreases, while the falloff rate in highlight intensity increases.



Related Examples

- "Model Binary Link" on page 2-57
- "Model Pivot Mount" on page 2-77

RGBA Color Vectors

The Solid block accepts an RGBA vector as input for the material color parameters. The RGBA model is based on three primary colors that you combine to obtain other colors in the spectrum. By varying the proportions of the three primary colors, it is possible to obtain colors throughout most of the visible spectrum. The model obtains its name from the first letter of the three primary colors — red (R), green (G), blue (B). The fourth letter (A) denotes the solid opacity, the degree to which the solid appears impenetrable to incident light.

The color parameters of the Solid block accept either 3- or 4-element vectors specifying the proportions of the primary colors. The 4–element vector has the form [R G B A] and includes a value for the solid opacity. The 3-element vector has the form [R G B], and assumes unity value for the solid opacity.

The values of the four parameters fall in the numerical range `0 1`. Depending on the vector element, a value of `0` indicates that the corresponding primary color is not used to obtain the final color, or that the solid opacity is a minimum (a completely transparent solid). Likewise, a value of `1` indicates that a maximum quantity of the corresponding primary color is used to obtain the final color, or that the solid opacity is a maximum (a completely opaque solid). To convert RGB values in the range `0-255`, divide the [R G B] vector elements by 255 — e.g. [255 0 0 255]/255 for a completely opaque red color.

The RGBA model applies to the four color types used in SimMechanics — diffuse, ambient, specular, and emissive. For more information, see “Solid Color” on page 2-28

Related Examples

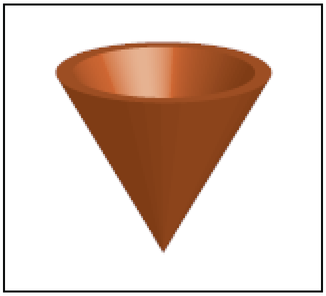
- “Model Binary Link” on page 2-57
- “Model Pivot Mount” on page 2-77

Model Cone

In this section...

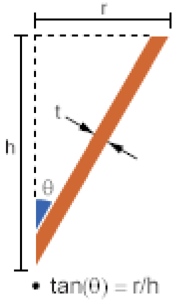
Model Overview

You can model solids of revolution using the SimMechanics `Revolution` shape. Examples of solids of revolution include cone and circular dome shapes. In this example, you model a simple solid with cone shape using the `Revolution` shape. For an example that shows you how to model a circular dome solid, see "Model Dome" on page 2-40.

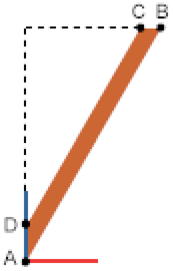


Modeling Approach

To represent the cone geometry, first identify its cross-section shape. This is the 2-D area that you can revolve about an axis to obtain the 3-D cone. Then, specify the cross-section coordinate using the Solid block. For more information, see "Cross-Section Coordinates" on page 2-14. The cone in this example has a trapezoidal cross-section. The figure shows this cross-section.



The [0 0] cross-section coordinate identifies the reference frame origin for this solid. To place the reference frame at the cone tip, specify the coordinates so that the [0 0] coordinate is at the tip. By parameterizing the cross-section coordinates in terms of the relevant cone dimensions, you can quickly change the cone dimensions without having to reenter the cross-section coordinates. The figure shows the cross-section dimensions and coordinates that you must enter to specify the cone.



- A = [0, 0]
- B = [r, h]
- C = [r-t/cos(θ), h]
- D = [0 t/sin(θ), h]

Build Model

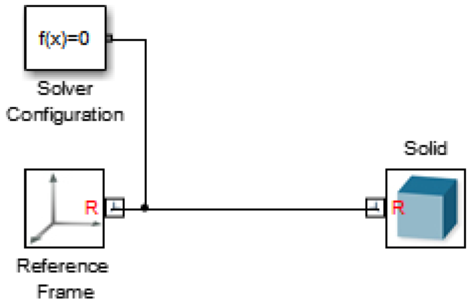
Add and connect the blocks to represent the cone. Include a Solver Configuration block so that you can visualize the solid in Mechanics Explorer during the modeling process.

1 Start a new model.

2 Add the following blocks to the model.

Library	Block	Quantity
Body Elements	Solid	1
Frames & Transforms	Reference Frame	1
Simscape Utilities	Solver Configuration	1

3 Connect the blocks as they appear in the figure.

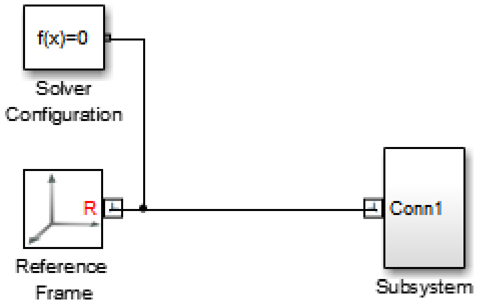


4 Double-click the Solid block.

5 In the dialog box, specify the following parameters.

Parameter	Value
Geometry > Shape	Select `Revolution`
Geometry > Cross-Section	Enter `xsection`. Specify units of `in`.
Graphic > Visual Properties > Color	Enter `rgb`

6 Right-click the Solid block and select **Create Subsystem from Selection**.



Specify Parameter Values

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

1. Click the Subsystem block and press **Ctrl+M** to create a subsystem mask.

2. In the **Parameters & Dialog** tab of the Mask Editor, drag edit boxes inside the **Parameters** group and specify the following parameters.

Prompt	Name
`Base Radius`	`r`
`Cone Height`	`h`
`Wall Thickness`	`t`
`Color`	`rgb`

3. In the **Initialization** tab of the Mask Editor, enter the initialization code for the `xsection` variable.

```
theta = atan(r/h);
xsection = [0 0; r h; r-t/cos(theta) h; 0 t/sin(theta)];
```

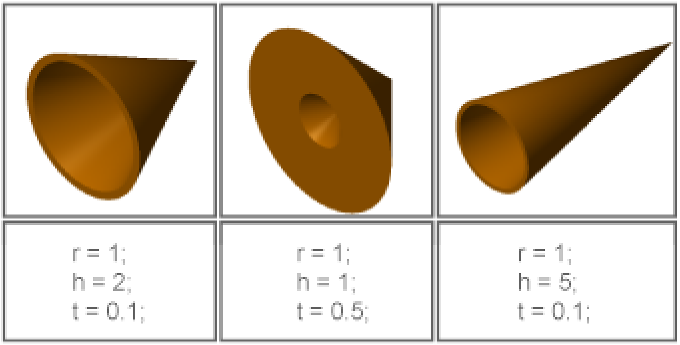
4. In the Subsystem block dialog box, specify these parameters

Parameter	Value
Base Radius	1
Cone Height	2
Wall Thickness	0.1
Color	[0.85 0.45 0]

Visualize Model

You can now visualize the cone that you modeled. To do this, on the Simulink menu bar, select **Simulation > Update Diagram**. Mechanics Explorer opens with a 3-D display of your model. Rotate, pan, and zoom to explore.

Try modifying the cone geometry. To do this, in the subsystem dialog box, change the dimension parameter values. Then, update the model. The figure shows some examples.



Related Examples

- “Model Dome” on page 2-40
- “Model I-Beam” on page 2-45
- “Model Box Beam” on page 2-51

Concepts

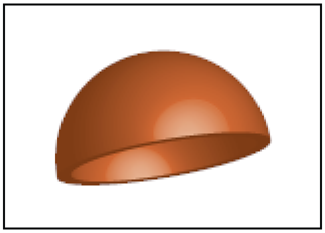
- “Advanced Solid Shapes” on page 2-7
- “Cross-Section Coordinates” on page 2-14
- “Revolution and General Extrusion Cross-Sections” on page 2-10

Model Dome

In this section...

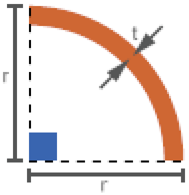
Model Overview

You can model a solid of revolution with a round cross-section. One example is the circular dome. In this example, you specify the cross-section coordinates of a circular dome using the MATLAB `cos` and `sin` functions. For an example that shows you how to model a cone-shaped solid, see "Model Cone" on page 2-35.

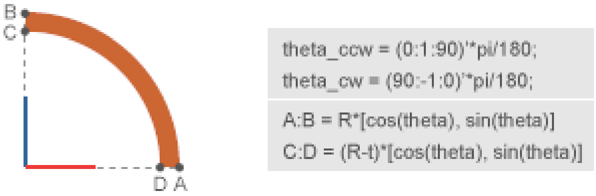


Modeling Approach

To represent the dome geometry, first identify its cross-section shape. This is the 2-D shape that you revolve about an axis to obtain the 3-D dome. You can then specify the cross-section coordinates using the Solid block. For more information, see "Cross-Section Coordinates" on page 2-14. The figure shows the cross-section shape for the dome in this example.



The [0 0] cross-section coordinate identifies the reference frame origin for this solid. To place the reference frame at dome base center, specify the coordinates so that the [0 0] coordinate is at 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 cross-section dimensions and coordinates that you must enter to specify the dome.



```
theta_ccw = (0:1:90)'*pi/180;
theta_cw = (90:-1:0)'*pi/180;
A:B = R*[cos(theta), sin(theta)]
C:D = (R-t)*[cos(theta), sin(theta)]
```

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 `cos` and `sin` MATLAB functions.

In this example, you name the counterclockwise angle array `theta_ccw`. You name the clockwise angle array `theta_cw`. You also name the outer cross-section coordinate array `outer_coords` and the inner cross-section coordinate array `inner_coords`. In the figure, points A through B denote the outer cross-section coordinates, and points C through D denote the inner cross-section coordinates.

Build Model

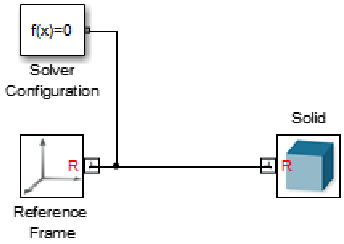
Add and connect the blocks to represent the dome. Include a Solver Configuration block so that you can visualize the solid in Mechanics Explorer during the modeling process.

1 Start a new model.

2 Add the following blocks to the model.

Library	Block	Quantity
Body Elements	Solid	1
Frames & Transforms	Reference Frame	1
Simscape Utilities	Solver Configuration	1

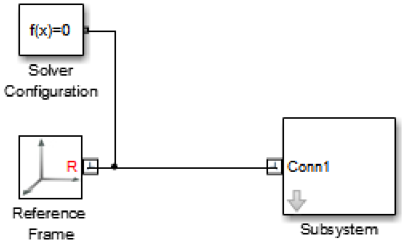
3 Connect the blocks as they appear in the figure.



4 Double-click the Solid block and specify the following parameters.

Parameter	Value
Geometry > Shape	Select `Revolution`
Geometry > Cross-Section	Enter `xsection`. Specify units of `in`.
Graphic > Visual Properties > Color	Enter `rgb`

5 Right-click the Solid block and select **Create Subsystem from Selection**.



Specify Parameter Values

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

1 Click the Subsystem block and press **Ctrl+M** to create a subsystem mask.

2 In the **Parameters & Dialog** tab of the Mask Editor, drag edit boxes inside the **Parameters** group and specify the following parameters.

Prompt	Name
`Radius`	`r`
`Wall Thickness`	`t`

3 In the **Initialization** tab of the Mask Editor, enter the initialization code for the xsection variable.

```
% Circular dome outer coordinates:
theta_ccw = (0:1:90)'*pi/180;
outer_coords = r*[cos(theta_ccw), sin(theta_ccw)];

% Circular dome inner coordinates:
theta_cw = (90:-1:0)'*pi/180;
inner_coords = (r-t)*[cos(theta_cw), sin(theta_cw)];

xsection = [outer_coords; inner_coords];
```

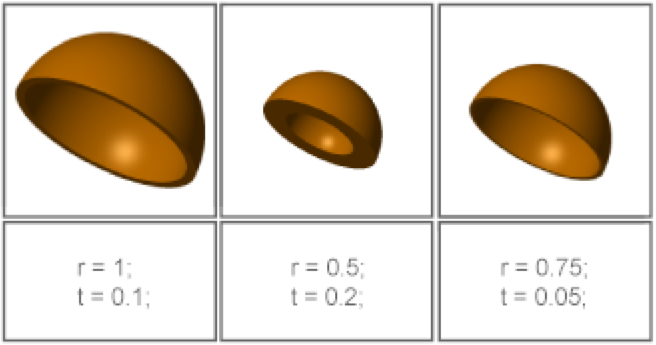
4 In the Subsystem block dialog box, specify these parameters.

Parameter	Value
Radius	`1`
Wall Thickness	`0.1`

Visualize Model

You can now visualize the dome that you modeled. To do this, on the Simulink menu bar, select **Simulation > Update Diagram**. Mechanics Explorer opens with a 3-D display of your model. Rotate, pan, and zoom to explore.

Try modifying the dome geometry. To do this, in the subsystem dialog box, change the dimension parameter values. Then, update the model. The figure shows some examples.



Related Examples

- "Model Cone" on page 2-35
- "Model I-Beam" on page 2-45
- "Model Box Beam" on page 2-51

Concepts

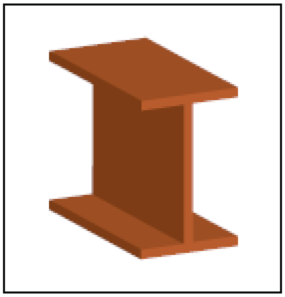
- "Advanced Solid Shapes" on page 2-7
- "Cross-Section Coordinates" on page 2-14
- "Revolution and General Extrusion Cross-Sections" on page 2-10

Model I-Beam

In this section...

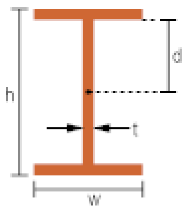
Model Overview

You can model an extrusion using the SimMechanics shape `General Extrusion`. Examples of extrusions include the I-beam and box-beam shapes. In this example, you model a simple solid with I-beam shape using the `General Extrusion` shape. For an example that shows you how to model a box beam, see "Model Box Beam" on page 2-51.



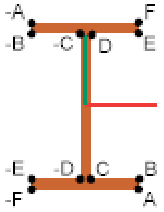
Modeling Approach

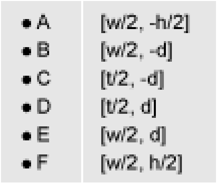
To represent the I-beam geometry, first identify its cross-section. This is the 2-D area that you sweep along an axis to obtain the 3-D I-beam. You can then specify the cross-section coordinates using the Solid block. The figure shows the I-beam cross-section that you specify in this example.



The [0 0] coordinate identifies the solid reference frame origin. To place the reference frame at the center of the I-beam, specify the coordinates so that the [0 0] coordinate is at the cross-section center. Because the I-beam cross-section is symmetric about the vertical axis, you need only define the coordinates for one cross-section half—e.g, the right half. You can then define the left half coordinates in terms of the right half coordinates.

By parameterizing the cross-section coordinates in terms of relevant I-beam dimensions, you can quickly change the I-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 I-beam.





● A	[w/2, -h/2]
● B	[w/2, -d]
● C	[t/2, -d]
● D	[t/2, d]
● E	[w/2, d]
● F	[w/2, h/2]

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

```
right_half = [A; B; C; D; E; F];
xsection = [right_half; -right_half];
```

Build Model

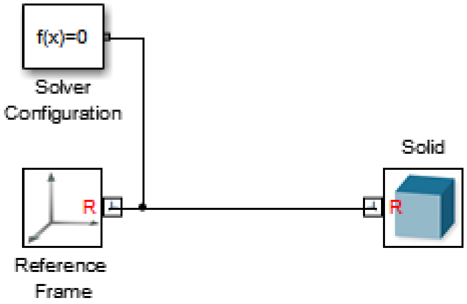
Add and connect the blocks to represent the I-beam. Include a Solver Configuration block so that you can visualize the solid in Mechanics Explorer during the modeling process.

1 Start a new model.

2 Add the following blocks to the model.

Library	Block	Quantity
Body Elements	Solid	1
Frames & Transforms	Reference Frame	1
Simscape Utilities	Solver Configuration	1

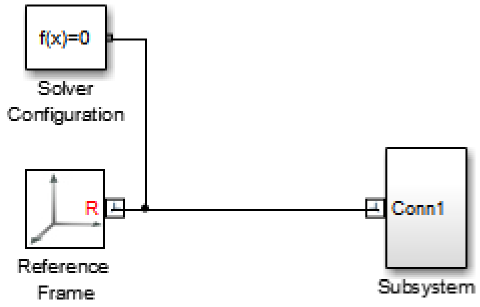
3 Connect the blocks as they appear in the figure.



4 Double-click the Solid block and specify the following parameters.

Parameter	Value
Geometry > Shape	Select `General Extrusion`
Geometry > Cross-Section	Enter `xsection`. Select units of `in`.
Geometry > Length	Enter `L`. Select units of `in`
Graphic > Visual Properties > Color	Enter `rgb`

5 Right-click the Solid block and select **Create Subsystem from Selection**.



Specify Parameter Values

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

1 Click the Subsystem block and press **Ctrl+M** to create a subsystem mask.

2 In the **Parameters & Dialog** tab of the Mask Editor, drag edit boxes inside the **Parameters** group and specify the following parameters.

Prompt	Name
`Length`	`L`
`Height`	`h`
`Width`	`w`
`Thickness`	`t`
`Color`	`rgb`

3 In the **Initialization** tab of the Mask Editor, enter the initialization code for the `xsection` variable and click **OK**:

```
d = h/2-t;
right_half = [w/2, -h/2; w/2, -d; t/2, -d;
t/2, d; w/2, d; w/2, h/2];
xsection = [right_half; -right_half];
```

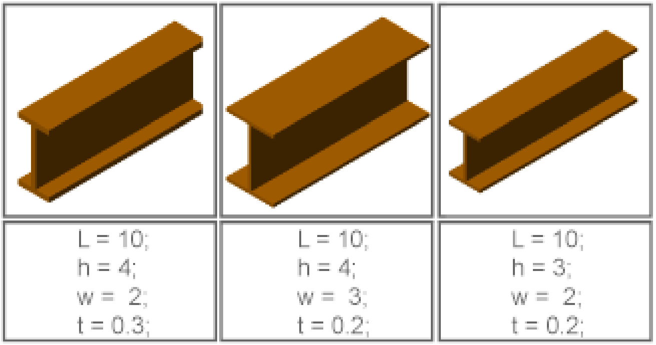
4 In the Subsystem block dialog box, specify these parameters.

Parameter	Value
Length	`10`
Height	`4`
Width	`2`
Thickness	`0.3`
Color	`[0.85 0.45 0]`

Visualize I-Beam in Mechanics Explorer

You can now visualize the I-beam that you modeled. To do this, on the Simulink menu bar, select **Simulation > Update Diagram**. Mechanics Explorer opens with a 3-D display of your model. Rotate, pan, and zoom to explore.

Try modifying the I-beam geometry. To do this, in the subsystem dialog box, change the dimension parameter values. Then, update the model. The figure shows some examples.



Related Examples

- "Model Cone" on page 2-35
- "Model Dome" on page 2-40
- "Model Box Beam" on page 2-51

Concepts

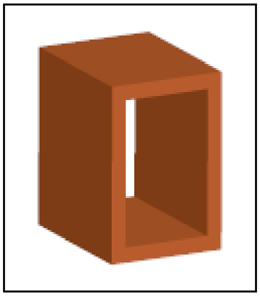
- “Advanced Solid Shapes” on page 2-7
- “Cross-Section Coordinates” on page 2-14
- “Revolution and General Extrusion Cross-Sections” on page 2-10

Model Box Beam

In this section...

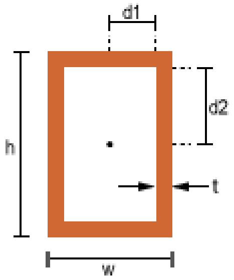
Model Overview

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 "Cross-Section Coordinates" on page 2-14. In this example, you specify the cross-section coordinates of a box beam. For an example that shows you how to model an I-beam extrusion, see "Model I-Beam" on page 2-45.

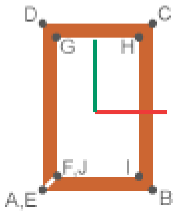


Modeling Approach

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 the 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 quickly 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.



• A,E	[-w/2, -h/2]
• B	[w/2, -h/2]
• C	[w/2, h/2]
• D	[-w/2, h/2]
• F,J	[-d1, -d2]
• G	[-d1, d2]
• H	[d1, d2]
• I	[d1, -d2]

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

```
d1 = w/2-t;
d2 = h/2-t;
outer_path = [-w/2,-h/2; w/2,-h/2; w/2,h/2; ...
-w/2,h/2; -w/2,-h/2];
inner_path = [-d1,-d2; -d1,d2; d1,d2; d1 -d2; -d1,-d2];
xsection = [outer_path; inner_path];
```

For more information about specifying the hollow cross-section coordinates, see “Hollow Cross-Sections” on page 2-16.

Build Model

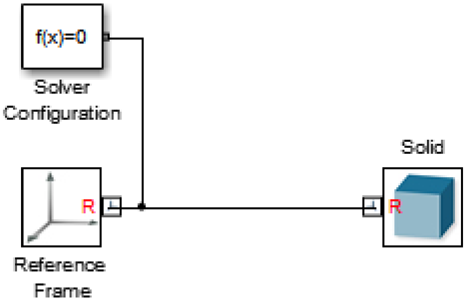
Add and connect the blocks to represent the box beam. Include a Solver Configuration block so that you can visualize the solid in Mechanics Explorer during the modeling process.

1 Start a new model.

2 Add the following blocks to the model.

Library	Block	Quantity
Body Elements	Solid	1
Frames & Transforms	Reference Frame	1
Simscape Utilities	Solver Configuration	1

3 Connect the blocks as they appear in the figure.

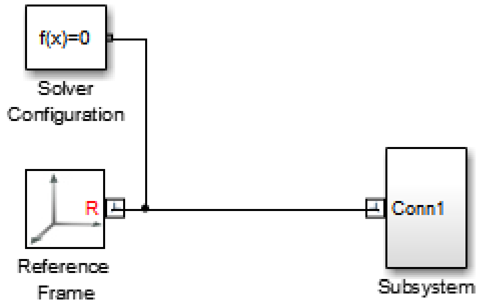


4 Double-click the Solid block and specify the following parameters.

Parameter	Value
Geometry > Shape	Select `General Extrusion`
Geometry > Cross-Section	Enter `xsection`. Select units of `in`.

Parameter	Value
Geometry > Length	Enter `L`. Select units of `in`
Graphic > Visual Properties > Color	Enter `rgb`

5 Right-click the Solid block and select **Create Subsystem from Selection**.



Specify Parameter Values

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

1 Click the Subsystem block and press **Ctrl+M** to create a subsystem mask.

2 In the **Parameters & Dialog** tab of the Mask Editor, drag edit boxes inside the **Parameters** group and specify the following parameters.

Prompt	Name
`Length`	`L`
`Height`	`h`
`Width`	`w`
`Thickness`	`t`
`Color`	`rgb`

3 In the **Initialization** tab of the Mask Editor, enter the initialization code for the `xsection` variable and click **OK**:

```
d1 = w/2-t;
d2 = h/2-t;
outer_path = [-w/2,-h/2; w/2,-h/2; w/2,h/2; ...
-w/2,h/2; -w/2,-h/2];
inner_path = [-d1,-d2; -d1,d2; d1,d2; d1 -d2; -d1,-d2];
xsection = [outer_path; inner_path];
```

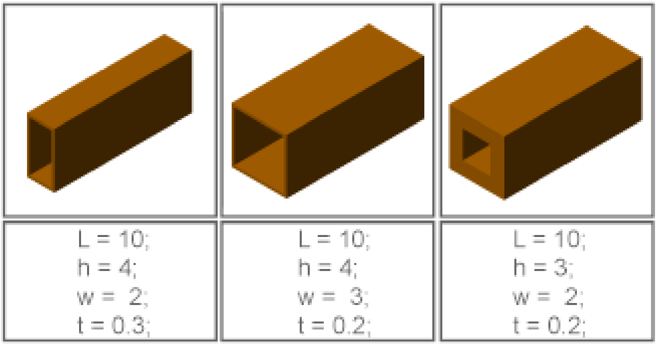
4 In the Subsystem block dialog box, specify these parameters.

Parameter	Value
Length	`10`
Height	`4`
Width	`2`
Thickness	`0.2`
Color	`[0.85 0.45 0]`

Visualize Box Beam in Mechanics Explorer

You can now visualize the box beam that you modeled. To do this, on the Simulink menu bar, select **Simulation > Update Diagram**. Mechanics Explorer opens with a 3-D display of your model. Rotate, pan, and zoom to explore.

Try modifying the box beam geometry. To do this, in the subsystem dialog box, change the dimension parameter values. Then, update the model. The figure shows some examples.



Related Examples

- “Model I-Beam” on page 2-45
- “Model Cone” on page 2-35
- “Model Dome” on page 2-40

Concepts

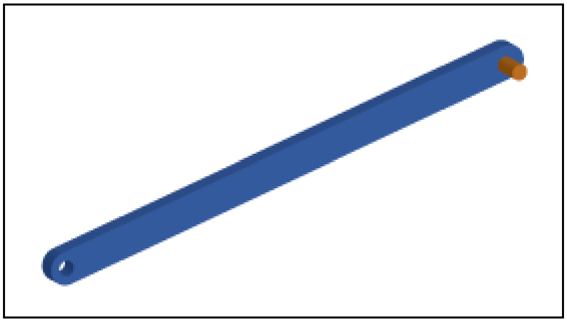
- “Advanced Solid Shapes” on page 2-7
- “Cross-Section Coordinates” on page 2-14
- “Revolution and General Extrusion Cross-Sections” on page 2-10

Model Binary Link

In this section...

Model Overview

In example "Represent Binary Link Frame Tree" on page 1-36, you modeled the frame tree of a binary link rigid body. In this example, you add to that frame tree the solid properties of the binary link: geometry, inertia, and color.



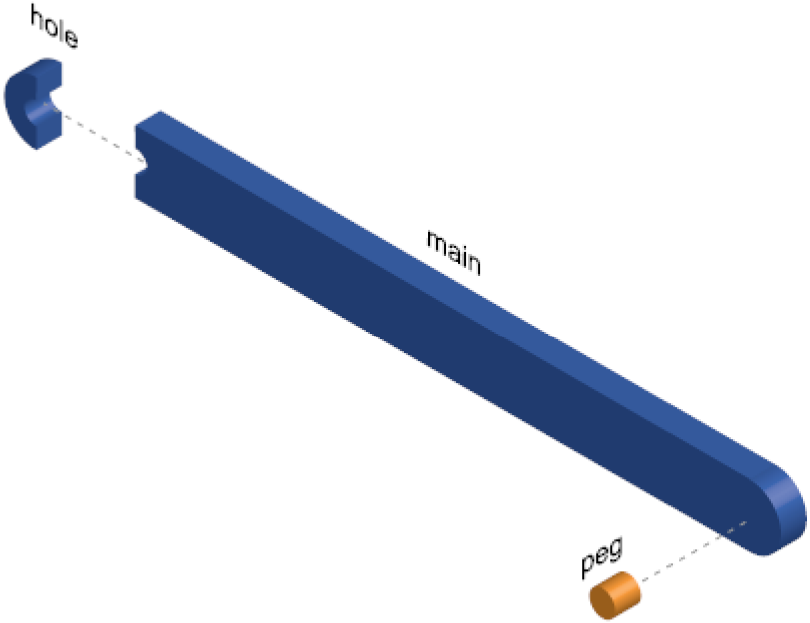
Modeling Approach

To model a binary link, you must use multiple Solid blocks. Each Solid block represents an elementary portion of the binary link. Rigid bodies that you model using multiple Solid blocks are called *compound rigid bodies*. The compound rigid body technique reduces a single complex task (modeling the entire binary link shape) into several simple tasks (modeling the main, hole, and peg sections of the binary link).

To use the compound rigid body technique:

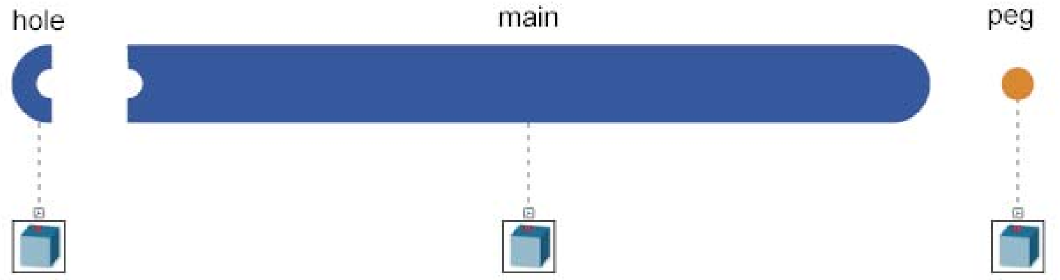
1 Divide shape into simple sections.

   Dividing the shape simplifies the modeling task in more complex cases. You can divide the binary link into three simple sections: main, peg, and hole, shown in the figure.



2 Represent each section with a Solid block.

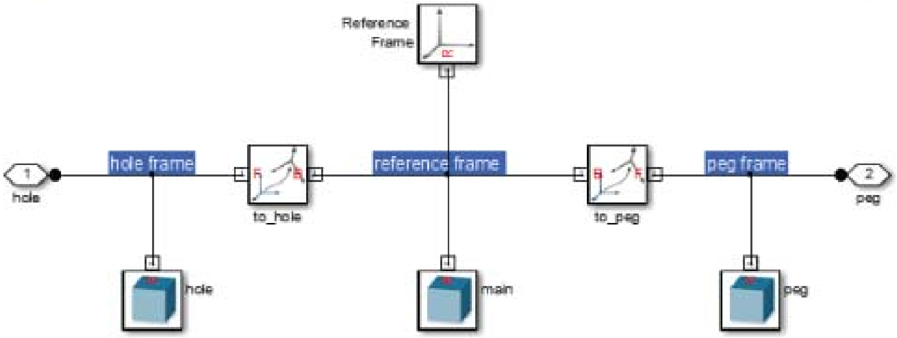
   Each section should be simple enough to model using a single Solid block. In the binary link example, you can represent sections main and hole using SimMechanics shape `General Extrusion`, and section peg with SimMechanics shape `Cylinder`.



3 Rigidly connect Solid blocks to rigid body frame tree.

Rigid connections ensure the different solid sections move as a single rigid body. Connect the Solid blocks to the binary link frame tree to apply the correct spatial relationships between the solid sections.

hole
main
peg



Solid Properties

You model the binary link as a compound rigid body subsystem. In this subsystem, three Solid blocks represent the basic solid sections of the binary link. Each solid section has a shape and a local reference frame that you connect to the binary link frame tree. Two SimMechanics shapes are used: `General Extrusion` and `Cylinder`.

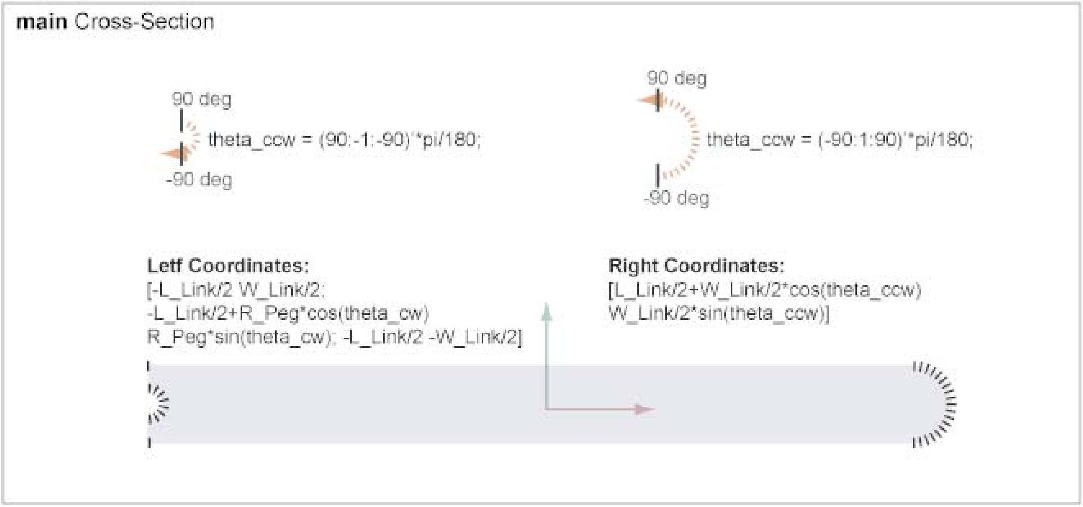
You can promote subsystem reusability by parameterizing solid properties in terms of MATLAB variables. In this example, you initialize the variables in a subsystem mask. You can then specify their numerical values in the subsystem dialog box. The table provides the dimensions needed to model the binary link solid sections. In the previous example, "Represent Binary Link Frame Tree" on page 1-36, you used the first three dimensions to specify the spatial relationships between the different binary link frames.

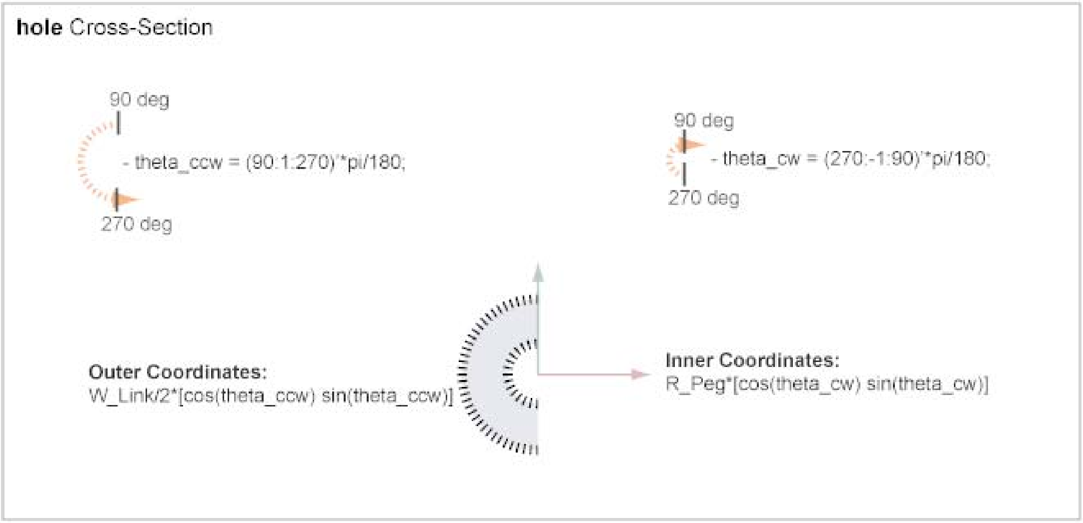
Dimension	MATLAB Variable
Length	`L_Link`
Width	`W_Link`

Dimension	MATLAB Variable
Thickness	T_Link
Peg Radius	R_Peg

SimMechanics shape General Extrusion requires you to specify a set of cross-section coordinates. This is a MATLAB matrix with all the [X Y] coordinate pairs needed to draw the cross-section. Straight line segments connect adjacent coordinate pairs.

Coordinate matrices must obey a set of rules. The most important rule is that the solid region must lie to the left of the line segment connecting adjacent coordinate pairs. For more information, see "Cross-Section Coordinates" on page 2-14. The figure shows the coordinates required to specify the cross-section shapes of solid sections main and hole.


main Cross-Section
90 deg
theta_ccw = (90:-1:-90)'*pi/180;
-90 deg
90 deg
theta_ccw = (-90:1:90)'*pi/180;
-90 deg
Letf Coordinates:
[-L_Link/2 W_Link/2;
-L_Link/2+R_Peg*cos(theta_cw)
R_Peg*sin(theta_cw); -L_Link/2 -W_Link/2]
Right Coordinates:
[L_Link/2+W_Link/2*cos(theta_ccw)
W_Link/2*sin(theta_ccw)]


hole Cross-Section
90 deg
- theta_ccw = (90:1:270)'*pi/180;
270 deg
90 deg
- theta_cw = (270:-1:90)'*pi/180;
270 deg
Outer Coordinates:
W_Link/2*[cos(theta_ccw) sin(theta_ccw)]
Inner Coordinates:
R_Peg*[cos(theta_cw) sin(theta_cw)]

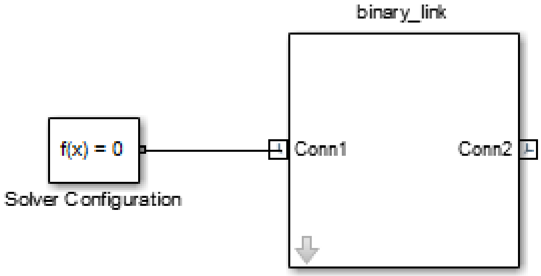
This example assumes the binary link is made of Aluminum, with a mass density of 2,700 kg/m^3. The binary link has a blue color, while the peg has an orange color. The orange color helps identify the peg when, in subsequent examples, you connect the peg of one link to the hole of another link. As with all parameters in this example, you specify density and color in terms of MATLAB variables. The table summarizes the variables and the values that you use in this example.

Solid Sections: Property	**MATLAB Variable**	**Value**
main/peg/hole: **Density**	`rho`	`2700`
main/hole: **Color**	`rgb_link`	`[0.25 0.4 0.7]`
peg: **Color**	`rgb_peg`	`[1 0.6 0.25]`

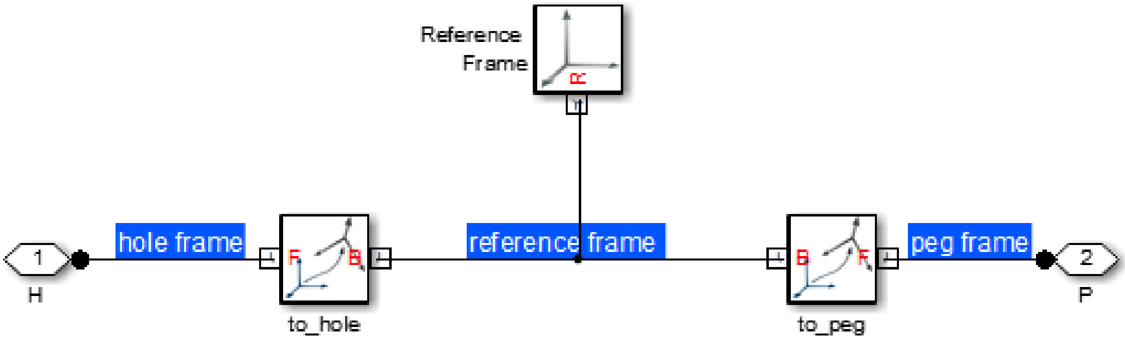
Build Model

Drag blocks into the model canvas and specify the relevant block parameters.

1 Open the frame tree model you built in example “Represent Binary Link Frame Tree” on page 1-36.

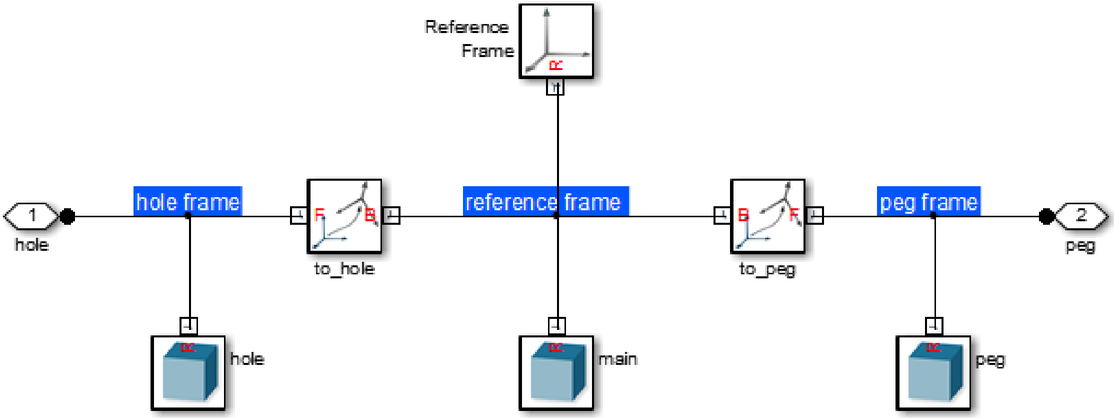


2 Open the binary_link subsystem.



3 From the SimMechanics Body Elements library drag three Solid blocks into the model.

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



5 In the Solid block dialog boxes, specify these parameters.

Parameter	hole	main	peg
Geometry > Shape	Select General Extrusion	Select General Extrusion	Select Cylinder
Geometry > Cross-section	Enter hole_coords	Enter main_coords	—
Geometry > Radius	—	—	Enter R_Peg
Geometry > Length	Enter T_Link	Enter T_Link	Enter 2*T_Link
Geometry > Density	Enter rho	Enter rho	Enter rho
Graphic > Color	Enter rgb_link	Enter rgb_Link	Enter rgb_Peg

Update Subsystem

In the subsystem mask, initialize the MATLAB variables you entered for the block parameters.

1 Select the subsystem block and press **Ctrl+M** to create a subsystem mask.

2 In the **Parameters & Dialog** tab of the Mask Editor, drag four edit boxes into the **Parameters** group and specify these parameters.

Prompt	Name
Peg Radius	R_Peg
Mass Density	rho
Link Color [R G B]	rgb_Link
Peg Color [R G B]	rgb_Peg

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

```
% Cross-section of main:
theta_ccw = (-90:1:90)'*pi/180;
theta_cw = (90:-1:-90)'*pi/180;
peg_end = [L_Link/2+W_Link/2*cos(theta_ccw)...
W_Link/2*sin(theta_ccw)];
hole_end = [-L_Link/2 W_Link/2; ...
-L_Link/2+R_Peg*cos(theta_cw)...
R_Peg*sin(theta_cw); -L_Link/2 -W_Link/2];
main_coords = [peg_end; hole_end];

% Cross-section of hole:
theta_ccw = (90:1:270)'*pi/180;
theta_cw = (270:-1:90)'*pi/180;
hole_coords = [W_Link/2*cos(theta_ccw) ...
W_Link/2*sin(theta_ccw);
R_Peg*cos(theta_cw) R_Peg*sin(theta_cw)];
```

4 In the binary_link subsystem block dialog box, specify these parameters.

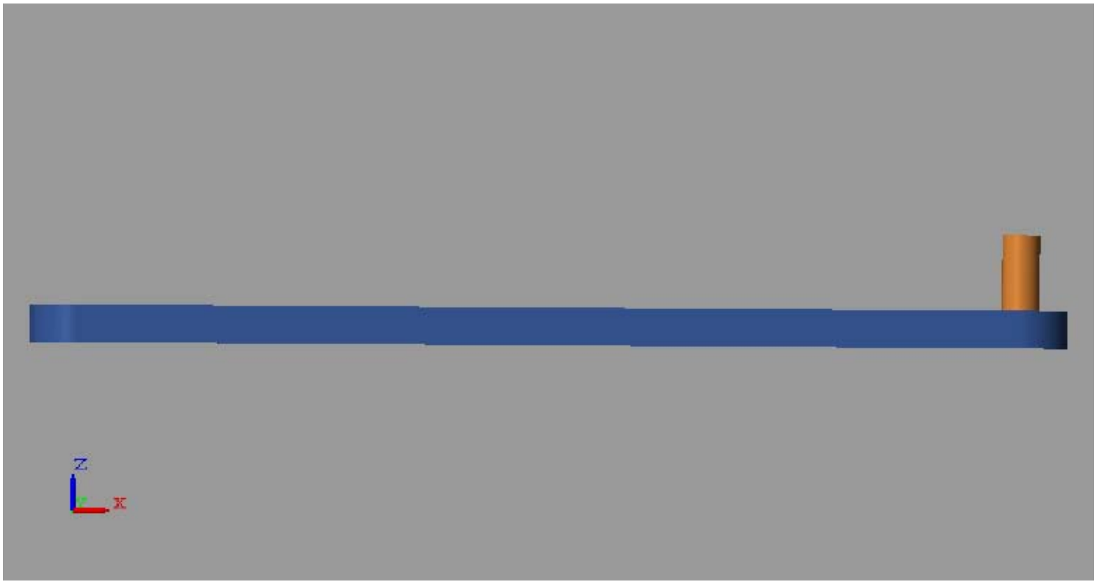
Parameter	Value
Length	0.2
Width	0.02
Thickness	0.008
Peg Radius	0.004
Mass Density	2700
Link Color [R G B]	[0.25 0.4 0.7]
Peg Color [R G B]	[1 0.6 0.25]

Note Values are in the default physical units: m for length quantities and kg/(m^3) for mass density.

Visualize Model

Update the model to visualize the binary link rigid body in Mechanics Explorer.

- Press **Ctrl+D** to update the diagram. The binary link appears in the visualization pane of Mechanics Explorer.

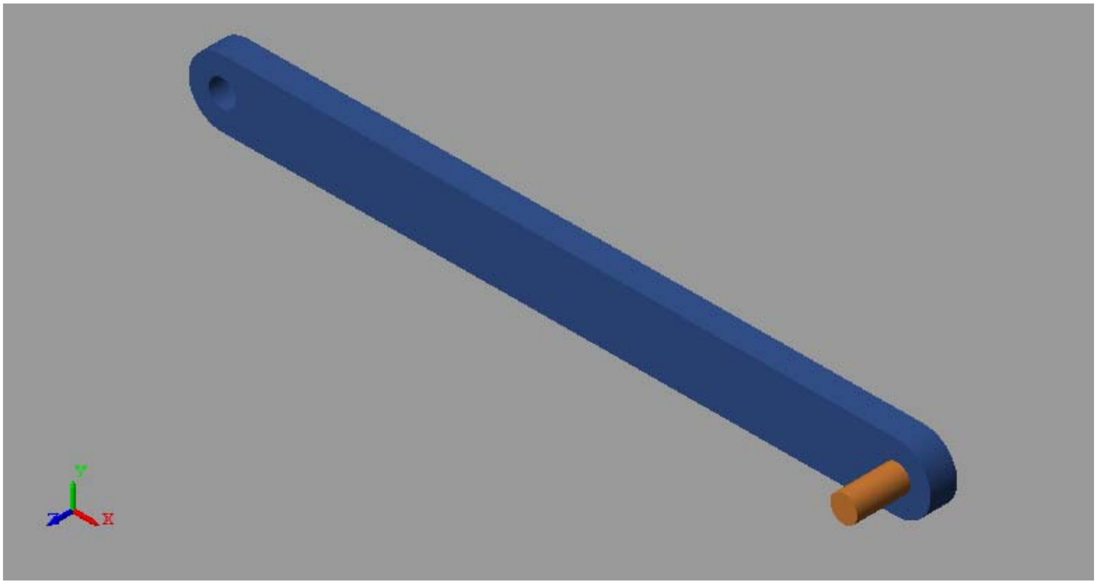


To obtain the view used in the illustrations for this example:

1 In the **View Convention** drop-down list, select `Y Up (XY Front)`.

2 In the Mechanics Explorer toolstrip, click the isometric view button .

Compare the result with the example schematics to confirm the validity of the solid properties specified.



Save Custom Library Block

So that you can use it in later examples, save the binary link subsystem as a custom library block.

1. Open the custom block library that you created in “Represent Binary Link Frame Tree” on page 1-36
2. Drag the binary_link subsystem block to the library.
3. Save the library as linkage_elements.

Related Examples

- “Model Two-Hole Binary Link” on page 2-69
- “Model Pivot Mount” on page 2-77
- “Model Double Pendulum” on page 3-29
- “Model Four-Bar Linkage” on page 3-37

Concepts

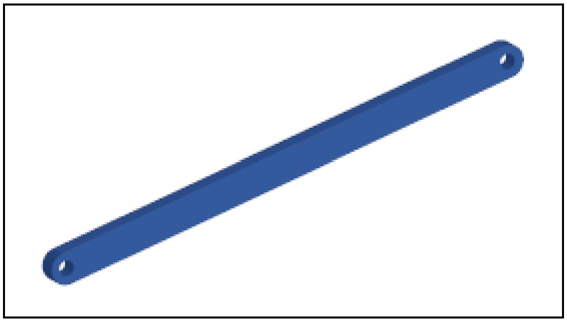
- “Representing Frames” on page 1-7
- “Specifying Solid Geometry” on page 2-2
- “Solid Color” on page 2-28
- “Cross-Section Coordinates” on page 2-14

Model Two-Hole Binary Link

In this section...

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. This example is a variation of “Model Binary Link” on page 2-57.



Build Model

Drag blocks onto a new model and specify their parameters:

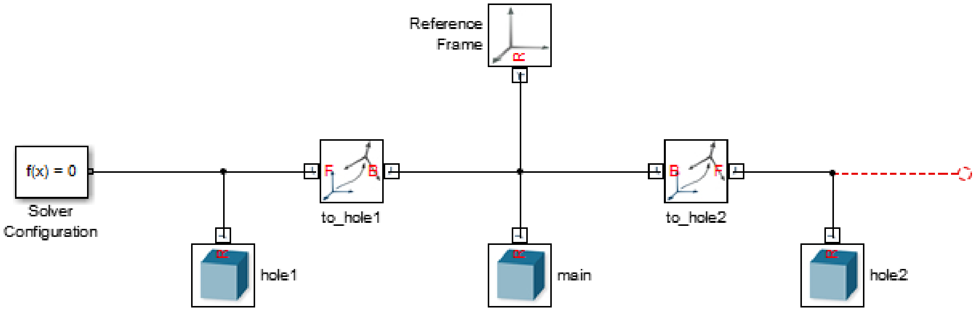
1 Start a new model.

2 Drag the following blocks to the model.

Block	Library	Quantity
Solver Configuration	**Simscape > Utilities**	1
Reference Frame	**SimMechanics Second Generation (SM 2G) > Frames and Transforms**	1
Rigid Transform	**SimMechanics Second Generation (SM 2G) > Frames and Transforms**	2
Solid	**SimMechanics Second Generation (SM 2G) > Body Elements**	3

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

Be sure to flip the to_hole1 block. Its B frame port should face the main Solid block.



4 For the Solid blocks, specify these parameters.

Parameter	**hole1**	**main**	**hole2**
Geometry > Shape	Select `General Extrusion`	Select `General Extrusion`	Select `General Extrusion`
Geometry > Cross-section	Enter `hole1_coords`	Enter `main_coords`	Enter `hole2_coords`
Geometry > Length	Enter `T_Link`	Enter `T_Link`	Enter `T_Link`
Inertia > Density	Enter `rho`	Enter `rho`	Enter `rho`
Graphic > Visual Properties > Color	Enter `rgb_link`	Enter `rgb_link`	Enter `rgb_link`

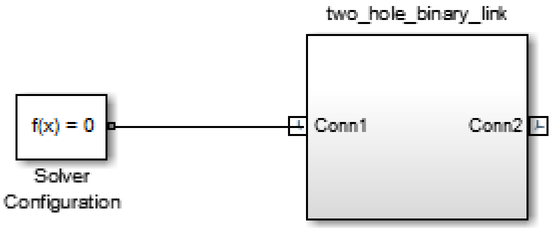
5 For the Rigid Transform blocks, specify these parameters.

Parameter	**to_hole1**	**to_hole2**
Translation > Method	`Standard Axis`	`Standard Axis`
Translation > Axis	`+X`	`+X`
Translation > Offset	`L_Link/2`	`+L_Link/2`

Generate Subsystem

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

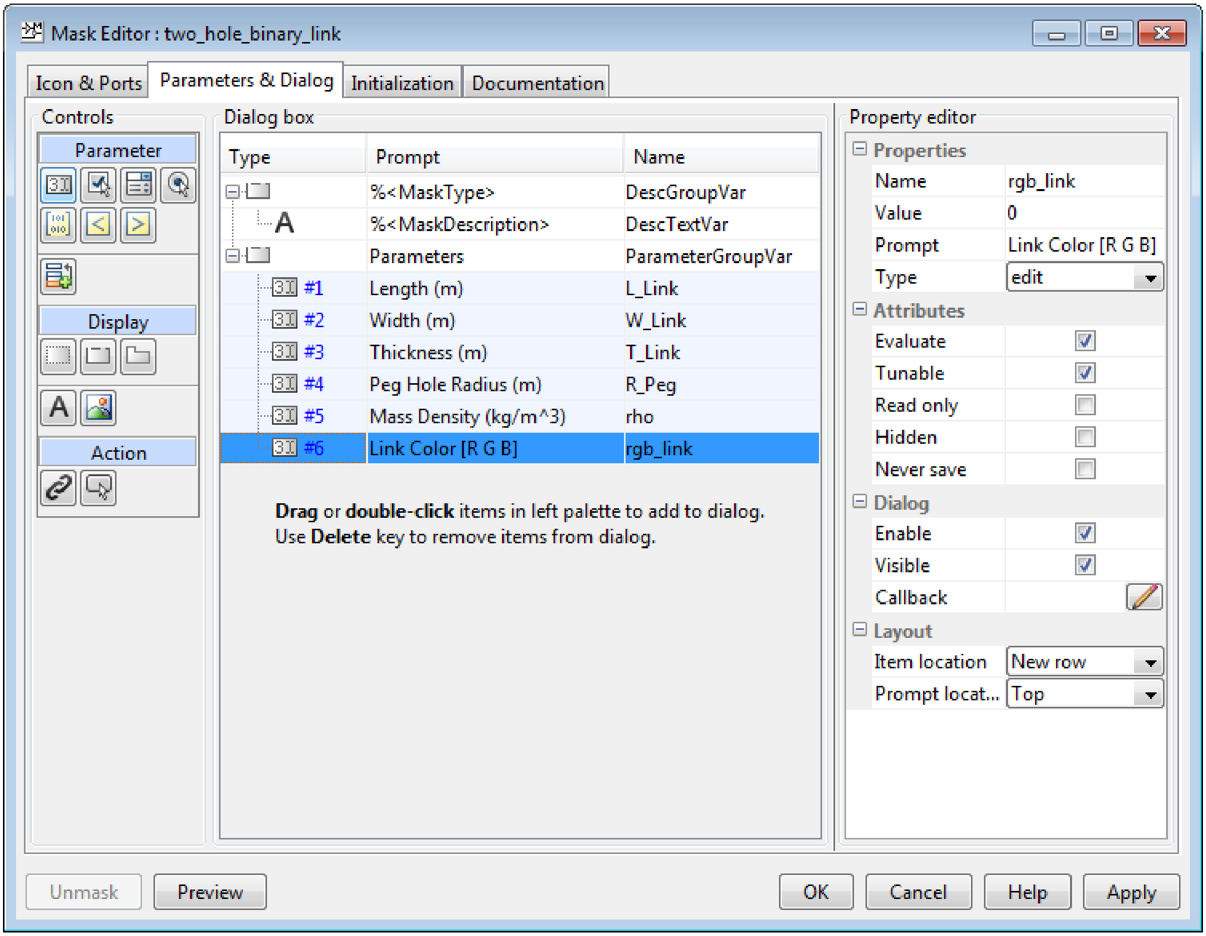
1 Select all blocks except Solver Configuration and press **Ctrl+G.**. Rename the subsystem block two_hole_binary_link for use in subsequent example.



2 Select the subsystem block and press **Ctrl+M** to create a subsystem mask.

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

Prompt	Name
`Length (m)`	`L_Link`
`Width (m)`	`W_Link`
`Thickness (m)`	`T_link`
`Peg Hole Radius (m)`	`R_Peg`
`Mass Density (kg/m^3)`	`rho`
`Link Color [R G B]`	`rgb_Link`



Type	Prompt	Name
	%<MaskType>	DescGroupVar
A	%<MaskDescription>	DescTextVar
	Parameters	ParameterGroupVar
#1	Length (m)	L_Link
#2	Width (m)	W_Link
#3	Thickness (m)	T_Link
#4	Peg Hole Radius (m)	R_Peg
#5	Mass Density (kg/m^3)	rho
#6	Link Color [R G B]	rgb_link

Drag or **double-click** items in left palette to add to dialog.
Use **Delete** key to remove items from dialog.

Properties	
Name	rgb_link
Value	0
Prompt	Link Color [R G B]
Type	edit
Attributes	
Evaluate	☑
Tunable	☑
Read only	□
Hidden	□
Never save	□
Dialog	
Enable	☑
Visible	☑
Callback	
Layout	
Item location	New row
Prompt locat...	Top

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

```
% Cross-section of main:
theta_hole1 = (90:-1:-90)'*pi/180;
theta_hole2 = (270:-1:90)'*pi/180;

hole1_end = [-L_Link/2 W_Link/2;...
-L_Link/2+R_Peg*cos(theta_hole1)...
R_Peg*sin(theta_hole1); -L_Link/2 -W_Link/2];
```

```
hole2_end = [L_Link/2 -W_Link/2;...
L_Link/2+R_Peg*cos(theta_hole2)...
R_Peg*sin(theta_hole2); L_Link/2 W_Link/2];

main_coords = [hole1_end; hole2_end];

% Cross-section of hole1:
theta_ccw = (90:1:270)'*pi/180;
theta_cw = (270:-1:90)'*pi/180;
hole1_coords = [W_Link/2*cos(theta_ccw) W_Link/2*sin(theta_ccw);...
R_Peg*cos(theta_cw) R_Peg*sin(theta_cw)]

% Cross-section of hole2:
theta_ccw = (-90:1:90)'*pi/180;
theta_cw = (90:-1:-90)'*pi/180;
hole2_coords = [W_Link/2*cos(theta_ccw) W_Link/2*sin(theta_ccw);...
R_Peg*cos(theta_cw) R_Peg*sin(theta_cw)];
```

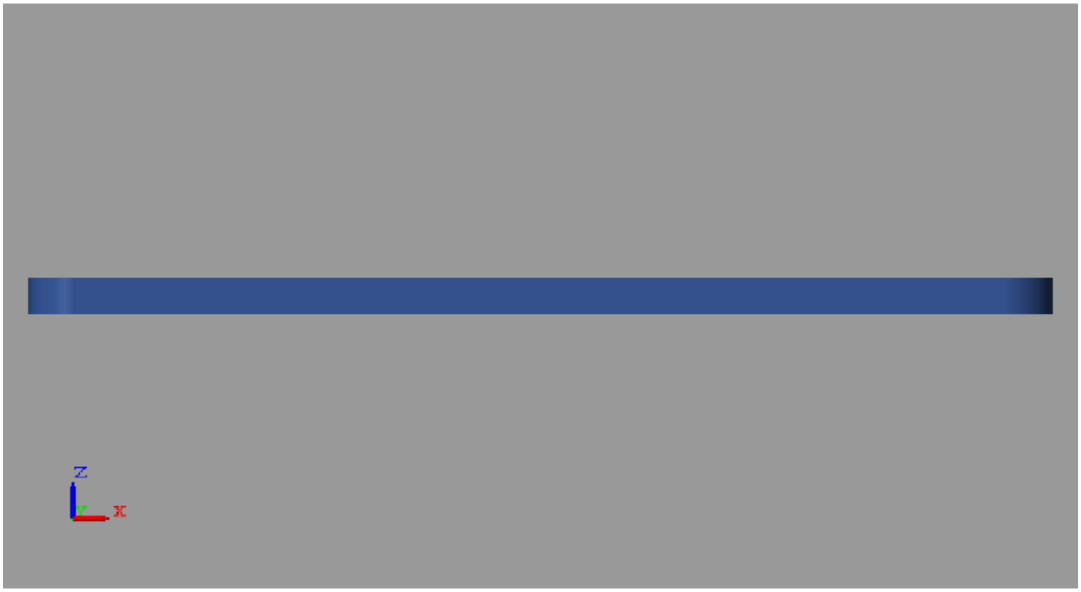
5 In the binary_link subsystem block dialog box, specify these parameters.

Parameter	Value
Length (m)	0.2
Width (m)	0.02
Thickness (m)	0.008
Peg Hole Radius (m)	0.004
Mass Density (kg/m^3)	2700
Link Color [R G B]	[0.25 0.4 0.7]

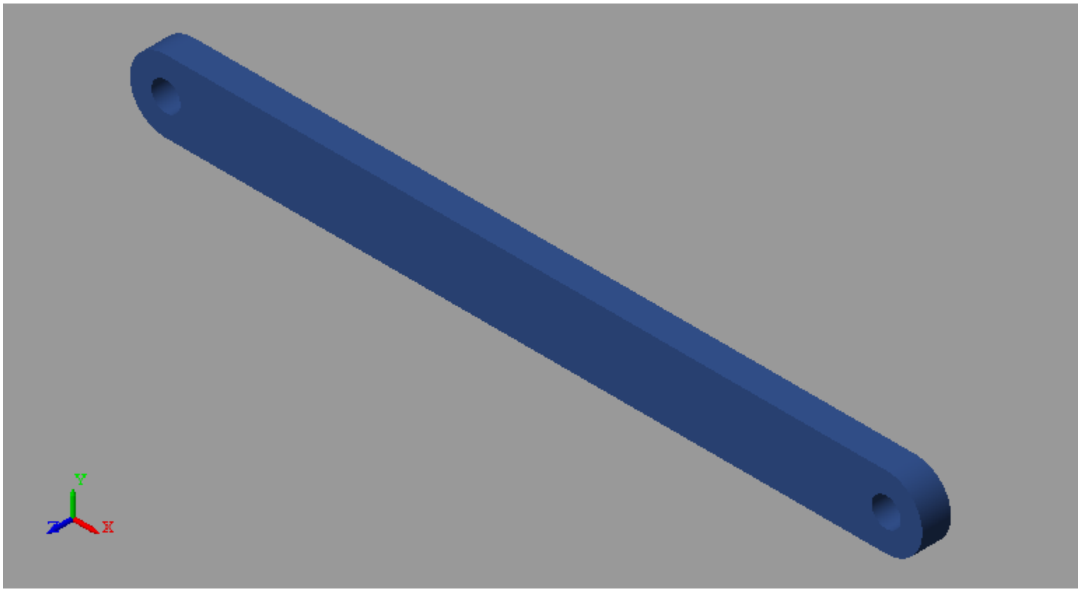
Visualize Model

Update the model to visualize the binary link rigid body in Mechanics Explorer.

1 Press **Ctrl+D** to update the diagram. The binary link appears in the visualization pane of Mechanics Explorer.



2 In the Mechanics Explorer toolstrip, change the **View convention** to `Y up (XY Front)`. Then, click the isometric button.



Save Custom Library Block

So that you can use it in later examples, save the binary link subsystem as a custom library block:

1. Create a custom block library, if you have not yet done so.

2. Drag the two_hole_binary_link subsystem block into the custom block library.

3. Save the custom block library as linkage_elements for use in subsequent examples.

Related Examples

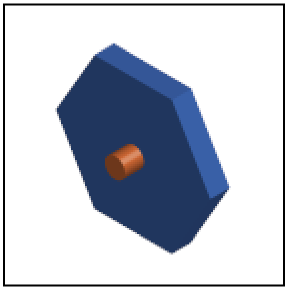
- "Model Binary Link" on page 2-57
- "Model Pivot Mount" on page 2-77
- "Model Four-Bar Linkage" on page 3-37

Model Pivot Mount

In this section...

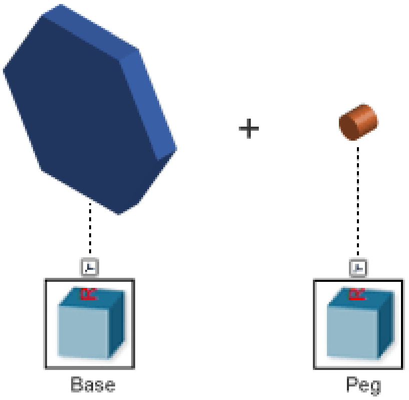
Model Overview

In this example, you model a simple pivot mount. This mount is a compound rigid body with a hexagonal shape and a protruding cylindrical peg. You represent the hexagonal shape using solid shape `Regular Extrusion`. You then offset the protruding peg from the hexagonal shape using a Rigid Transform block. In later examples, you use this mount to support mechanical linkages like the double pendulum and the four bar system.



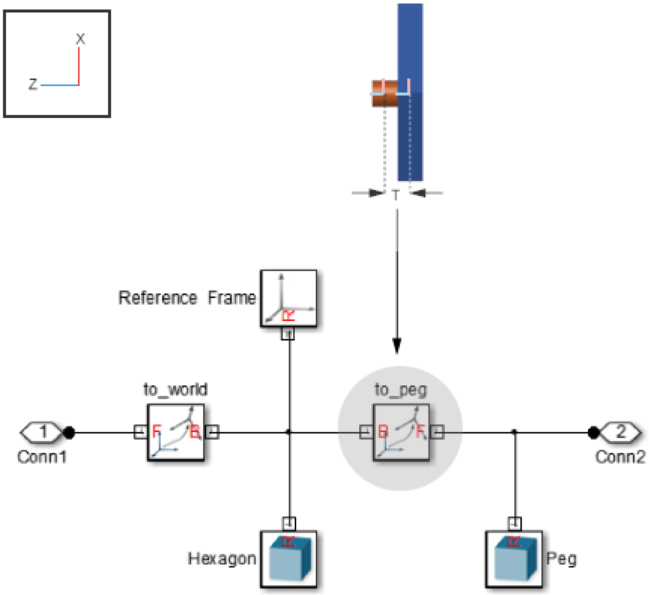
Modeling Approach

To model the pivot mount, you use two Solid blocks. Because the pivot mount has a hexagonal shape, you can model it using the `Regular Extrusion` shape. To represent the cylindrical peg, you use the `Cylinder` shape.



Each shape has a reference frame with origin at the geometry center. To offset the cylindrical peg with respect to the hexagonal mount, you apply a rigid transform between the two reference frames. You do this using the Rigid Transform block.

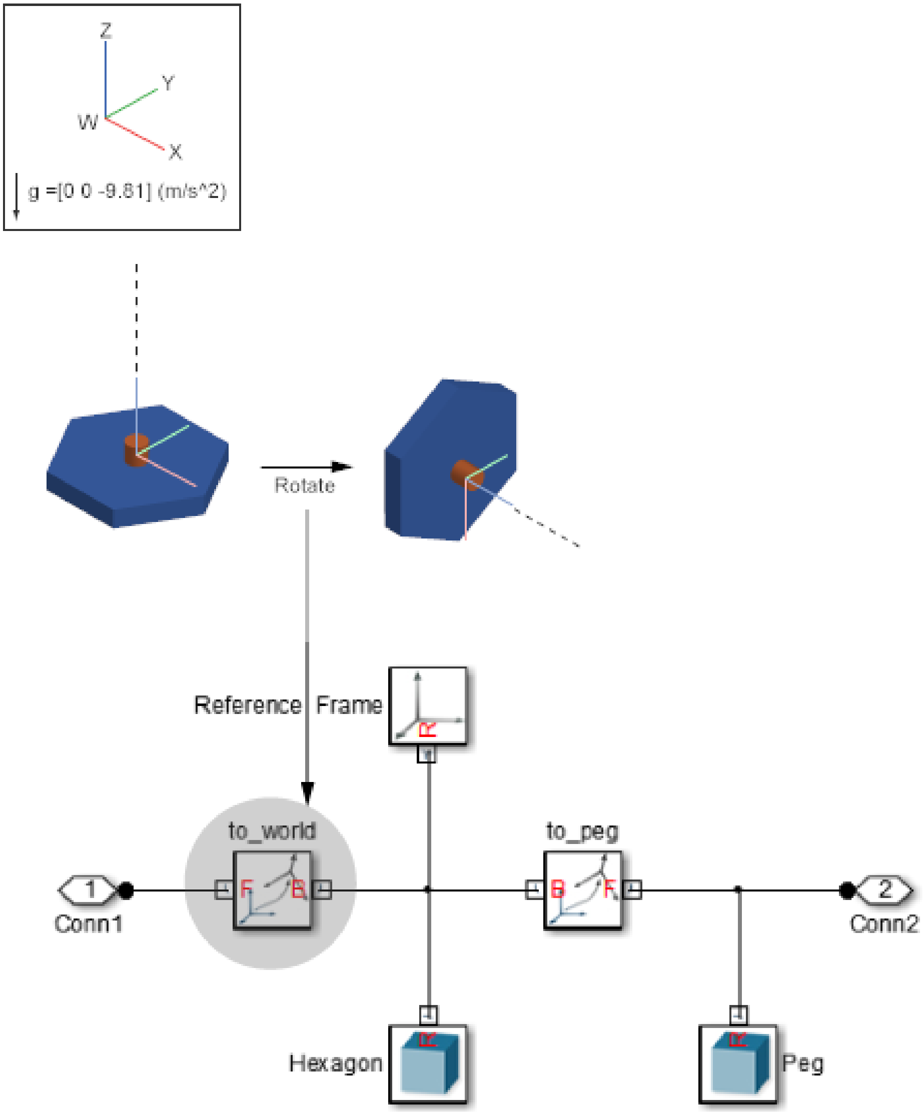
The Z axes of the two reference frames align with the cylindrical and extrusion axes of the peg and mount, respectively. Assuming the two solids have both have thickness *T*, the rigid transform between the two reference frames is a translation *T* along the common Z axis.



In later examples, you connect the pivot mount to a binary link using a revolute joint. One example is a double pendulum that moves due to gravity. For this example, it helps to rotate the Z axis of the mount so that it is orthogonal to the World frame Z axis. This task:

- Aligns the pendulum rotation plane with the default gravity vector `[0 0 -9.81] m/s^2`
- Aligns the pivot mount so that the pendulum appears vertically in Mechanics Explorer.

To perform this task, you rotate the pivot mount by 90° about the Y axis of the World frame using a Rigid Transform block. The figure illustrates the effect of this transform.


Z
Y
W
X
g =[0 0 -9.81] (m/s^2)
Rotate
Reference Frame
R
to_world
F
B
to_peg
B
F
1
Conn1
2
Conn2
Hexagon
R
Peg
R

Build Model

To model the pivot mount:

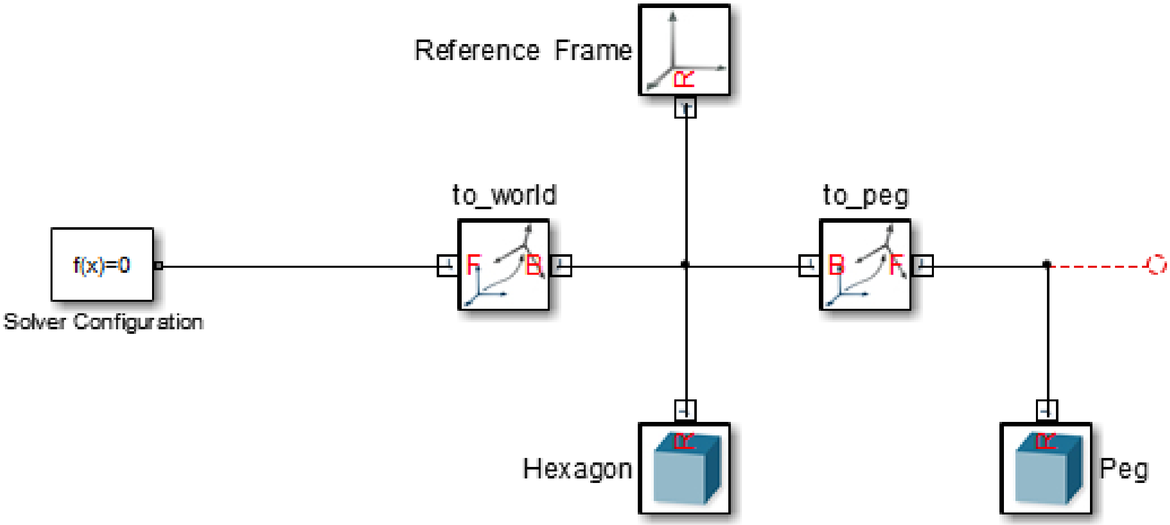
1 Start a new model.

2 Drag the following blocks to the model.

Block	Library	Quantity
Solid	**SimMechanics > Second Generation (SM 2G) > Body Elements**	2
Rigid Transform	**SimMechanics > Second Generation (SM 2G) > Frames and Transforms**	2
Reference Frame	**SimMechanics > Second Generation (SM 2G) > Frames and Transforms**	1
Solver Configuration	**Simscape > Utilities**	1

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

Note Include the disconnected frame line (red dashed line). This line becomes important when you generate a subsystem for the pivot mount. To add this line, right-click on the solid frame line and drag to the right.



4 Double-click Solid block Hexagon and specify the parameters in the table.

Parameter	Value
Geometry > Shape	`Regular Extrusion`
Geometry > Number of Sides	`6`
Geometry > Outer Radius	`R_Hexagon`
Geometry > Length	`T`
Inertia > Density	`rho`
Graphic > Color	`rgb_Hexagon`

5 Double-click Solid block Peg and specify the parameters in the table.

Parameter	Value
Geometry > Shape	`Cylinder`
Geometry > Radius	`R_Peg`
Geometry > Length	`2*T`

Parameter	Value
Inertia > Density	rho
Graphic > Color	rgb_Peg

6 Double-click Rigid Transform block to_peg and specify the parameters in the table.

Parameter	Value
Translation > Method	Standard Axis
Translation > Axis	+Z
Translation > Offset	3/2*T

7 Double-click Rigid Transform block to_world and specify the parameters in the table.

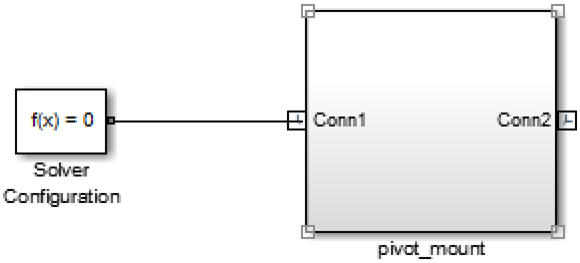
Parameter	Value
Rotation > Method	Standard Axis
Rotation > Axis	Y
Rotation > Angle	90

Generate Subsystem

You can now generate a subsystem to encapsulate the pivot mount block diagram. The subsystem mask provides a convenient place to initialize the MATLAB variables that you defined the block parameters with. To generate the subsystem:

1 Select all blocks except Solver Configuration.

2 Press **Ctrl+G** to enclose the blocks in a subsystem. Name the subsystem pivot_mount.



3 Select the pivot_mount subsystem block and press **Ctrl+M** to create a subsystem mask.

4 In the **Parameters & Dialog** tab of the Mask Editor, drag six edit boxes into the **Parameters** group and specify their properties. Click **OK**.

Prompt	Name
`Hexagon Outer Radius (m):`	`R_Hexagon`
`Hexagon Thickness (m):`	`T`
`Mass Density (kg/m^3):`	`rho`
`Hexagon Color [R G B]:`	`rgb_Hexagon`
`Peg Radius (m):`	`R_Peg`
`Peg Color [R G B]:`	`rgb_Peg`

5 In the pivot_mount block dialog box, specify these parameters.

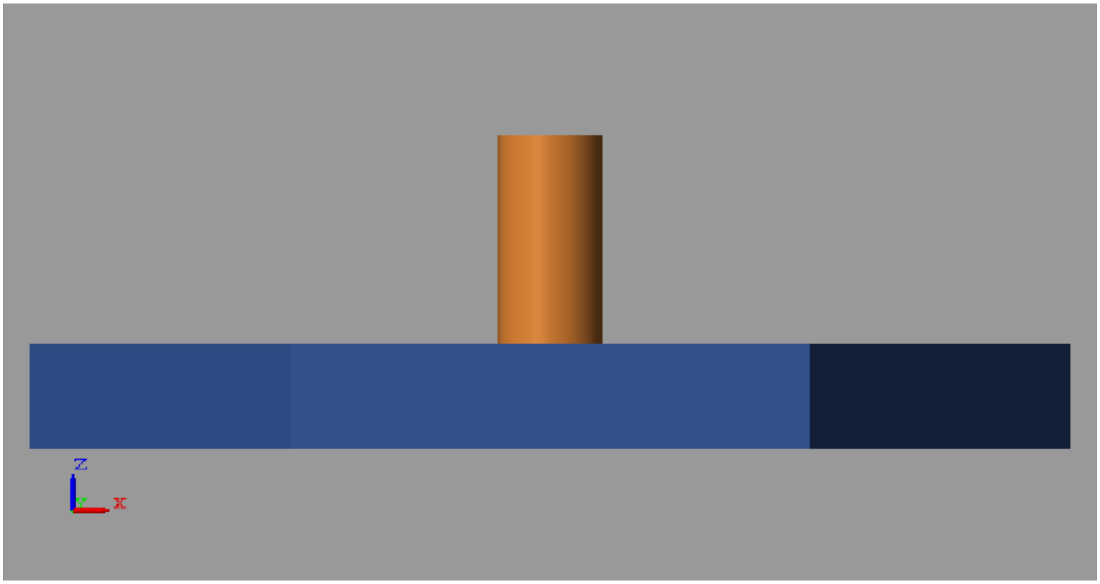
Parameter	Value
Hexagon Outer Radius (m):	`0.04`
Hexagon Thickness (m):	`0.008`
Mass Density (kg/m^3):	`2700`
Hexagon Color [R G B]:	`[0.25 0.4 0.7]`

Parameter	Value
Peg Radius (m):	`0.004`
Peg Color [R G B]:	`[1 0.6 0.25]`

Visualize Model

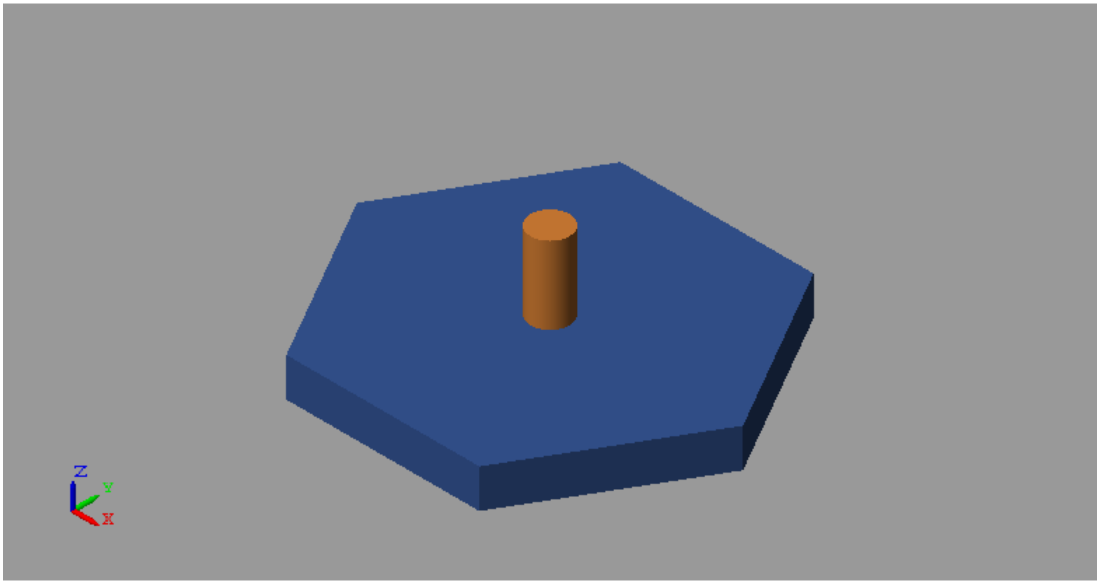
Update the model to visualize the pivot mount in Mechanics Explorer.

- With the model window active, press **Ctrl+D**. The pivot mount appears in the visualization pane of Mechanics Explorer.



To obtain the view used in the illustrations for this example:

- In the Mechanics Explorer toolstrip, click the isometric view button .



Save Custom Library Block

So that you can use it in later examples, save the pivot mount subsystem as a custom library block. If you have not done so, create a new library to save the block in:

1. In the Simulink menu bar, click **File > New > Library**.
2. Drag the pivot_mount block to the new library.
3. Save the library as linkage_elements.

Related Examples

- “Represent Binary Link Frame Tree” on page 1-36
- “Model Binary Link” on page 2-57

Concepts

- “Representing Frames” on page 1-7
- “Cross-Section Coordinates” on page 2-14
- “Specifying Solid Inertia” on page 2-20
- “Solid Color” on page 2-28

Multibody Systems

Assembling a Multibody Model

In this section...

To model a mechanism or machine, you connect rigid bodies with joints that constrain their relative degrees of freedom. This process is known as **multibody assembly**.

Note Before you can assemble a multibody model, you must create the rigid body subsystems for that model.

Workflow

Assembling a multibody system involves the following steps:

1. Connect rigid bodies subsystems with joint blocks.

2. Specify joint state targets.

3. Check assembly.

4. Adjust frames and joint state targets if necessary.

Identify Joint Requirements

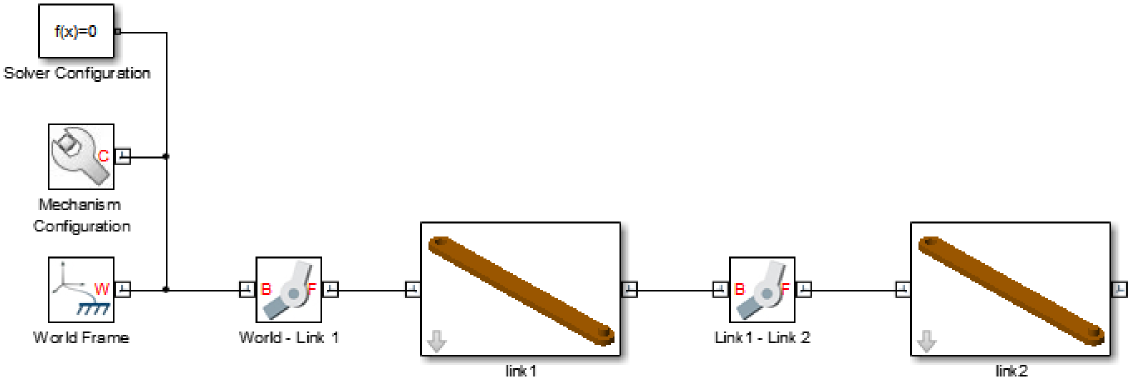
Each joint block connects precisely two rigid body frames. The type of joint block determines how the two rigid bodies can move with respect to each other. Two types of motion are possible:

- Rotation — provided by revolute and spherical joint primitives
- Translation — provided by prismatic joint primitives

To identify the correct joint block for your application, see “Modeling Joints” on page 3-8. The most commonly used joint blocks are Prismatic Joint, Revolute Joint, and Spherical Joint.

Connect Rigid Bodies with Joints

Drag the rigid body subsystem blocks onto the SimMechanics model. Then, select and drag the appropriate joint blocks from the Joints library. Connect the base and follower frames of each joint block to two frames on two distinct rigid body subsystems. The following figure shows the assembly of a double-pendulum model.



Caution Carefully check the position and orientation of the rigid body frames that you connect to joint blocks. Joint frames that possess either incorrect position or orientation can cause the model to assemble in an unexpected configuration.

In severe cases, joint frames cause kinematic conflicts that lead to assembly failure and simulation errors. Kinematic conflicts due to incorrect joint frames are more likely in closed kinematic loops such as the four-bar linkage.

Specify Joint State Targets

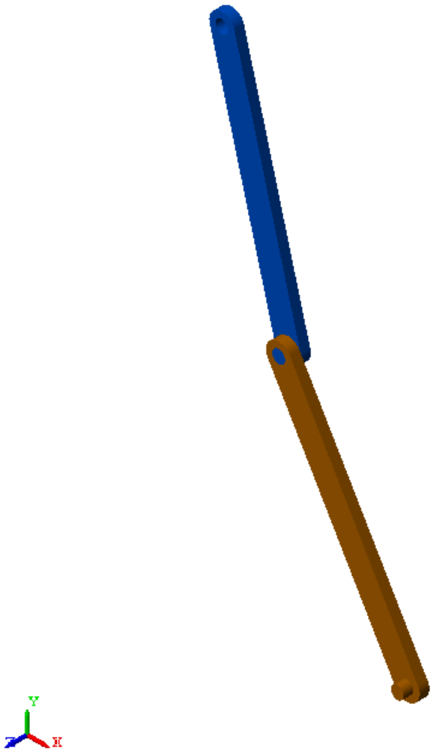
You can guide model assembly at time `t=0`. Each joint block provides a **State Targets** menu where you can specify the initial position and velocity of each joint primitive. You can select high or low priority for each joint state target.

During assembly, SimMechanics attempts to meet all specified state targets. The following rules apply to joint state targets:

- SimMechanics attempts to meet all state target values *precisely*.
- If two or more joint state targets are mutually incompatible, SimMechanics attempts to meet high priority targets *precisely*, and low priority targets *approximately*.
- If a state target is not met, the Model Report utility identifies the joint with the unsatisfied state target.

Note Unsatisfied state targets do not cause assembly failure or simulation errors, but can cause the model to assemble with an unexpected configuration.

The following figure shows the double-pendulum model with joint state targets of `+15 deg` for each revolute joint.



Check Assembly

Joints assemble base and follower frames according to a well-defined set of rules. For example, the Revolute Joint block makes the +Z axes of base and follower frames coincident in space. If the +Z axes of base and follower frames are not properly positioned, an unexpected joint configuration can result.

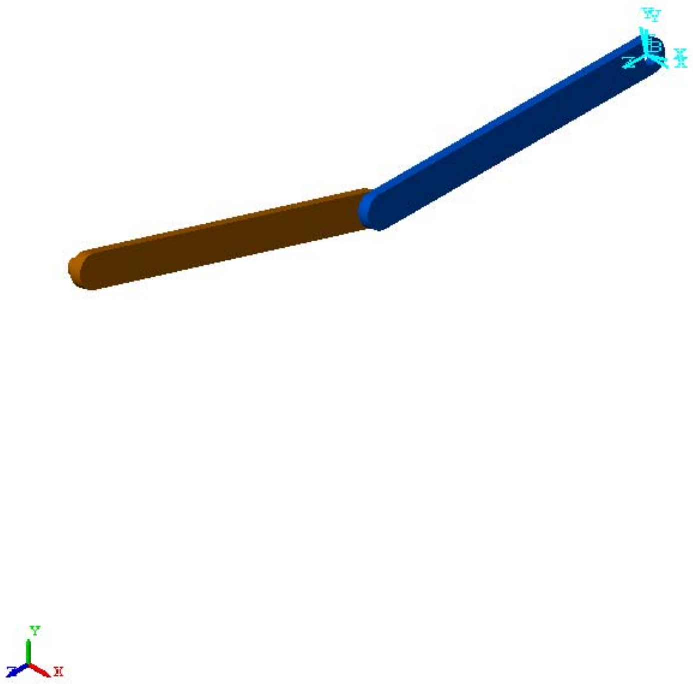
Use Mechanics Explorer to check the base and follower frames of each joint in a model.

1 In the tree-browser pane of Mechanics Explorer, click the name of each joint block.

Mechanics Explorer highlights the base and follower frames of the selected joint in the visualization pane.

2 Check the base and follower frames of the selected joint for an unexpected position or orientation.

The following figure shows an improperly configured double-pendulum model. The Z-axis of the top revolute joint, **World-Link1**, points along the length axis of the binary link. Since the +Z axis specifies the rotational axis of the Revolute Joint block, **link1** rotates about its length axis, producing unexpected results during simulation.



Related Examples

- “Model Double Pendulum” on page 3-29
- “Model Four-Bar Linkage” on page 3-37
- “Correct Aiming Mechanism Assembly Error” on page 3-48

Concepts

- "Identifying Assembly Issues" on page 3-25
- "Modeling Joints" on page 3-8

Modeling Joints

In this section...

Joints constrain the mechanical degrees of freedom between two connecting rigid bodies. The primary purpose of joints is to limit motion of a mechanism or machine so an end effector can move along a specified path. Rigid bodies can contain the following degrees of freedom:

- Translational — linear displacement of one rigid body frame relative to another along a common axis.
- Rotational — angular displacement of one rigid body frame relative to another about a common axis

A free rigid body contains exactly six degrees of freedom: three rotational and three translational. The free rigid body can translate along any combination of three mutually orthogonal axes, and rotate about any combination of the same axes. When you connect two rigid bodies with a joint, you remove degrees of freedom between the two. Depending on the joint, you can remove anywhere from zero-six degrees of freedom. A joint that removes all six degrees of freedom is called **Weld** joint.

Note The Rigid Transform block is similar to the Weld Joint block. Both blocks remove all six mechanical degrees of freedom between the two connecting rigid bodies. However, the Rigid Transform block also allows you to maintain a specified distance and angle between the two rigid bodies.

Joint Frames

The joint block contains two frame ports, B and F. The ports identify the base and follower frames of a joint, respectively. You connect the base frame port to one frame on one rigid body, and the follower frame port to another frame on a second rigid body. Switching the base and follower frames of a joint block has no effect on model assembly or simulation.

During simulation, joint blocks apply a time-varying transformation to the follower frame with respect to the base frame. The transformation depends on dynamic inputs (forces and torques) and the kinematic configuration of the model. Transformation components include rotation and translation about or along the joint primitive axes.

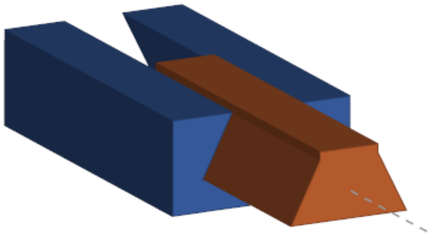
Joint Primitives

Each joint block contains a combination of *joint primitives* — elementary joint constructs that make up more advanced joints. The joint primitives represent the simplest joints you can find in SimMechanics. Three joint primitives exist: prismatic, revolute, and spherical. The following three sections briefly describe each primitive. The final section lists the primitives that make up each joint block.

Prismatic

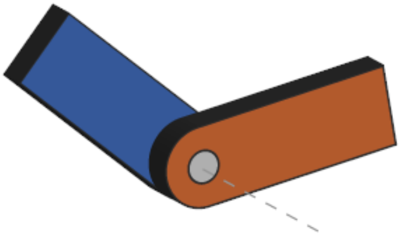
Joint primitive with one translational degree of freedom. The prismatic primitive allows the joint base and follower frames to translate relative to each other along a common axis. Joints with two prismatic primitives allow translation in a 2-D plane that contains the prismatic axes. Joints with three prismatic primitives allow translation in 3-D space.

The following figure shows a schematic of the prismatic joint primitive.



Revolute

Joint primitive with one rotational degree of freedom. The revolute primitive allows the joint base and follower frames to rotate relative to each other about a common axis. Joints with three revolute primitives allow rotation in 3-D space. The frames must each connect to a non-degenerate mass. The following figure shows a schematic of the revolute joint primitive.

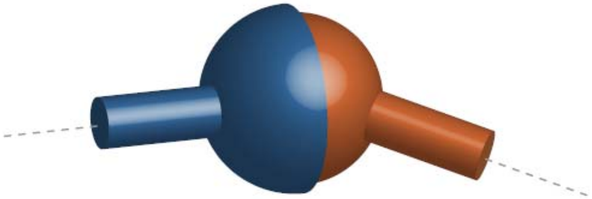


Spherical

Joint primitive with three rotational degrees of freedom. The spherical joint allows the joint base and follower frames to rotate about three mutually orthogonal axes.

The Spherical primitive is not a serial combination of revolute primitives. Such a combination is susceptible to Gimbal lock — an event in which two revolute axes align, resulting in the loss of one rotational degree of freedom.

The Spherical primitive is not susceptible to Gimbal lock at any time. The following figure shows a schematic of the spherical joint primitive.



Joint Primitive Composition

Joint primitives are the basic elements of joint blocks. Each joint block can contain multiple joint primitives. The number and type of joint primitives that a joint block contains defines the degrees of freedom that joint provides. The table summarizes the joint primitives and degrees of freedom (DOF) for each joint block.

Joint Block	Degrees of Freedom		Joint Primitives		
	Rotation	Translation	Prismatic	Revolute	Spherical
6–DOF Joint	3	3	3	0	1
Bearing Joint	3	1	1	3	0
Bushing Joint	3	3	3	3	0
Cartesian Joint	0	3	3	0	0
Cylindrical Joint	1	1	1	1	0
Pin Slot Joint	1	1	1	1	0

Joint Block	Degrees of Freedom		Joint Primitives		
	Rotation	Translation	Prismatic	Revolute	Spherical
Gimbal Joint	3	0	0	3	0
Planar Joint	1	2	2	1	0
Prismatic Joint	0	1	1	0	0
Rectangular Joint	0	2	2	0	0
Revolute Joint	1	0	1	0	0
Spherical Joint	3	0	0	0	1
Telescoping Joint	3	1	1	0	1
Universal Joint	2	0	0	2	0
Weld Joint	0	0	0	0	0

Assembling Joints

During assembly, joint blocks position and orients base and follower frames according to rules that depend on the joint type. The table summarizes the position and orientation constraints that each joint primitive imposes on the base and follower frames of a joint.

Joint primitive	Constraint
Prismatic	• Aligns base and follower frame prismatic axes. For example, the Z Prismatic Primitive aligns the Z axes of the base and follower frames. • Holds the remaining base and follower frame axes parallel to each other. For example, the Z Prismatic Primitive keeps the base and follower frame X and Y axes parallel to each other.
Revolute	• Aligns base and follower frame revolute axes. For example, the Z Revolute Primitive aligns the Z axes of the base and follower frames. • Holds base and follower frame origins coincident.
Spherical	• Holds base and follower frame origins coincident.

Guiding Joint Assembly

Each joint primitive provides the option to specify a state target: the desired initial state for that joint primitive. You can specify state targets for the position and velocity of the joint primitive, both of which can be either rotational (for revolute and spherical joint primitives), or translational (for prismatic joint primitives). The value of the state target represents the relative state of the follower port frame with reference to the base port frame. For example, when you enter a value for the velocity state target of a joint block, you specify the velocity of the follower port frame relative to the base port frame.

It is not always possible to set the initial state of a joint to the specified state target. This is especially true of closed loops containing state targets specified for multiple joints. However, during assembly, SimMechanics attempts to satisfy as many state targets as possible, and with a maximum level of precision. In the event that all state targets cannot be met, SimMechanics prioritizes state targets according to the priority level you specify. Joints with a priority level of `High (desired)` assemble earlier, followed by joints with a priority level of `Low (approximate)`.

In the event that it is not possible to set all state targets to their exact values, SimMechanics relaxes the low-priority state targets, and searches for the best-fit approximate values that still allow assembly. Should assembly still fail, SimMechanics begins to relax high-priority state targets, searching for the nearest approximate values that allow for successful assembly. If assembly fails, check that the model is kinematically valid. Check also that closed-loop systems do not contain state targets for every joint in the loop, which by default causes an assembly error.

Related Examples

- “Model Double Pendulum” on page 3-29
- “Model Four-Bar Linkage” on page 3-37
- “Correct Aiming Mechanism Assembly Error” on page 3-48

Concepts

- “Identifying Assembly Issues” on page 3-25

Modeling Gear Constraints

In this section...

You can represent gear constraints in a multibody model. To do this, SimMechanics provides a Gears, Couplings and Drives 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.

B
F
Common Gear
B
F
Rack and
Pinion

Gear Types

The Gears, Couplings and Drives library provides two blocks that you can represent gear constraints with: Common Gear, and Rack and Pinion. The table summarizes what you can do with each block.

Block	Description
Common Gear Constraint	Transmit angular motion from one rigid body frame to another
Rack and Pinion Constraint	Convert angular motion from one rigid body frame (pinion) to linear motion of another rigid body frame (rack)

Featured Examples

SimMechanics provides two featured examples that highlight the use of gear blocks. The table lists 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 Model Contains
Cardan Gear	`sm_cardan_gear`	Common Gear
Windshield Wiper	`sm_windshield_wiper`	Rack and Pinion

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

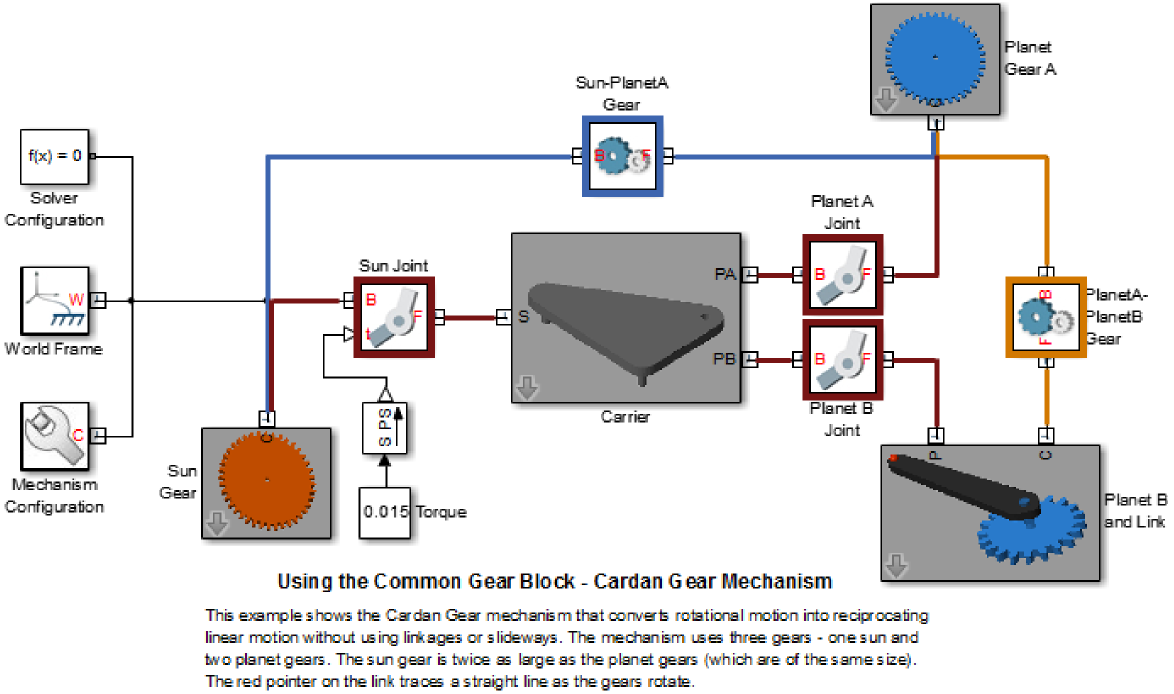
Inertia, Geometry, and Efficiency

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 SimMechanics 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.

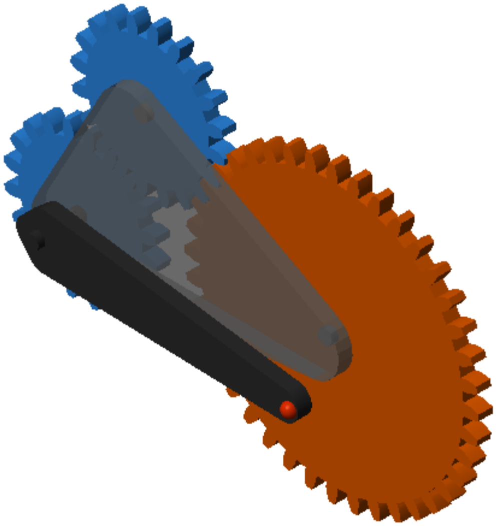
Featured example `sm_cardan_gear` illustrates an application of the Common Gear block. In this model, two Common Gear blocks connect three gear rigid bodies. Subsystems Planet Gear A, Planet B and Link, and Sun Gear represent these rigid bodies. One Common Gear block constrains the motion of subsystem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



Using the Common Gear Block - Cardan Gear Mechanism

This example shows the Cardan Gear mechanism that converts rotational motion into reciprocating linear motion without using linkages or slideways. 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 summarize these constraints.

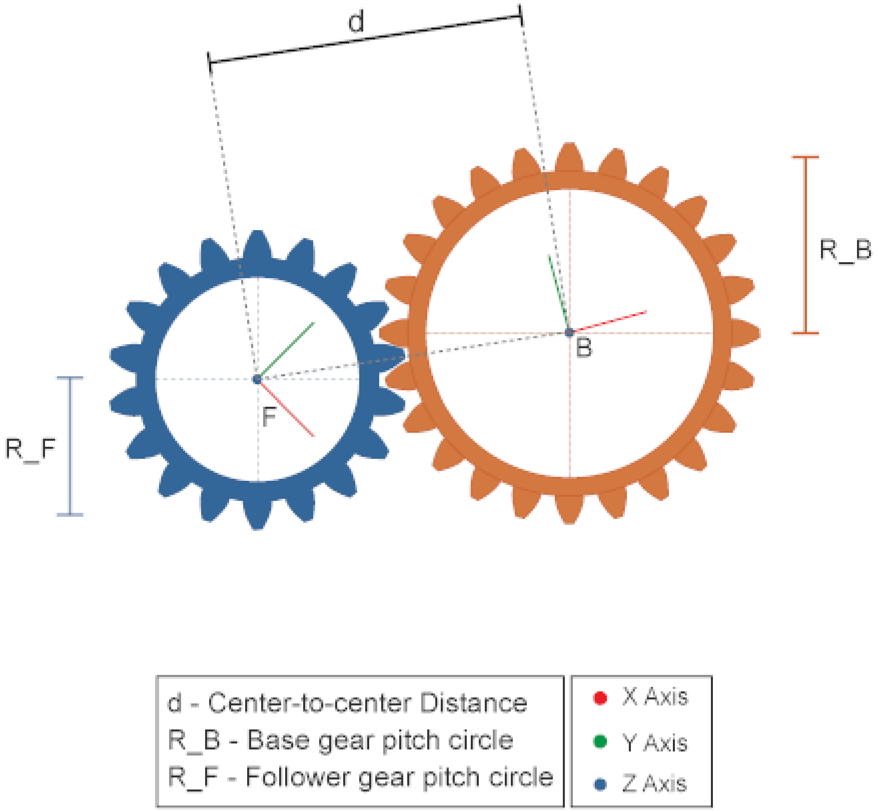
Gear Constraint	Description
Frame Distance	The model must maintain a fixed distance between the base and follower gear frames. The value of this distance depends on the gear block that you use.
Frame Orientation	The model must orient the base and follower gear frames according to rules that are specific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

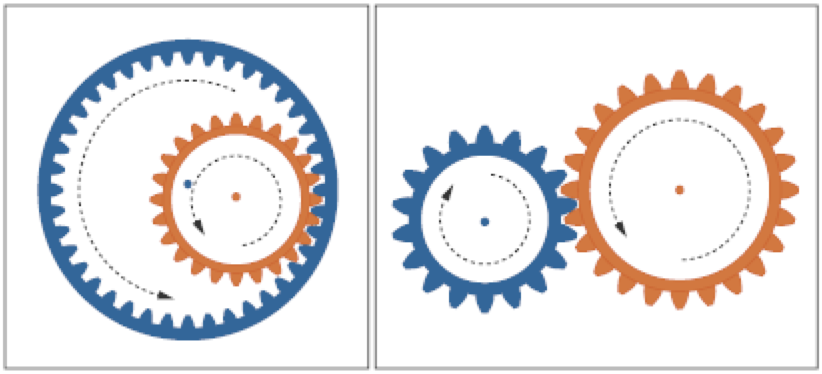
During assembly, the Common Gear block requires that the base and follower frame Z axes align. These are the rotation axes of the two gear frames. Failure to align the Z axes of the two gear frames results in assembly failure during model update. The figure illustrates the common gear rigid bodies, frames,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


d
R_B
B
F
R_F
d - Center-to-center Distance
R_B - Base gear pitch circle
R_F - Follower gear pitch circle
● X Axis
● Y Axis
● Z Axis

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

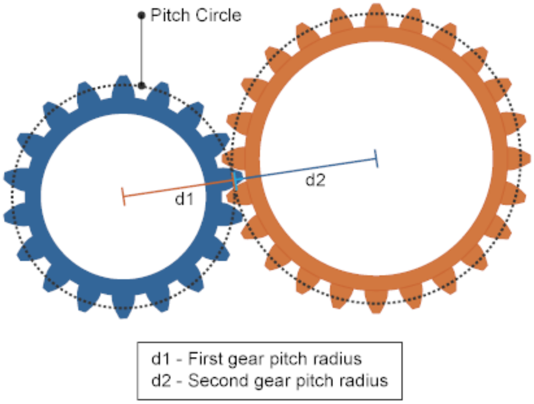
In the block dialog box, you specify the gear dimensions. Depending on the specification method that you choose, you can specify the center-to-center distance between gears or the pitch circle radii. During model assembly, the Common Gear block imposes this distance constraint between the two gear frames. 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.



Simulation

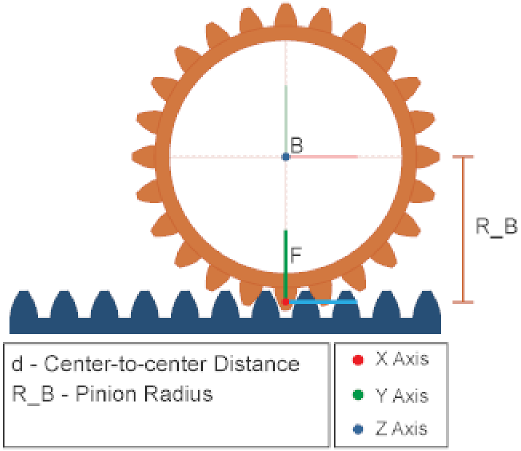
During simulation, the Common Gear block requires that the model maintain the proper distance between gear frames. This distance must equal either the center-to-center distance or the sum of base and follower gear pitch radii that you specify in the block dialog box. The structure of the model must be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

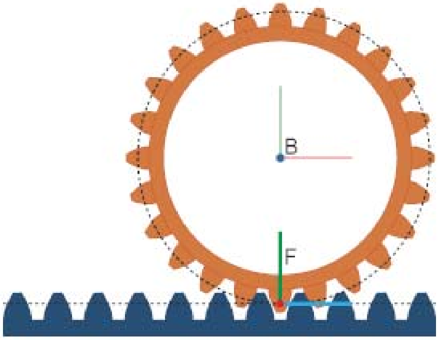
When the gear is in its zero configuration—a configuration in which the angle and displacement between base and follower frames are taken as zero—the follower frame Z axis is also parallel to the base frame X axis, and base and follower frame Y axes are parallel to each other. The follower frame origin 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.

Identifying Assembly Issues

In this section...

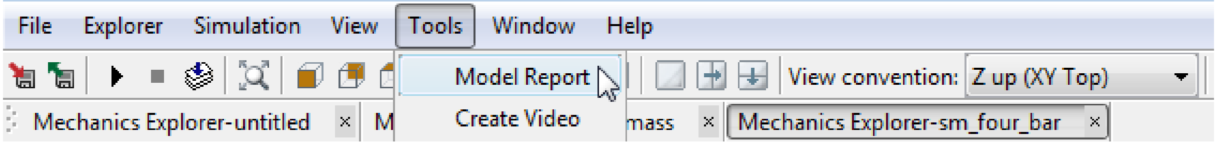
Model Report is a SimMechanics tool that provides model assembly status and parameters. Use Model Report to:

- Identify joints and constraints with assembly issues.
- Identify joints with unsatisfied state targets.
- Compare specified and actual joint state targets.
- Obtain relevant statistics for a model

Open Model Report

Model Report is accessible from Mechanics Explorer. To open Model Report:

1. Update or simulate a SimMechanics model.
2. In the Mechanics Explorer menu bar, click **Tools > Model Report**.



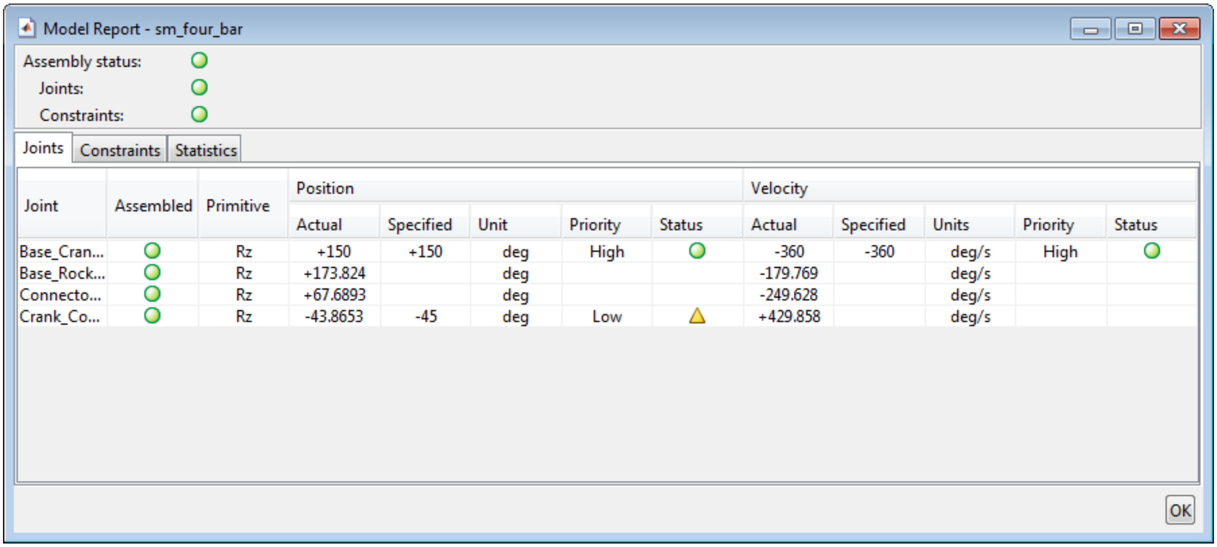
3. Model Report opens with assembly status and parameters relevant to the current model.

Model Report Tabs

Model Report contains one header section and three tabs:

- **Header** — Provides model-wide assembly status.

- **Joints** — Provides assembly status and state target values for each joint block in a model. The following figure shows the **Joints** tab for example `sm_four_bar`.



Model Report - sm_four_bar

Assembly status: ●

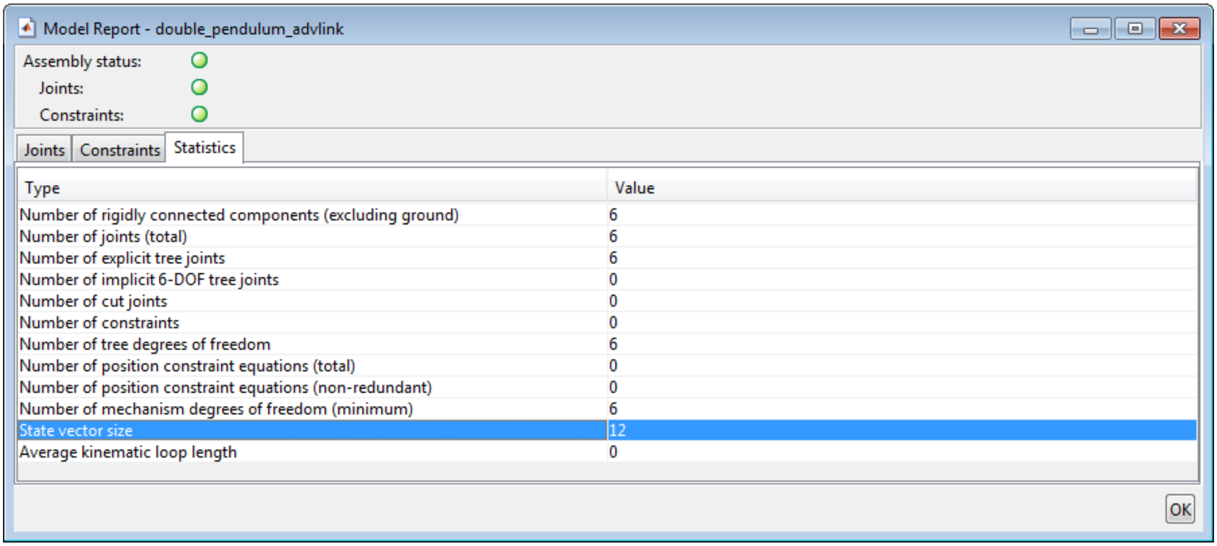
Joints: ●

Constraints: ●

Joint	Assembled	Primitive	Position					Velocity				
			Actual	Specified	Unit	Priority	Status	Actual	Specified	Units	Priority	Status
Base_Cran...	●	Rz	+150	+150	deg	High	●	-360	-360	deg/s	High	●
Base_Rock...	●	Rz	+173.824		deg			-179.769		deg/s		
Connecto...	●	Rz	+67.6893		deg			-249.628		deg/s		
Crank_Co...	●	Rz	-43.8653	-45	deg	Low	⚠	+429.858		deg/s		

- **Constraints** — Provides assembly status for each constraint block in a model. The following figure shows the **Constraints** tab for example `sm_four_bar`.

- **Statistics** — Provides model-wide statistics. Parameters include number of joints, constraints, and kinematic degrees of freedom in a model. The following figure shows the **Statistics** tab for example `sm_four_bar`.



Assembly status: 🟢
Joints: 🟢
Constraints: 🟢

Type	Value
Number of rigidly connected components (excluding ground)	6
Number of joints (total)	6
Number of explicit tree joints	6
Number of implicit 6-DOF tree joints	0
Number of cut joints	0
Number of constraints	0
Number of tree degrees of freedom	6
Number of position constraint equations (total)	0
Number of position constraint equations (non-redundant)	0
Number of mechanism degrees of freedom (minimum)	6
State vector size	12
Average kinematic loop length	0

Status Icons

Model Report uses three icons to identify the assembly status of a model, joint block, or constraint block.

Status Icon	Description
🟢	Assembled without issues. In joint blocks, the icon indicates state target was successfully met.
⚠	Assembled with issues. In Joint blocks, the icon indicates the state target was approximately met.
🟥	Not assembled. In Joint blocks, the icon indicates the state target was not met.

Related Examples

- “Model Double Pendulum” on page 3-29
- “Model Four-Bar Linkage” on page 3-37
- “Correct Aiming Mechanism Assembly Error” on page 3-48

Concepts

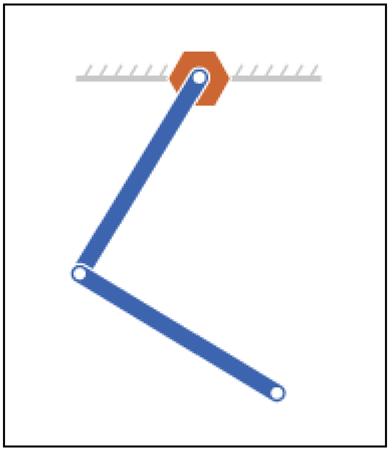
- "Visualizing and Inspecting a Model" on page 6-2
- "Identifying Assembly Issues" on page 3-25
- "Modeling Joints" on page 3-8

Model Double Pendulum

In this section...

Model Overview

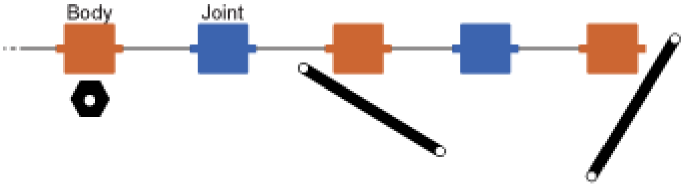
The double pendulum is a simple multibody system. It contains two links and a pivot mount that connect with joints. This system is nonlinear and does under certain conditions exhibit chaos. In this example, you assemble a double pendulum using custom blocks for the links and the pivot mount. You can later use this model to study the chaotic motion of a double pendulum.



Before continuing, you must have completed examples "Model Binary Link" on page 2-57 and "Model Pivot Mount" on page 2-77.

Modeling Approach

To model the double pendulum, you represent each physical component and constraint using a SimMechanics block. The double pendulum system contains three rigid bodies—one pivot mount and two binary links— that connect in series through a pair of revolute joints. You represent the pivot mount and the binary links using the custom library blocks that you created in previous examples. You represent the two joints using two Revolute Joint blocks from the Joints library.



You can guide model assembly. By specifying joint state targets, you can instruct SimMechanics to assemble a joint in the configuration you want. State targets that you can specify include position and velocity, both angular and linear. At times, a state target may conflict with other state targets, or even with other kinematic constraints in the model. In these cases, you can prioritize the most important state targets by assigning them a high priority level. During assembly, if two targets conflict with each other, SimMechanics assembles the high priority target first. To specify both state target values and priority levels, you use the **State Targets** menu of the joint block dialog boxes.

Build Model

To model the double pendulum system:

1 Start a new model.

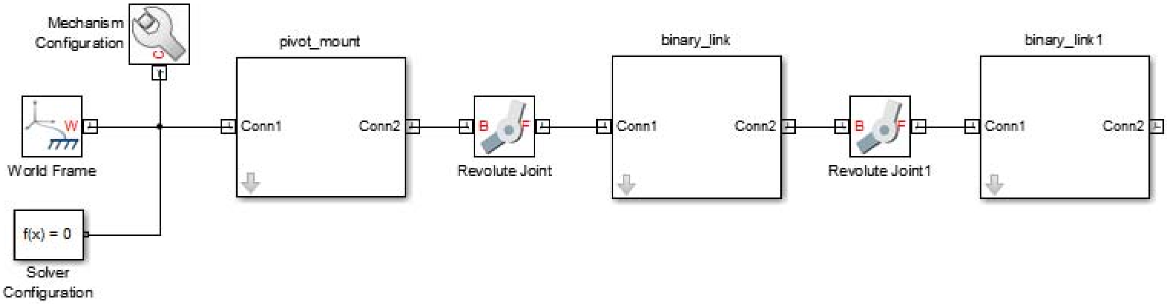
2 Drag these custom blocks into the model. See the modeling tutorials if you have not created these custom blocks.

Block	Quantity	Modeling Tutorial
pivot_mount	1	"Model Pivot Mount" on page 2-77
binary_link	2	"Model Binary Link" on page 2-57

3 Drag these blocks into the model.

Block	Library	Quantity
Revolute Joint	**SimMechanics > Second Generation (SM 2G) > Joints**	2
World Frame	**SimMechanics > Second Generation (SM 2G) > Frames and Transforms**	1
Mechanism Configuration	**SimMechanics > Second Generation (SM 2G) > Utilities**	1
Solver Configuration	**Simscape > Utilities**	1

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



Guide Model Assembly

The model is now complete. However, before visualizing and simulating the model, specify joint state targets to guide model assembly.

1 Double-click the two Revolute Joint blocks.

2 In each block dialog box, click **State Targets > Specify Position Target**.

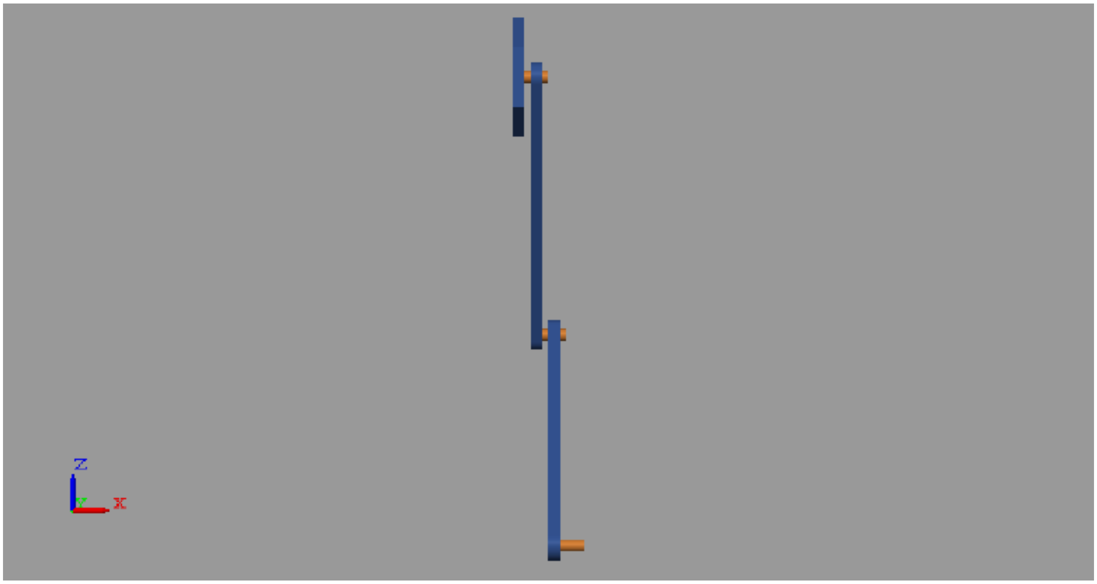
3 In the **Value** fields, enter the following values and press **OK**.

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

Visualize Model and Check Assembly Status

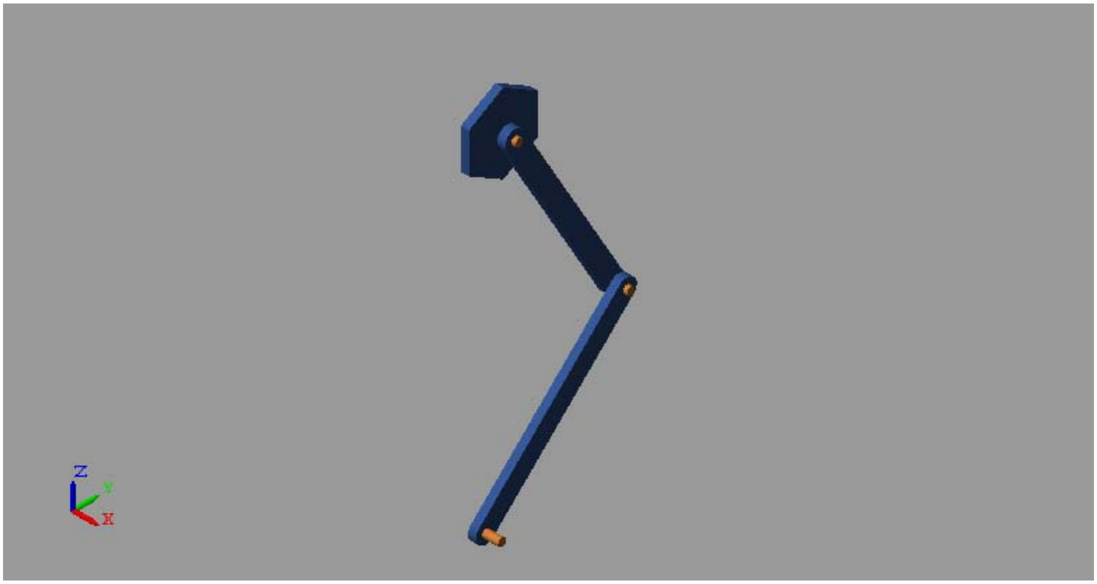
If the state targets are consistent with each other and with the rest of the model, SimMechanics will assemble each joint in the specified state. To check if a state target was met and what it’s actual value is, use the Model Report tool in Mechanics Explorer after updating the model.

1 With the model window active, press **Ctrl+D** to update the model.



2 In the Mechanics Explorer tool bar, click the isometric view button, .

The visualization pane of Mechanics Explorer displays the assembled double pendulum model using an isometric view.



From the visualization pane, it appears that SimMechanics successfully met both joint state targets. To confirm, open the Model Report tool:

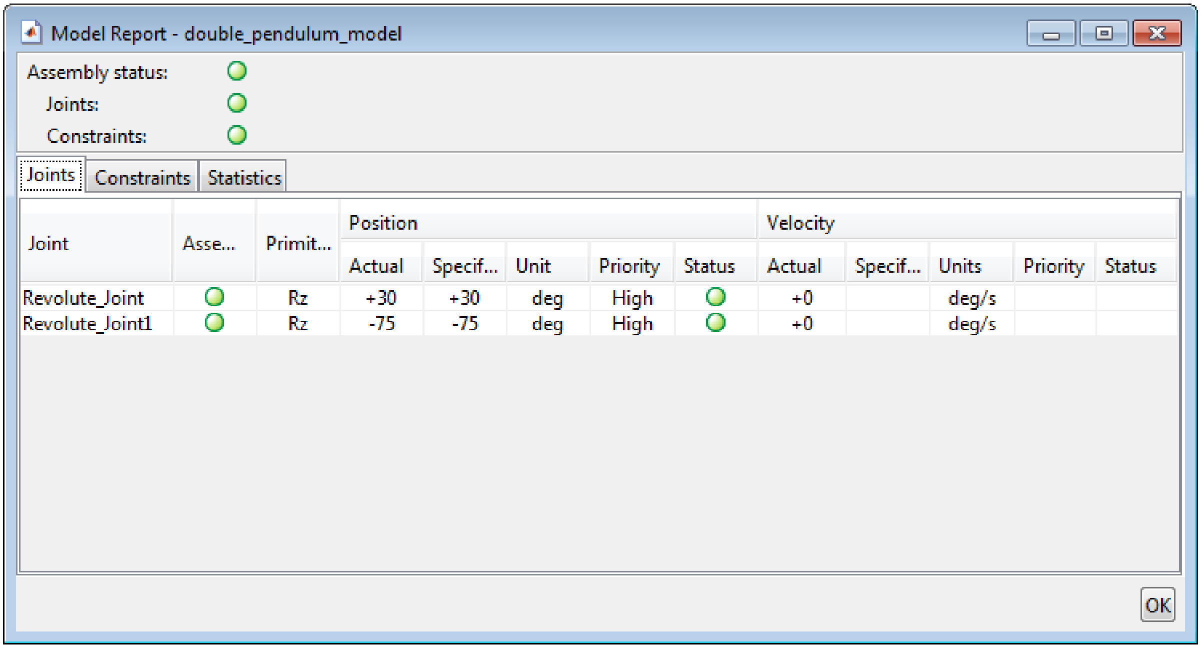
1 In the Mechanics Explorer menu bar, click **Tools > Model Report**.

2 In the Joints tab of the Model Report window, check for yellow or red lights.

   These lights identify joints with assembly issues. In this example, you should see none.

3 Under **Position**, compare the values of **Actual** and **Specified**.

   These are the actual angle of the joint at time zero, and the specified angle entered in the joint block dialog box, respectively. A green light in the **Status** column indicates that the two values are equal.



Model Report - double_pendulum_model

Assembly status: ●
Joints: ●
Constraints: ●

Joint	Asse...	Primit...	Position					Velocity				
			Actual	Specif...	Unit	Priority	Status	Actual	Specif...	Units	Priority	Status
Revolute_Joint	●	Rz	+30	+30	deg	High	●	+0		deg/s		
Revolute_Joint1	●	Rz	-75	-75	deg	High	●	+0		deg/s		

Simulate Model

If assembly was successful, you are now ready to simulate the double pendulum model:

- With model or Mechanics Explorer window active, press **Ctrl+T** to simulate the model.

The visualization pane of Mechanics Explorer shows the double pendulum simulation. When the simulation ends, you can replay it without rerunning the simulation. You can also adjust playback speed and loop the animation so that it begins again once it reaches the end. To do this, use the animation tool bar at the bottom of the visualization pane.

Save Model

For use in subsequent examples, save the model you created as `double_pendulum` in a convenient folder.

Related Examples

- “Sense Double-Pendulum Motion” on page 4-43
- “Model Four-Bar Linkage” on page 3-37
- “Correct Aiming Mechanism Assembly Error” on page 3-48

Concepts

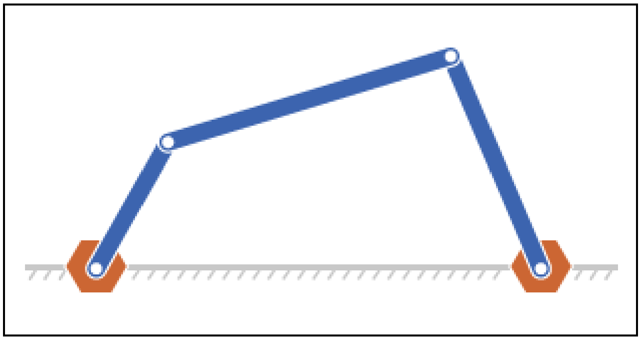
- “Visualizing and Inspecting a Model” on page 6-2
- “Identifying Assembly Issues” on page 3-25
- “Modeling Joints” on page 3-8

Model Four-Bar Linkage

In this section...

Model Overview

The four-bar linkage is a planar closed-loop linkage used extensively in mechanical machinery. This linkage has four coplanar bars that connect end-to-end with four revolute joints. In this example, you model a four-bar linkage using the Binary Link and Pivot Mount custom blocks that you created in previous examples. For an advanced application of the four-bar linkage, see the bucket actuating mechanism of the Backhoe featured example.



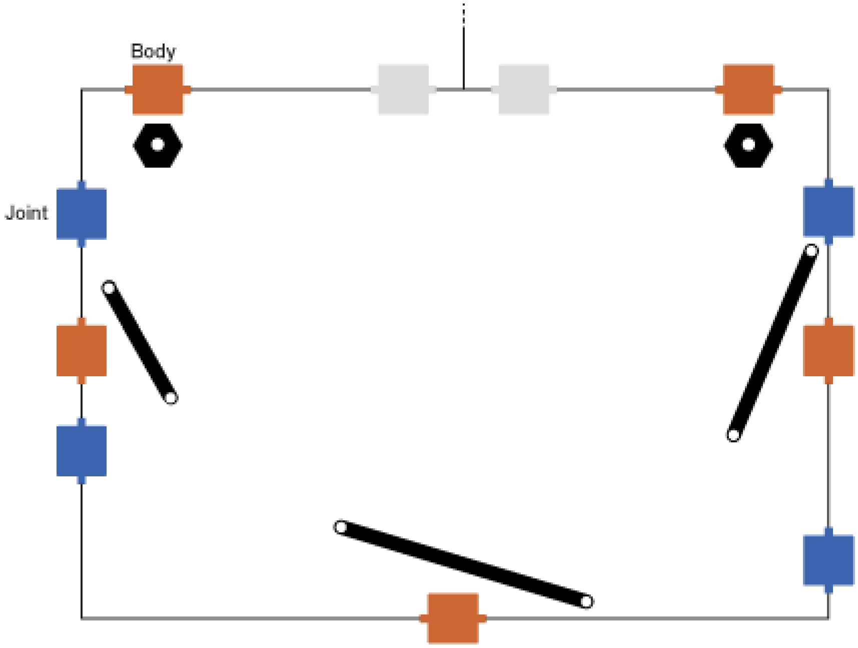
Before continuing, you must complete the following examples:

- "Model Binary Link" on page 2-57
- "Model Two-Hole Binary Link" on page 2-69
- "Model Pivot Mount" on page 2-77

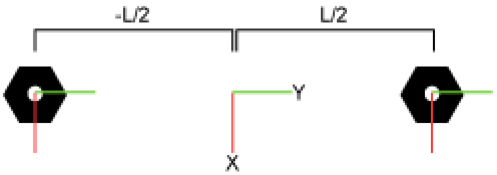
Modeling Approach

To model the four-bar linkage, you represent each physical component with a SimMechanics block. The linkage in this example has five rigid bodies—three binary links and two pivot mounts—that connect in a closed loop through four revolute joints. Two of the binary links have one peg and one hole. The third binary link has two holes. The fourth link is implicit: the fixed distance between the two coplanar pivot mounts represents this link.

You represent the binary links and pivot mounts using the custom library blocks that you created in previous examples. You represent the four revolute joints using four Revolute Joint blocks from the SimMechanics Joints library.



The two pivot mounts connect rigidly to the World frame. For this reason, the fourth (implicit) link acts as the ground link. Two Rigid Transform blocks provide the rigid connection between the two pivot mounts and the World frame. A translation offset in each Rigid Transform block displaces the two pivot mounts symmetrically along the World frame Y axis.



To guide model assembly, you can specify the desired initial state for one or more joints in the model. To do this, you use the **State Targets** menu of the joint blocks. The state targets that you can specify are the joint position and velocity. These are angular quantities in revolute joints. You can specify state targets for all but one of the joints in a closed loop.

Build Model

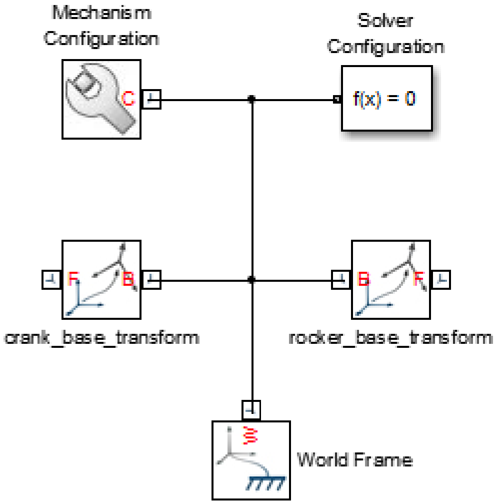
To model the four-bar linkage:

1 Start a new model.

2 Drag these blocks to the model.

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

3 Connect and name the blocks as shown in the figure. Be sure to flip the frame ports of the crank_base_transform block.

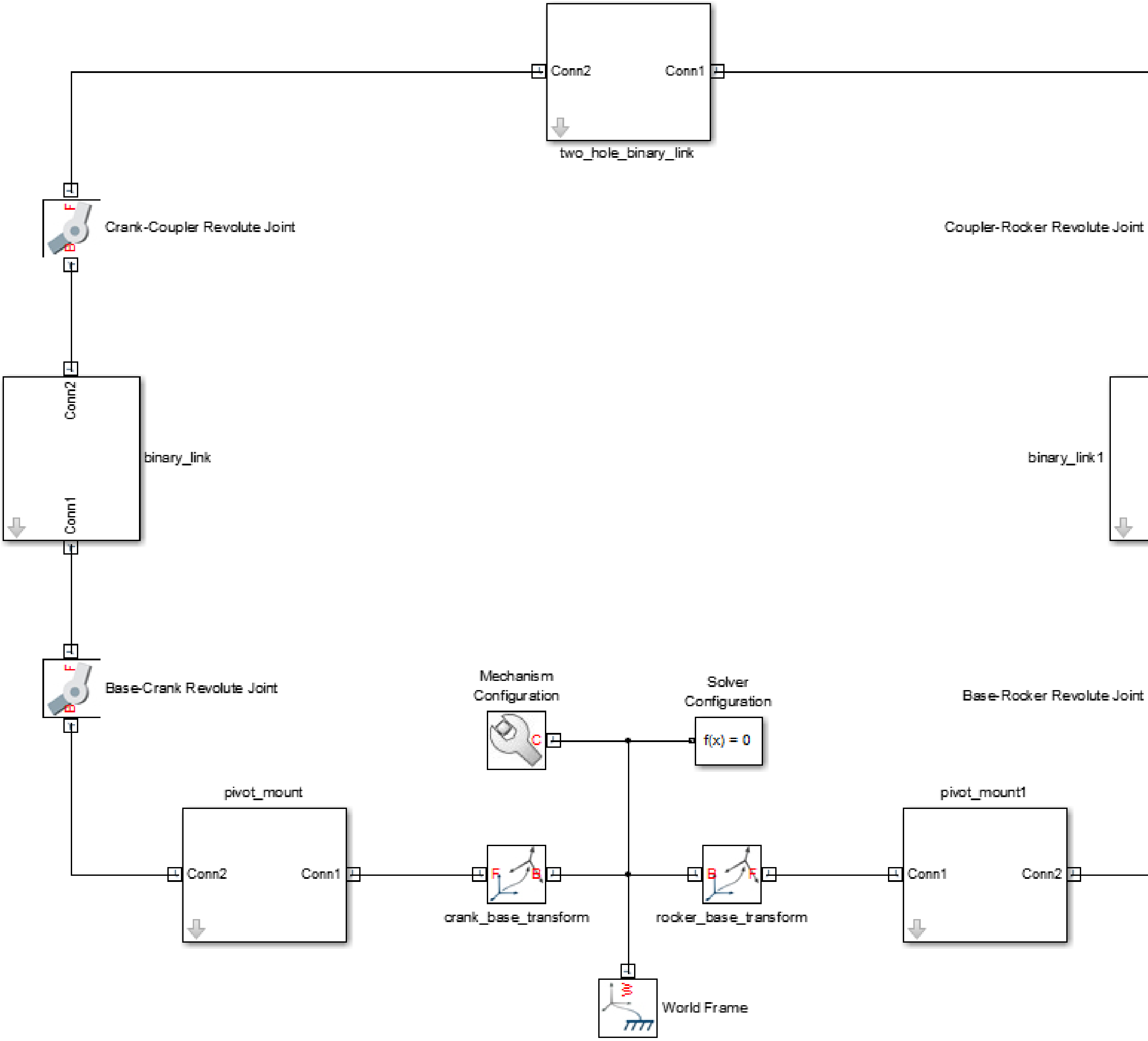


4 Drag four Revolute Joint blocks from the **SimMechanics Second Generation (SM2G) > Joints** library into the model.

5 Drag these custom blocks into the model. See the modeling tutorials if you have not created these custom blocks.

Block	Quantity	Modeling Tutorial
binary_link	2	"Model Binary Link" on page 2-57
two_hole_binary_link	1	"Model Two-Hole Binary Link" on page 2-69
pivot_mount	2	"Model Pivot Mount" on page 2-77

6 Connect the blocks as shown in the figure, paying close attention to the port names. Rotate the blocks as needed.


two_hole_binary_link
Conn2
Conn1
Crank-Coupler Revolute Joint
Coupler-Rocker Revolute Joint
binary_link
Conn2
Conn1
binary_link1
Base-Crank Revolute Joint
Mechanism Configuration
Solver Configuration
f(x) = 0
Base-Rocker Revolute Joint
pivot_mount
Conn2
Conn1
crank_base_transform
rocker_base_transform
pivot_mount1
Conn1
Conn2
World Frame
B
F
C
W

Specify Block Parameters

Specify the binary link dimensions and the spatial relationships between the pivot_mount blocks and the World frame.

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

Parameter	crank_base_transform	rocker_base_transform
Translation > Method	`Standard Axis`	`Standard Axis`
Translation > Axis	`-Y`	`+Y`
Translation > Offset	`0.15`	`0.15`

2 In the binary link block dialog boxes, specify the length parameter.

Binary Link	Length (m)
binary_link	`0.10`
two_hole_binary_link	`0.35`
binary_link1	`0.20`

Guide Assembly and Visualize Model

The model is now complete. You can now specify the desired initial state for one or more joints in the model. In this example, you specify an initial angle of 30° for the Base-Crank joint. To do this:

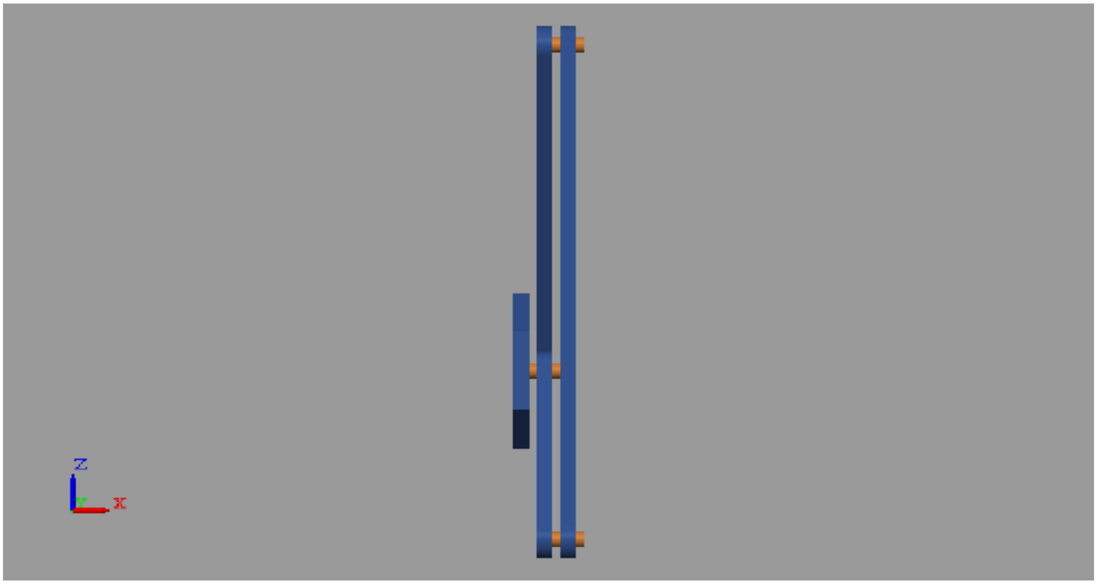
1 Double-click the Base-Crank Revolute Joint block.

2 In the block dialog box, expand **State Targets** and select **Position**.

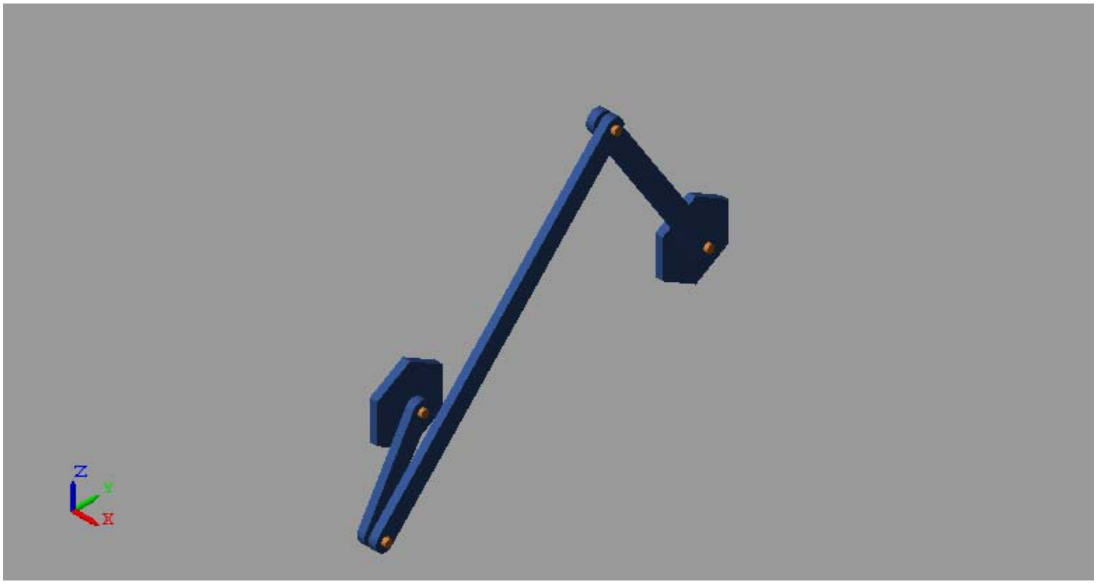
3 In **Value**, enter `-30` and press **OK**.

4 With the model window active, press **Ctrl+D**.

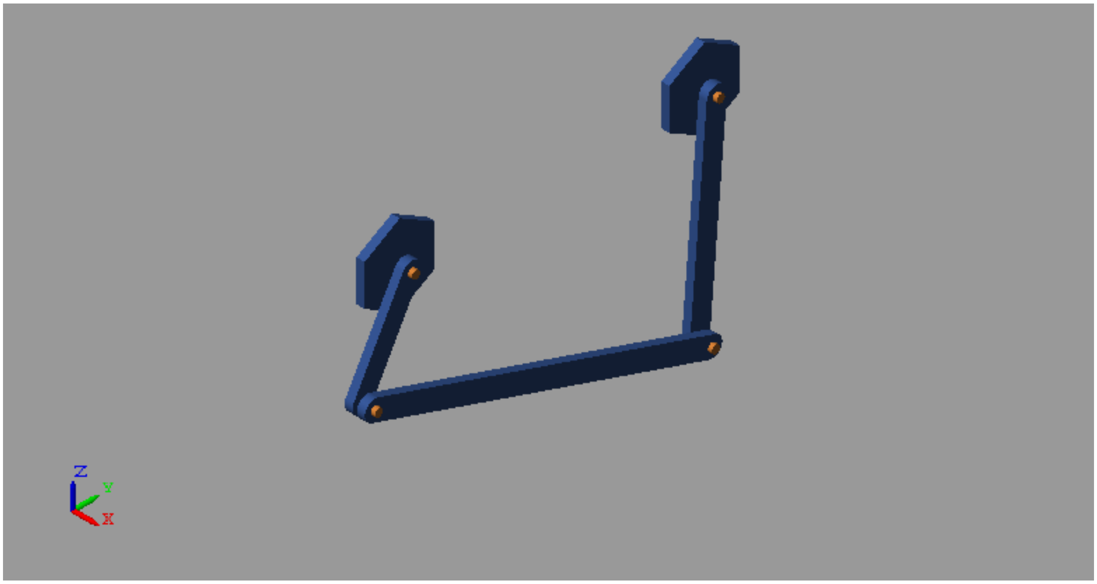
Mechanics Explorer opens with a static display of the four-bar linkage in its initial configuration. If the joint state targets that you specified are valid and compatible, the initial configuration matches those state targets. The figure shows the static display that you see in Mechanics Explorer after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



5 In the Mechanics Explorer toolstrip, click the isometric view button .



You can guide assembly so that the four-bar linkage assembles in an open configuration instead. To do this, you must specify a position state target for at least one more joint. You do not have to specify this target precisely. If you have a general idea of what the target should be, you can enter an approximate value and select a low priority level for that target. The figure shows the open initial configuration that results when you specify an additional position state target of 0 degrees for the Base-Rocker Revolute Joint block.



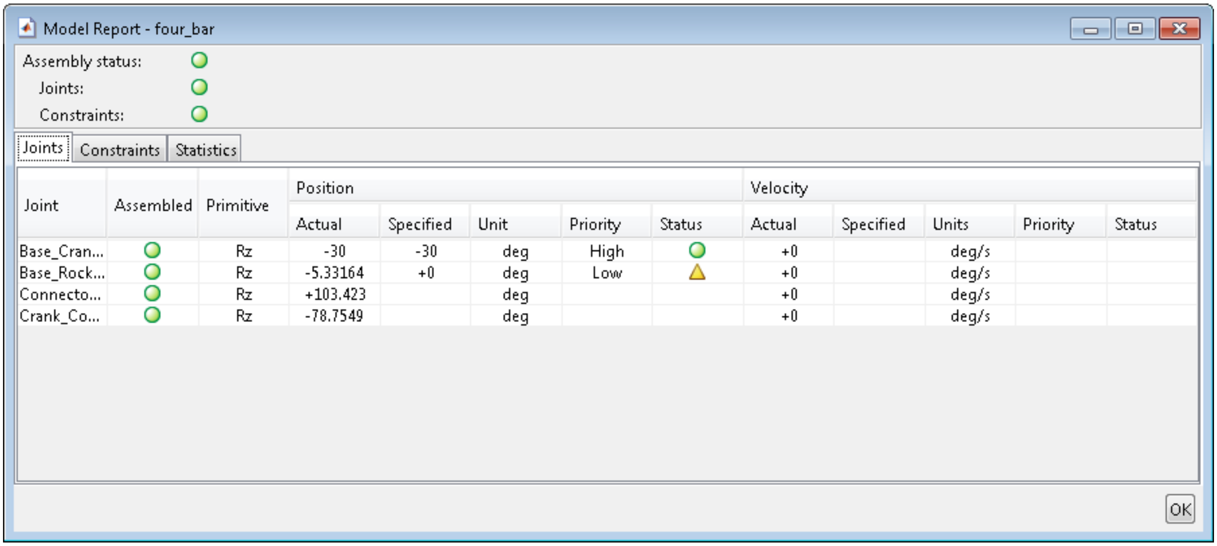
Closed-loop kinematic chains like the four-bar linkage are especially vulnerable to assembly issues. Even when the model assembles, SimMechanics may fail to meet one or more state targets. You can check the assembly status of the model and of the joints using the Model Report tool:

1 In the Mechanics Explorer menu bar, click **Tools > Model Report**.

2 Scan the model report for circles that are yellow or red.

   These circles identify issues in the assembly or in the joint state targets. A yellow or red circle in the **Position > Status** column identifies state targets that SimMechanics could not satisfy.

The figure shows the model report for the four bar linkage in the open configuration. A green circle indicates that SimMechanics satisfied the Base-Crank Revolute Joint state target precisely. A yellow circle indicates that SimMechanics satisfied the Base-Rocker Revolute Joint state target only approximately.



Assembly status: 🟢
Joints: 🟢
Constraints: 🟢

Joint	Assembled	Primitive	Position					Velocity				
			Actual	Specified	Unit	Priority	Status	Actual	Specified	Units	Priority	Status
Base_Cran...	🟢	Rz	-30	-30	deg	High	🟢	+0		deg/s		
Base_Rock...	🟢	Rz	-5.33164	+0	deg	Low	⚠	+0		deg/s		
Connecto...	🟢	Rz	+103.423		deg			+0		deg/s		
Crank_Co...	🟢	Rz	-78.7549		deg			+0		deg/s		

Simulate Model

You can now simulate the model. To do this, In the Simulink tool bar, click the **Run** button.

Save Model

In a subsequent example, you actuate the four-bar linkage and sense its motion. So that you add the actuation inputs and motion sensors to this model, save it as four_bar_linkage in a convenient folder. Then, see example "Prescribe Four-Bar Actuation Torque" on page 4-50.

Related Examples

- "Model Double Pendulum" on page 3-29
- "Correct Aiming Mechanism Assembly Error" on page 3-48

Concepts

- “Visualizing and Inspecting a Model” on page 6-2
- “Identifying Assembly Issues” on page 3-25
- “Modeling Joints” on page 3-8

Correct Aiming Mechanism Assembly Error

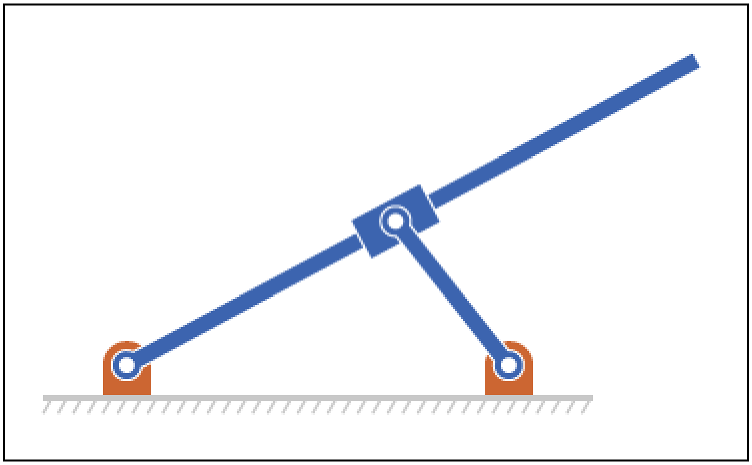
In this section...

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, SimMechanics 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.

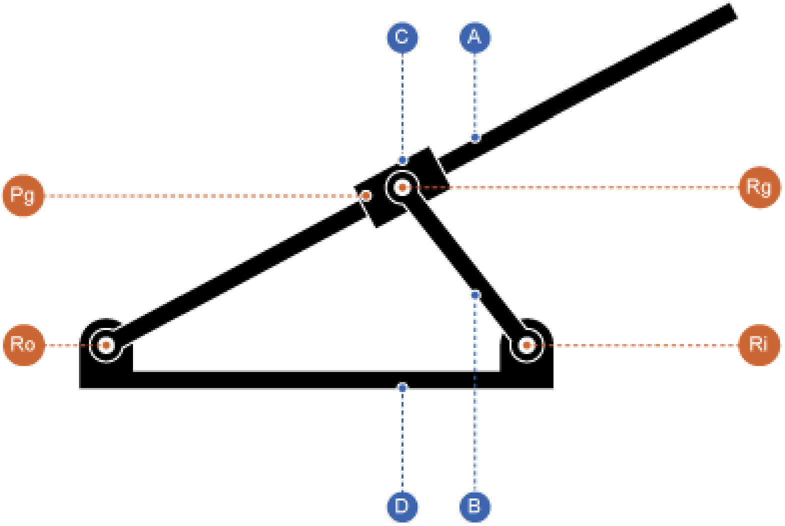
In this example, you identify the assembly error source in an aiming mechanism model using Model Report. Then, using Mechanics Explorer, you determine how to correct that error source. The `sm_dcrankaim_assembly_with_error` featured example provides the basis for this example.



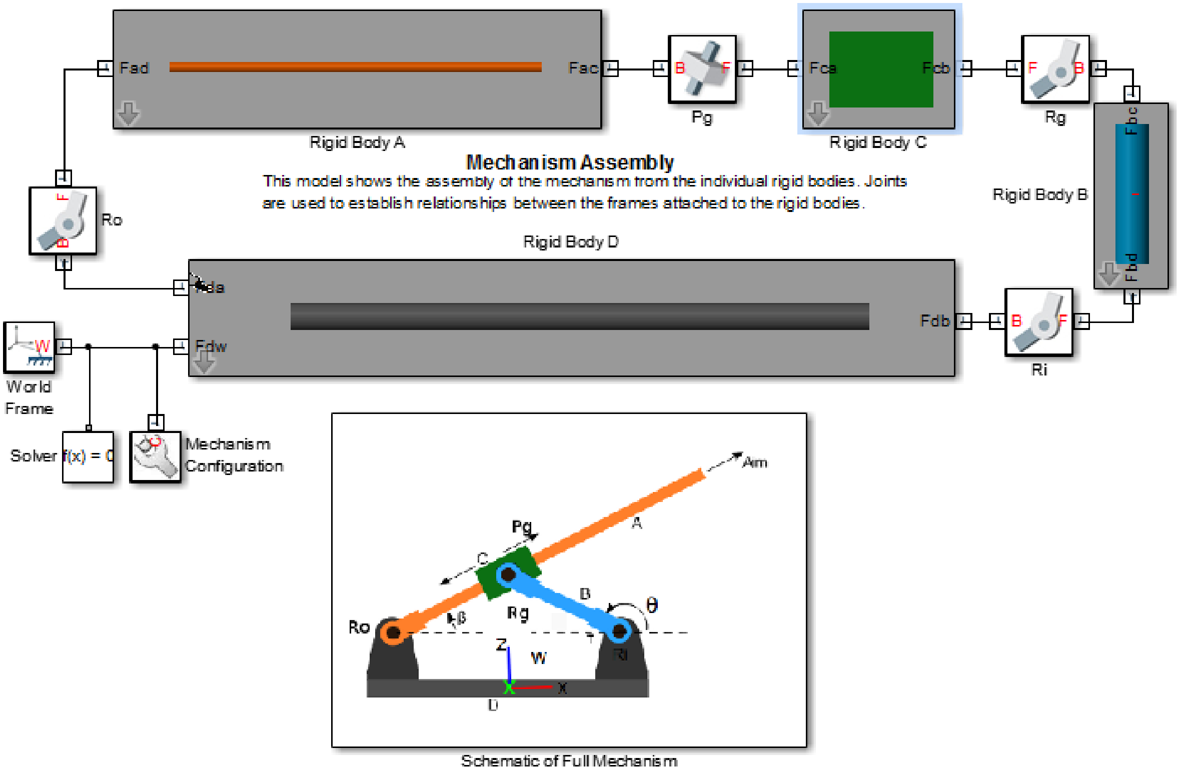
Explore Model

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.

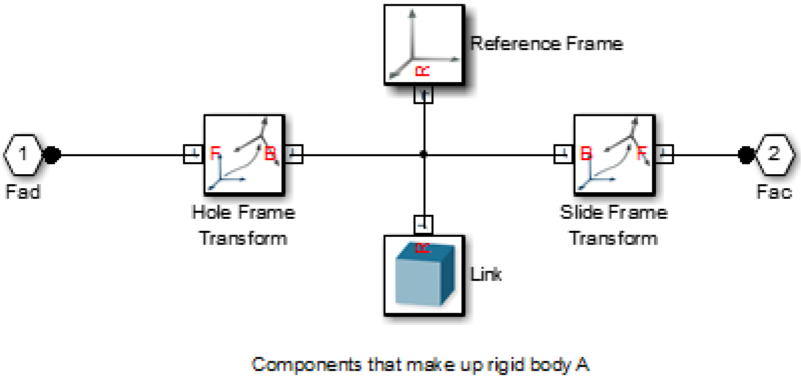


Mechanism Assembly

This model shows the assembly of the mechanism from the individual rigid bodies. Joints are used to establish relationships between the frames attached to the rigid bodies.

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.

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**.

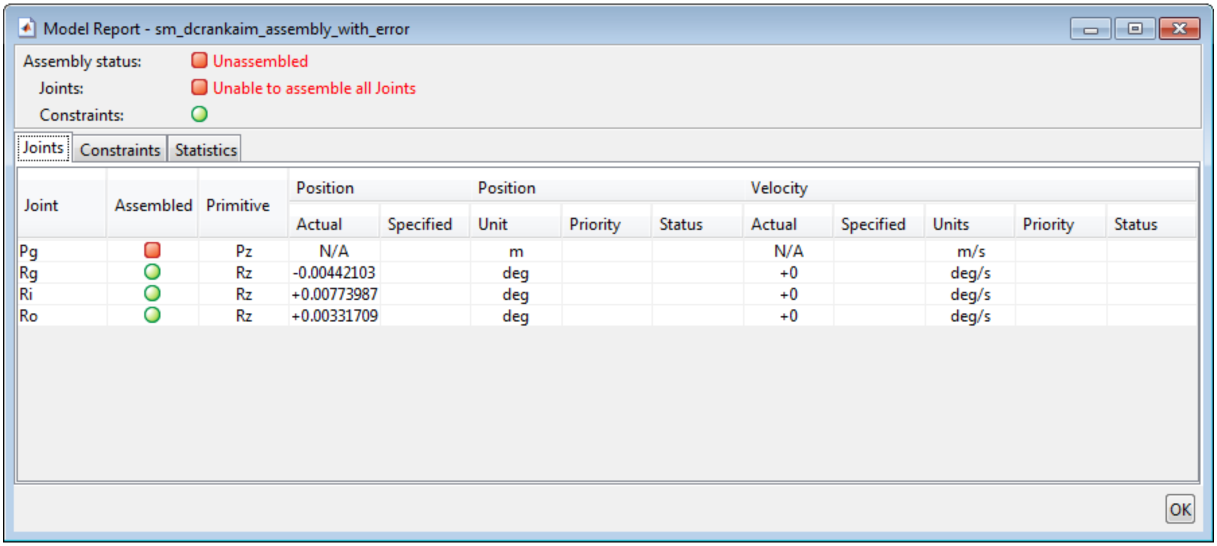
  Mechanics Explorer opens with a static display of your model in its initial state. Because the model contains an assembly error, SimMechanics issues an error message. Ignore that message for now.

Troubleshoot Assembly Error

The Mechanics Explorer utility contains a Model Report tool which identifies the status of joints and constraints. Use the Model Report utility to determine if joints and constraints have successfully assembled. The report also provides information on the configuration of joints and constraints. To see the report:

- On the Mechanics Explorer menu bar, select **Tools > Model Report**.

The model report opens in a new window. The top section of this window identifies the assembly status of the model as a whole. Because assembly failed, the assembly status for this model is **Unassembled**. Because the source of the assembly error is a joint block, the assembly status for joints is "Unable to assembly all joints".



Assembly status: 🟥 Unassembled
Joints: 🟥 Unable to assemble all Joints
Constraints: 🟢

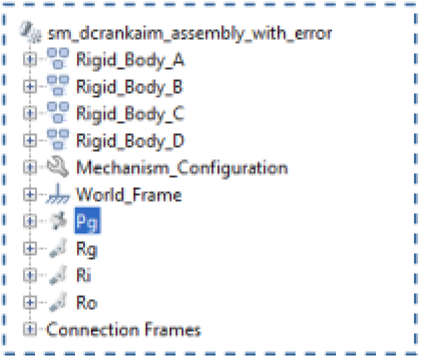
Joint	Assembled	Primitive	Position		Position			Velocity				
			Actual	Specified	Unit	Priority	Status	Actual	Specified	Units	Priority	Status
Pg	🟥	Pz	N/A		m			N/A		m/s		
Rg	🟢	Rz	-0.00442103		deg			+0		deg/s		
Ri	🟢	Rz	+0.00773987		deg			+0		deg/s		
Ro	🟢	Rz	+0.00331709		deg			+0		deg/s		

To identify the joint block that caused assembly fail, examine the model report **Joints** tab. A red square identifies the problematic joint—Pg. This joint did not assemble, and the model report provides neither position nor velocity information for that joint.

Identifying Error Root Cause

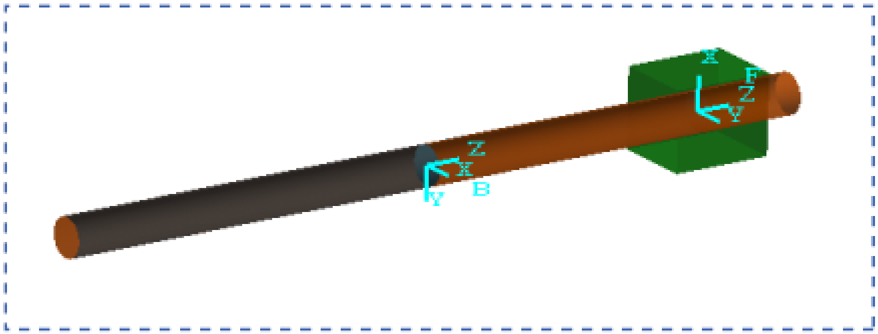
The error message that SimMechanics issued during model update identifies position violation as the root cause of assembly failure. This suggests that the frames which joint Pg connects to are improperly aligned. To confirm this hypothesis, check the orientation of these frames in Mechanics Explorer.

1 In the Mechanics Explorer tree view pane, click joint Pg.



sm_dcrankaim_assembly_with_error
- Rigid_Body_A
- Rigid_Body_B
- Rigid_Body_C
- Rigid_Body_D
- Mechanism_Configuration
- World_Frame
- Pg
- Rg
- Ri
- Ro
- Connection Frames

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



The two frames are offset along the Z axis. Because joint Pg provides a translational degree of freedom along the Z axis, this offset is valid. However, the follower frame has a different orientation than the base frame. It contains a rotation angle of 90 degrees about the common Z axis with respect to the base frame. Because joint Pg provides no rotational degrees of freedom, this rotation angle is invalid. This angle prevents the two joint frames from aligning correctly, causing assembly to fail.

Correct Assembly Error

To correct the assembly error, you must change the orientation of either base or follower joint frames. In this example, you change the orientation of the follower frame so that the model can assemble successfully. To change the orientation of this frame:

1 Right-click the Rigid Body C subsystem block and select **Mask > Look Under Mask**.

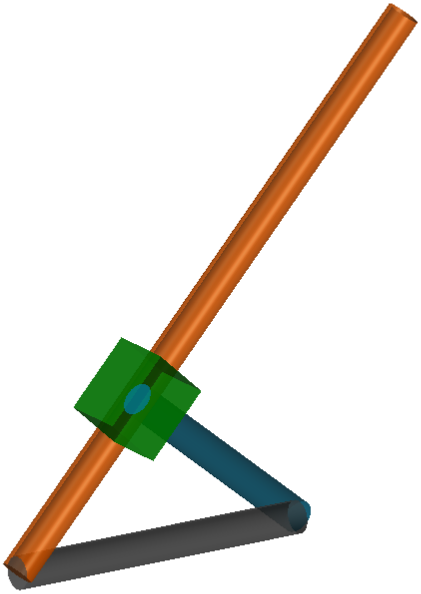
2 Double-click the **Slide Frame Transform** block and select the new parameter values that the table provides.

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

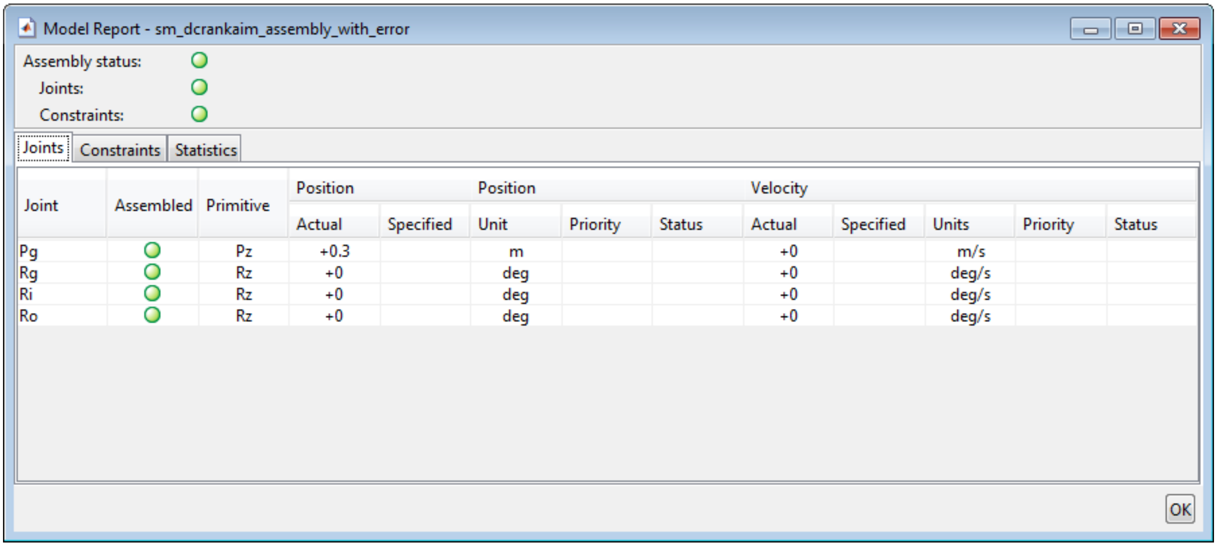
3 Click **OK**.

Simulate Model

You can now simulate the model. On the Simulink menu bar, select **Simulation > Run**. Mechanics Explorer opens with a dynamic display of your model. The figure shows this display. Rotate, pan, and zoom to explore.



You can use the Model Report tool to verify the assembly status. To do this, on the Mechanics Explorer menu bar, select **Tools > Model Report**. The model report opens in a new window. Check that the assembly status icon for the model and its joints is a green square. The green square indicates that assembly has been successful.



Model Report - sm_dcrankaim_assembly_with_error

Assembly status: ●

Joints: ●

Constraints: ●

Joint	Assembled	Primitive	Position		Position			Velocity				
			Actual	Specified	Unit	Priority	Status	Actual	Specified	Units	Priority	Status
Pg	●	Pz	+0.3		m			+0		m/s		
Rg	●	Rz	+0		deg			+0		deg/s		
Ri	●	Rz	+0		deg			+0		deg/s		
Ro	●	Rz	+0		deg			+0		deg/s		

Save Model

Save the model as aiming_mechanism in a convenient folder. In a subsequent example, you prescribe a joint trajectory using this model.

Related Examples

- “Model Double Pendulum” on page 3-29
- “Model Four-Bar Linkage” on page 3-37

Concepts

- “Visualizing and Inspecting a Model” on page 6-2
- “Identifying Assembly Issues” on page 3-25
- “Modeling Joints” on page 3-8

Internal Mechanics, Actuation and Sensing

Forces and Torques

In this section...

Force and Torque Blocks

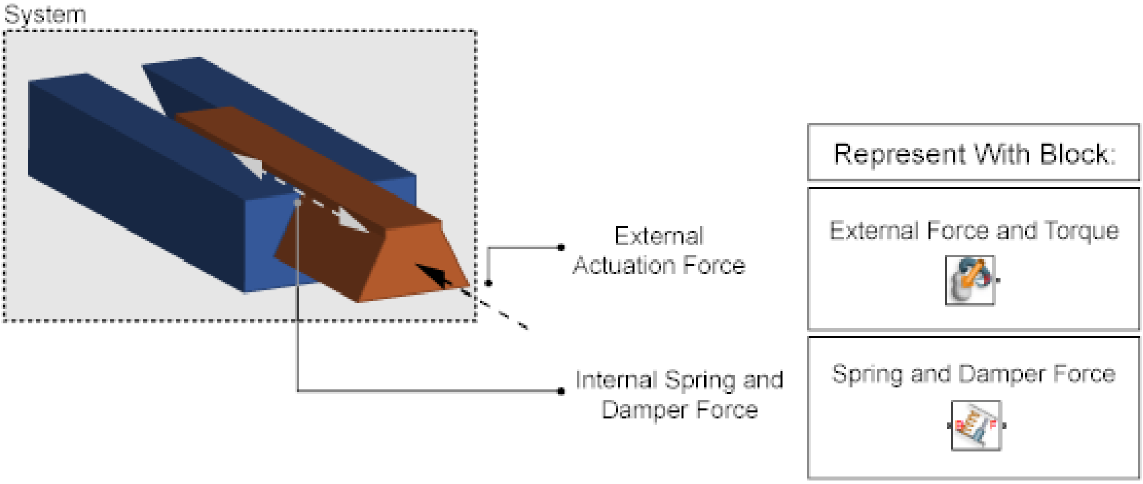
You can apply different forces and torques to a model. The table summarizes the different forces and torques that you can represent using SimMechanics blocks. For detailed information about these blocks, see the block reference pages.

Block	Description
External Force and Torque	General force and torque arising outside a model
Internal Force	General force acting between two rigid bodies in a model
Spring and Damper Force	Internal force, acting between two rigid bodies in a model, that accounts for energy storage and dissipation in your model
Inverse Square Law Force	Internal force, acting between two rigid bodies in a model, with a 1/R^2 dependence. Examples include gravity and Coulomb forces.

Actuating Rigid Bodies

You can actuate a rigid body directly using blocks from the Forces and Torques library. Use the External Force and Torque block to represent an actuation input that arises outside your model. Use the remaining blocks to represent forces that are internal to your model.

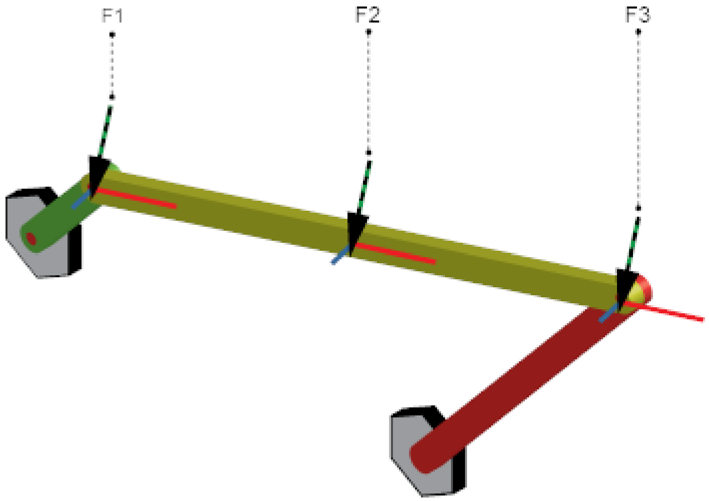
The figure illustrates external and internal forces acting on a mechanical system. An external force provides the actuation input to the system. This can be a constant or a general time-dependent input. A spring and damper force acting between the two bodies in the system accounts for energy storage and dissipation. You represent the actuation input using the External Force and Torque block. You represent the internal spring and damper force using the Spring and Damper Force block.



The Forces and Torques blocks contain frame ports. These ports identify the rigid body frames the forces/torques act on. If the block represents an internal force, the block contains two frame ports. Connect these ports to the two rigid bodies the force/torque acts on. If the block represents an external force or torque, the block contains one frame port. Connect this port to the rigid body frame the external force or torque acts on.

The frame origin identifies the point of application for a force or torque. The frame axes identify the directions of the X, Y, and Z force/torque vector components that you specify. Changing the frame position changes also the force/torque application point. Likewise, changing the frame orientation changes also the force/torque direction.

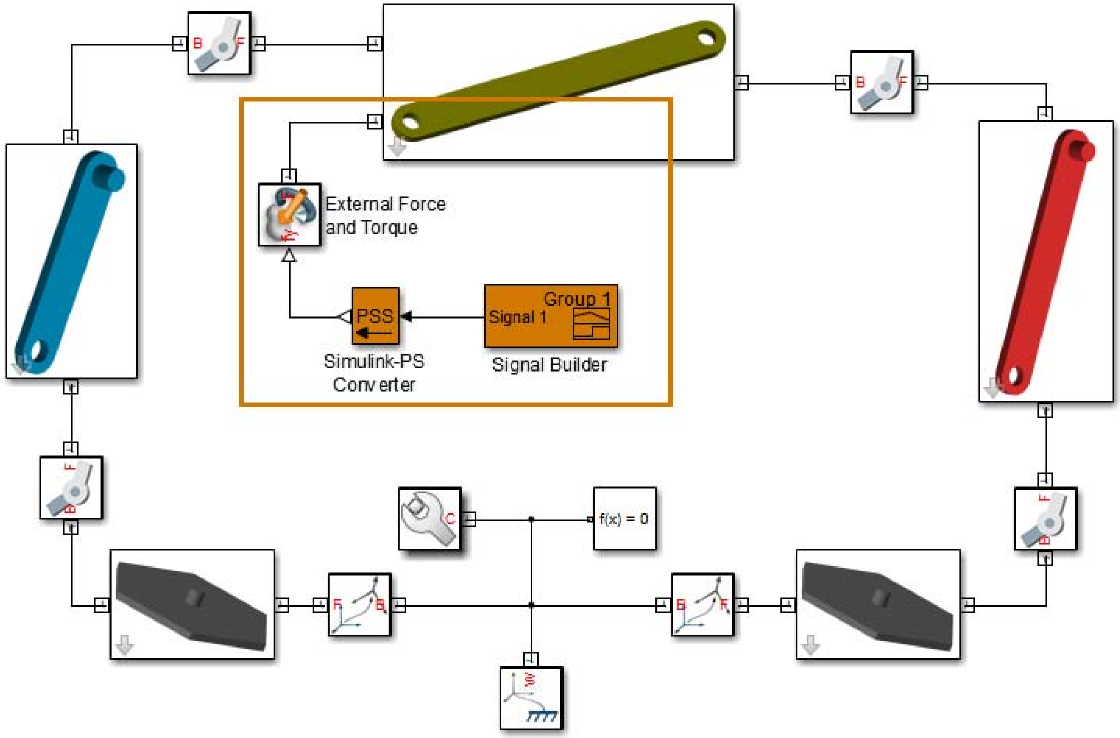
The figure shows three external forces that you can apply to the rocker link of a four-bar mechanism—F1, F2, and F3. Forces F1 and F3 act at the ends of the link, while force F2 acts at its mass center.



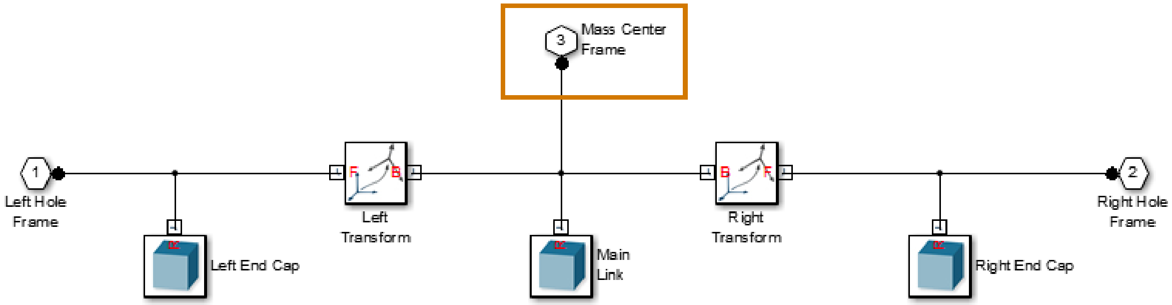
To represent one of these forces in a SimMechanics model, you first define the frame to apply that force to. Example "Represent Binary Link Frame Tree" on page 1-36 shows you how to do this. Then, in the block diagram for your model, connect the frame port of an External Force and Torque block to the frame entity that represents that frame—frame port, line, or node. For more information, see "Representing Frames" on page 1-7.

Finally, in the block dialog box, select the force component(s) that you want to specify. For example, to specify a force acting along the -Y axis of the frame it connects to, select **Force > Force (Y)**. Then, use the physical signal port that the block exposes to input the value of that force component. That value is negative for a force acting along the -Y axis.

The figure shows the modified block diagram of a four-bar model that is present in your SimMechanics installation. You can open the original model by typing `sm_four_bar` at the MATLAB command line.



The rectangular frame in the image highlights the blocks that you can use to apply an external force. The frame port that the External Force and Torque block connects to represents the binary link mass center. The block diagram of the binary link subsystem provides this frame. The figure shows the block diagram.



In the External Force and Torque block, physical signal port fy identifies the force component that the block represents—in this case, a force in the Y direction of the frame that the block connects to.

Related Examples

- “Prescribe Four-Bar Actuation Torque” on page 4-50

Concepts

- “Joint Actuation” on page 4-7
- “Actuating and Sensing with Physical Signals” on page 4-18
- “Representing Frames” on page 1-7

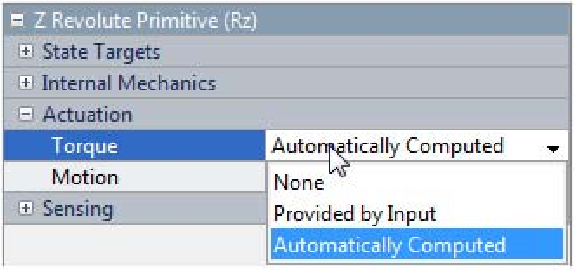
Joint Actuation

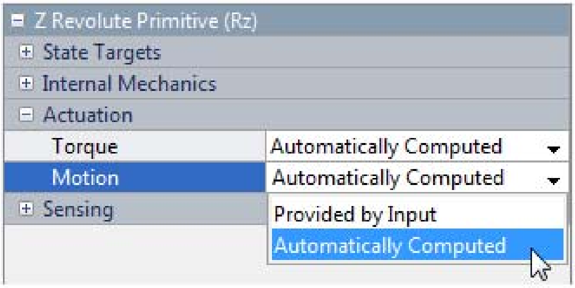
In this section…

Actuation Modes

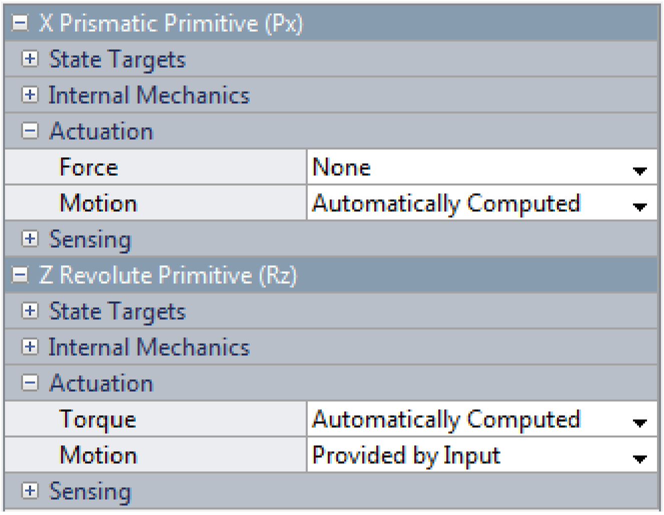
Joint blocks provide two actuation parameters. These parameters, **Force/Torque** and **Motion**, govern how the joint behaves during simulation. Depending on the parameter settings you select, a joint block can accept either actuation parameter as input or automatically compute its value during simulation.

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.


Z Revolute Primitive (Rz)
State Targets
Internal Mechanics
Actuation
Torque: Automatically Computed
Motion
Sensing


Z Revolute Primitive (Rz)
State Targets
Internal Mechanics
Actuation
Torque: Automatically Computed
Motion: Automatically Computed
Sensing

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

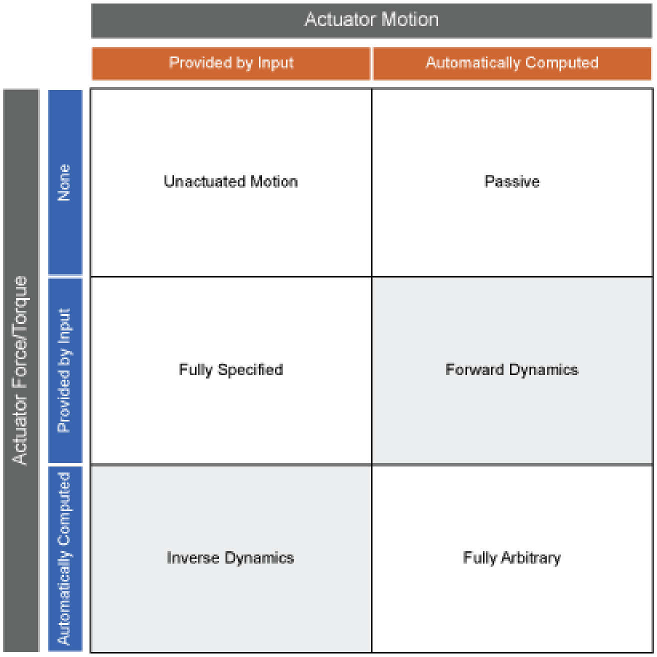
⊟ Actuation

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



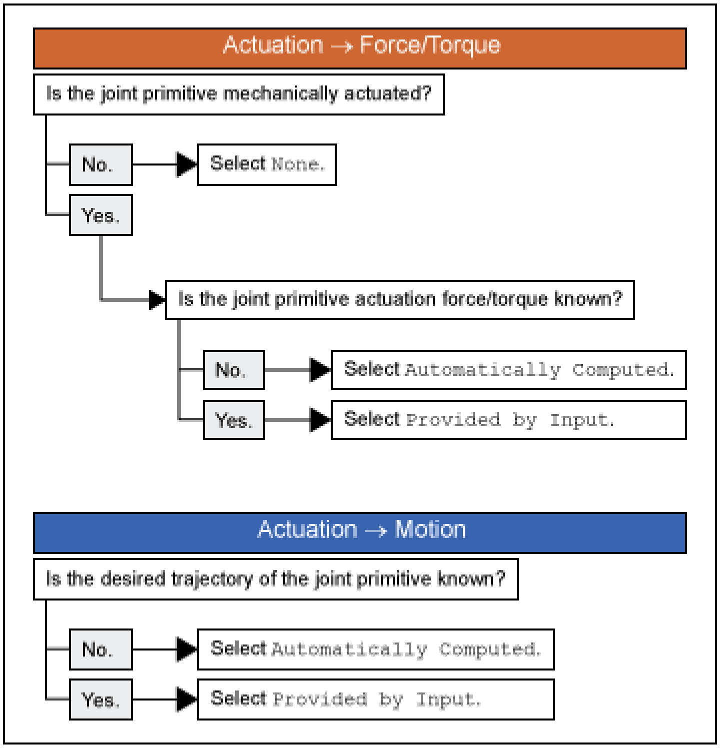
		Actuator Motion	
		Provided by Input	Automatically Computed
Actuator Force/Torque	None	Unactuated Motion	Passive
	Provided by Input	Fully Specified	Forward Dynamics
	Automatically Computed	Inverse Dynamics	Fully Arbitrary

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.


Actuation → Force/Torque
Is the joint primitive mechanically actuated?
No.
Select None.
Yes.
Is the joint primitive actuation force/torque known?
No.
Select Automatically Computed.
Yes.
Select Provided by Input.
Actuation → Motion
Is the desired trajectory of the joint primitive known?
No.
Select Automatically Computed.
Yes.
Select Provided by Input.

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.

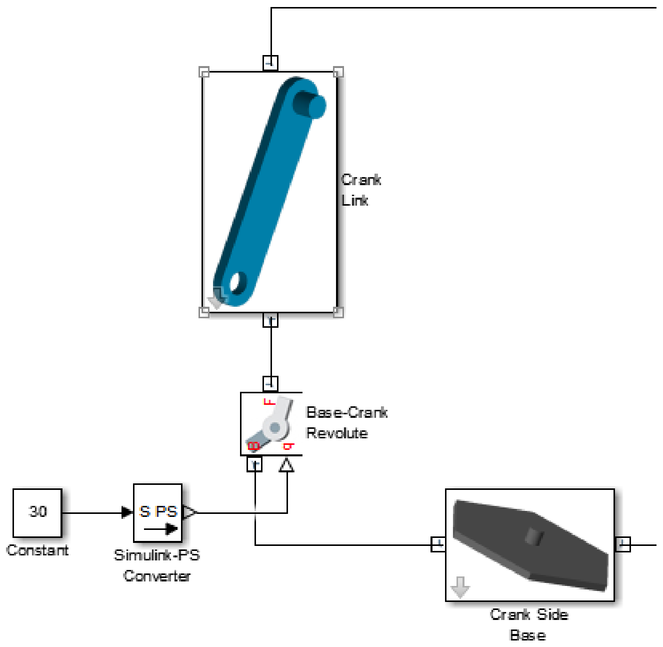
Spherical joint primitives provide no motion actuation options. You can specify actuation torque for these primitives, but you cannot prescribe their trajectories. Those trajectories are always automatically computed from the model dynamics during simulation.

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 SimMechanics, 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 than 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 SimMechanics 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.

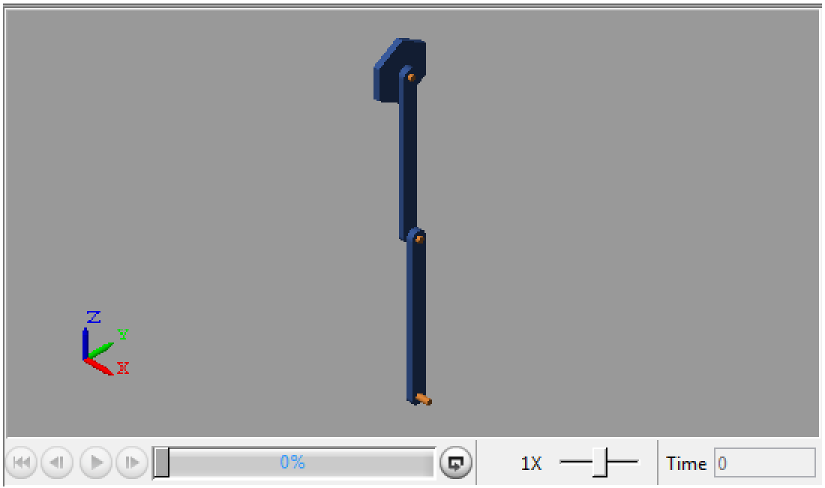
Assembly and Simulation

SimMechanics 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.

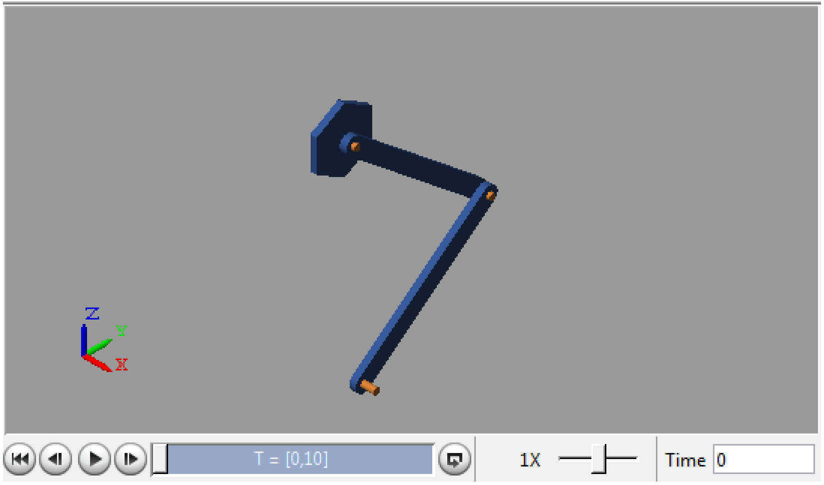
Note You obtain the assembled state each time you update the block diagram, e.g., by pressing **Ctrl+D**. You obtain the initial simulation state each time you run the simulation, e.g., by pressing **Ctrl+T**, and pausing at time zero.

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.



Time 0



Time 0

Related Examples

- “Prescribe Two-Bar Motion” on page 4-78
- “Prescribe Four-Bar Motion” on page 4-66
- “Specify Motion Input Derivatives” on page 4-15

Specify 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 Two-Bar Motion" on page 4-78
- "Prescribe Four-Bar Motion" on page 4-66

Concepts

- "Joint Actuation" on page 4-7

Joint Motion Actuation Restrictions

In this section...

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 Two-Bar Motion" on page 4-78 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 SimMechanics 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 Two-Bar Motion” on page 4-78
- “Prescribe Four-Bar Motion” on page 4-66
- “Specify Motion Input Derivatives” on page 4-15

Concepts

- “Joint Actuation” on page 4-7

Actuating and Sensing with Physical Signals

In this section...

Some SimMechanics blocks provide physical signal ports for actuation input or sensing output. These ports accept 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 SimMechanics, 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.

Providing Actuation Signals

To provide an actuation signal based on Simulink blocks, you use the Simulink-PS Converter block:

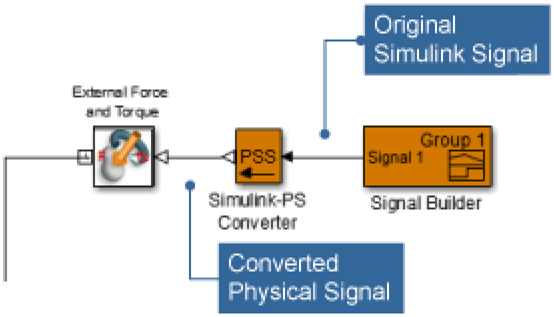
1 Build the Simulink block diagram to represent the actuation signal

   This diagram can be as simple as a single block.

2 Connect the Simulink signal from the block diagram 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 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 SimMechanics blocks.



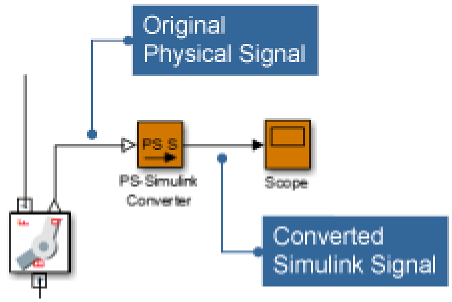
Extracting Sensing Signals

To connect the sensing signal of a SimMechanics block to a Simulink block, you use the PS-Simulink Converter block:

1 Connect the SimMechanics 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 SimMechanics sensing signal to a Simulink Scope block.



Related Examples

- "Prescribe Four-Bar Actuation Torque" on page 4-50

Concepts

- "Forces and Torques" on page 4-2
- "Representing Frames" on page 1-7

Sensing Spatial Relationships

In SimMechanics, 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 SimMechanics 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 Relationship 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 Relationship Between Arbitrary Frames" on page 4-23.

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 Relationship 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 4-40. For more information about frame lines and ports, see “Representing Frames” on page 1-7.

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 don't 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
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

Parameter	Rotation Port	Angular Velocity Port	Angular Acceleration Port
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 Double-Pendulum Motion" on page 4-43
- "Prescribe Four-Bar Actuation Torque" on page 4-50

Concepts

- "Rotation Measurements" on page 4-26
- "Translation Measurements" on page 4-31
- "Measurement Frames" on page 4-40

Rotation Measurements

In this section...

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 [1]. The choice of measurement format depends on the model. Select the format that is most convenient for the application.

Measuring Rotation

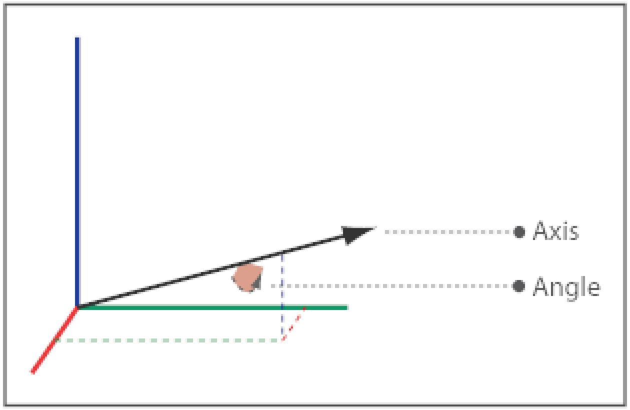
Rotation is a relative quantity. The rotation of one frame is meaningful only with respect to another frame. As such, blocks with rotation sensing capability require two frames to make a measurement: measured and reference frames. In these blocks, the follower frame port identifies the measured frame; the base frame port identifies the reference frame of the measurement.

SimMechanics 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.

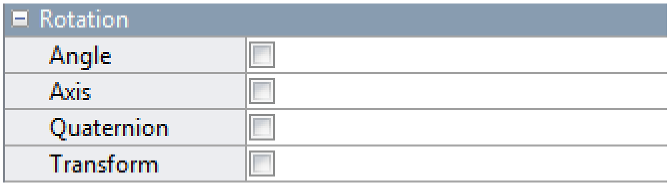
Axis-Angle Measurements

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 theorem, any 3–D rotation or rotation sequence can be described as a pure rotation about a single fixed axis.

[1] Weld Joint is an exception



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 axs and q, respectively). Because the axis-angle parameters are listed separately, you can choose to measure the axis, the angle, or both.



⊟ Rotation

Angle	□
Axis	□
Quaternion	□
Transform	□

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–π. 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.

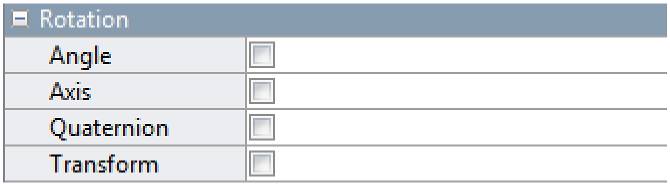
Quaternion Measurements

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 SimMechanics, gimbal lock causes numerical errors that lead to simulation failure. The absence of singularities means that quaternions are more robust for simulation purposes.

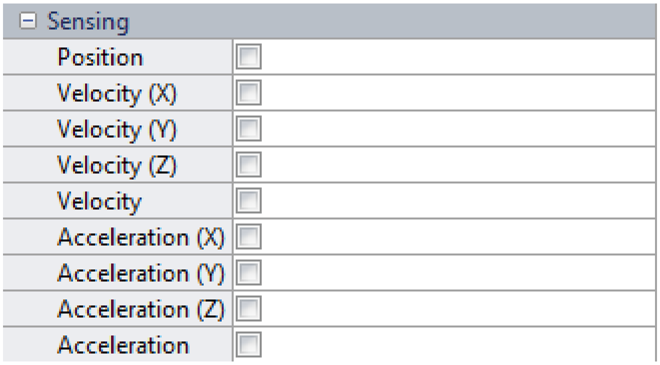
To measure frame rotation in quaternion format, use:

- 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	□
Axis	□
Quaternion	□
Transform	□

- 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	□
Velocity (X)	□
Velocity (Y)	□
Velocity (Z)	□
Velocity	□
Acceleration (X)	□
Acceleration (Y)	□
Acceleration (Z)	□
Acceleration	□

Note The Transform Sensor block provides a quaternion option for rotation and its first two time-derivatives (angular velocity and acceleration). On the other hand, the Spherical Joint block provides a quaternion option only for rotation — angular velocity and acceleration are measured only in terms of rotation axis and angle.

The quaternion output is a 4-element row vector $Q = \left(S \quad \boldsymbol{V}\right)$, where:

$$S = \cos\left(\theta/2\right)$$

and

$$\mathbf{V} = [V_x V_y V_z]\sin\left(\frac{\theta}{2}\right)$$

θ is the rotation angle. The angle can take any value between 0–π. [V_x, V_y, V_z] is the rotation axis. Axis components can take any value between 0–1.

Transform Measurements

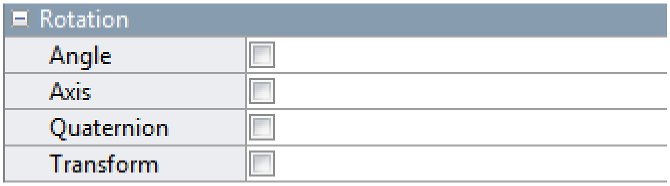
The rotation transform is a 3×3 matrix that encodes frame rotation. In terms of base frame axes (X, Y, and Z), the follower frame axes (X', Y', and Z') are:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

The transform describes the rotation required to bring one frame into coincidence with another frame. In terms of Transform Sensor port frames, the rotation transform describes the rotation that brings the base port frame into coincidence with the follower port frame.

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 measured 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 base frame to the follower frame.

To measure frame rotation in transform format, use the Transform Sensor block. The block dialog box contains a **Transform** option that you can select to expose the corresponding physical signal port (labeled R). The transform output is a nine-element matrix with elements valued between 0–1.



⊟ Rotation

Angle	□
Axis	□
Quaternion	□
Transform	□

Related Examples

- “Sense Double-Pendulum Motion” on page 4-43
- “Prescribe Four-Bar Actuation Torque” on page 4-50

Concepts

- “Sensing Spatial Relationships” on page 4-21
- “Translation Measurements” on page 4-31
- “Measurement Frames” on page 4-40

Translation Measurements

In this section...

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, in joint blocks [2]. The choice of coordinate system depends on the model. Select the coordinate system that is most convenient for the application.

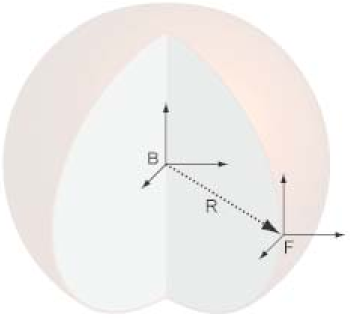
Measuring Translation

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.

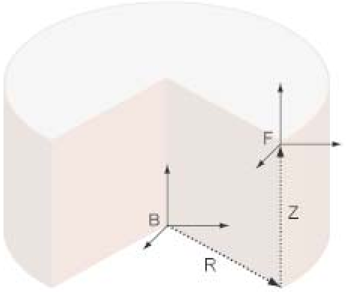
Some measurements are common to multiple coordinate systems. One example is the Z-coordinate, which exists in both Cartesian and cylindrical systems. In the Transform Sensor dialog box, coordinates that make up more than one coordinate system appear only once. Selecting **Z** outputs translation along 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.

[2] Weld Joint is an exception



Spherical Radius



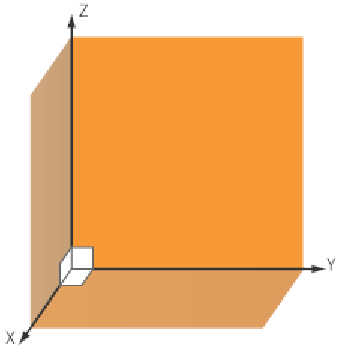
Cylindrical Radius

To differentiate between the two radial coordinates, SimMechanics uses the following convention:

- Radius — Cylindrical radial coordinate
- Distance — Spherical radial coordinate

Cartesian Measurements

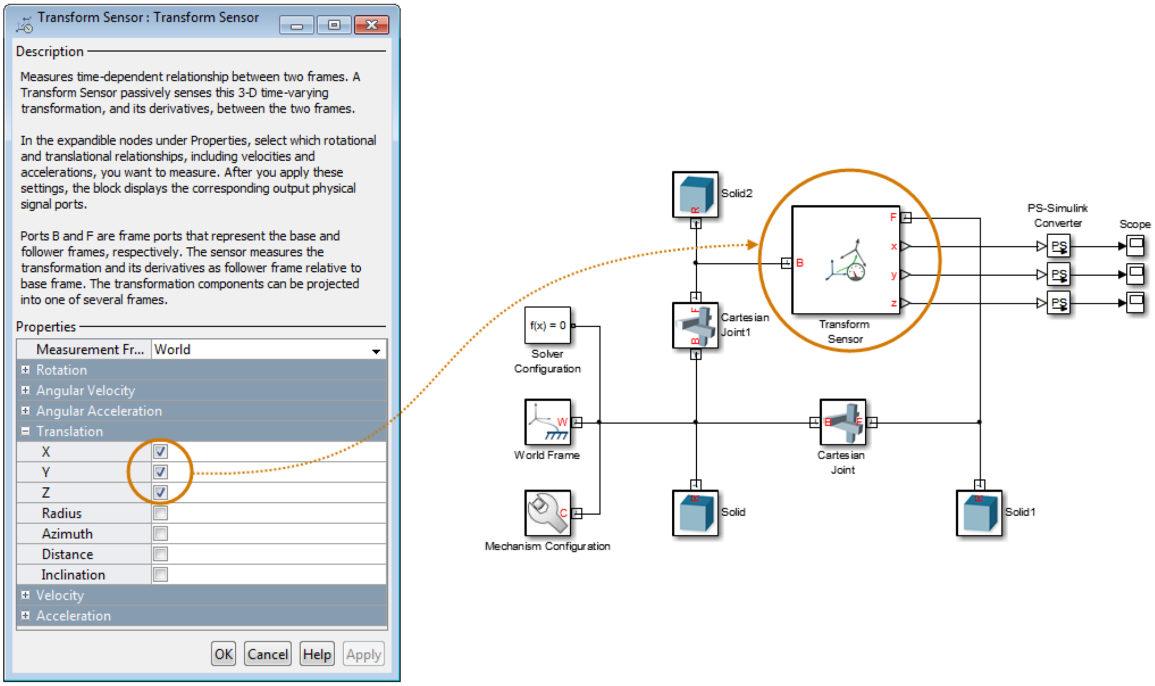
The Cartesian coordinate system uses three linear coordinates — X, Y, and Z — corresponding to three mutually orthogonal axes. Cartesian translation measurements have units of distance, with meter being the default. You can use the PS-Simulink Converter block to select a different physical unit when 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 1-7.



Transform Sensor : Transform Sensor

Description

Measures time-dependent relationship between two frames. A Transform Sensor passively senses this 3-D time-varying transformation, and its derivatives, between the two frames.

In the expandible nodes under Properties, select which rotational and translational relationships, including velocities and accelerations, you want to measure. After you apply these settings, the block displays the corresponding output physical signal ports.

Ports B and F are frame ports that represent the base and follower frames, respectively. The sensor measures the transformation and its derivatives as follower frame relative to base frame. The transformation components can be projected into one of several frames.

Properties

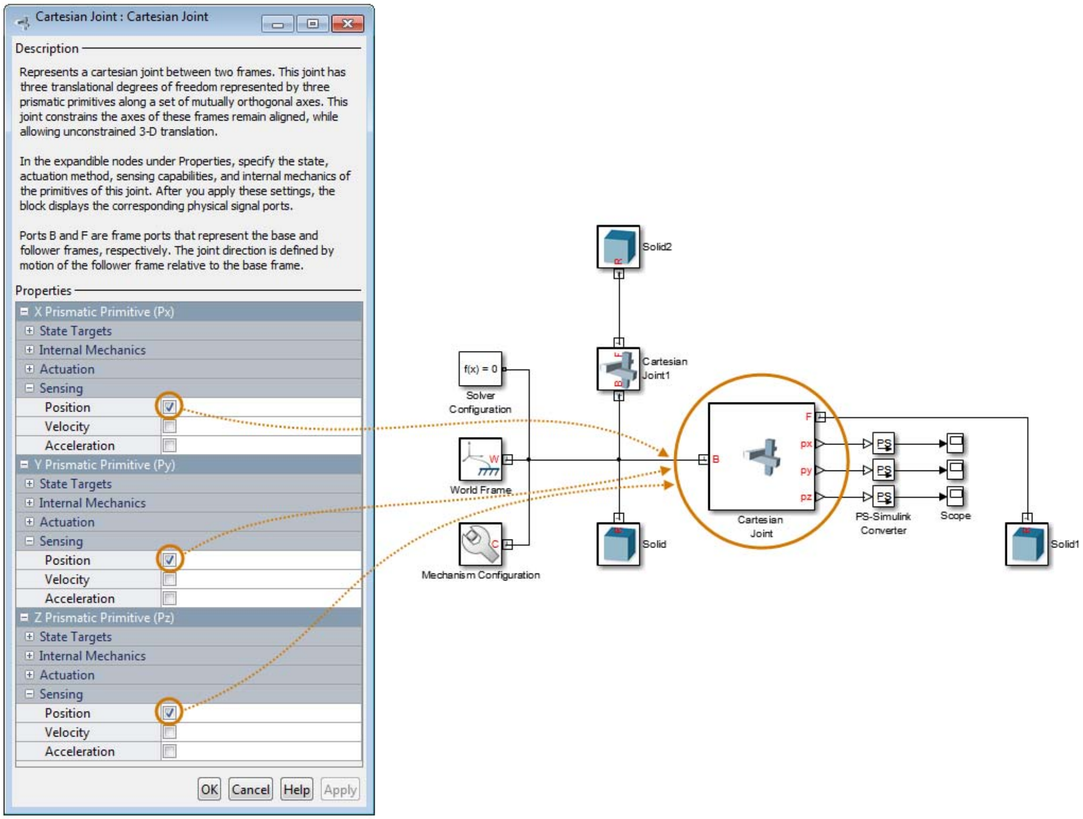
Measurement Fr...	World
⊞ Rotation	
⊞ Angular Velocity	
⊞ Angular Acceleration	
⊟ Translation	
X	☑
Y	☑
Z	☑
Radius	☐
Azimuth	☐
Distance	☐
Inclination	☐
⊞ Velocity	
⊞ Acceleration	

OK Cancel Help Apply

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. If the joint contains zero prismatic primitives, the joint frames cannot translate with respect to each other, and translation sensing is not necessary. For this reason, joint blocks with zero prismatic primitives do not sense translation.

The figure shows a simple model using a Cartesian Joint 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 Solid reference port frames.



Cartesian Joint : Cartesian Joint

Description

Represents a cartesian joint between two frames. This joint has three translational degrees of freedom represented by three prismatic primitives along a set of mutually orthogonal axes. This joint constrains the axes of these frames remain aligned, while allowing unconstrained 3-D translation.

In the expandible nodes under Properties, specify the state, actuation method, sensing capabilities, and internal mechanics of the primitives of this joint. After you apply these settings, the block displays the corresponding physical signal ports.

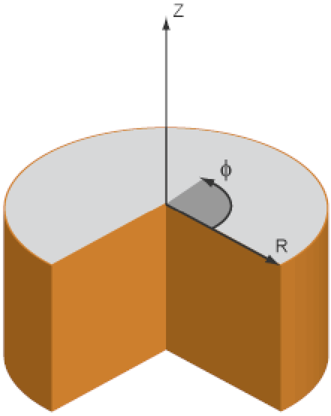
Ports B and F are frame ports that represent the base and follower frames, respectively. The joint direction is defined by motion of the follower frame relative to the base frame.

Properties

- X Prismatic Primitive (Px)
  - State Targets
  - Internal Mechanics
  - Actuation
  - Sensing
    - Position ☑
    - Velocity □
    - Acceleration □
- Y Prismatic Primitive (Py)
  - State Targets
  - Internal Mechanics
  - Actuation
  - Sensing
    - Position ☑
    - Velocity □
    - Acceleration □
- Z Prismatic Primitive (Pz)
  - State Targets
  - Internal Mechanics
  - Actuation
  - Sensing
    - Position ☑
    - Velocity □
    - Acceleration □

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, φ, 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.

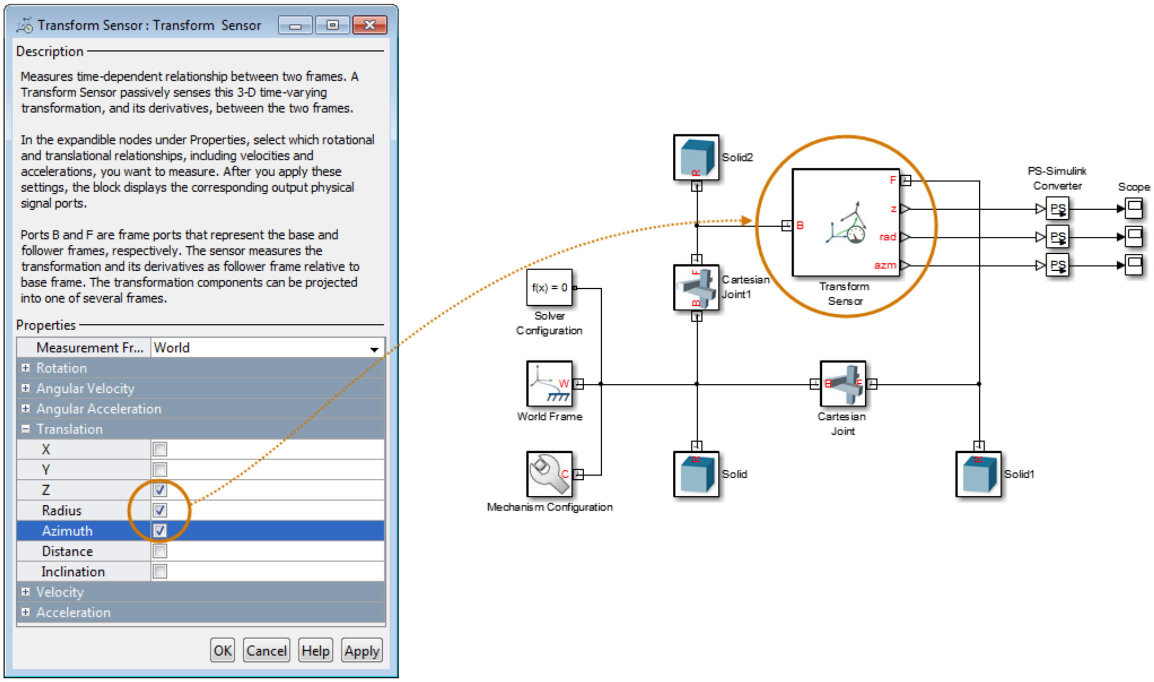


Transform Sensor

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.



Transform Sensor : Transform Sensor

Description

Measures time-dependent relationship between two frames. A Transform Sensor passively senses this 3-D time-varying transformation, and its derivatives, between the two frames.

In the expandible nodes under Properties, select which rotational and translational relationships, including velocities and accelerations, you want to measure. After you apply these settings, the block displays the corresponding output physical signal ports.

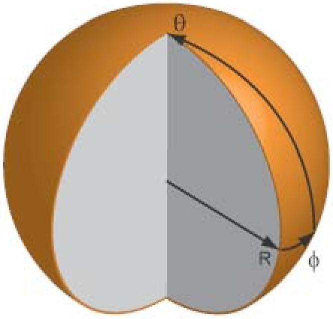
Ports B and F are frame ports that represent the base and follower frames, respectively. The sensor measures the transformation and its derivatives as follower frame relative to base frame. The transformation components can be projected into one of several frames.

Properties

Measurement Fr...	World
⊞ Rotation	
⊞ Angular Velocity	
⊞ Angular Acceleration	
⊟ Translation	
X	□
Y	□
Z	☑
Radius	☑
Azimuth	☑
Distance	□
Inclination	□
⊞ Velocity	
⊞ Acceleration	

Spherical Measurements

The spherical coordinate system uses two angular and one linear coordinates. The linear coordinate is the spherical radius, R. The angular coordinates are the azimuth, φ, and inclination, θ. 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.

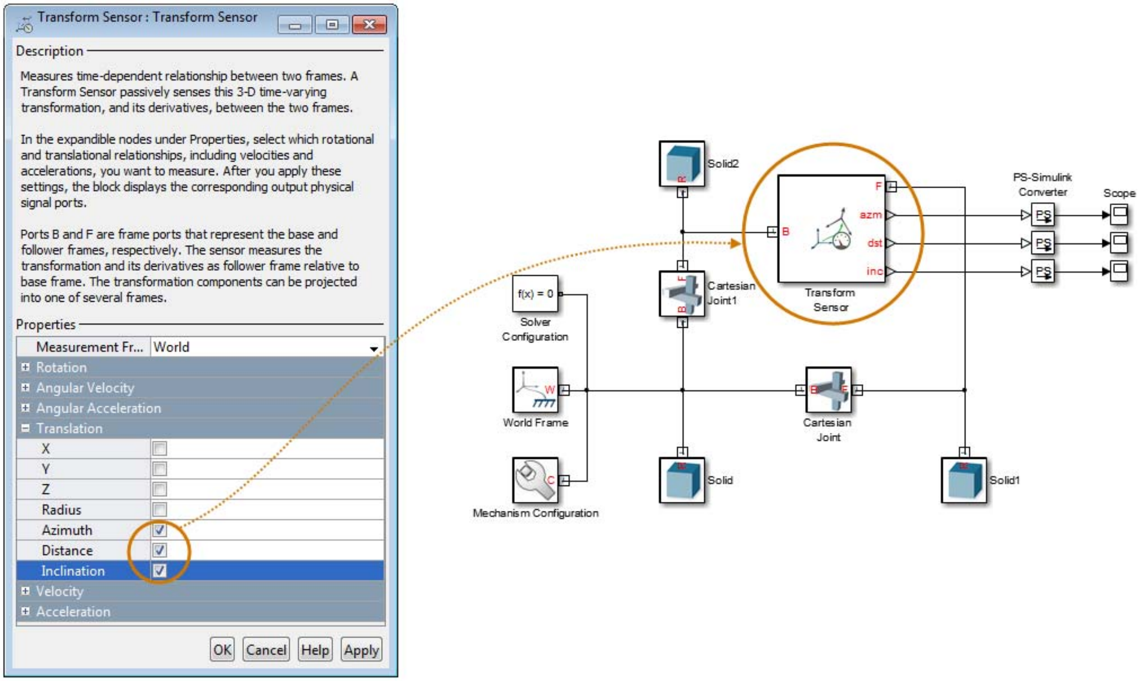


Transform Sensor

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.



Transform Sensor : Transform Sensor

Description

Measures time-dependent relationship between two frames. A Transform Sensor passively senses this 3-D time-varying transformation, and its derivatives, between the two frames.

In the expandible nodes under Properties, select which rotational and translational relationships, including velocities and accelerations, you want to measure. After you apply these settings, the block displays the corresponding output physical signal ports.

Ports B and F are frame ports that represent the base and follower frames, respectively. The sensor measures the transformation and its derivatives as follower frame relative to base frame. The transformation components can be projected into one of several frames.

Properties

Measurement Fr...	World
⊞ Rotation	
⊞ Angular Velocity	
⊞ Angular Acceleration	
⊟ Translation	
X	□
Y	□
Z	□
Radius	□
Azimuth	☑
Distance	☑
Inclination	☑
⊞ Velocity	
⊞ Acceleration	

Related Examples

- “Sense Double-Pendulum Motion” on page 4-43
- “Prescribe Four-Bar Actuation Torque” on page 4-50

Concepts

- “Sensing Spatial Relationships” on page 4-21
- “Rotation Measurements” on page 4-26
- “Measurement Frames” on page 4-40

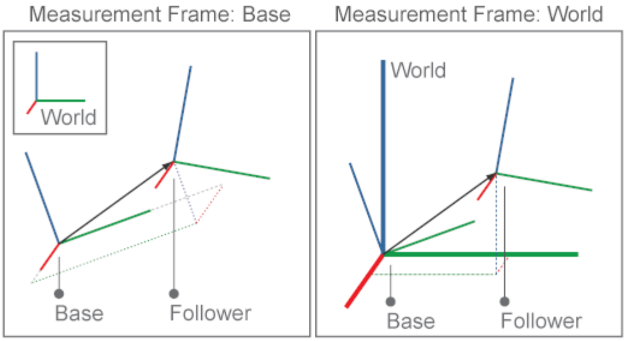
Measurement Frames

In this section...

You can sense the spatial relationship between two frames. When you do so, SimMechanics 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 includes `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 SimMechanics 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. SimMechanics 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, SimMechanics resolves that translation vector along the axes of the base frame. If you select the World frame as the measurement frame, SimMechanics instead resolves the translation vector along the axes of the World frame. The translation vector remains the same, but the frame SimMechanics 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

SimMechanics 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 World Frame 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.

Measurement Frame	Description
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	Non-rotating versions of the Base and follower frames. A non-rotating frame is a virtual frame which, at every point in time, SimMechanics holds coincident with the rotating frame, but which has zero angular velocity with respect to the World frame. Measurements that can differ between rotating and non-rotating frames are the linear velocity and linear acceleration.

Related Examples

- "Sense Double-Pendulum Motion" on page 4-43
- "Prescribe Four-Bar Actuation Torque" on page 4-50

Concepts

- "Sensing Spatial Relationships" on page 4-21
- "Rotation Measurements" on page 4-26
- "Translation Measurements" on page 4-31

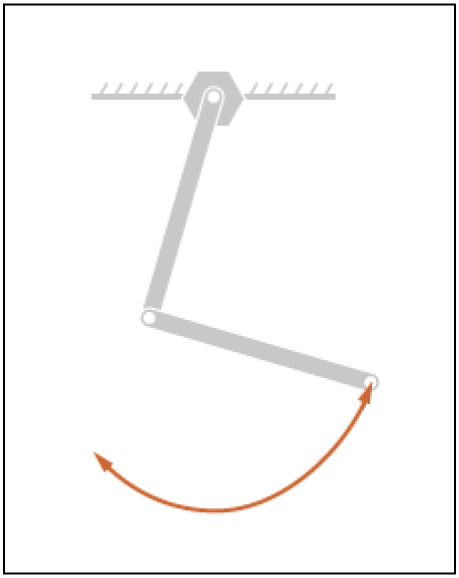
Sense Double-Pendulum Motion

In this section...

Model Overview

The Transform Sensor block provides the broadest motion-sensing capability in SimMechanics 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.



Before continuing, you must have completed example "Model Double Pendulum" on page 3-29.

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

To sense motion in the double-pendulum model:

1 Open the `double_pendulum` that you created in example “Model Double Pendulum” on page 3-29.

2 Drag these blocks to the model.

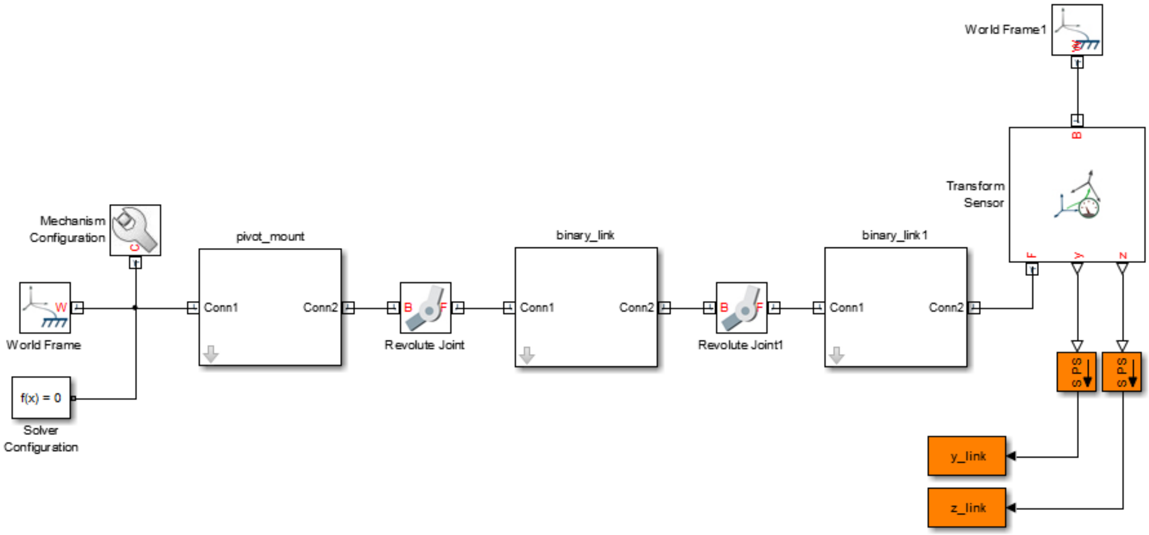
Block	Library	Quantity
Transform Sensor	**SimMechanics Second Generation (SM 2G) > Frames and Transforms**	1
World Frame	**SimMechanics Second Generation (SM 2G) > Frames and Transforms**	1
PS-Simulink Converter	**Simscape > Utilities**	2
To Workspace	**Simulink > Sinks**	2

3 In the Transform Sensor block dialog box, select **Translation > Y** and **Translation > Z**.

4 In the PS-Simulink Converter blocks, specify `cm` physical units.

5 In the two To Workspace blocks, enter the variable names `y_link` and `z_link`.

6 Connect the blocks to the model as shown in the figure.



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. Double-click block Revolute Joint.

   This is the block between Pivot Mount and Binary Link subsystem blocks.

2. In the **State Targets** section of the block dialog box, select **Specify Position Target**.

3. In **Value**, enter `10`.

   Check that the physical unit is `deg` (degrees).

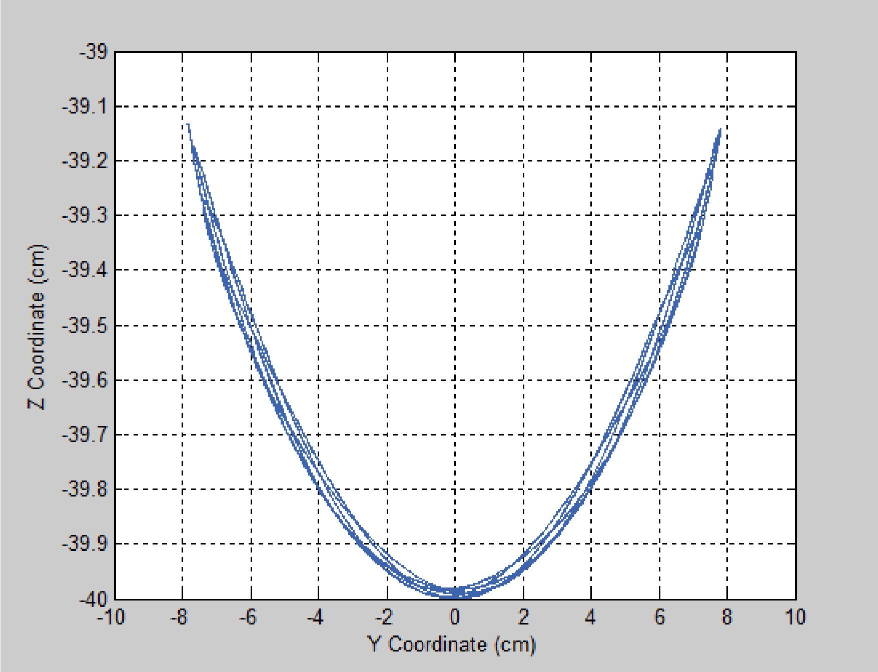
Simulate Model

To simulate the model, in the Simulink tool bar click the **Run** button. Alternatively, with the model window active, press **Ctrl+T**. Mechanics Explorer displays the model simulation in the visualization pane.

You can now plot the position coordinates of the lower link. At the MATLAB command line, enter:

```
figure(1);
hold;
plot(y_link.data, z_link.data, 'color', [60 100 175]/255);
axis([-10 10 -40 -39])
xlabel('Y Coordinate (cm)');
ylabel('Z Coordinate (cm)');
grid on;
```

The figure shows the plot that opens. This plots shows that lower link path is nearly, but not quite, the same with each oscillation. This behavior is characteristic of quasi-periodic systems.



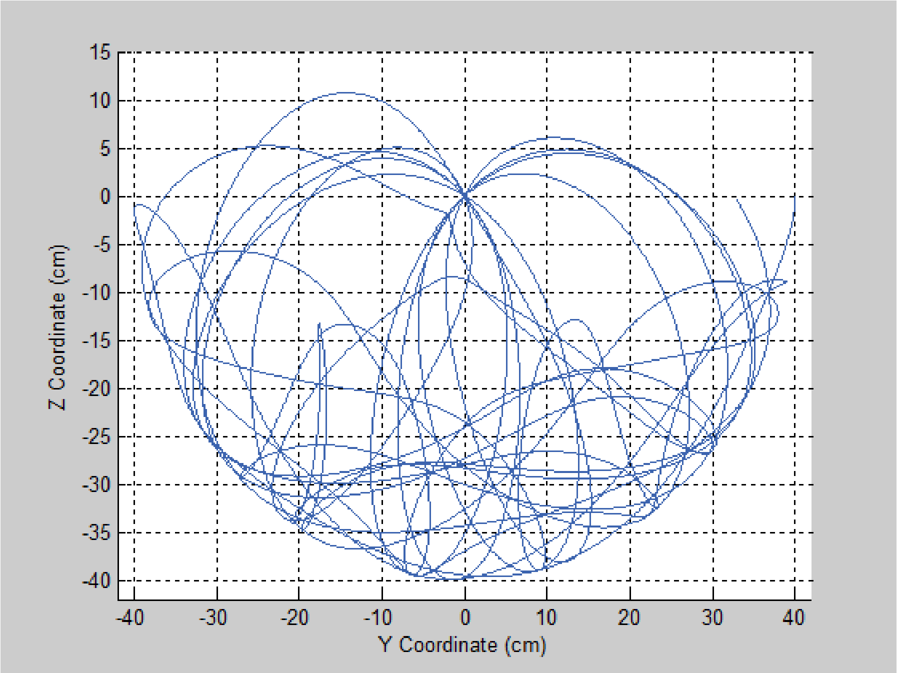
Simulate Chaotic Motion

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 line enter this code:

```
figure(2);
hold;
plot(y_link.data, z_link.data, 'color', [60 100 175]/255);
axis([-42 42 -42 15])
xlabel('Y Coordinate (cm)');
ylabel('Z Coordinate (cm)');
grid on;
```

The figure shows the plot that opens. This plots shows that lower link path is very different with each oscillation. This behavior is characteristic of chaotic systems.



Save Model

So that you can reuse this model in subsequent examples, save it in a convenient folder as double_pendulum.

Related Examples

- “Prescribe Four-Bar Actuation Torque” on page 4-50

Concepts

- “Forces and Torques” on page 4-2
- “Actuating and Sensing with Physical Signals” on page 4-18
- “Sensing Spatial Relationships” on page 4-21

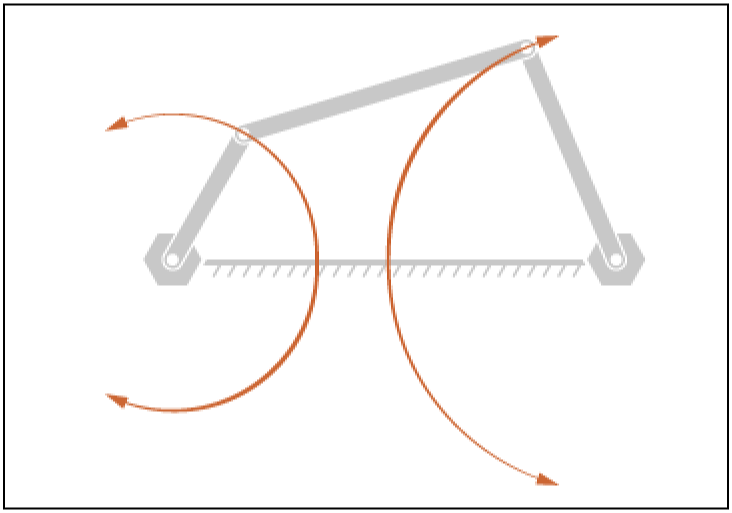
Prescribe Four-Bar Actuation Torque

In this section...

Model Overview

In SimMechanics, 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.



Before continuing, you must have completed example “Model Four-Bar Linkage” on page 3-37.

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 Four-Bar Linkage" on page 3-37, 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)
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:

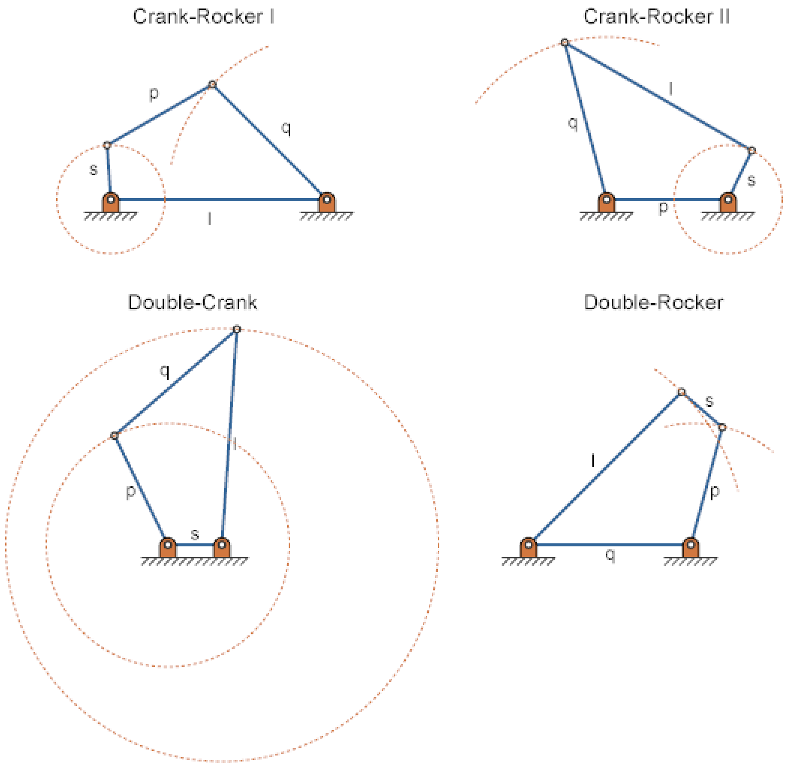
- s is the shortest link
- l is the longest link
- p and q are the two remaining links

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 SimMechanics can use. For more information, see "Actuating and Sensing with Physical Signals" on page 4-18.

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 binary_link and binary_link1 end frames.

Provide Joint Actuation Input

To add an actuation input to the four-bar linkage:

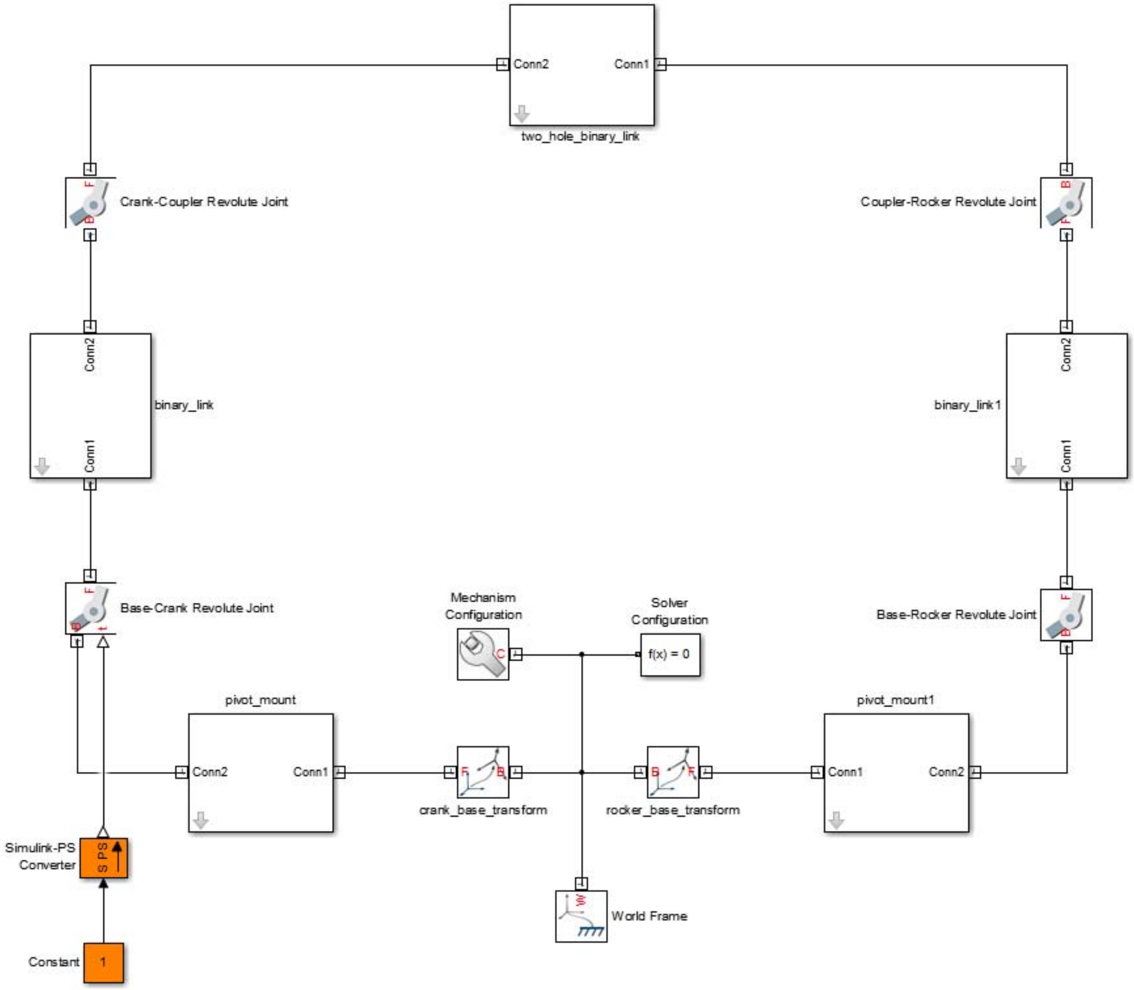
1. Open model the `four_bar_linkage` model that you created in example "Model Four-Bar Linkage" on page 3-37.

2. Drag these blocks to the model.

Block	Library	Quantity
Simulink-PS Converter	**Simscape > Utilities**	1
Constant	**Simulink > Sources**	1

3 In the **Actuation > Torque** drop-down list, select `Provided by Input`. The block displays the physical signal input port t.

4 Connect the blocks as shown in the figure.



5. In the **Input Signal Unit** parameter of the Simulink-PS Converter block dialog box, enter `N*m` and click **OK**.

Specify Joint Internal Mechanics

Real joints dissipate kinetic energy as heat due to damping. Increase the fidelity of the four-bar model by specifying the joint damping coefficient. You can specify this coefficient directly from the joint block dialog boxes.

1. Open the four Revolute Joint block dialog boxes.
2. In **Internal Mechanics > Damping** enter `5e-4` and press **OK**.

   Keep the `N*m/(deg/s)` default units.

Sense Link Position Coordinates

Complete the model with additional blocks to sense the position coordinates of binary_link and binary_link1 end frames.

1. Add these blocks to the model.

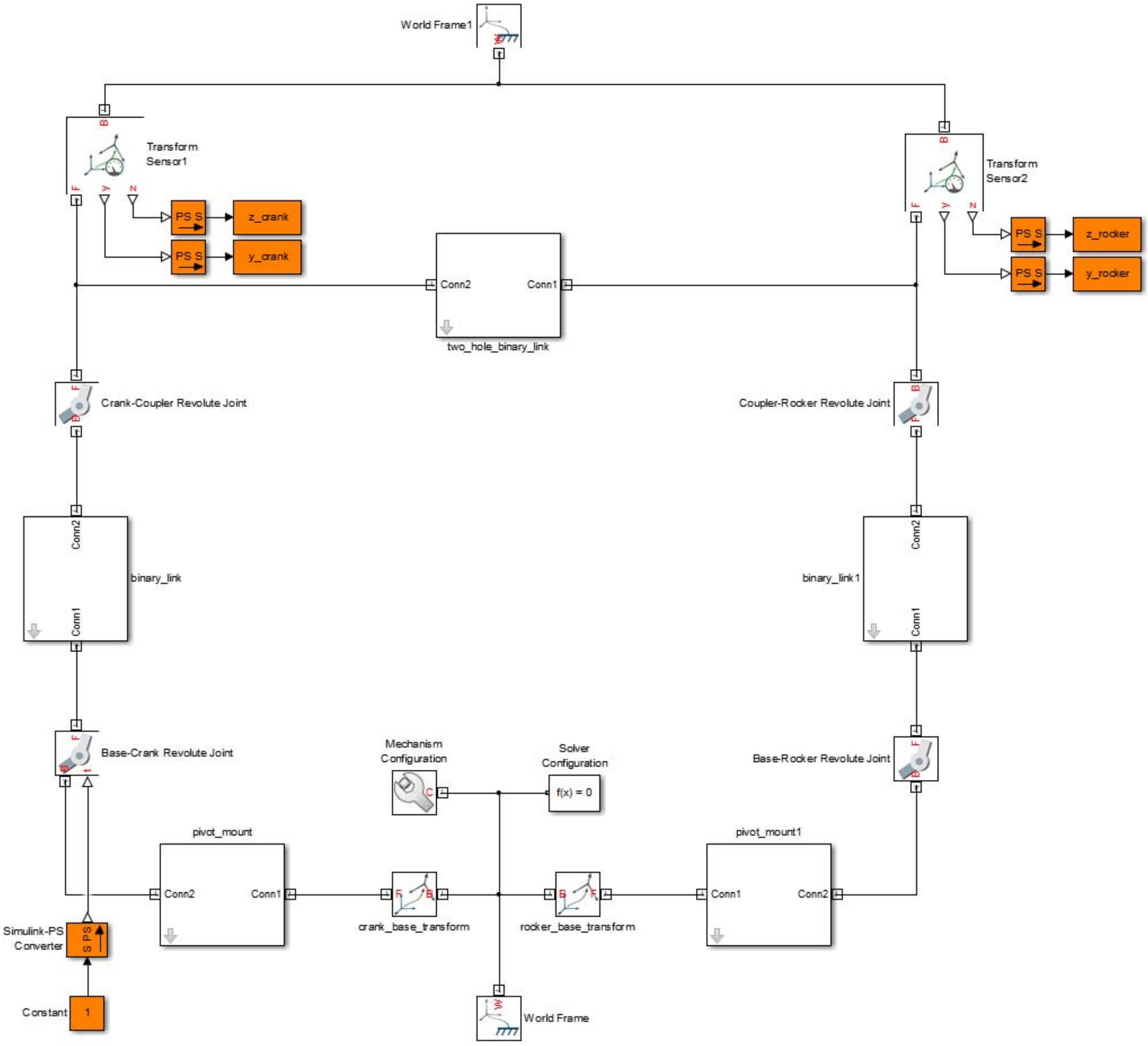
Block	Library	Quantity
Transform Sensor	**SimMechanics > Frames and Transforms**	2
World Frame	**SimMechanics > Frames and Transforms**	1
PS-Simulink Converter	**Simscape > Utilities**	4
To Workspace	**Simulink > Sinks**	4

2. In the Transform Sensor block dialog boxes, select **Translation > Y** and **Translation > Z**.
3. In the **Input 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 these values:

- `y_crank`
- `z_crank`
- `y_rocker`
- `z_rocker`

5. Connect and name the blocks as they appear in the figure.

   The new blocks appear in the top portion of the figure.



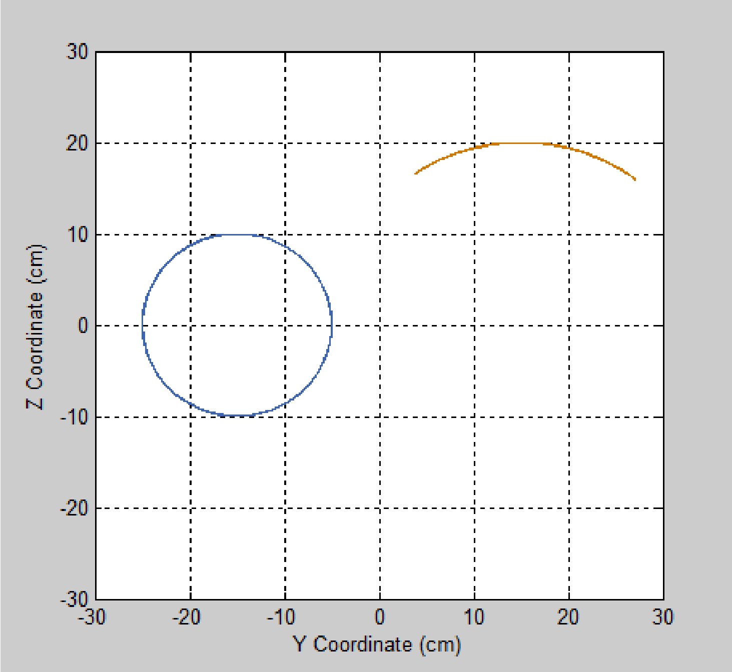
Simulate Model

To simulate the model, in the Simulink tool bar click the **Run** button. Alternatively, with the model window active, press **Ctrl+T**. Mechanics Explorer displays the model simulation in the visualization pane.

Plot the position coordinates of the binary_link and binary_link1 end frames. At the MATLAB command line, enter this code:

```
figure(1);
hold;
plot(y_crank.data, z_crank.data, 'color', [60 100 175]/255);
plot(y_rocker.data, z_rocker.data, 'color', [210 120 0]/255);
axis([-30 30 -30 30]);
xlabel('Y Coordinate (cm)');
ylabel('Z Coordinate (cm)');
grid on;
```

The figure shows the plot that opens. This plots shows that binary_link completes a full revolution, while binary_link1 completes only a partial revolution, e.g. it oscillates. This behavior is characteristic of crank-rocker systems.

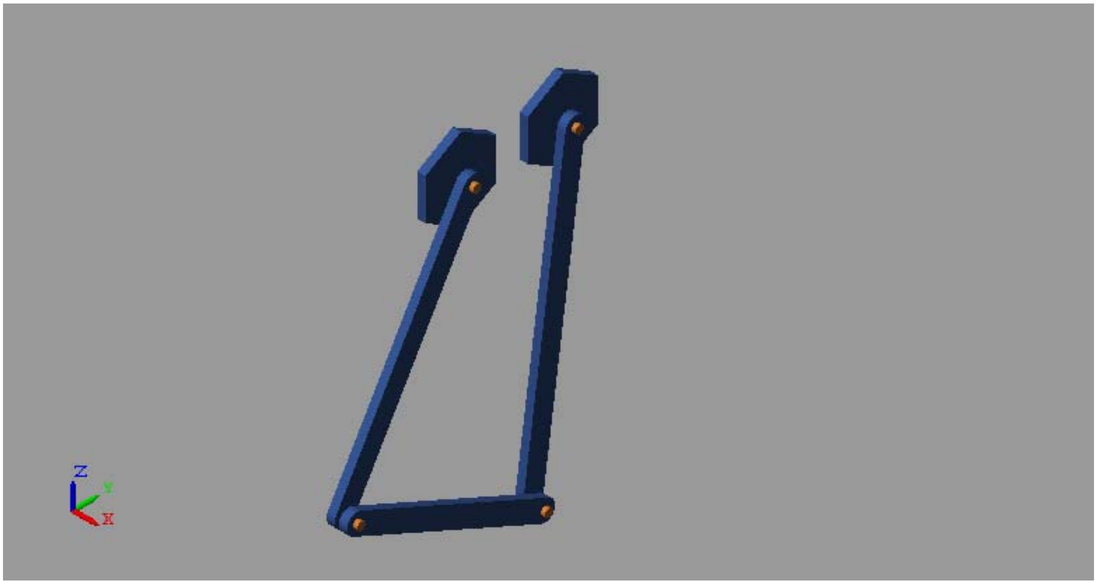


Simulate Model in Double-Crank Mode

Try simulating the model in double-crank mode. You can change the four-bar linkage into a double-crank linkage by adjusting the relative link lengths. The following lengths ensure the model represents a double-crank linkage.

Block	Parameter	Value
binary_link	**Length**	`0.25`
two_hole_binary_link	**Length**	`0.20`
binary_link1	**Length**	`0.30`
crank_base_transform	**Translation > Offset**	`0.05`
rocker_base_transform	**Translation > Offset**	`0.05`

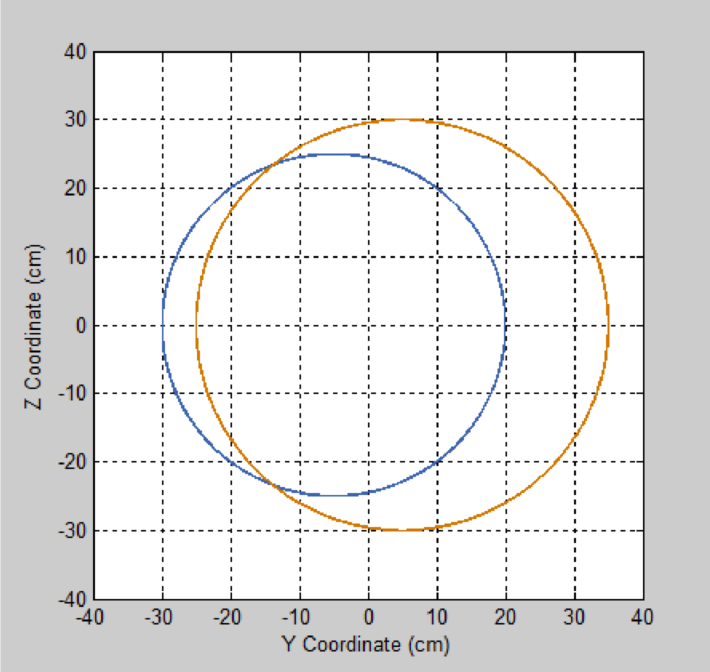
Update and simulate the model. The figure shows the updated visualization display in Mechanics Explorer.



Plot the position coordinates of the binary_link and binary_link1 end frames. At the MATLAB command line, enter:

```
figure(2);
hold;
plot(y_crank.data, z_crank.data, 'color', [60 100 175]/255);
plot(y_rocker.data, z_rocker.data, 'color', [210 120 0]/255);
axis([-40 40 -40 40]);
xlabel('Y Coordinate (cm)');
ylabel('Z Coordinate (cm)');
grid on;
```

The figure shows the plot that opens. This plots shows that both links complete a full revolution. This behavior is characteristic of double-crank linkages.



Simulate Model in Double-Rocker Mode

Try also simulating the model in double-rocker mode. As before, you can change the four-bar linkage into a double-rocker linkage by adjusting the relative link lengths. The following lengths ensure the model represents a double-crank linkage.

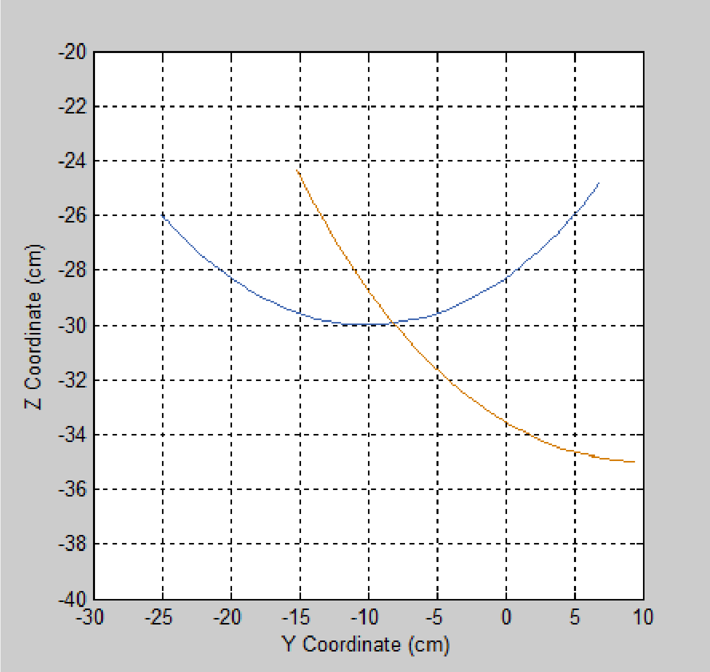
Block	Parameter	Value
binary_link	**Length**	`0.30`
two_hole_binary_link	**Length**	`0.10`
binary_link1	**Length**	`0.35`
crank_base_transform	**Translation > Offset**	`0.10`
rocker_base_transform	**Translation > Offset**	`0.10`

Update and simulate the model. The figure shows the updated visualization display in Mechanics Explorer.

Plot the position coordinates of the binary_link and binary_link1 end frames. At the MATLAB command line, enter:

```
figure(3);
hold;
plot(y_crank.data, z_crank.data, 'color', [60 100 175]/255);
plot(y_rocker.data, z_rocker.data, 'color', [210 120 0]/255);
axis([-30 10 -40 -20]);
xlabel('Y Coordinate (cm)');
ylabel('Z Coordinate (cm)');
grid on;
```

The figure shows the plot that opens. This plots shows that neither link completes a full revolution. This behavior is characteristic of double-rocker linkages.



Related Examples

- “Sense Double-Pendulum Motion” on page 4-43
- “Correct Aiming Mechanism Assembly Error” on page 3-48

Concepts

- “Forces and Torques” on page 4-2
- “Actuating and Sensing with Physical Signals” on page 4-18
- “Sensing Spatial Relationships” on page 4-21

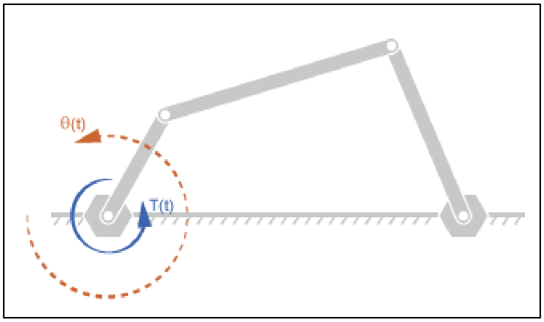
Prescribe Four-Bar Motion

In this section...

Model Overview

For certain applications, you must specify joint motion directly. Inverse dynamic analysis is one example. In this analysis mode, you prescribe joint motion and determine the actuation forces and torques required to achieve that motion.

In this example, you prescribe the angular trajectory of a four-bar revolute joint. You prescribe that trajectory directly using the Revolute Joint block. You then sense and plot the actuation torque required to achieve the prescribed trajectory.



Before continuing, you must have completed example "Model Four-Bar Linkage" on page 3-37.

Build Model

This example is based on the four_bar_linkage model that you created in example "Model Four-Bar Linkage" on page 3-37. To add joint motion input and actuation torque output to that model:

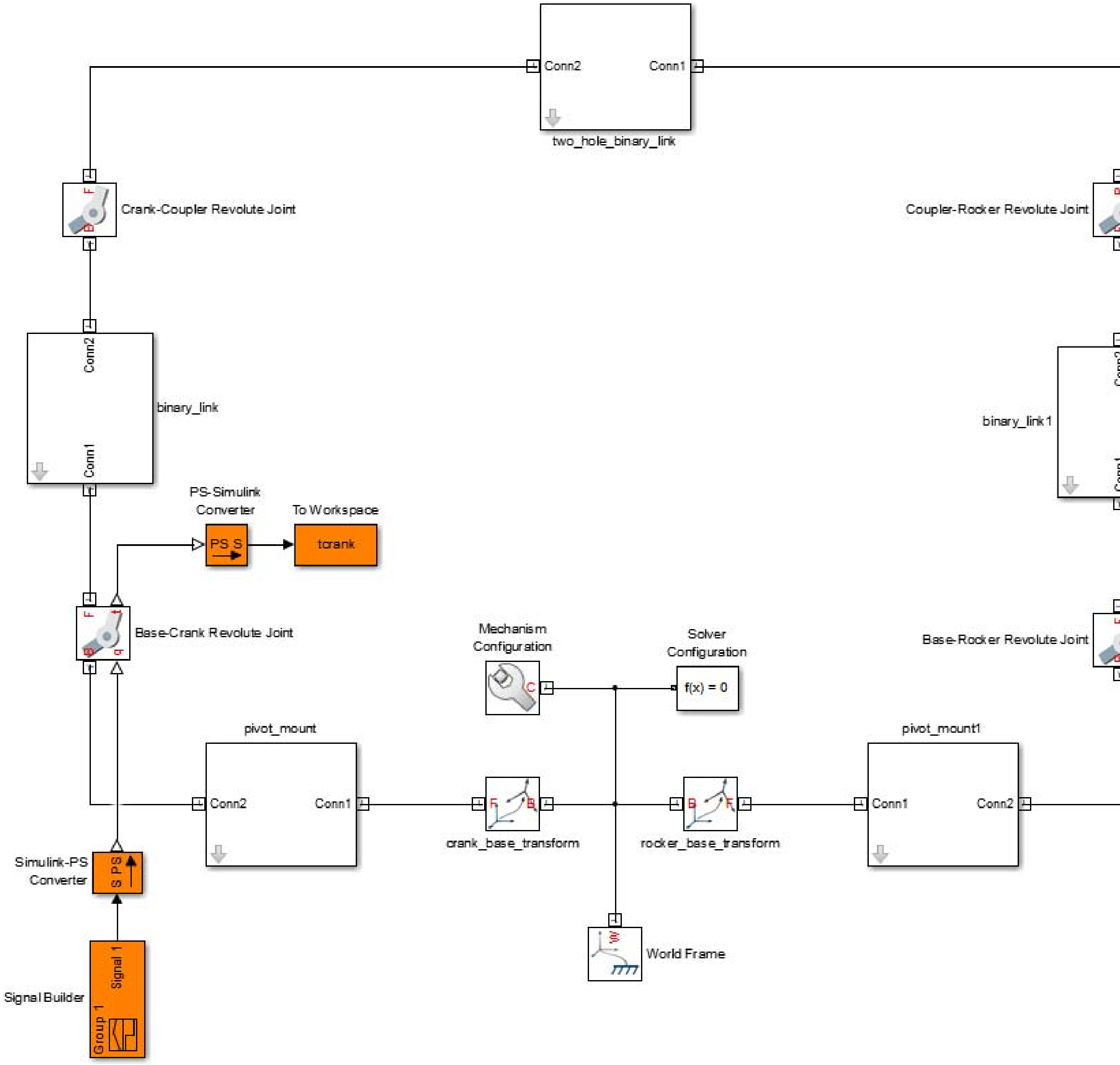
1. Open the four_bar_linkage model.
2. In the dialog box of the Base-Crank Revolute Joint block, specify these parameter settings.

Parameter	Setting
Actuation > Torque	`Automatically Computed`
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 the **Internal Mechanics > Damping Coefficient** parameter of the Revolute Joint block dialog boxes, enter `5e-4`.
4. Drag these blocks into the model.

Block	Library	Quantity
Simulink-PS Converter	**Simscape > Utilities**	1
PS-Simulink Converter	**Simscape > Utilities**	1
To Workspace	**Simulink > Sinks**	1
Scope	**Simulink > Sinks**	1
Signal Builder	**Simulink > Sources**	1

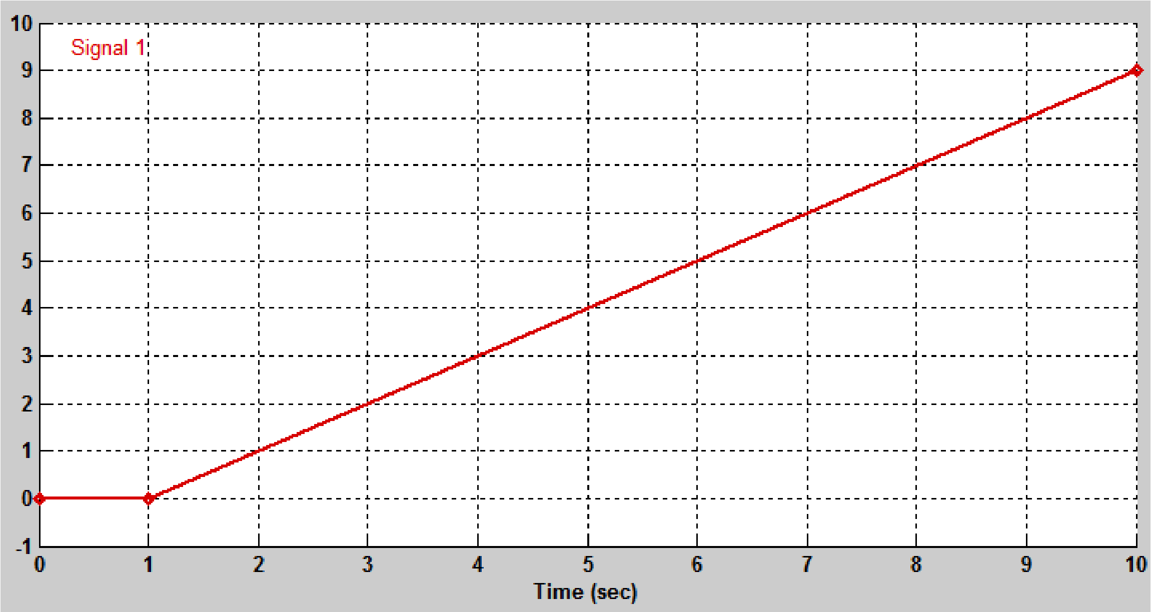
5. Connect the bocks as shown in the figure.


Conn2
Conn1
two_hole_binary_link
Crank-Coupler Revolute Joint
Coupler-Rocker Revolute Joint
Conn2
Conn1
binary_link
binary_link1
PS-Simulink Converter
PS S
To Workspace
tcrank
Base-Crank Revolute Joint
Base-Rocker Revolute Joint
Mechanism Configuration
Solver Configuration
f(x) = 0
pivot_mount
Conn2
Conn1
pivot_mount1
Conn1
Conn2
crank_base_transform
rocker_base_transform
Simulink-PS Converter
S PS
World Frame
Signal Builder
Signal 1
Group 1

6 Specify these block parameters.

Block	Parameter	Value
To Workspace	**Variable name**	`tcrank`
PS-Simulink Converter	**Output signal unit**	`N*m`
Simulink-PS Converter	**Units > Input signal unit**	`rev`
	Input Handling > Filtering and derivatives	`Filter input`
	Input Handling > Input filtering order	`Second-order filtering`

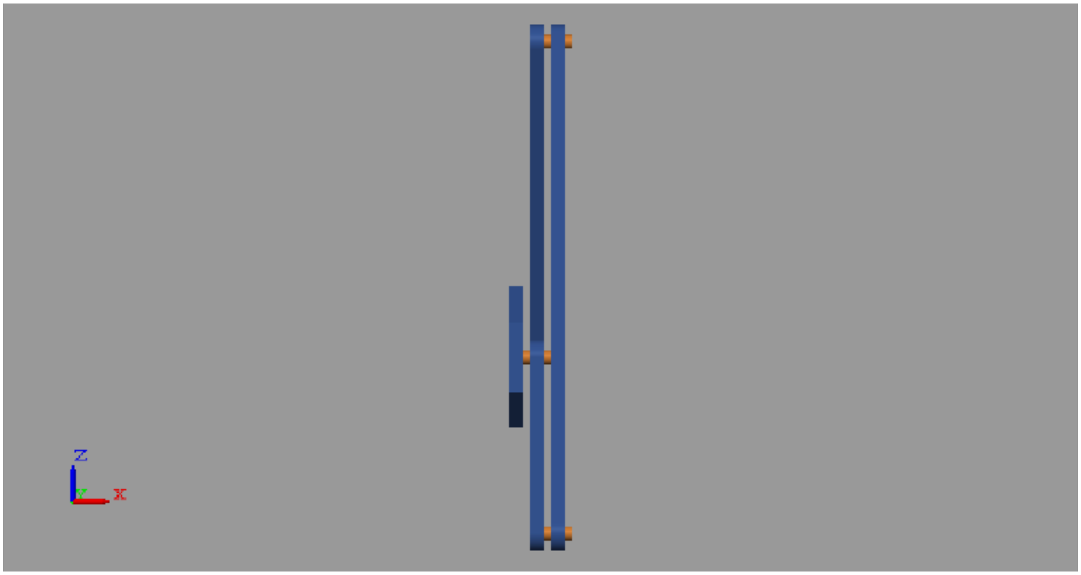
7 In the Signal Builder window, specify the joint angular trajectory as shown in the figure.



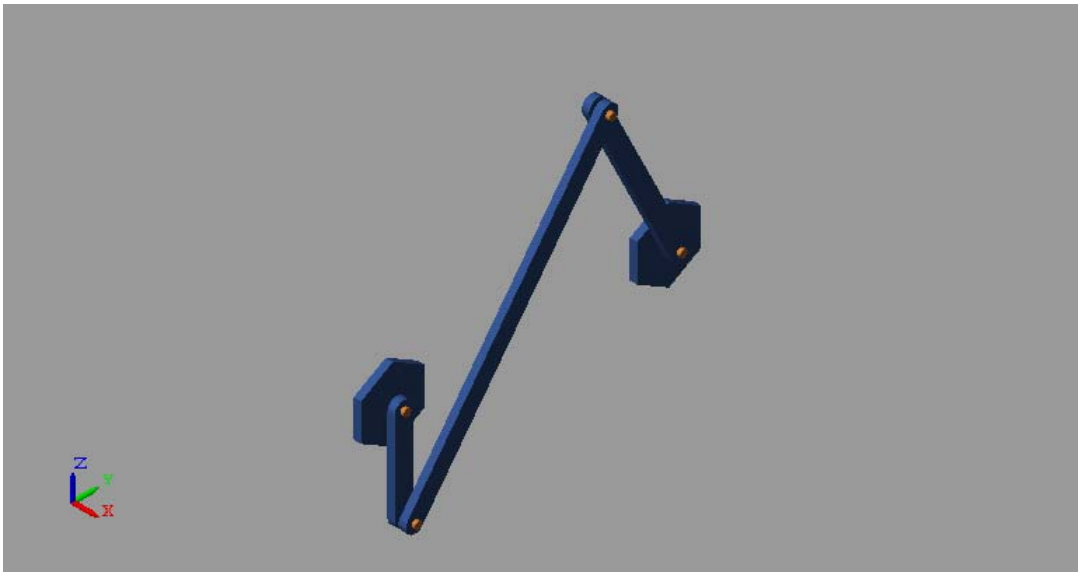
This signal corresponds to a constant angular speed of 1 rev/s from t = 1s onwards.

Simulate Model

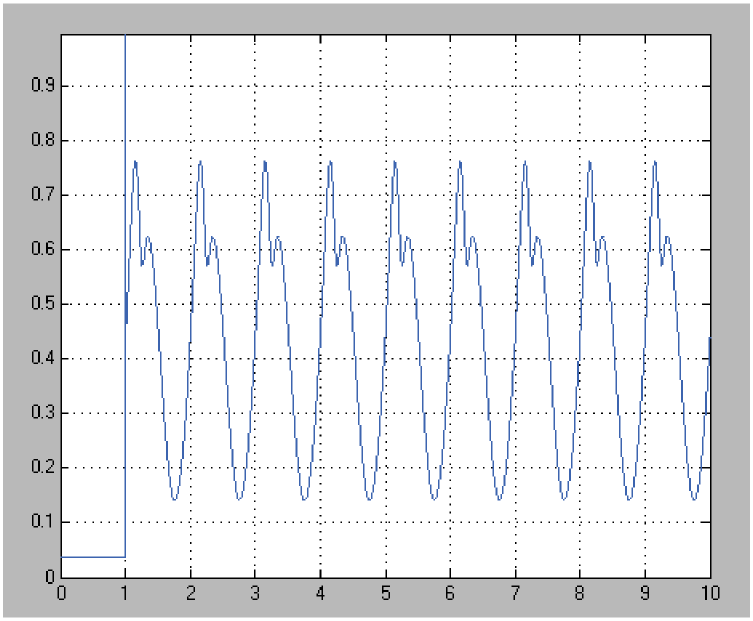
Run the simulation. Mechanics Explorer opens with a dynamic display of the four-bar model.



In the Mechanics Explorer toolstrip, click the isometric view button for a 3-D viewpoint.



Open the Scope window. It displays the joint actuation torque required to achieve the prescribed motion input.



Actuate Model Using Sensed Torque

Change actuation mode from motion input to torque input. Then, verify that the angular velocity of the Base-Crank Revolute Joint block equals 1 rev/s. This angular velocity corresponds to the original motion input you prescribed.

1 Disconnect the Signal Builder and To Workspace blocks from the model.

2 In the dialog box of the Base-Crank Revolute Joint block, change these parameter settings.

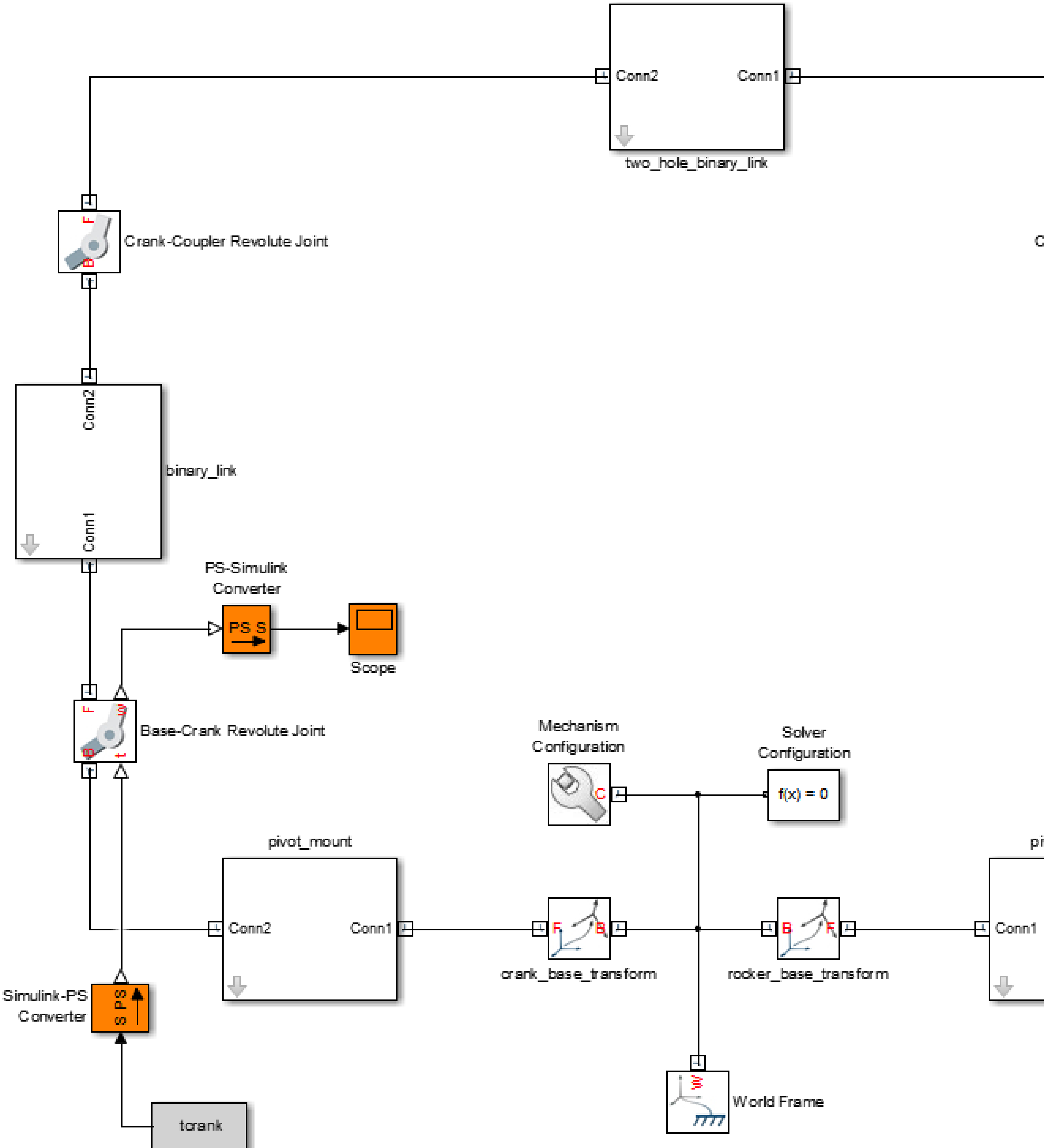
Parameter	Original Setting	New Setting
Actuation > Torque	`Automatically computed`	`Provided by Input`
Actuation > Motion	`Provided by Input`	`Automatically Computed`

Parameter	Original Setting	New Setting
Sensing > Velocity	Unselected	Selected
Sensing > Actuator Torque	Selected	Unselected

3 In the PS-Simulink Converter block dialog box, change **Output signal unit** to `rev/s`.

4 In the Simulink-PS Converter block dialog box, change **Input signal unit** to `N*m`.

5 From the Simulink Sinks library, drag a From Workspace block and connect it as shown in the figure.


Conn2
Conn1
two_hole_binary_link
F
B
Crank-Coupler Revolute Joint
C
Conn2
Conn1
binary_link
PS-Simulink Converter
PS S
Scope
F
W
B
t
Base-Crank Revolute Joint
Mechanism Configuration
C
Solver Configuration
f(x) = 0
pivot_mount
Conn2
Conn1
F
B
crank_base_transform
B
F
rocker_base_transform
pi
Conn1
Simulink-PS Converter
S PS
W
World Frame
tcrank

Guide Model Assembly

For the sensed actuation torque to yield the original prescribed motion, the initial joint states of the two model versions (one with the original prescribed motion as input, the other with the sensed actuation torque as input) must be the same.

When a model contains joints with motion inputs, the initial state is dictated by the motion inputs. If the model contains no motion inputs, the initial state is dictated solely by joint state targets, if any.

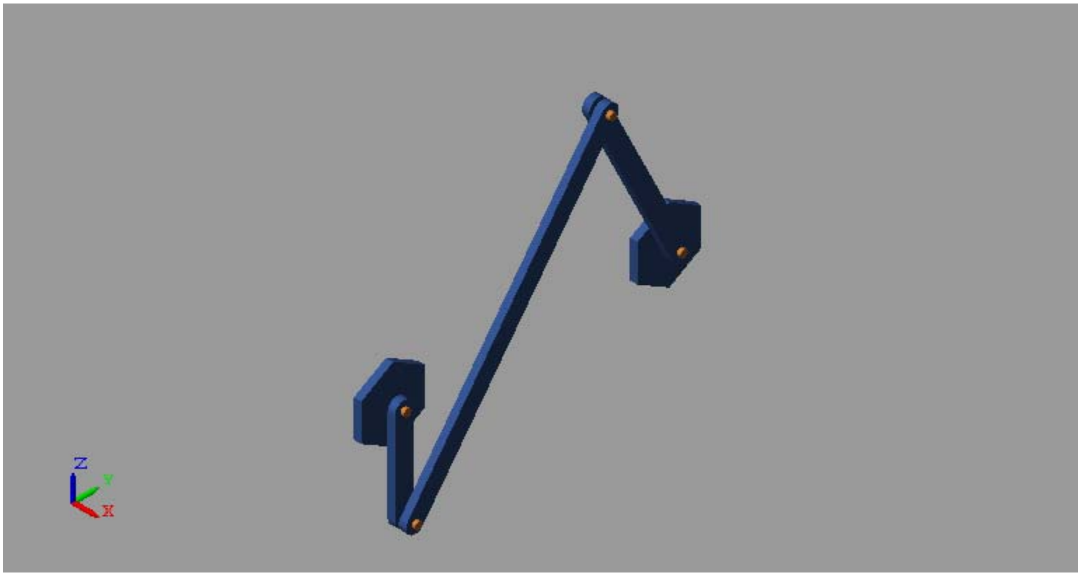
In this example, the motion input sets the initial state of the Base-Crank Revolute Joint block at zero degrees. To ensure that simulation of the torque-actuated model starts from the same initial state, you must specify a position state target of zero degrees for the Base-Crank Revolute Joint block.

1. In the dialog box of the Base-Crank Revolute Joint block, select **State Targets > Specify Position Target**.

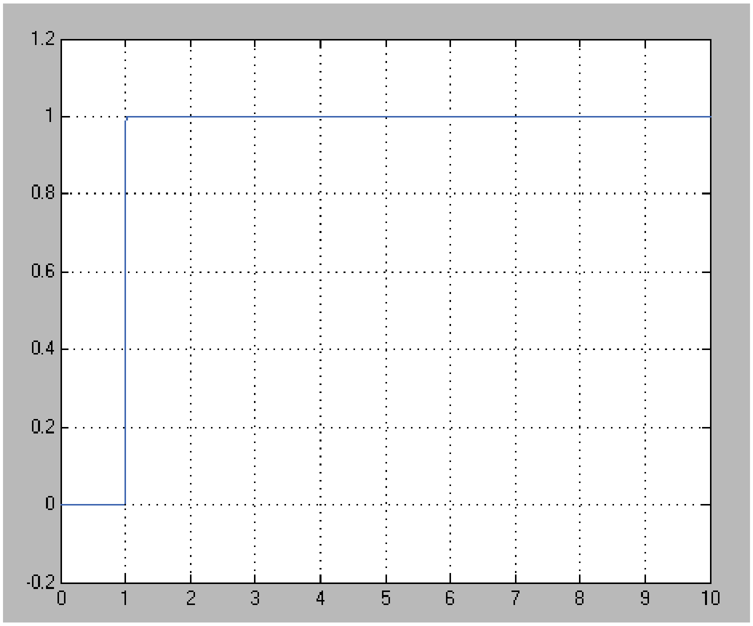
2. In **Value**, enter `0` and click **OK**.

Simulate Updated Model

Run the simulation. Mechanics Explorer displays the updated model. Click the isometric view button for a 3-D viewpoint of the model.



Open the Scope window. It displays the joint angular velocity, in rev/s units, due to the prescribed torque input.



The angular velocity remains constant at 1 rev/s from t = 1s onwards—precisely as prescribed in the original motion input.

Related Examples

- “Sense Double-Pendulum Motion” on page 4-43
- “Prescribe Four-Bar Motion” on page 4-66
- “Specify Motion Input Derivatives” on page 4-15

Concepts

- “Forces and Torques” on page 4-2
- “Joint Actuation” on page 4-7
- “Actuating and Sensing with Physical Signals” on page 4-18

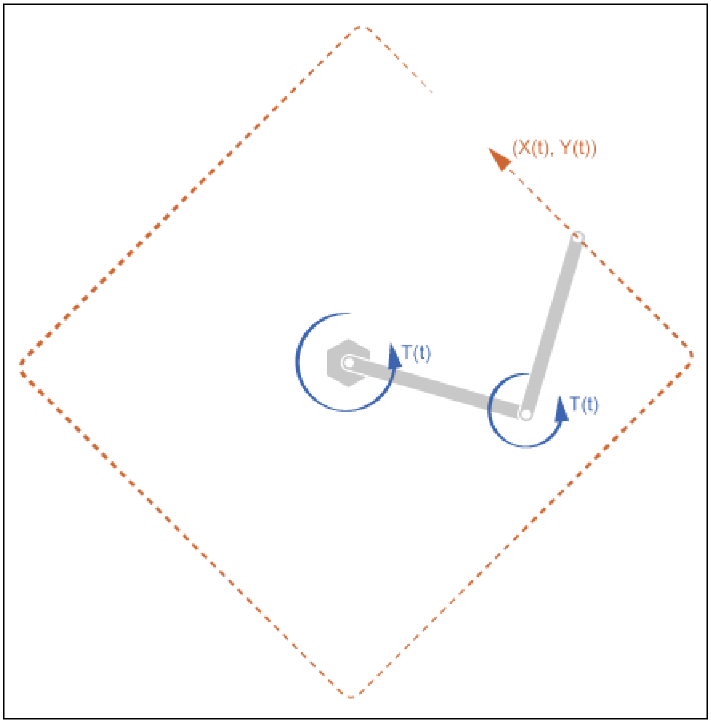
Prescribe Two-Bar Motion

In this section...

Model Overview

You can prescribe an end effector trajectory relative to the world frame. To perform this task, you must connect the end effector and world frames using a joint block. This block provides the two frames with the required degrees of freedom, but it does not represent a physical joint. The joint is said to be virtual.

In this example, you prescribe a square trajectory for the end effector of a two-arm linkage relative to the world frame. A 6-DOF Joint block represents the virtual joint between the end effector and world frames. Using this block, you sense and plot the actuation torques acting at the two revolute joints present in the linkage.



Before continuing, you must have completed example "Sense Double-Pendulum Motion" on page 4-43.

Add Virtual Joint

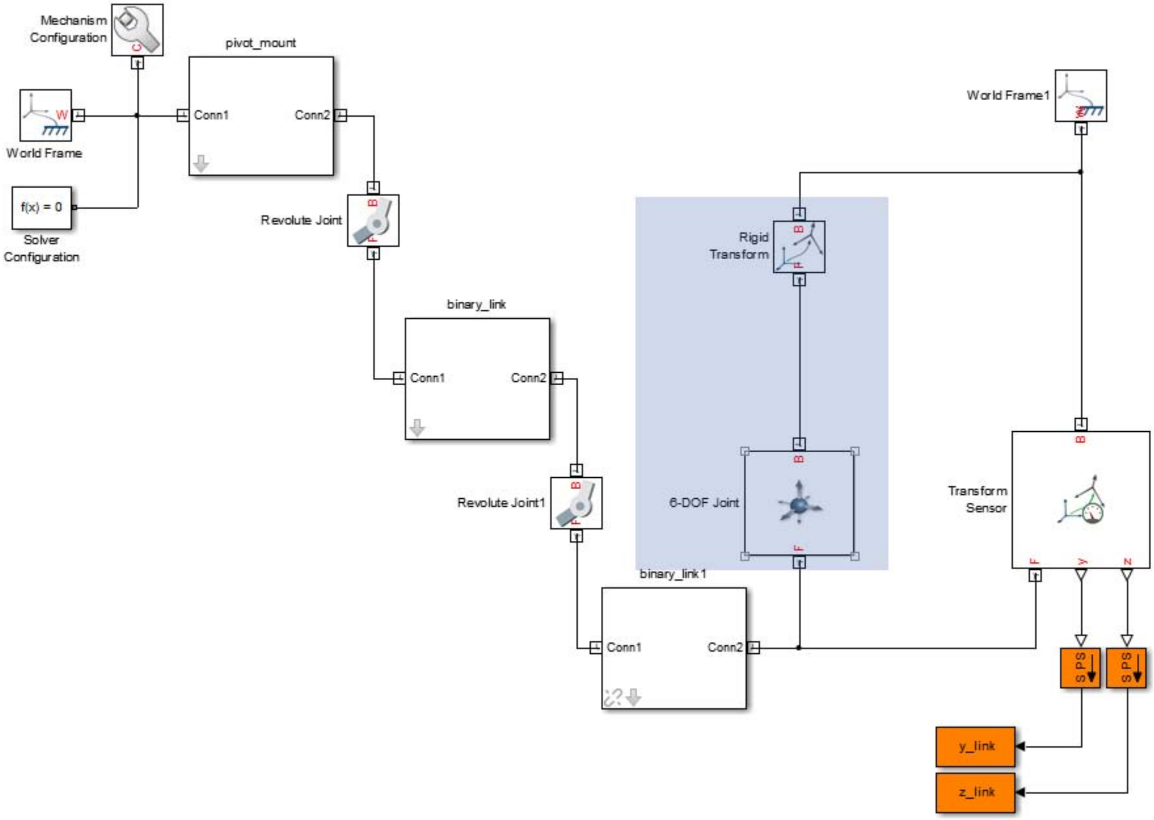
So that you can specify the end effector trajectory with respect to the world frame, connect the two using a virtual 6-DOF joint.

1 Open the double_pendulum model.

2 Drag these blocks into the model.

Block	Library	Quantity
6-DOF Joint	**SimMechanics Second Generation (SM2G) > Joints**	1
Rigid Transform	**SimMechanics Second Generation (SM2G) > Frames and Transforms**	1

3 Connect the blocks as shown in the figure.



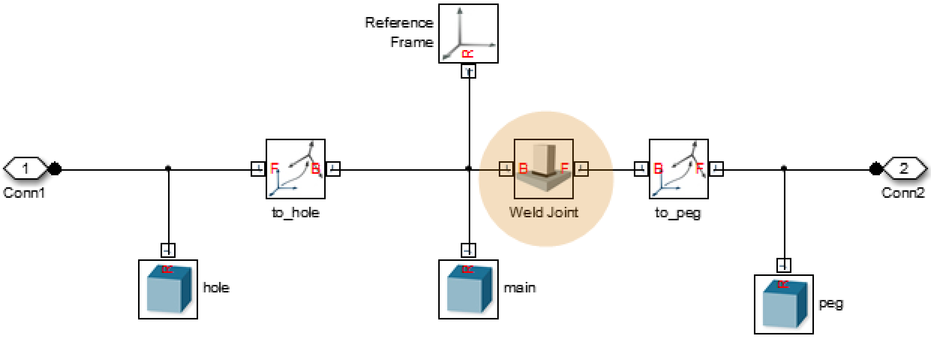
Note The blocks in the figure have been rearranged to conserve space.

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

Parameter	Value
Rotation > Method	`Standard Axis`
Rotation > Axis	`+Y`
Rotation > Angle	`90`

These parameters specify the rotation transform required to make the Z axes of the world and binary_link1 Conn2 frames parallel to each other. This rotation transform ensures that the model assembles properly.

5 Inside the binary_link1 subsystem, replace a rigid connection line with a Weld Joint block.



Adding the Weld Joint block ensures that the now-closed-loop system contains at least one joint with neither motion inputs nor computed actuation torque, a SimMechanics simulation requirement.

Add Motion Inputs

Drag and connect the required blocks to specify a square trajectory.

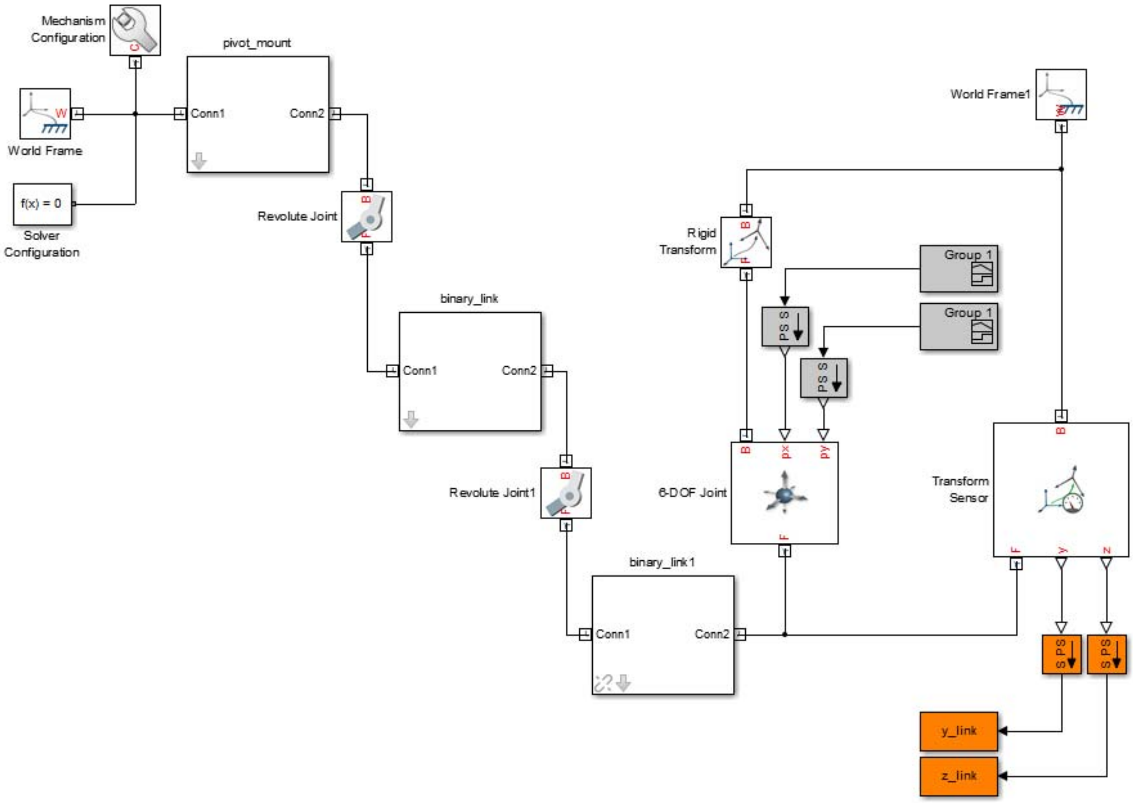
1 In the 6-DOF Joint block dialog box, specify these parameters.

Parameter	Value
X Prismatic Primitive (Px) > Actuation > Motion	`Provided by Input`
Y Prismatic Primitive (Py) > Actuation > Motion	`Provided by Input`

2 Drag these blocks into the model.

Block	Library	Quantity
Simulink-PS Converter	**Simscape > Utilities**	2
Signal Builder	**Simulink > Sources**	2

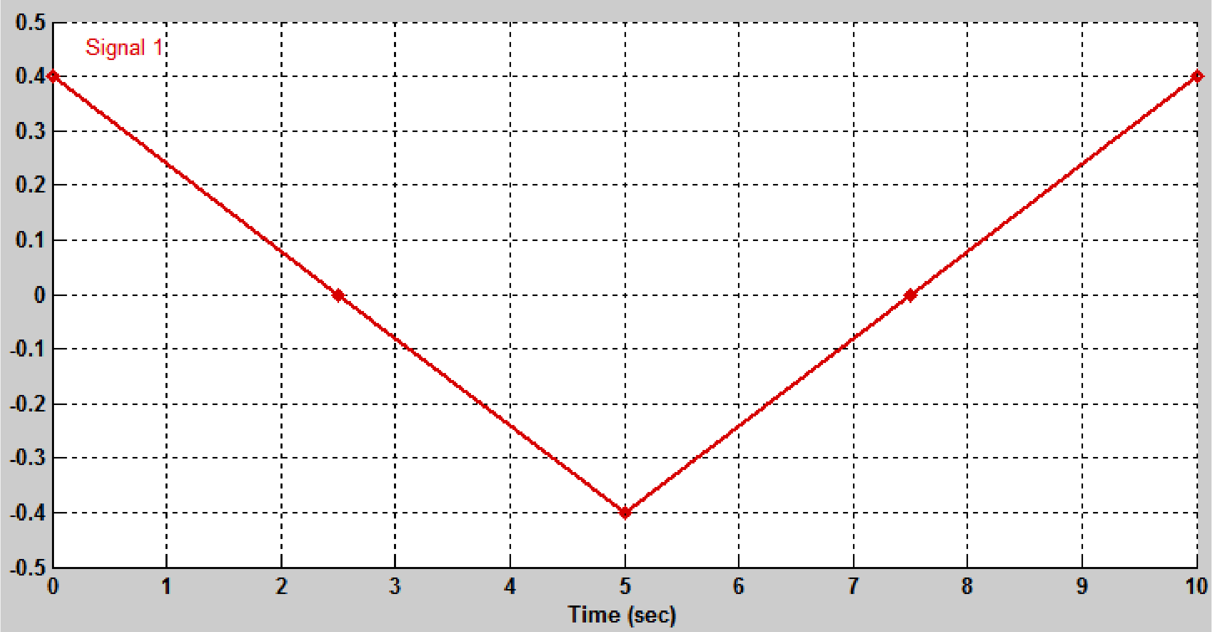
3 Connect the blocks as shown in the figure.



4 In the dialog boxes of the Simulink-PS Converter blocks, specify these parameters.

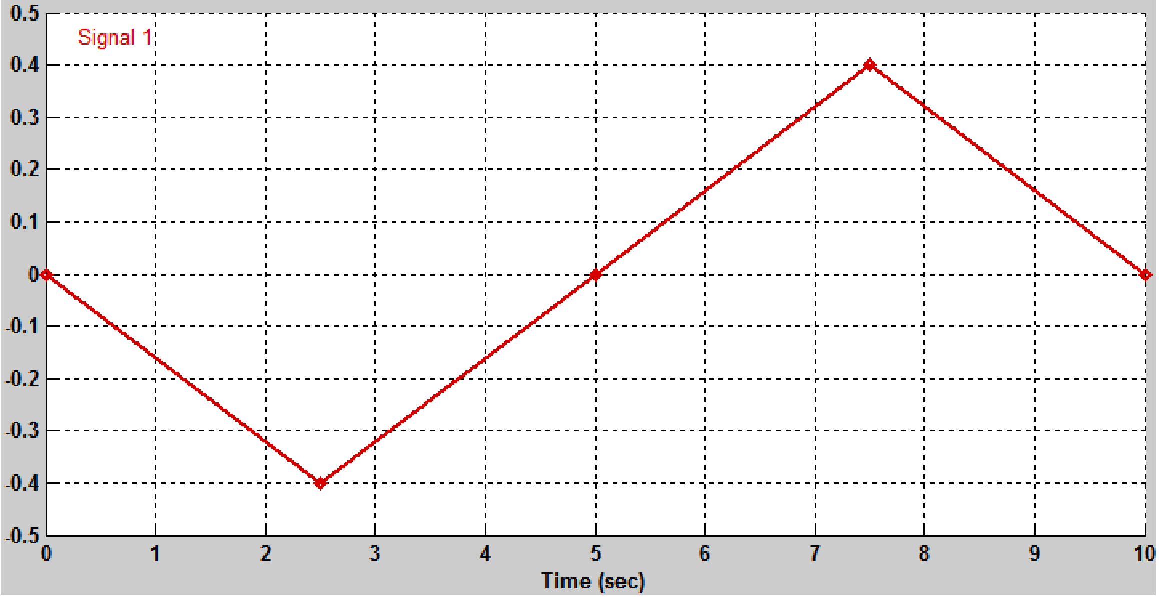
Parameter	Value
Units > Input signal unit	`m`
Input Handling > Filtering and derivatives	`Filter input`
Input Handling > Input filtering order	`Second-order filtering`

5 For the Signal Builder block that connects to the px port of the 6-DOF Joint block, specify this signal.



This signal provides the -Z coordinates, as seen in the world frame, of the square trajectory that the end effector is to follow.

6 For the Signal Builder block that connects to the py port of the 6-DOF Joint block, specify this signal.



This signal provides the Y coordinates, as seen in the world frame, of the square trajectory that the end effector is to follow.

7 In the dialog boxes of the Revolute Joint and Revolute Joint1 blocks, set **Actuation > Torque** to `Automatically Computed`.

Add Actuation Torque Outputs

Drag and connect the required blocks to sense the actuation torque outputs.

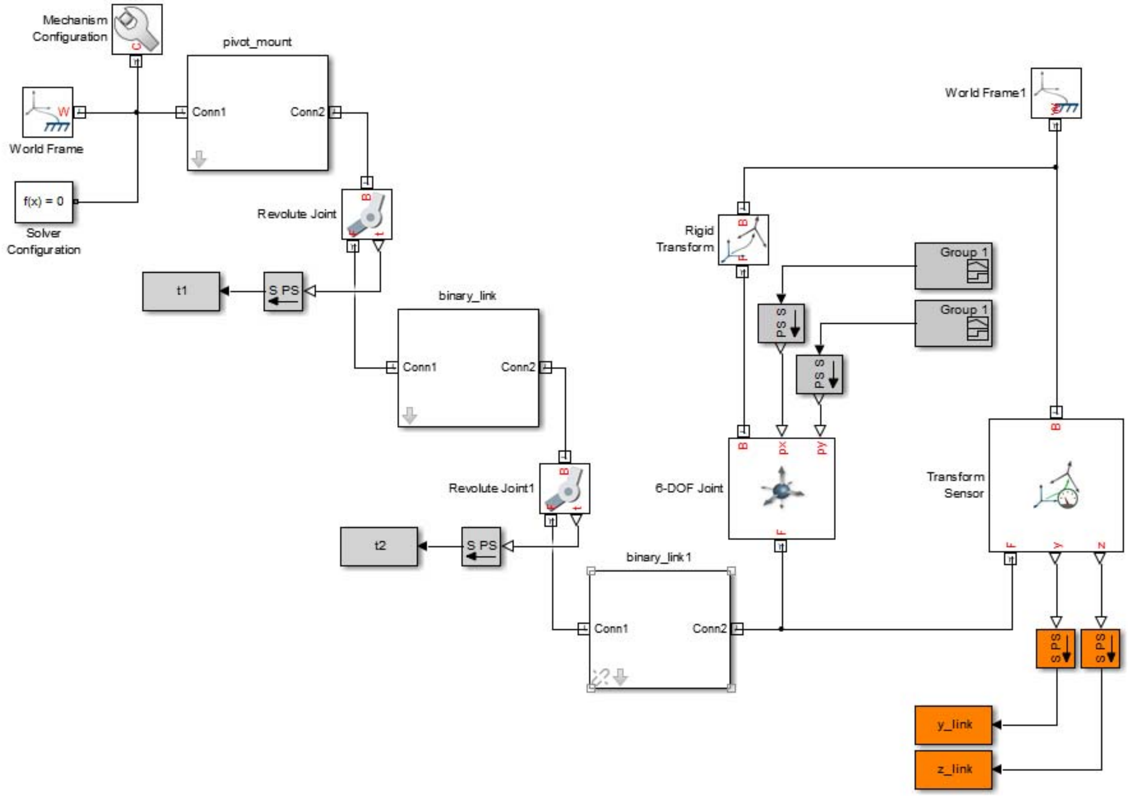
1 In the dialog boxes of the Revolute Joint and Revolute Joint1 blocks, select **Sensing > Actuator Torque**.

2 Drag these blocks into the model.

Block	Library	Quantity
PS-Simulink Converter	**Simscape > Utilities**	2
To Workspace	**Simulink > Sinks**	2

3 In the dialog boxes of the two To Workspace blocks, enter the variable names `t1` and `t2`.

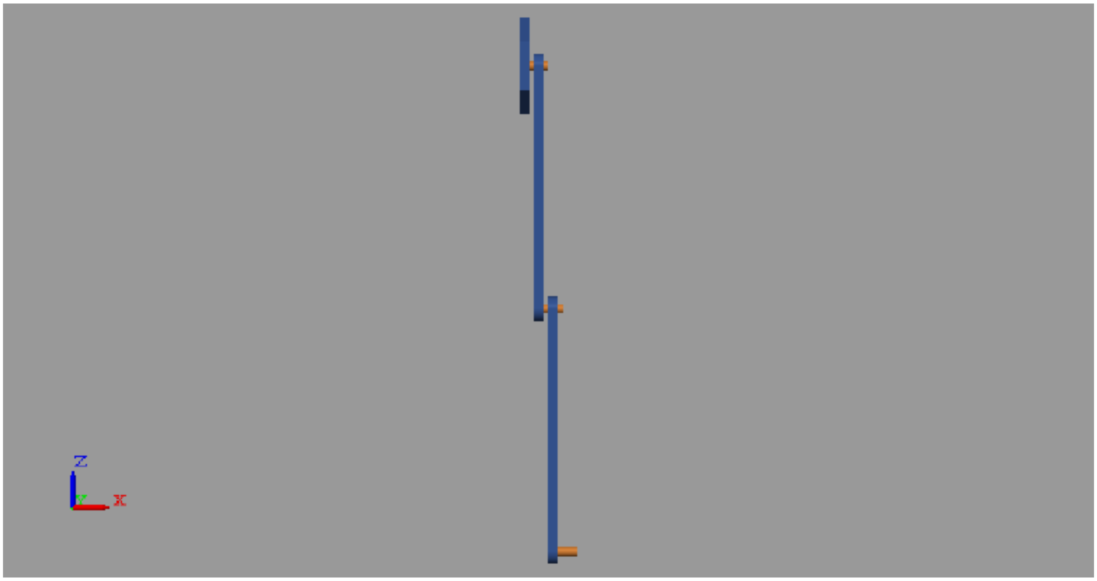
4 Connect the blocks as shown in the figure.



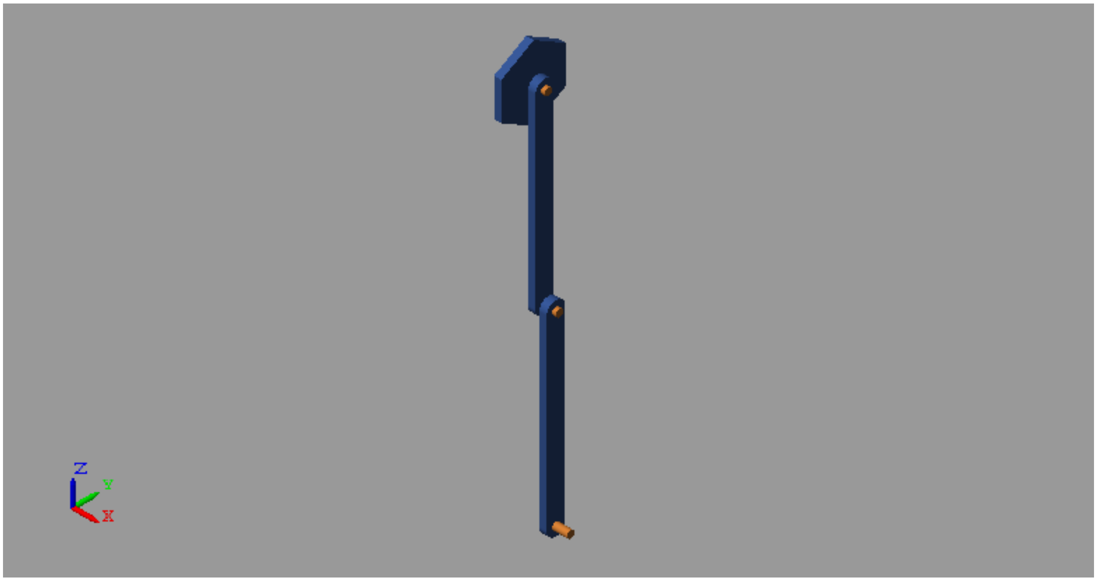
5 In the dialog boxes of the PS-Simulink Converter blocks, specify units of `N*m`.

Simulate Model

Run the simulation. Mechanics Explorer opens with a dynamic display of the two-bar model.



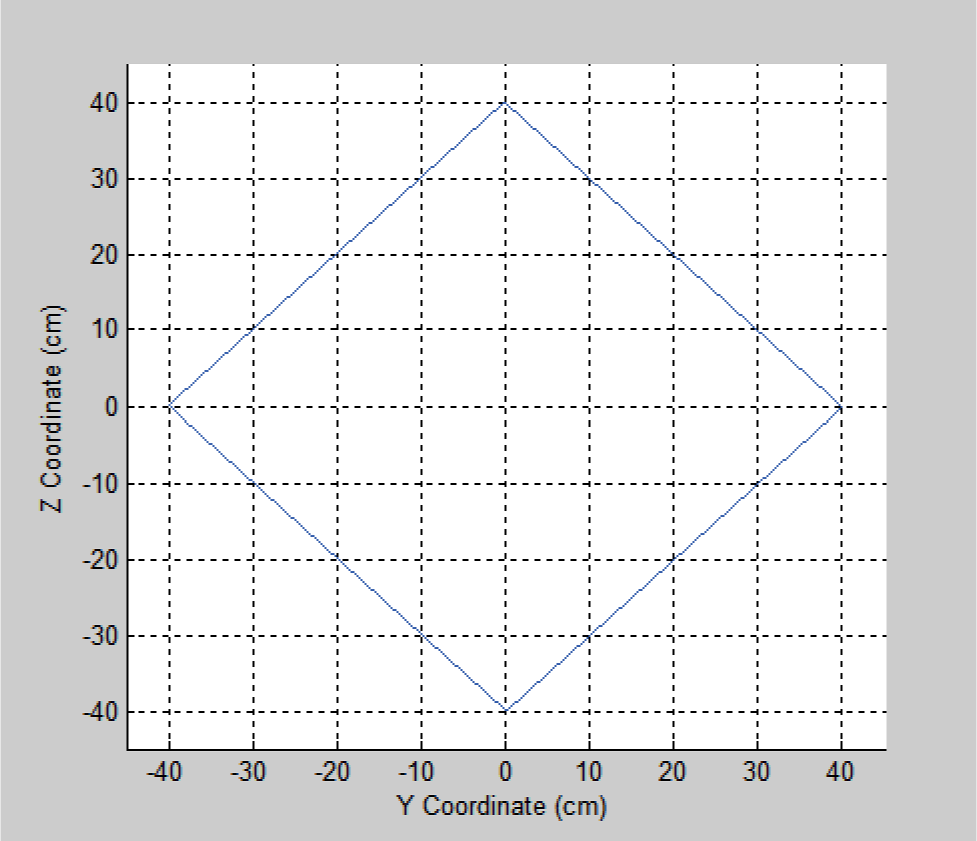
In the Mechanics Explorer toolstrip, click the isometric view button for a 3-D viewpoint.



Plot the position coordinates of the binary_link1 peg frame with respect to the world frame. At the MATLAB command line, enter this code:

```
figure(1);
plot(y_link.data, z_link.data, 'color', [60 100 175]/255);
axis([-45 45 -45 45]);
xlabel('Y Coordinate (cm)');
ylabel('Z Coordinate (cm)');
grid on;
```

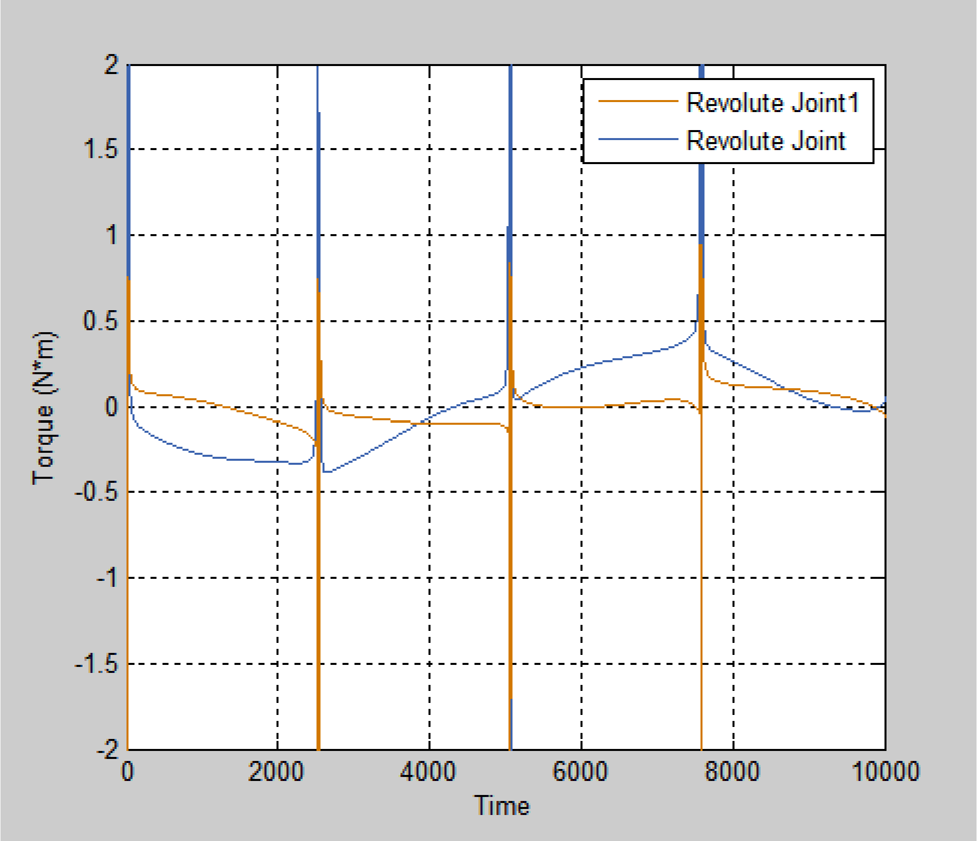
The figure shows the square trajectory that the binary_link1 peg frame traces during simulation.



Plot the computed actuation torques acting at the two revolute joints in the linkage. At the MATLAB command line, enter this code:

```
figure(2);
hold on;
plot(t1.data, 'color', [60 100 175]/255);
plot(t2.data, 'color', [210 120 0]/255);
xlabel('Time');
ylabel('Torque (N*m)');
grid on;
```

The figure shows the actuation torques required to achieve the square trajectory that you prescribed.



Related Examples

- "Sense Double-Pendulum Motion" on page 4-43
- "Prescribe Four-Bar Motion" on page 4-66
- "Specify Motion Input Derivatives" on page 4-15

Concepts

- "Joint Actuation" on page 4-7
- "Actuating and Sensing with Physical Signals" on page 4-18

Simulation and Analysis

Simulation

Configure Model for Simulation

During simulation, SimMechanics employs a Simulink global solver to determine the configuration of a model as a function of time. You can select the best solver for your application from a list of solvers that Simulink provides. Simulation parameters include the numerical step used to progress through the simulation and the solver tolerance values. Adjust the parameters to optimize speed and accuracy of the simulation.

For solver selection and parameter specification, see:

- “Choose a Solver” in the Simulink documentation.
- “Setting Up Solvers for Physical Models” in the Simscape documentation.

Specify Solver Settings

To select a global solver for your model:

1 On the Simulink menu bar, click **Simulation > Model Configuration Parameters**.

2 On the Tree View pane, select **Solver**.

3 In **Solver Options**, click **Type** and select `Variable-step` or `Fixed-step`.

Note For best performance, select `Variable-step`. For model deployment, select `Fixed-step`.

4 Click **Solver** and select the appropriate solver for your application. The default solver is `ODE45 (Dormand-Prince)`.

To modify the global solver parameters for your model:

1 In the **Solver options** pane of the **Model Configuration Parameters** window, enter the desired values for step size and tolerance parameters.

Reducing the values of the step size and tolerance parameters enhances simulation accuracy, but decreases simulation speed. Adjust the parameters to obtain an optimal trade-off between simulation speed and accuracy.

Related Examples

- "Configure Model for Simulation" on page 5-2
- "Configure Model for Rapid Acceleration Mode" on page 8-8
- "Find and Fix Simulation Issues" on page 5-4

Concepts

- "Visualizing and Inspecting a Model" on page 6-2

Find and Fix Simulation Issues

In this section...

Under certain conditions, a model that you simulate can behave in unexpected ways. Some issues that you can encounter while simulating a SimMechanics 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 SimMechanics

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—SimMechanics 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, SimMechanics issues a warning at the MATLAB command line. The Mechanics Explorer visualization utility does not open.

Simscape local solvers do not work with SimMechanics

SimMechanics 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 SimMechanics portion of a model. SimMechanics blocks continue to use the Simulink global solver that you select in **Model Configuration Parameters** for your model.

Note SimMechanics requires the Simulink global solver to be *continuous*. If the global solver is discrete, SimMechanics issues an error and the model does not simulate. This requirement applies to both fixed- and variable-step solvers.

Related Examples

- “Configure Model for Simulation” on page 5-2
- “Configure Model for Rapid Acceleration Mode” on page 8-8

Concepts

- “Visualizing and Inspecting a Model” on page 6-2

Visualization and Animation

Visualizing and Inspecting a Model

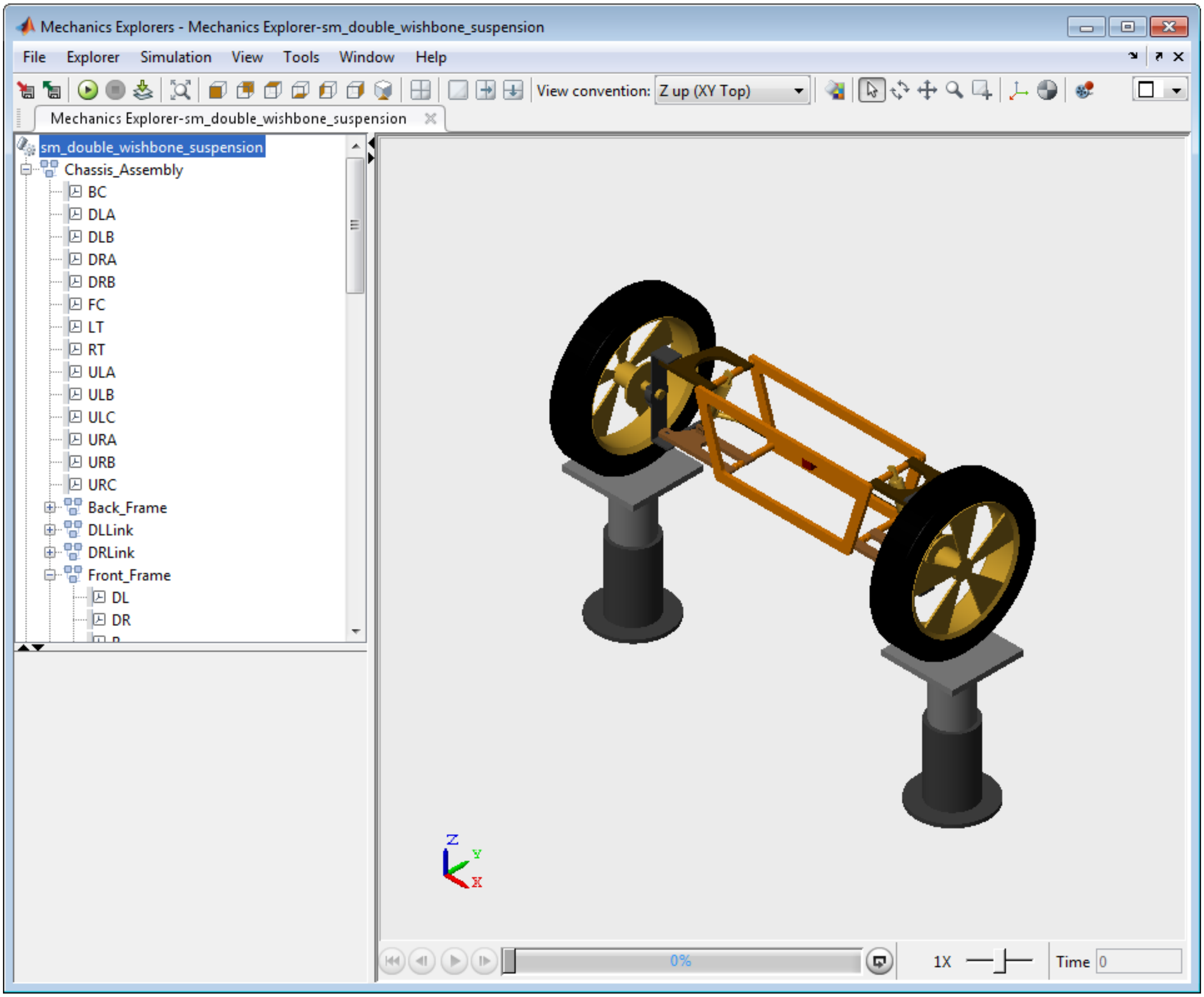
In this section...

Mechanics Explorer is a utility that provides 3-D visualization, model navigation, and troubleshooting tools for SimMechanics models. By default, each time you update or simulate a SimMechanics model, Mechanics Explorer displays the updated model. Use Mechanics Explorer frequently throughout the modeling process to uncover errors in rigid body geometry and frames or in multibody assembly.

Mechanics Explorer Window

The Mechanics Explorer window contains three primary panes:

- Visualization—Display a 3-D graphic representation of a multibody model.
- Model Navigation—Navigate the model by subsystem, block or port.
- Model Property—Inspect block properties and port connections in a model.



Visualization

The visualization pane of Mechanics Explorer displays a 3-D view of a SimMechanics model. The 3-D view is static when you update a model (**Ctrl+D**), or dynamic when you simulate a model (**Ctrl+T**). You can choose from seven preset views: front, back, top, back, left, right, and isometric. You can rotate, pan, and zoom a model. See "Visualizing and Inspecting a Model" on page 6-2.

Model Navigation

Identify a subsystem, block, or port with the model navigation pane. When you click the name of a subsystem, block, or port in the model navigation pane, the visualization pane highlights the corresponding entity with a light blue color. Use the model navigation pane to highlight multibody subsystems, rigid body subsystems, and frames in the visualization pane.

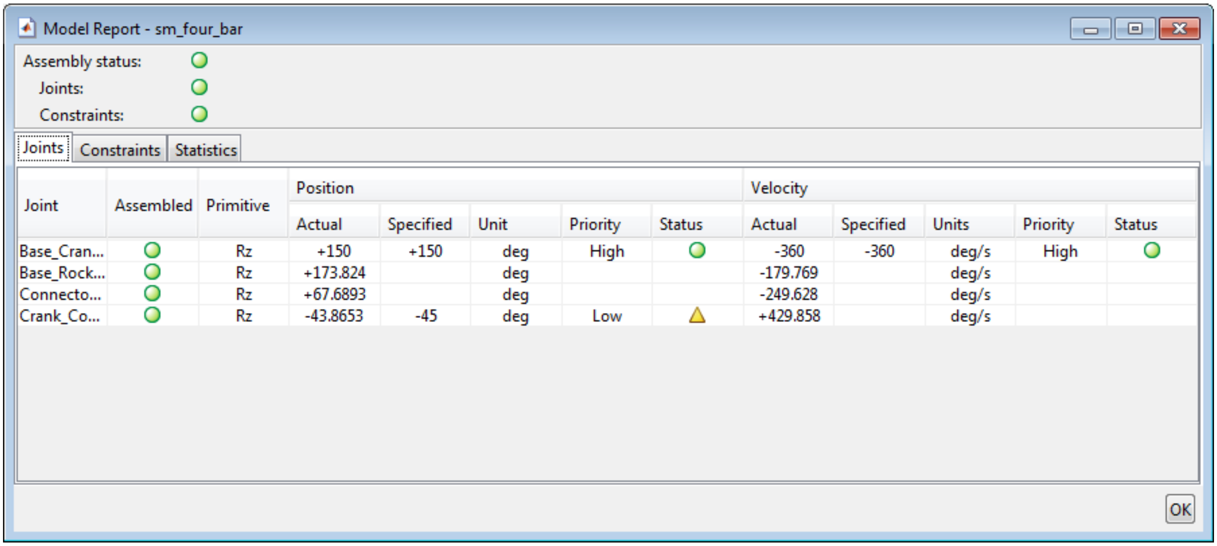
Model Property

Each time you click the name of an entity in the model navigation pane, the model property pane displays the parameters and frames associated with the selected entity. Use the model property pane to review the parameters and frames that belong to a subsystem, block, or port.

Model Report

Mechanics Explorer provides the Model Report tool to uncover model assembly problems. Model Report identifies the status of each joint and constraint in a model, and flags assembly errors. For joints with state targets, Model Report includes the actual and specified state targets. The report flags joints that have unsatisfied state targets. The following image shows the Model Report window for model `sm_four_bar`.

Note Model Report provides the actual state values for the assembled configuration. In models with prescribed motion inputs, this configuration may differ from the simulation time zero, making the actual state values unreliable. As a rule, avoid using Model Report in models with prescribed inputs.



Model Report - sm_four_bar

Assembly status: 🟢

Joints: 🟢

Constraints: 🟢

Joint	Assembled	Primitive	Position					Velocity				
			Actual	Specified	Unit	Priority	Status	Actual	Specified	Units	Priority	Status
Base_Cran...	🟢	Rz	+150	+150	deg	High	🟢	-360	-360	deg/s	High	🟢
Base_Rock...	🟢	Rz	+173.824		deg			-179.769		deg/s		
Connecto...	🟢	Rz	+67.6893		deg			-249.628		deg/s		
Crank_Co...	🟢	Rz	-43.8653	-45	deg	Low	⚠	+429.858		deg/s		

Animation

You can record the video of a simulation for future reference or to share.

Mechanics Explorer provides a **Record** button so that you can create the simulation video. All videos have a quality setting of 30 frames per second (fps) and AVI format. You can open a simulation video externally with any video player that supports AVI files. See "Record Animation Video" on page 6-22.

Related Examples

- "Configure Visualization Settings" on page 6-6
- "Rotate, Pan, and Zoom View" on page 6-18
- "Find and Fix Visualization Issues" on page 6-29

Concepts

- "Visualizing and Inspecting a Model" on page 6-2
- "Identifying Assembly Issues" on page 3-25

Configure Visualization Settings

In this section...

You can customize the display of Mechanics Explorer. Settings you can change include:

- Background color
- View point
- View convention
- Number of display windows for a model
- Visibility of frames and centers of mass

Change Background Color

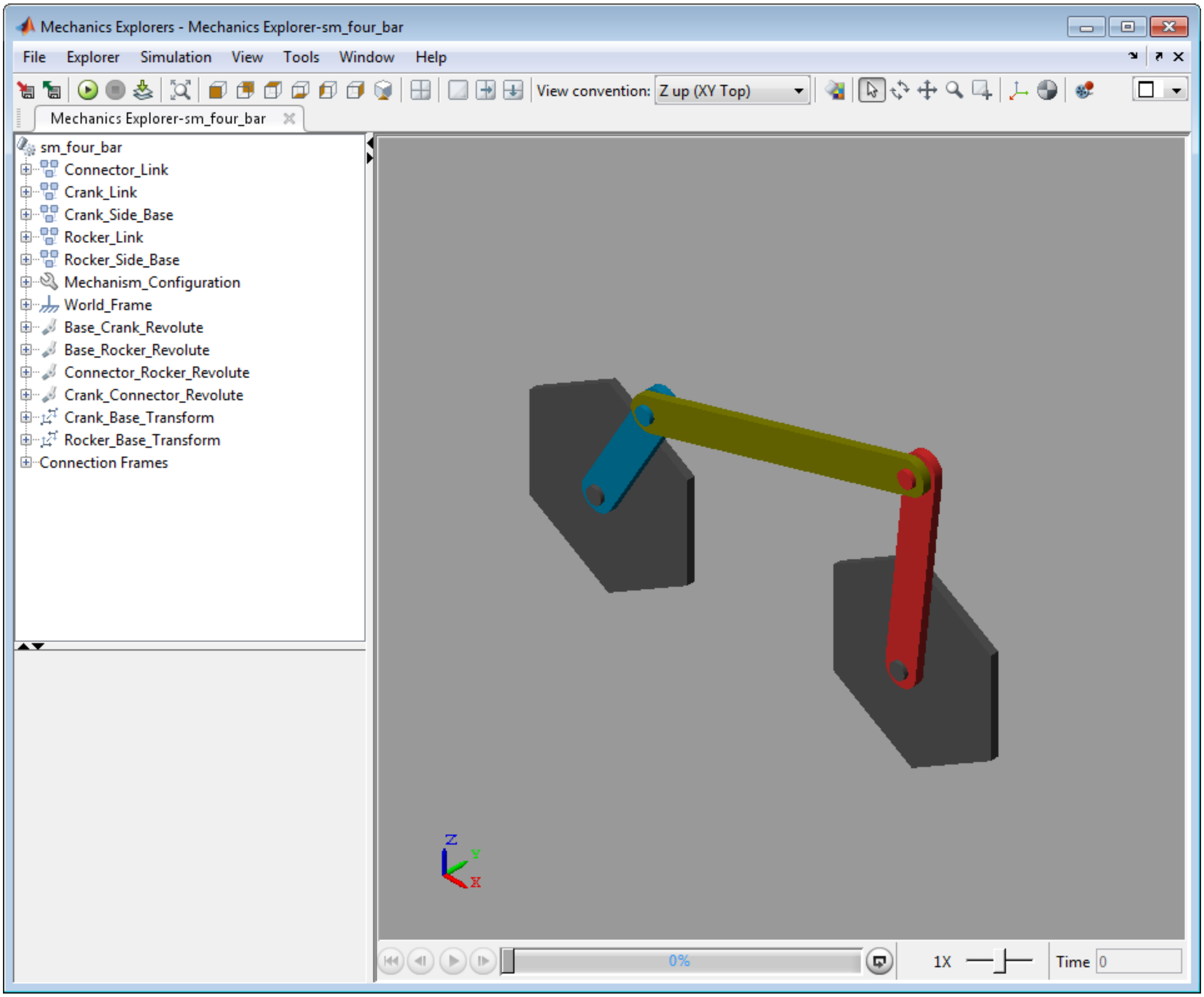
To change the background color, use the following procedure. The procedure uses the `sm_four_bar` as an example.

1 At the MATLAB command line, enter `sm_four_bar`.

Note Alternatively, open a SimMechanics model of your choice.

2 In the Simulink Editor window for the model, select **Simulation > Update Diagram**.

Note Mechanics Explorer opens with a display of your model against the default grey background.

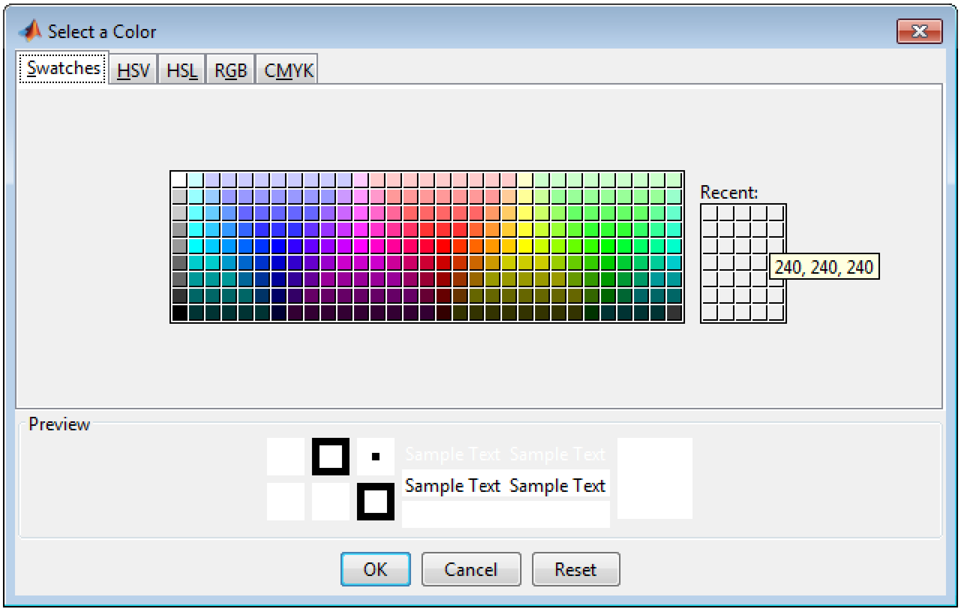


View convention: Z up (XY Top)

Time 0

3 In the Mechanics Explorer toolbar, click the icon.

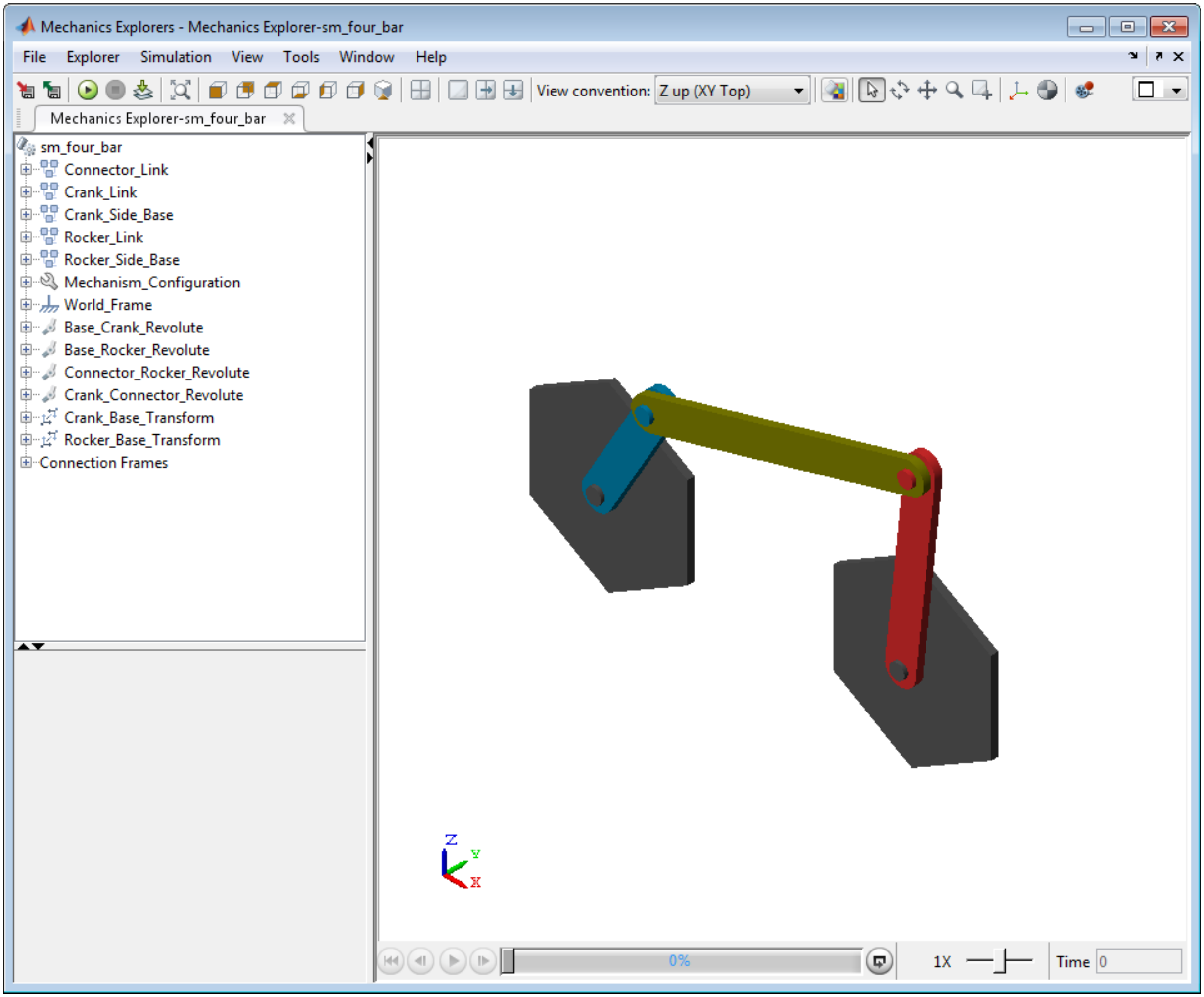
4 In the **Select a Color** dialog box, select a color. Click **HSV**, **HSL**, **RGB**, or **CMYK** tabs to specify a color in these formats.



5 Click **OK**.

6 In the Mechanics Explorer toolbar, click the icon.

Clicking the icon saves the current Mechanics Explorer configuration to the SimMechanics model. If you close the Mechanics Explorer window and update the model, Mechanics Explorer opens with the new configuration.



View convention: Z up (XY Top)

Time 0

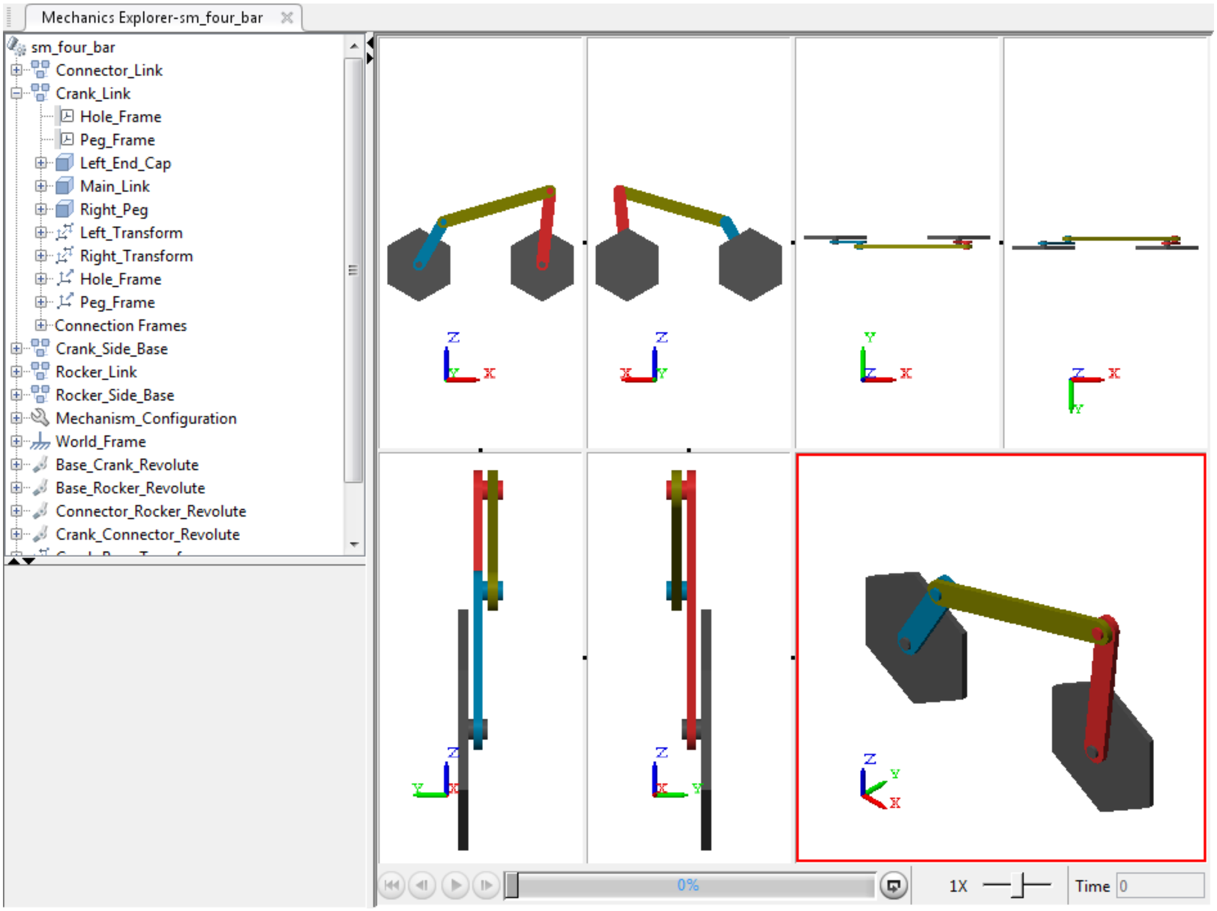
Change View Point

Mechanics Explorer provides seven view presets that you can use to change the perspective of a model. Each preset has an icon in the Mechanics Explorer toolbar .

Click an icon to select the corresponding view preset. The following table describes the seven presets in Mechanics Explorer.

View Icon	View Name	View Description
■	Front view	Display model ZX plane with Y axis pointing into screen
□	Back view	Display model ZX plane with Y axis pointing out of screen
□	Top view	Display model XY plane with Z axis pointing out of screen.
□	Bottom view	Display model XY plane with Z axis pointing into screen
□	Left view	Display model YZ plane with X axis pointing into screen
□	Right view	Display model YZ plane with X axis pointing out of screen
◈	Isometric view	Display model in 3-D with axes X, Y, and Z at 120° to each other.

The following figure shows the seven view presets in Mechanics Explorer. The top row shows the following four presets ordered left to right: front, bottom, top, bottom. The bottom row shows the following three presets ordered left to right: left, right, isometric.

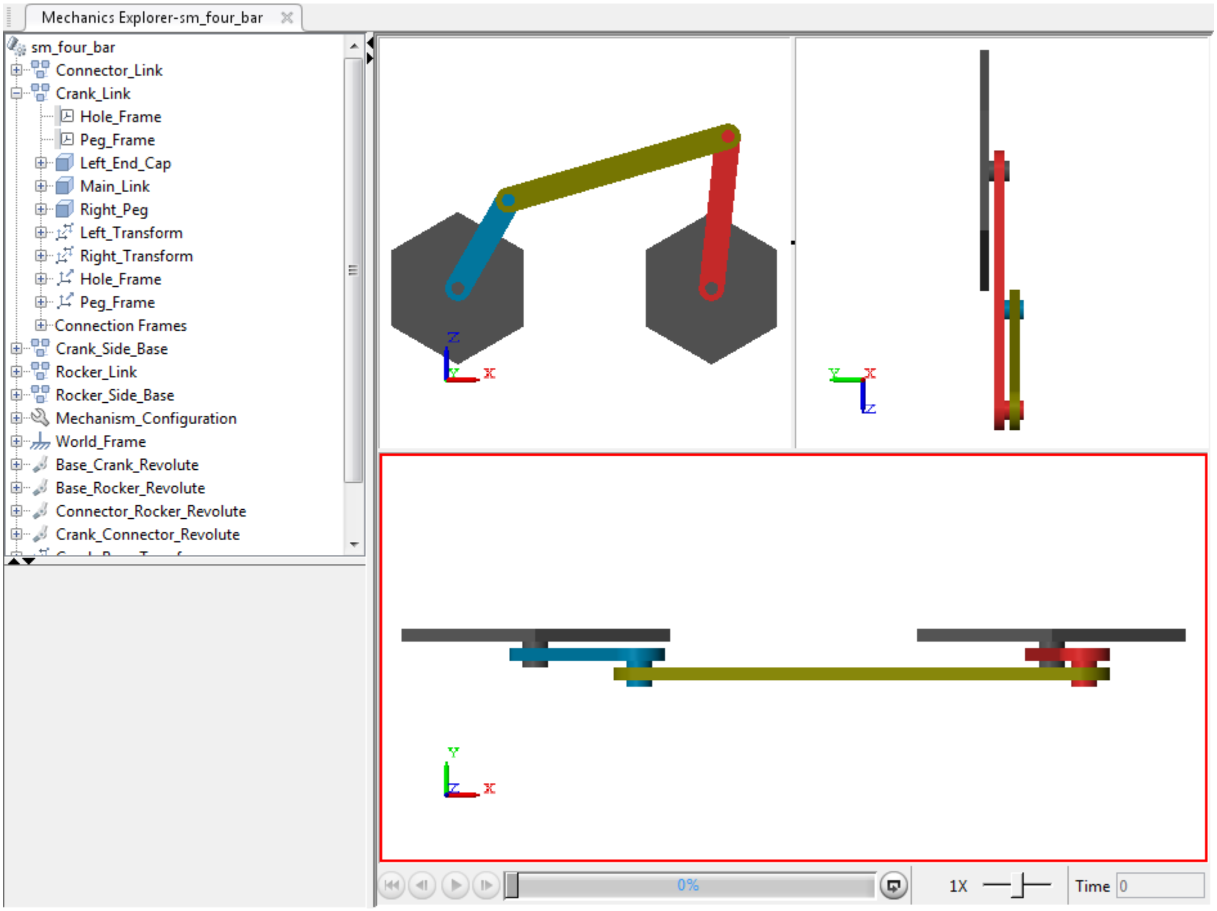


Change View Convention

You can choose from three view conventions:

- Z axis up—displays the model ZX plane in front view
- Z axis down—displays the model YZ plane in front view
- Y axis up—displays the model XY plane in front view

To select a view convention, click the **View convention** drop-down menu, and select one of the three view conventions. The following figure shows a four-bar model in front view using the three view conventions. The top row shows view conventions `Z up` and `Z down` ordered left to right. The bottom row shows view convention `Y up`.



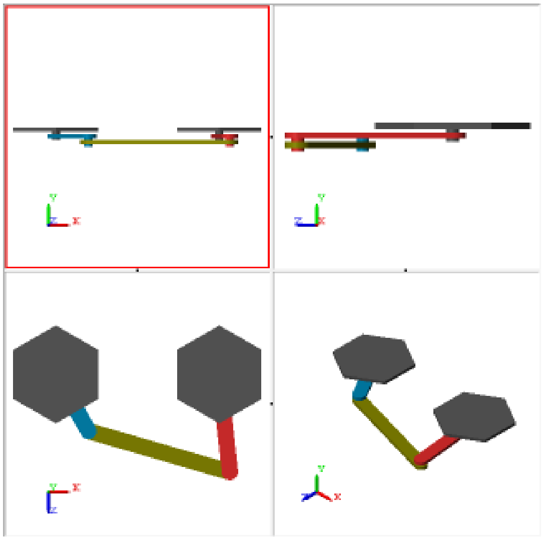
Display Multiple Screens

You can divide the Mechanics Explorer screen into multiple screens, each with an independent view of a model. The Mechanics Explorer toolbar provides icons ⊞ | □ → ↓ to split the active window into two windows vertically or horizontally.

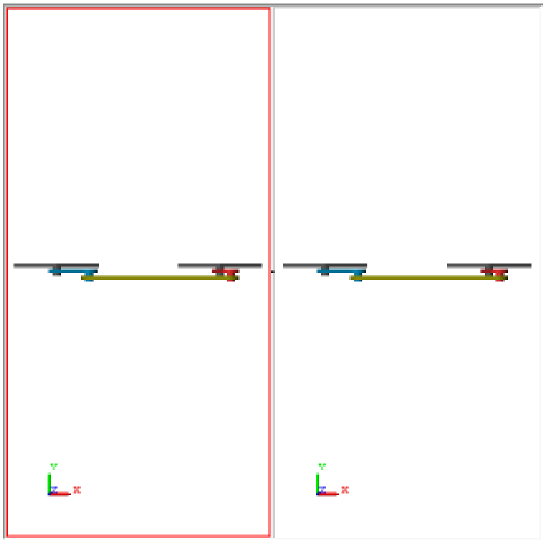
Each time you split the active window, you generate two smaller windows. You can split the active window an arbitrary number of times to generate as many view screens as you need. The following table describes the screen split icons.

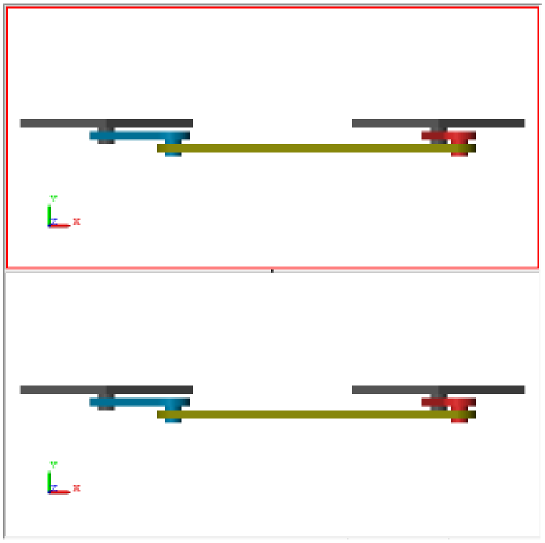
Icon	Icon Description
⊞	Split the active screen into four standard views
□	Display a single screen
→	Split the active screen vertically into two screens
↓	Split the active screen horizontally into two screens

The following image shows Mechanics Explorer with four standard views, in single screen mode, with two vertically split screens, and with two horizontally split screens.


y
z
x
y
z
x
y
x
z
y
z
x

y
z
x


y
z
x
y
z
x


y
z
x
y
z
x

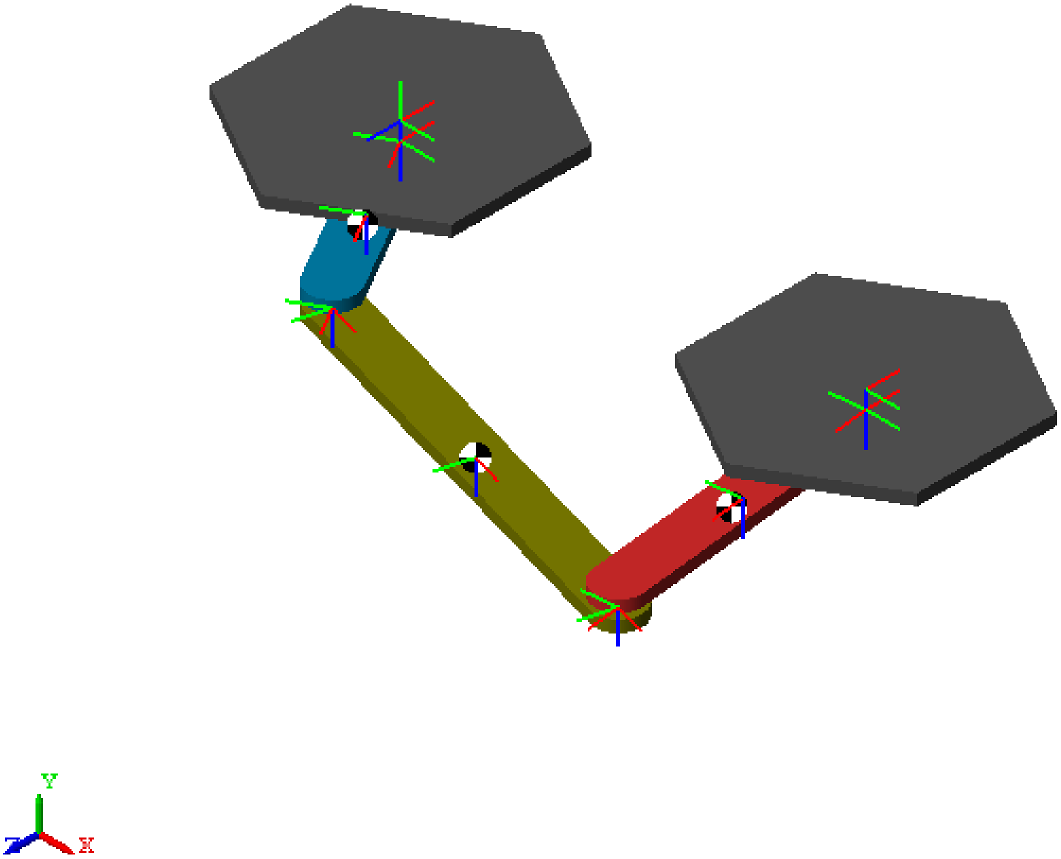
Toggle Visibility of Frames and Mass Centers

The Mechanics Explorer provides icons ∟ ◕ so that you can display and hide frames and center-of-mass markers.

To toggle frame visibility, click the ∟ icon.

To toggle the visibility of center-of-mass markers, click the ◕ icon.

The following figure shows a four-bar model that displays frames and center-of-mass markers.


Y
Z
X

Related Examples

- "Configure Visualization Settings" on page 6-6
- "Rotate, Pan, and Zoom View" on page 6-18
- "Find and Fix Visualization Issues" on page 6-29

Concepts

- "Visualizing and Inspecting a Model" on page 6-2
- "Identifying Assembly Issues" on page 3-25

Rotate, Pan, and Zoom View

In this section...

You can rotate, pan, and zoom your model in Mechanics Explorer. To do this, you use three buttons in the Mechanics Explorer toolbar: ⟳ ✥ 🔍. Select the button for the task you want to perform. Then, use the mouse to perform that task. You can also use mouse shortcuts to rotate, pan, and zoom.

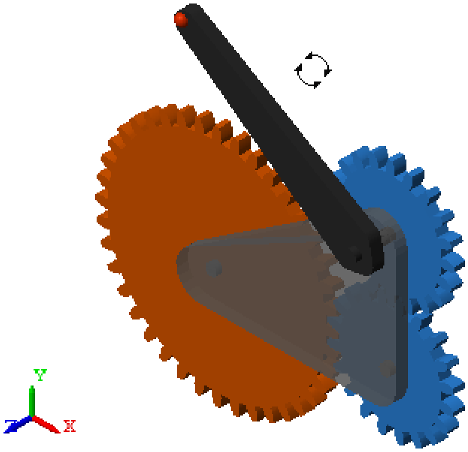
Rotate, Pan, and Zoom Shortcuts

The following table summarizes the mouse shortcuts that you can use to rotate, pan, and zoom a model.

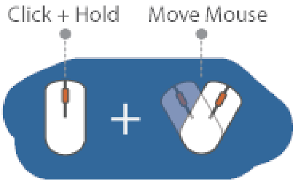
Function	Mouse Shortcut
Rotate	Press Scroll Wheel + Move Mouse
Pan	Press Scroll Wheel + Shift + Move Mouse
Zoom	Press Scroll Wheel + Ctrl + Move Mouse

Rotate View

In the Mechanics Explorer tool bar, click the **Rotate view** button ⟳. In the visualization pane, click the mouse to set the rotation pivot point. Then, move the mouse to rotate about that pivot. A rotation icon denotes the position of the mouse.

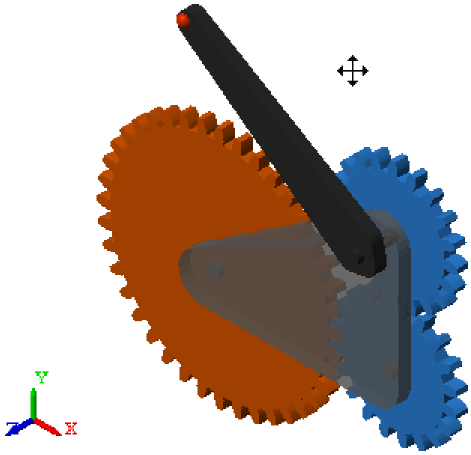


If you use a mouse with a scroll wheel, you can also use a shortcut to rotate your model. Click and hold the scroll wheel while moving the mouse in the visualization pane. As you move your mouse, the model rotates.

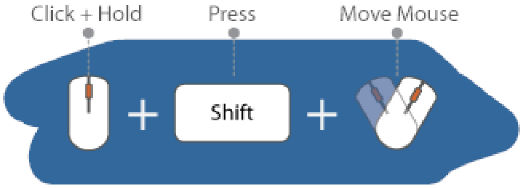


Pan View

In the Mechanics Explorer tool bar, click the **Pan view** button ✥. In the visualization pane, click and move the mouse to pan the model. A pan icon denotes the position of the mouse.

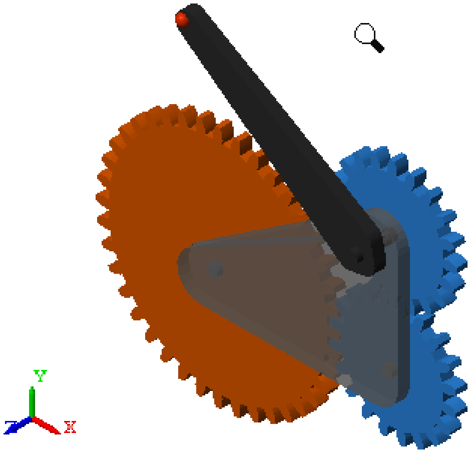


If you use a mouse with a scroll wheel, you can also use a shortcut to pan your model. In the visualization pane, click and hold the scroll wheel and press **Shift** while moving the mouse. As you move the mouse, the model pans.

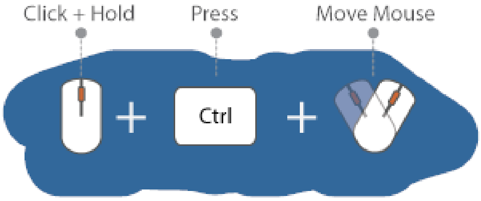


Zoom View

In the Mechanics Explorer tool bar, click the **Zoom in/out** button 🔍. In the visualization pane, click the mouse in the part that you want to zoom. Then, move the mouse to zoom that part. Move the mouse up to zoom in or down to zoom out. A zoom icon denotes the position of the mouse.



If you use a mouse with a scroll wheel, you can also use a shortcut to zoom your model. In the visualization pane, click and hold the scroll wheel and press **Ctrl** while moving the mouse. As you move the mouse, the model zooms. Move the mouse up to zoom in or down to zoom out.



Related Examples

- "Configure Visualization Settings" on page 6-6
- "Find and Fix Visualization Issues" on page 6-29
- "Record Animation Video" on page 6-22

Concepts

- "Visualizing and Inspecting a Model" on page 6-2
- "Identifying Assembly Issues" on page 3-25

Record Animation Video

With Mechanics Explorer, you can record a 3-D animation of your SimMechanics simulation. You can then play back the animation video without running the simulation again—or even opening the original model.

To record an animation, Mechanics Explorer provides a record button, . Recorded videos are in AVI format. The video playback speed is 30 frames per second.

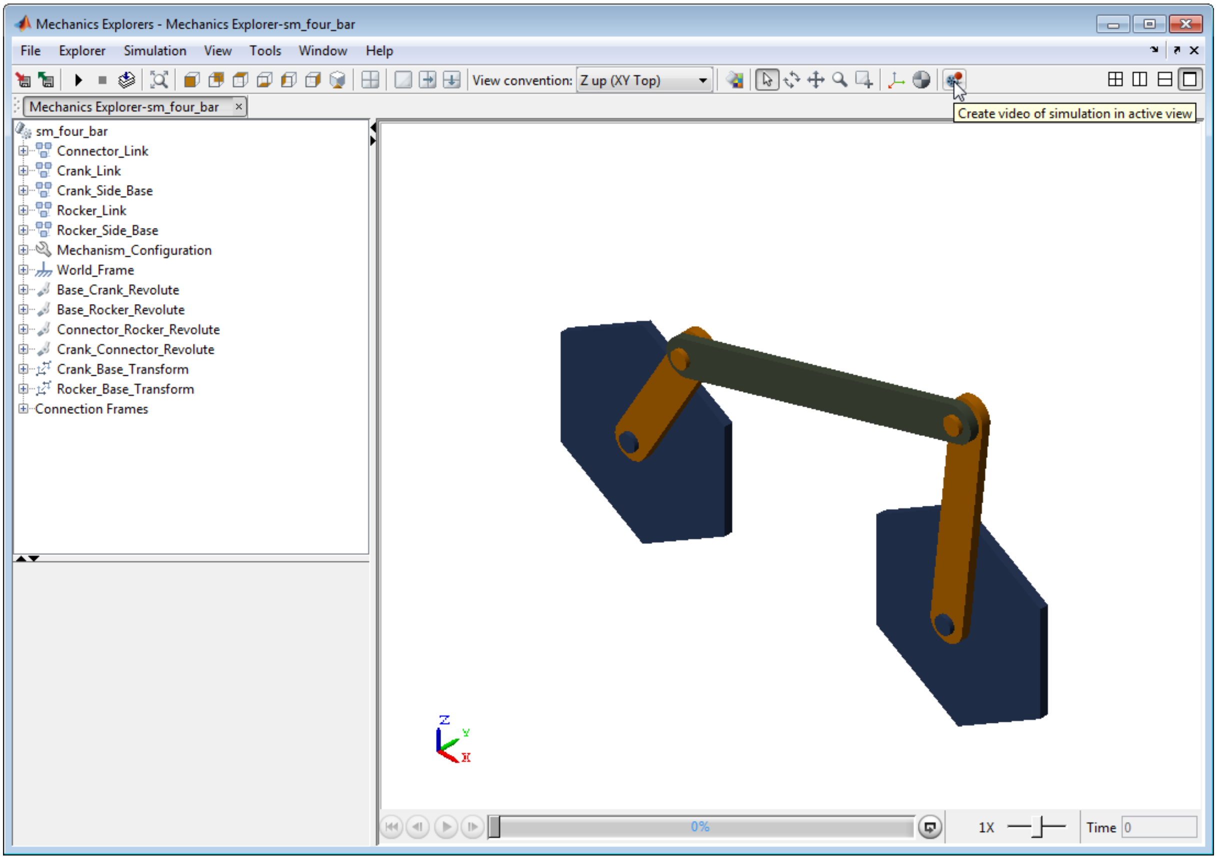
Record Video

This example shows how you can record a 3-D animation. The model in this example is sm_four_bar, which accompanies your SimMechanics installation.

1 At the MATLAB command line, enter `sm_four_bar`.

2 In the Simulink Editor window, select **Simulation > Run**.

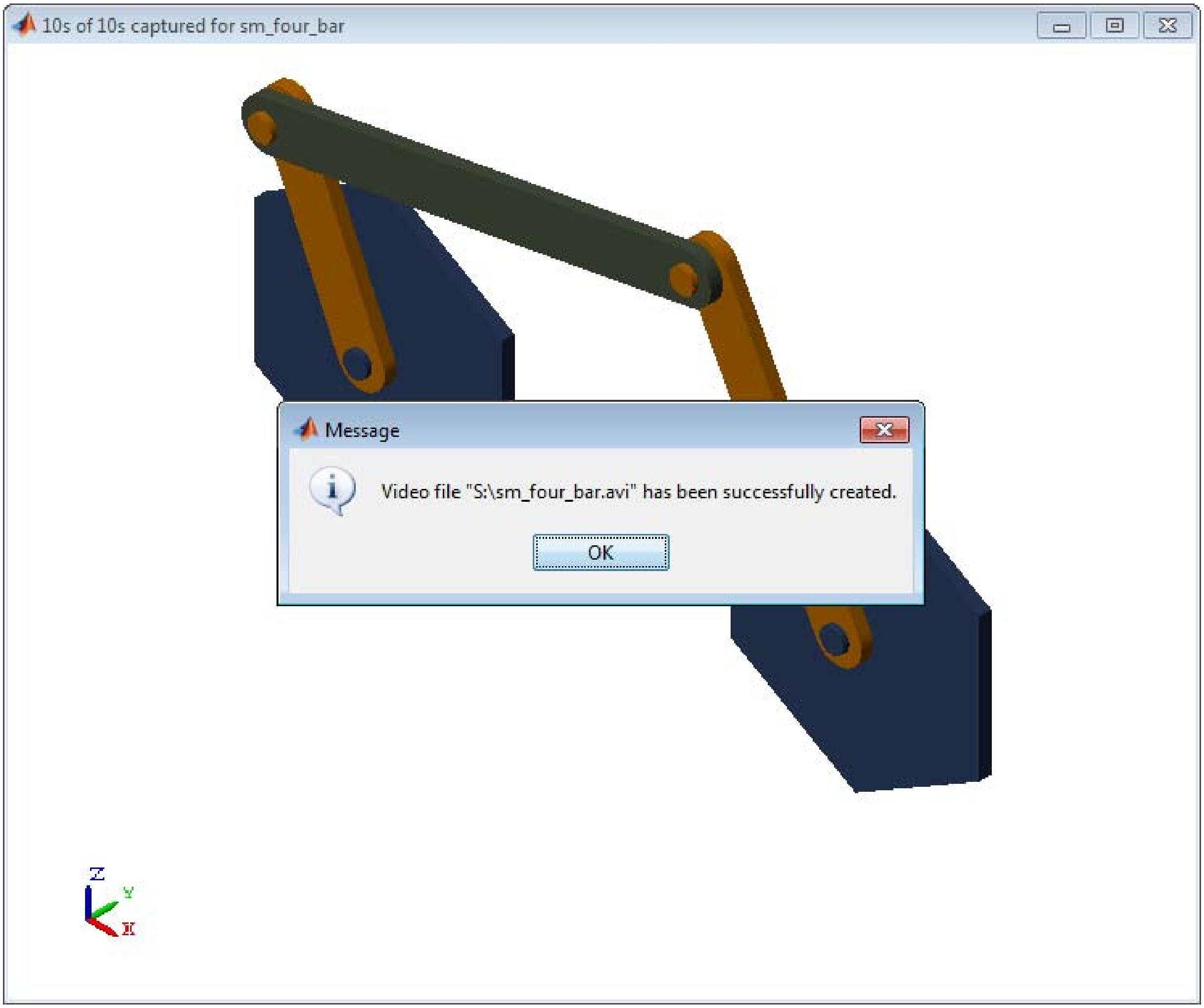
3 In the Mechanics Explorer window, press the **Record** button.



4 In the **Select video file** window, specify the name of the file.

5 Press **Save**.

A new animation window opens when you press **Save**. The title bar of the new window provides the recording progress status. When a new window opens informing you that the recording has finished, click **OK**



10s of 10s captured for sm_four_bar

Message

Video file "S:\sm_four_bar.avi" has been successfully created.

Related Examples

- "Adjust Video Playback Speed" on page 6-25
- "Configure Visualization Settings" on page 6-6
- "Rotate, Pan, and Zoom View" on page 6-18
- "Find and Fix Visualization Issues" on page 6-29

Concepts

- "Visualizing and Inspecting a Model" on page 6-2
- "Identifying Assembly Issues" on page 3-25

Adjust Video Playback Speed

In this section...
"Variable-Step Solvers" on page 6-25
"Fixed-Step Solvers" on page 6-27

SimMechanics animation videos play at a fixed speed of 30 frames per second (fps), with each frame corresponding to a simulation time step. When the step size differs from the 1/30 second duration of a video frame, the video speed differs from the simulation speed. To ensure that the two speeds are equal, you must adjust the configuration parameters for your model. The exact approach depends on the type of solver that you select: variable-step or fixed-step.

Variable-Step Solvers

Variable-step solvers are commonly used in SimMechanics simulations. With a variable-step solver, the step size can vary between minimum and maximum values that you specify in your model's **Configuration Parameters** menu. Because each video frame corresponds to a simulation step, a variable step size can introduce time distortion into the video.

For example, when the step size is larger than 1/30 second, it must shrink to fit the 1/30 second duration of a video frame, causing the video to appear faster than the simulation. Similarly, when the step size is smaller than 1/30 second, it must expand to fit the 1/30 second duration of a video frame, causing the video to appear slower than the simulation.

To avoid time distortion in the video, you must sample the simulation at regularly spaced intervals. By using a 1/30 second sampling time interval, you can ensure that the resulting video plays at the simulation speed:

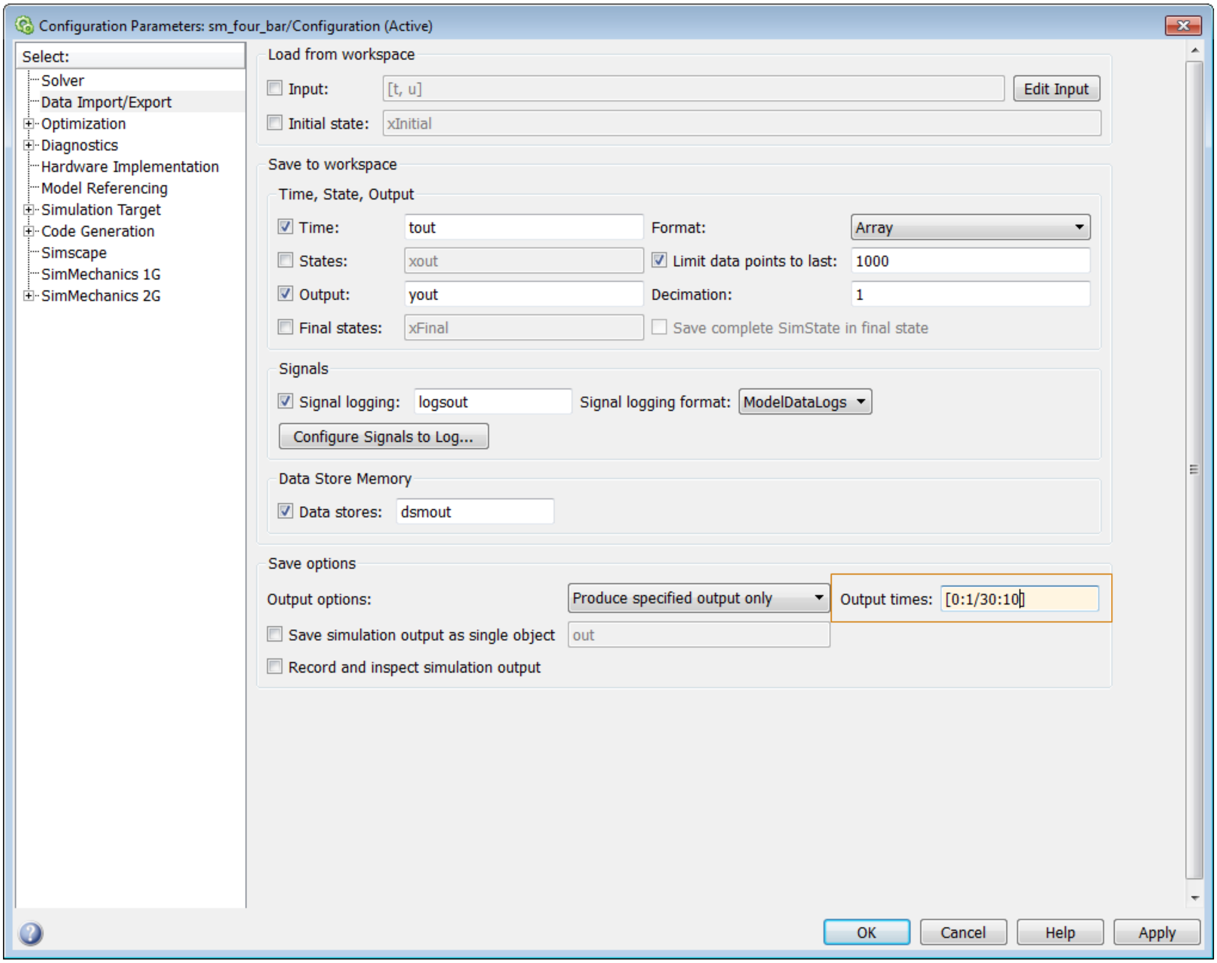
1 On the Simulink menu bar, select **Simulation > Model Configuration Parameters**.

2 On the **Configuration Parameters** tree browser, select **Data Import/Export**.

3 In the **Output Options** drop-down list of the **Save options** pane, select `Produce specified output only`.

4 In **Output times**, enter `(1:N)*dt`, replacing *N* with the number of data samples, and *dt* with the sampling time interval in seconds, `1/30`. This array specifies the times at which to record the frames of the video.

In terms of the duration of a simulation *T*, the number of samples *N* equals T/dt. For example, if the simulation lasts ten seconds (T = 10) and the sampling time interval is 1/30 second (dt = 1/30), then N = 300. In this case, the array you enter in **Output times** must be `(1:300)*1/30`.



Load from workspace

□ Input: [t, u]

□ Initial state: xInitial

Save to workspace

Time, State, Output

☑ Time: tout — Format: Array

□ States: xout — ☑ Limit data points to last: 1000

☑ Output: yout — Decimation: 1

□ Final states: xFinal — □ Save complete SimState in final state

Signals

☑ Signal logging: logsout — Signal logging format: ModelDataLogs

Data Store Memory

☑ Data stores: dsmout

Save options

Output options: Produce specified output only — Output times: [0:1/30:10]

□ Save simulation output as single object: out

□ Record and inspect simulation output

If you change the time interval *dt* in the array from 1/30, the video playback speed changes accordingly. Changing *dt* to 1/15 causes the resulting video to play at twice the simulation speed. Similarly, changing *dt* to 1/60 second causes the resulting video to play at half the simulation speed.

Fixed-Step Solvers

Fixed-step solvers are less commonly used in SimMechanics simulations. With a fixed-step solver, the simulation step size remains constant at a value that you specify in your model's **Configuration Parameters** menu. Because the step size is constant, the resulting video displays no time distortion. It can, however, play at a different speed than the simulation.

To change the playback speed of the video, you must change the step size of the simulation. Change the step size to 1/30 second to ensure that the video plays at the same speed as the simulation:

1. On the Simulink menu bar, select **Simulation > Model Configuration Parameters**.
2. On the **Solver options** pane, check that **Type** is set to `Fixed-step`.
3. In **Fixed-step size (fundamental sample time)**, enter `1/30`.

Changing the step size from `1/30` causes the animation video to play at a different speed. The effect of changing the step size is similar to the effect of changing the sampling time interval for a variable-step solver. Changing the step size to `2/30` causes the resulting video to play at twice the simulation speed. Similarly, changing the step size to `1/60` causes the resulting video to play at half the simulation speed.

Note Model dynamics take precedence over video playback considerations. Select a solver and step size based on the dynamics of your model. Then, if possible, adjust the time step to control the video playback speed.

Related Examples

- "Record Animation Video" on page 6-22
- "Configure Visualization Settings" on page 6-6
- "Rotate, Pan, and Zoom View" on page 6-18

Find and Fix Visualization Issues

In this section...
"Mechanics Explorer Fails to Open" on page 6-29
"Model appears with different orientation in Mechanics Explorer" on page 7-41
"Part appears invisible in Mechanics Explorer" on page 7-43

Under certain conditions, a model that you visualize can behave in unexpected ways. Some issues that you can encounter while attempting to visualize a model include:

- Mechanics Explorer fails to open
- Model appears with different orientation in Mechanics Explorer
- Part appears invisible in Mechanics Explorer

Mechanics Explorer Fails to Open

By default, Mechanics Explorer is set to open the first time you update a model. If a Mechanics Explorer window is already open for your model, the open window updates the model display. Note, however, that updating a model does not automatically bring the Mechanics Explorer window to the front. If the Mechanics Explorer window is hidden during model update, you must bring that window to the front to see the updated model.

Set Mechanics Explorer to Open on Model Update

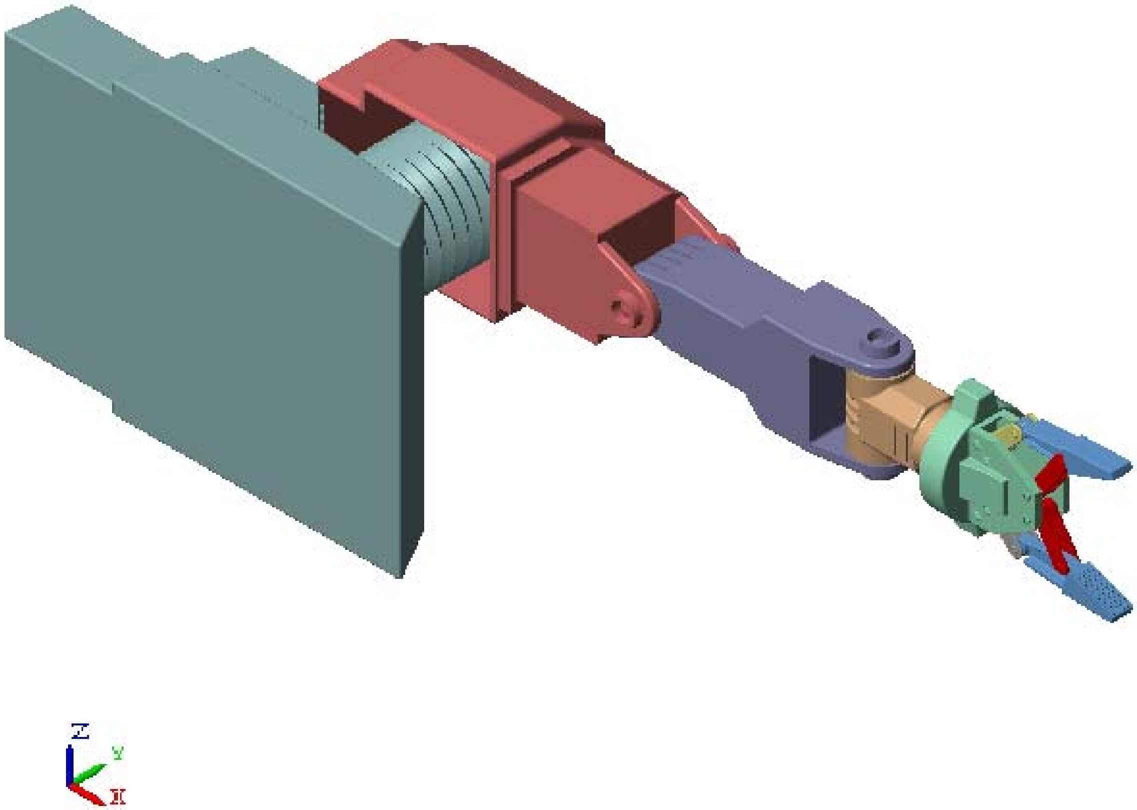
If Mechanics Explorer fails to open during model update, check that Mechanics Explorer is set to open on model update:

1. In the Simulink Editor menu bar, select **Simulation > Model Configuration Parameters**.
2. Expand the **SimMechanics 2G** node.
3. Click **Explorer**.

4 Verify that **Open Mechanics Explorer on model update or simulation** is selected.

Model appears with different orientation in Mechanics Explorer

By default, Mechanics Explorer displays a model with the Z axis of the World frame pointing up. Using this convention, the default gravity vector `[0 0 -9.81] m/s^2` points down, a direction that is practical for most applications. However, this convention differs from that which CAD platforms commonly use, Y axis up, causing Mechanics Explorer to display some models sideways. If this happens, you can manually change the view convention to that used in the original CAD assembly. The figure shows the default Mechanics Explorer display of an imported robot arm model.



Change View Convention

To change the view convention of a model:

1. In the Mechanics Explorer toolbar, click the **View Convention** drop-down menu.
2. Select **Y up (ZX Top)**.

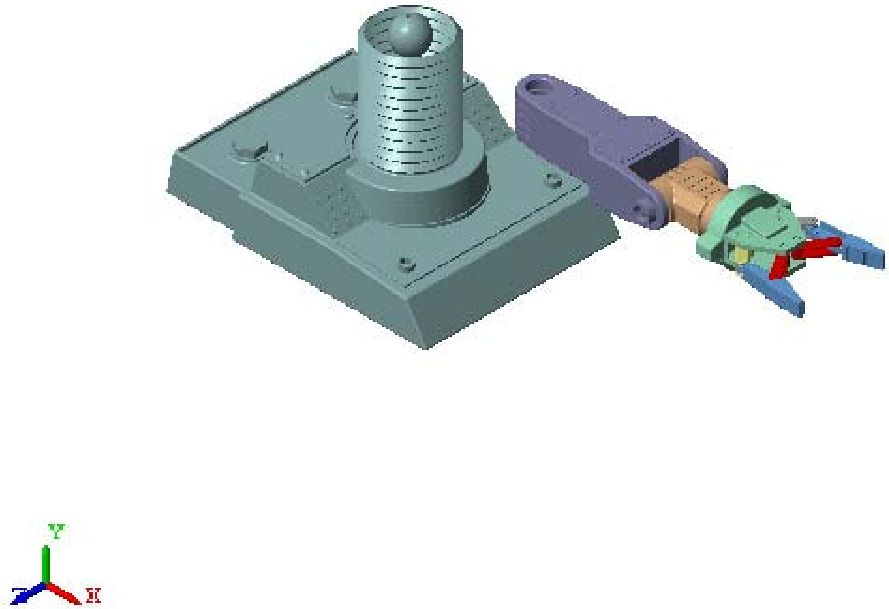
3 Refresh the Mechanics Explorer display by selecting a view point from the Mechanics Explorer tool bar.

Mechanics Explorer displays the model using the new view convention.

Part appears invisible in Mechanics Explorer

During CAD import, SimMechanics uses a set of stereolithographic (STL) files to generate the 3-D surface geometry of each CAD part. If SimMechanics cannot load the STL file for a part, that part appears invisible in Mechanics Explorer. This issue does not affect model update or simulation.

The figure shows the Mechanics Explorer display of an imported model containing an invalid STL file.



Correct Visualization Issue

If a part of an imported model appears invisible in Mechanics Explorer:

1 In Mechanics Explorer, identify the name of each invisible part.

2 In the block diagram, open the dialog boxes of the associated Solid blocks.

3 In the **Geometry** section, check that the name and location of the STL files are correct.

   If either is incorrect, enter the correct information and update the model. Check that Mechanics Explorer displays the invisible part. If not, check if the STL files are valid.

STL File Issues

To visualize a CAD assembly that you import, SimMechanics relies on a set of STL files that specify the 3-D surface geometry of the CAD parts. Each STL file specifies the surface geometry of one CAD part as a set of 2-D triangles. To do this, the STL files contain:

- [X Y Z] coordinates of the triangle vertices
- [X Y Z] components of the normal vectors for the triangles.

If an STL file specifies a normal vector with zero length, SimMechanics issues a warning. The STL file fails to load.

Related Examples

- “Configure Visualization Settings” on page 6-6
- “Rotate, Pan, and Zoom View” on page 6-18

Concepts

- “Visualizing and Inspecting a Model” on page 6-2
- “Identifying Assembly Issues” on page 3-25

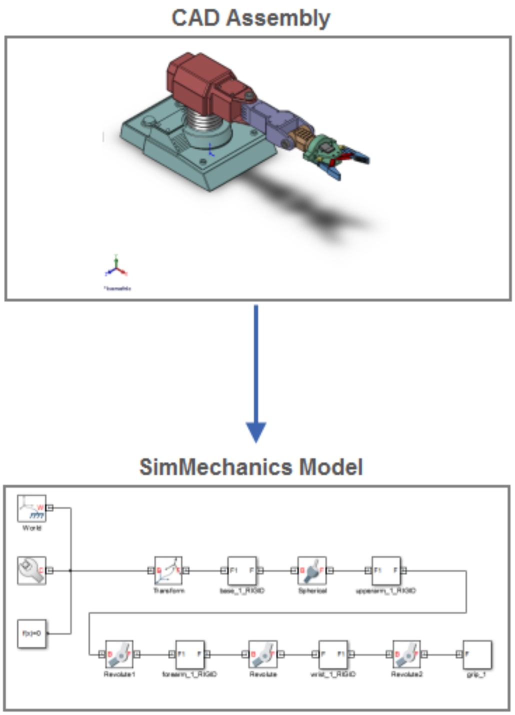
CAD Import

About CAD Import

CAD Translation

In this section...

You can translate a CAD assembly into a SimMechanics model for simulation and analysis. This process is called CAD translation. By translating a CAD assembly into a SimMechanics model, you leverage the strengths of your CAD platform with the strengths of SimMechanics software. You can modify any model that you translate—for example, adding actuators and sensors—to fit the needs of your application. CAD translation is especially useful for control system design.

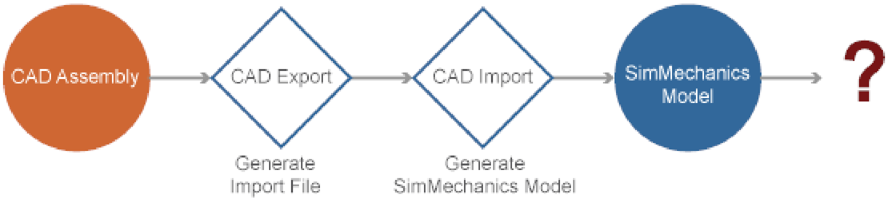


CAD Translation Steps

CAD translation is a two-step process. First, you export a CAD assembly in XML format. Then, you import the XML file into SimMechanics. SimMechanics uses the XML file to automatically generate a model that replicates the original CAD assembly. If the CAD assembly contains only supported constraints, CAD import requires no additional work on your part. Once SimMechanics generates your model, you are ready to simulate and analyze that model. The table summarizes the two CAD translation steps.

Translation Step	Description
CAD Export	Generate XML import file from CAD assembly
CAD Import	Generate SimMechanics model from import files

You must export a CAD assembly before you import it into SimMechanics. The schematic shows the CAD translation step sequence. A CAD assembly is the starting point of CAD translation. Exporting that assembly in XML format and importing the resulting XML file into SimMechanics produces an equivalent SimMechanics model.



Software Requirements

The table provides the software requirements for CAD translation. The requirements depend on the CAD translation step—export or import. For example, a CAD platform is a requirement only for CAD export.

Software	Notes	CAD Export	CAD Import
CAD Platform		✓	
MATLAB	Registration as computing server required	✓	✓
SimMechanics			✓
SimMechanics Link		✓	

The software requirements for CAD translation are optimized for cooperation between CAD and SimMechanics engineers. A CAD engineer can export the CAD assembly without an active SimMechanics installation. Likewise, a SimMechanics engineer can import the CAD assembly without an active CAD platform installation.

See Also

`smimport`

Related Examples

- “Install and Register SimMechanics Link Software” on page 7-9
- “Import Robot Arm Model” on page 7-27
- “Import Stewart Platform Model” on page 7-33
- “Find and Fix CAD Import Issues” on page 7-40

Concepts

- “CAD Import” on page 7-5

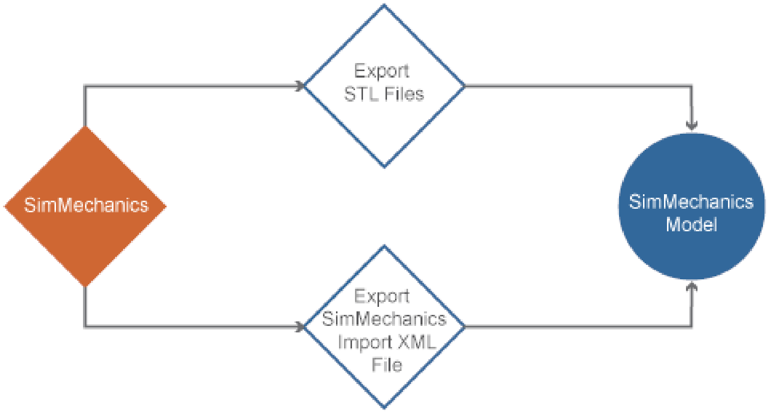
CAD Import

In this section...

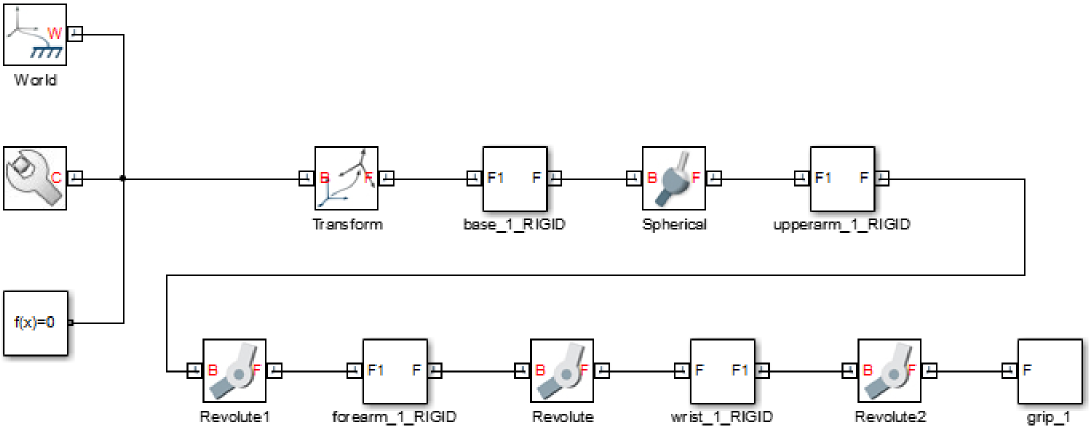
CAD Import is the second and final step of CAD translation. During CAD import, SimMechanics interprets the SimMechanics Import XML file generated during CAD Export. Then, based on the structure and parameters that the XML file provides, SimMechanics automatically generates model that replicates the original CAD assembly.

Importing a Model

CAD Import does not require access to the original CAD assembly or associated CAD platform. Access to the surface-geometry STL files is not required for simulation, but it is required for visualization. You can simulate an imported model that contains no STL files. However, the Mechanics Explorer visualization utility cannot display a representation of a model without the STL files.



In the model, each CAD part maps into a rigid body subsystem. Each CAD constraint or set of CAD constraints, map into a joint. Block names for SimMechanics subsystems are based on the original CAD parts and subassemblies which the subsystems represent. SimMechanics appends the suffix `RIGID` to the stem of a rigid body name. For example, CAD part `base` translates into rigid body subsystem `base_RIGID`. The following figure shows the imported SimMechanics model of a CAD robot assembly.

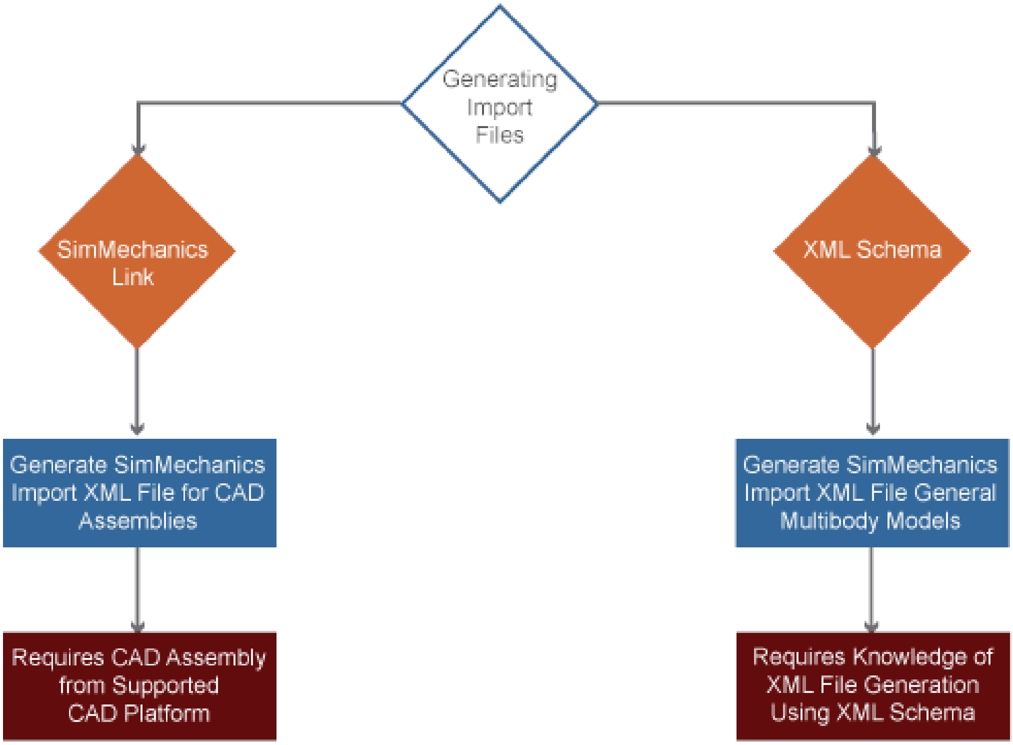


Modify SimMechanics model to fit the needs of your application.

Generating Import Files

To import a multibody model into SimMechanics, you must first generate the SimMechanics Import XML file. You can generate this file automatically, using the SimMechanics Link utility, or manually, using the XML schema that MathWorks® provides. The method that you use depends on the type of model that you want to import. The table summarizes the two methods and their limitations.

Import File Generation Method	Limitations
SimMechanics Link	Works only for CAD assemblies. CAD assembly must come from one of three supported CAD platforms.
XML Schema	Requires knowledge of XML file generation based on XML schema


Generating Import Files
SimMechanics Link
XML Schema
Generate SimMechanics Import XML File for CAD Assemblies
Generate SimMechanics Import XML File General Multibody Models
Requires CAD Assembly from Supported CAD Platform
Requires Knowledge of XML File Generation Using XML Schema

SimMechanics Link is a free utility that MathWorks provides. Use this utility to generate the SimMechanics Import XML file that you need to import a CAD assembly into SimMechanics. For more information about SimMechanics Link , see "Install and Register SimMechanics Link Software" on page 7-9.

SimMechanics XML Schema

The XML Schema is a set of files written according to the W3C XML Schema specification. MathWorks provides these files so that you can generate a SimMechanics Import XML file manually or using an external application. Use the XML Schema to generate the SimMechanics Import XML file for a CAD assembly or other multibody model.

The XSD files describe the elements and attributes that a SimMechanics Import XML file can contain and the order in which they must appear. Generating an XML file in accordance with the XML schema ensures that SimMechanics can successfully import it. Once you have generated the XML file, validate it against the schema to ensure SimMechanics can import it without issue.

To access the SimMechanics XML schema, visit the SimMechanics product website. Follow instructions to download the XSD files.

See Also

`smimport`

Related Examples

- “Install and Register SimMechanics Link Software” on page 7-9
- “Import Robot Arm Model” on page 7-27
- “Import Stewart Platform Model” on page 7-33
- “Find and Fix CAD Import Issues” on page 7-40

Concepts

- “CAD Translation” on page 7-2

Install and Register SimMechanics Link Software

In this section...

SimMechanics Link Installation Requirements

Before installing the SimMechanics Link utility, check that an active installation of the following software exists on your computer:

- MATLAB
- Supported CAD platform

MATLAB and SimMechanics Link must belong to the same release. For example, if your MATLAB release is R2012b, then your SimMechanics Link release must also be R2012b. Combining different release numbers can cause installation errors.

SimMechanics Link supports three CAD platforms:

- SolidWorks®
- Autodesk Inventor®
- PTC® Creo™ (Pro/ENGINEER®

You can use the SimMechanics Link utility to export a CAD assembly from any of these CAD platforms. Note that MATLAB, SimMechanics Link, and your CAD platform must chare the same architecture (e.g. 64-bit).

Download SimMechanics Link Software

You can download SimMechanics Link software directly from the MathWorks website:

1 Visit the SimMechanics Link download website at `http://www.mathworks.com/products/simmechanics/download_smlink.html`.

2 Select the software version to install.

3 Click **Submit**.

4 Save the installation files in a convenient folder.

Do not extract the zip file.

Install SimMechanics Link Software

Install SimMechanics Link software from the MATLAB command line:

1 Start MATLAB.

Note You may need administrator privileges to complete the installation.

2 At the MATLAB command line enter:

```
path(path, '<installation_file_folder>')
```

replacing `<installation_file_folder>` with the path to the folder with the installation files.

3 At the MATLAB command line, enter:

```
install_addon('<zip_file_name>.zip')
```

replacing `<zip_file_name>` with the name of the zip file that you downloaded (e.g., smlink.r2012b.win64). The command extracts the zip archive files to the MATLAB root directory.

Register SimMechanics Link Utility with CAD Platform

Complete the installation by registering your the SimMechanics Link utility with your CAD platform. The registration procedure makes SimMechanics Link available in your CAD platform as an Add-In tool. Once you have completed the linking procedure, you can use the Add-In tool to export a CAD assembly directly from your CAD platform.

The registration procedure is different for each supported CAD platform. The following table provides platform-specific registration information. Click the link that matches your CAD platform, and complete the registration procedure.

To register with CAD platform...	...click here
Autodesk Inventor	"Register SimMechanics Link with Inventor®"
PTC Creo (Pro/ENGINEER)	"Register SimMechanics Link with Creo"
SolidWorks	"Register SimMechanics Link with SolidWorks"

Link External Application to SimMechanics Link Software

You can link an unsupported CAD platform or other external application to SimMechanics software. For this task, SimMechanics Link provides an application programming interface (API) with a set of functions that you can use to create a C/C++ custom export module. For an overview of custom export using the API, see "Custom Export with SimMechanics Link API".

Register MATLAB as Automation Server

Each time you use the SimMechanics Link utility with a CAD platform or other external application, the utility attempts to connect to MATLAB.

Registration Requirements

Successful connection requires the following to be true:

- Matching MATLAB and SimMechanics Link release numbers (e.g. both release numbers R2012b)
- MATLAB registration as automation server.

Enable Automation Server Mode

You can register MATLAB as an automation server in two ways:

Condition	Registration Procedure
MATLAB session open in desktop mode	At the MATLAB command line, enter `regmatlabserver`. The command registers the current MATLAB session as an automation server.
	At the MATLAB command line, enter: `enableservice ('AutomationServer',true)` The command enables the current MATLAB session as an automation server.
MATLAB session not open	At the operating system command prompt, enter `matlab -automation -desktop` The prompt starts a new MATLAB session in automation server mode.
	At the operating system command prompt, enter command `matlab -regserver`. The command opens a new MATLAB session in automation server mode. You can close the MATLAB session.

A single MATLAB automation server registration can be active at a time. If multiple MATLAB sessions are open in your system, you must *first* disable the active registration and *then* register the desired MATLAB session as an automation server using the `regmatlabserver` command.

Caution If your system does not have an active MATLAB automation server registration, SimMechanics Link issues a error when it attempts to connect. In the event of a connection error, check that a MATLAB automation server is active in your system. If necessary, register MATLAB as an automation server.

Connection from External Application to MATLAB Automation Server

Invoking the SimMechanics Link utility from an external application produces one of the following results:

Condition	Required Action	Result
No MATLAB session open	None	• New MATLAB session opens in automation server mode • SimMechanics Link connects to MATLAB automation server
MATLAB server open in automation server mode	None	• SimMechanics Link connects to MATLAB automation server
MATLAB session open in desktop mode	Register MATLAB session as automation server. See “Enable Automation Server Mode” on page 7-12.	• SimMechanics Link connects to MATLAB automation server

Unregister SimMechanics Link Software

SimMechanics Link contains no uninstaller. If you no longer wish to use the SimMechanics Link utility in your CAD platform, you can unregister the utility. The following table provides information on the unlinking procedure for each CAD platform. Click the link that matches your CAD platform.

To link CAD platform...	...click here
Autodesk Inventor	"Register SimMechanics Link with Inventor"
PTC Creo (Pro/ENGINEER)	"Register SimMechanics Link with Creo"
SolidWorks	"Register SimMechanics Link with SolidWorks"

To register a different version of SimMechanics Link with your CAD platform, first unregister any currently registered version you may have. Then, register the desired version. To register and unregister the utility, follow the links provided in the previous table.

SimMechanics Import XML File

In this section...

The SimMechanics XML import file specifies the hierarchical structure of a CAD assembly and the physical parameters that describe each CAD part. SimMechanics imports the file to automatically generate an equivalent SimMechanics model with little or no additional work on your part.

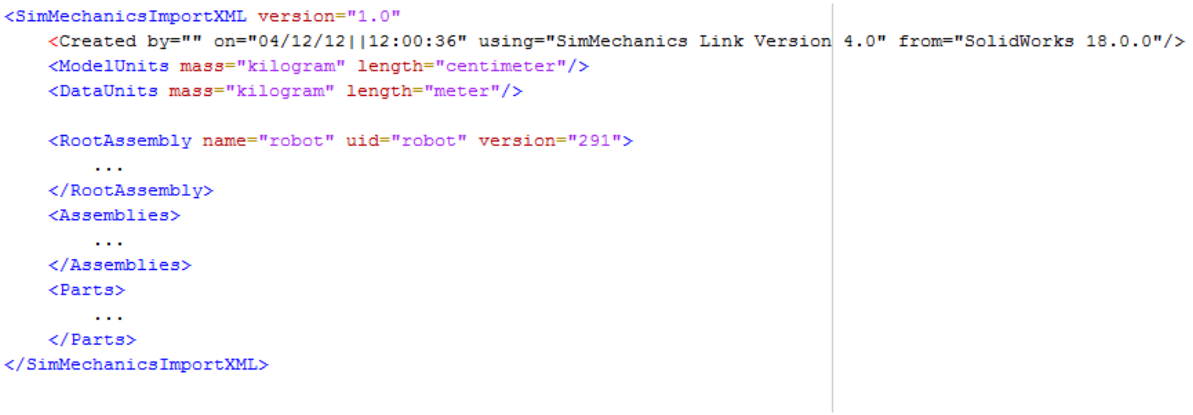
Each block in a model that you import has a unique name and a complete set of parameters. The SimMechanics Import XML file provides the name and parameters of a block based on the original CAD assembly. Once you have imported a model, you can modify the name and parameters of a block to fit your needs. You can also add and remove blocks from the model, or replace one block with another.

Note The following sections describe the structure and parameters of the SimMechanics Import XML file using a robot arm CAD assembly as an example. The actual structure and parameters of your SimMechanics Import XML file can differ from that shown here.

Organization of SimMechanics XML Import File

CAD assemblies are hierarchical systems: a CAD root assembly contains other CAD subassemblies, each made of CAD parts. The SimMechanics XML import file mirrors the hierarchical structure of a CAD assembly. The file organizes CAD assembly information in the order Root Assembly→Assemblies→Parts.

The following figure shows the SimMechanics XML import file for a CAD assembly with name `robot`. Content in sections `RootAssembly`, `Assemblies`, and `Parts` is removed for clarity.



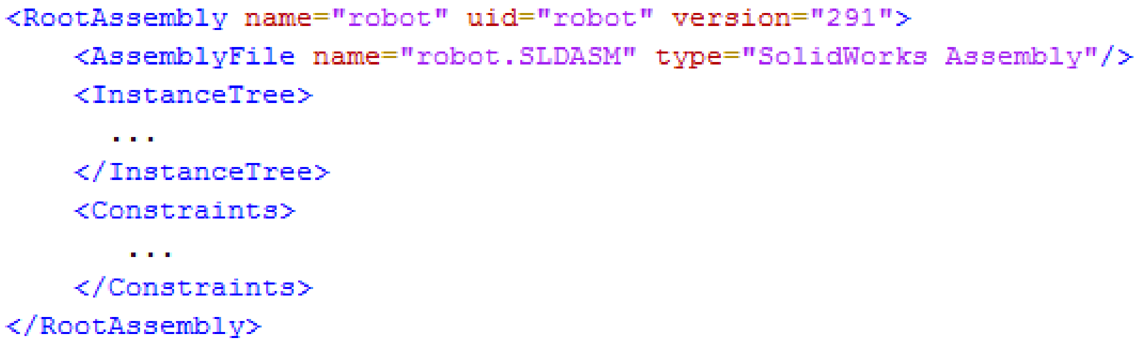
```
<SimMechanicsImportXML version="1.0"
    <Created by="" on="04/12/12||12:00:36" using="SimMechanics Link Version 4.0" from="SolidWorks 18.0.0"/>
    <ModelUnits mass="kilogram" length="centimeter"/>
    <DataUnits mass="kilogram" length="meter"/>

    <RootAssembly name="robot" uid="robot" version="291">
        ...
    </RootAssembly>
    <Assemblies>
        ...
    </Assemblies>
    <Parts>
        ...
    </Parts>
</SimMechanicsImportXML>
```

Root Assembly

The section `RootAssembly` of the SimMechanics XML import file organizes information into two separate subsections:

- `InstanceTree`
- `Constraints`

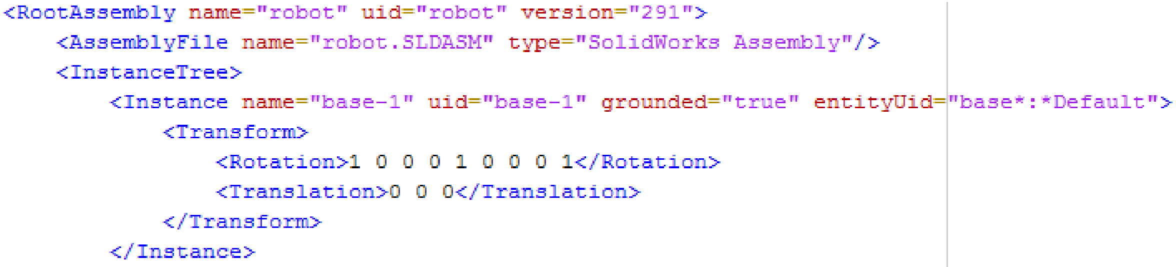


```
<RootAssembly name="robot" uid="robot" version="291">
    <AssemblyFile name="robot.SLDASM" type="SolidWorks Assembly"/>
    <InstanceTree>
      ...
    </InstanceTree>
    <Constraints>
       ...
    </Constraints>
</RootAssembly>
```

`InstanceTree`

Each part contains one body-fixed reference frame that represents a unique set of position and orientation coordinates.`InstanceTree` defines a reference frame for each assembly found in the root assembly. One frame provides an ultimate reference frame with origin coordinates (0,0,0). Rigid transformations translate and rotate the previous frame in `InstanceTree` to obtain the reference frame for another CAD assembly.

`Instance` sections contain the rigid transformation that defines the reference frame for a CAD part. The following figure shows an instance section in the SimMechanics XML import file for a root assembly with name `robot`.



```
<RootAssembly name="robot" uid="robot" version="291">
    <AssemblyFile name="robot.SLDASM" type="SolidWorks Assembly"/>
    <InstanceTree>
        <Instance name="base-1" uid="base-1" grounded="true" entityUid="base*:*Default">
            <Transform>
                <Rotation>1 0 0 0 1 0 0 0 1</Rotation>
                <Translation>0 0 0</Translation>
            </Transform>
        </Instance>
```

The `InstanceTree` section defines the hierarchical organization of the CAD assembly. The section organizes CAD assemblies and parts according to their place in the root assembly hierarchy. The following figure displays a SimMechanics XML import file for a CAD root assembly with name `Robot`. The root assembly contains five assemblies:

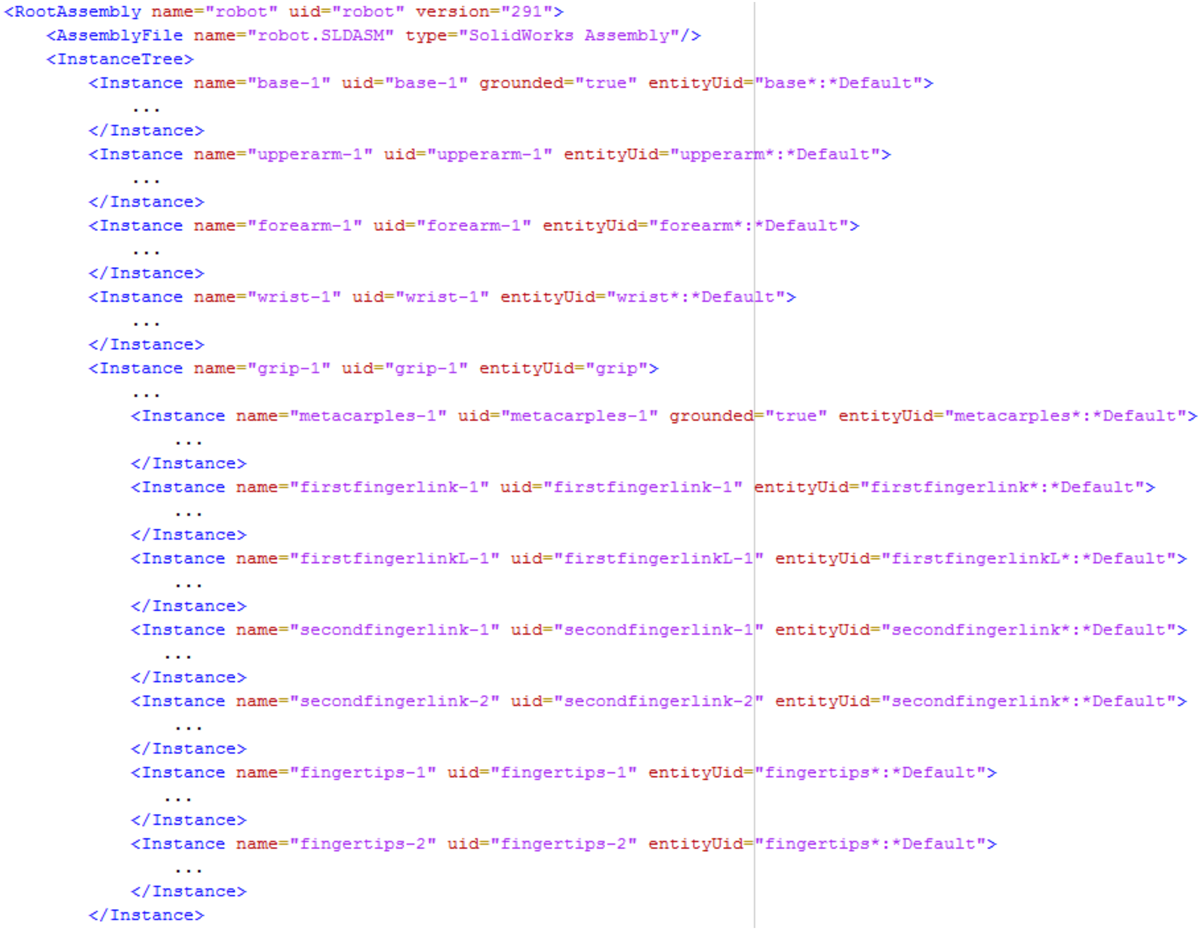
- `base-1`
- `upperarm-1`
- `forearm-1`
- `wrist-1`
- `grip-1`

All assemblies contain a single part, except assembly `Grip`. The assembly `Grip` is a multibody system that connects multiple parts with joints. `Grip` contains seven distinct parts:

- `metacarples-1`
- `firstfingerlink-1`
- `firstfingerlinkL-1`
- `secondfingerlink-1`

- `secondfingerlink-2`
- `fingertips-1`
- `fingertips-2`

`Instance` content is removed for clarity.



```
<RootAssembly name="robot" uid="robot" version="291">
    <AssemblyFile name="robot.SLDASM" type="SolidWorks Assembly"/>
    <InstanceTree>
        <Instance name="base-1" uid="base-1" grounded="true" entityUid="base*:*Default">
            ...
        </Instance>
        <Instance name="upperarm-1" uid="upperarm-1" entityUid="upperarm*:*Default">
            ...
        </Instance>
        <Instance name="forearm-1" uid="forearm-1" entityUid="forearm*:*Default">
            ...
        </Instance>
        <Instance name="wrist-1" uid="wrist-1" entityUid="wrist*:*Default">
            ...
        </Instance>
        <Instance name="grip-1" uid="grip-1" entityUid="grip">
            ...
            <Instance name="metacarples-1" uid="metacarples-1" grounded="true" entityUid="metacarples*:*Default">
                ...
            </Instance>
            <Instance name="firstfingerlink-1" uid="firstfingerlink-1" entityUid="firstfingerlink*:*Default">
                ...
            </Instance>
            <Instance name="firstfingerlinkL-1" uid="firstfingerlinkL-1" entityUid="firstfingerlinkL*:*Default">
                ...
            </Instance>
            <Instance name="secondfingerlink-1" uid="secondfingerlink-1" entityUid="secondfingerlink*:*Default">
               ...
            </Instance>
            <Instance name="secondfingerlink-2" uid="secondfingerlink-2" entityUid="secondfingerlink*:*Default">
                ...
            </Instance>
            <Instance name="fingertips-1" uid="fingertips-1" entityUid="fingertips*:*Default">
               ...
            </Instance>
            <Instance name="fingertips-2" uid="fingertips-2" entityUid="fingertips*:*Default">
                ...
            </Instance>
        </Instance>
```

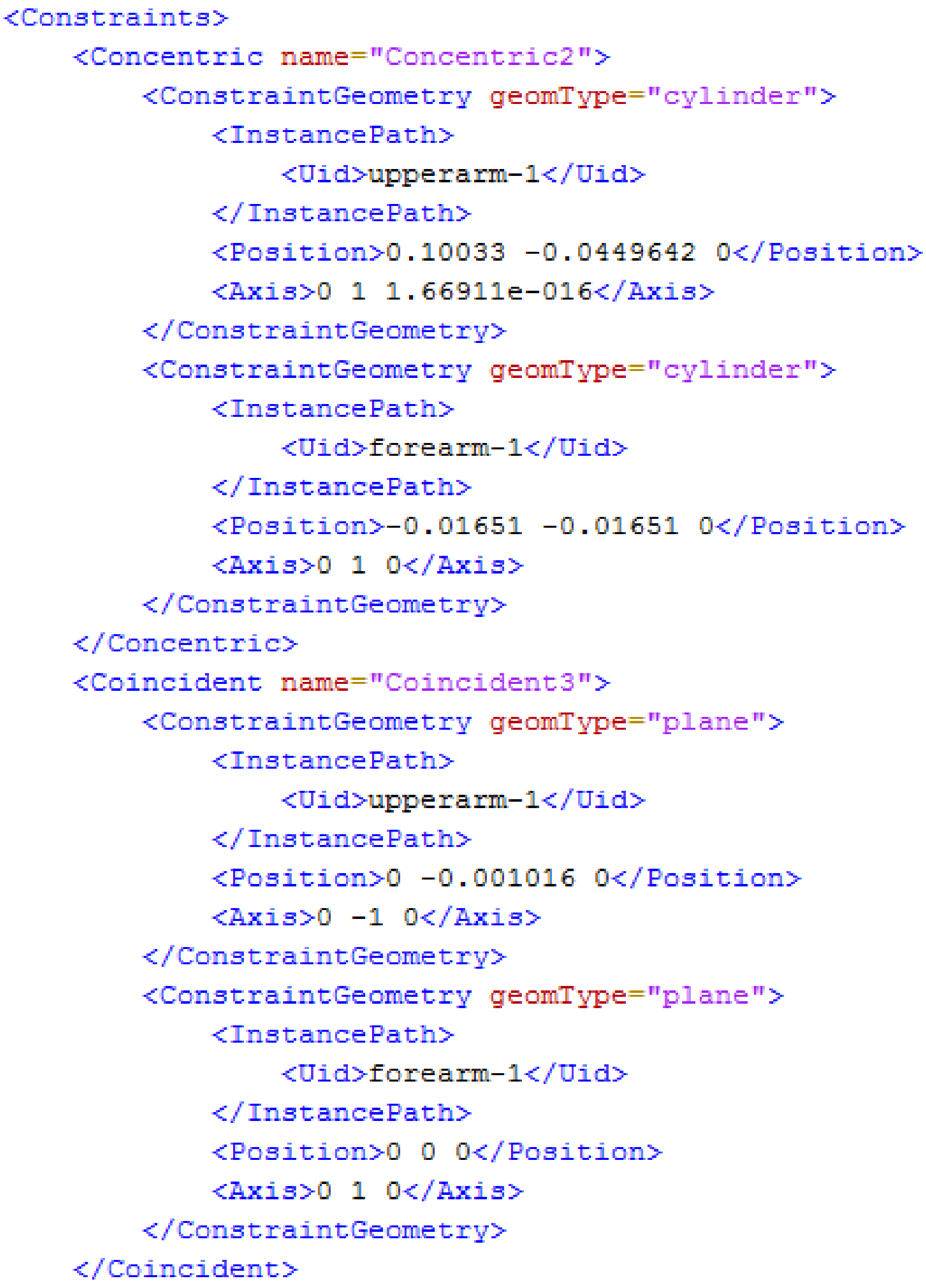
Constraints

CAD constraints define how two CAD parts can move relative to each other. One CAD constraint connects two CAD parts. Each CAD constraint specifies the mechanical degrees of freedom present between two CAD parts. Two CAD parts can translate along, and rotate about, up to three mutually orthogonal axes.

During CAD import, SimMechanics translates the CAD constraints into SimMechanics joints. Most CAD constraints have a SimMechanics equivalent, but the equivalence may not be a one-to-one-correspondence. A single SimMechanics joint may require a combination of multiple CAD constraints providing the same degrees of freedom.

Note Not all CAD constraints have a SimMechanics equivalent. CAD gear constraints are one example. You cannot translate a CAD gear constraint into SimMechanics Second Generation models.

The `Constraints` section specifies the position, orientation, and type of joint that connects each pair of CAD assemblies. Two constraints specify one joint. The following figure shows the constraints section of the SimMechanics XML import file for a joint between two CAD parts with names `upperarm-1` and `forearm-1`. In the figure, two constraints define a revolute joint that connects the two CAD parts: `Concentric` and `Coincident`. Each constraint specifies the position and orientation of the revolute joint relative to the reference frame for each CAD part.



```
<Constraints>
    <Concentric name="Concentric2">
        <ConstraintGeometry geomType="cylinder">
            <InstancePath>
                <Uid>upperarm-1</Uid>
            </InstancePath>
            <Position>0.10033 -0.0449642 0</Position>
            <Axis>0 1 1.66911e-016</Axis>
        </ConstraintGeometry>
        <ConstraintGeometry geomType="cylinder">
            <InstancePath>
                <Uid>forearm-1</Uid>
            </InstancePath>
            <Position>-0.01651 -0.01651 0</Position>
            <Axis>0 1 0</Axis>
        </ConstraintGeometry>
    </Concentric>
    <Coincident name="Coincident3">
        <ConstraintGeometry geomType="plane">
            <InstancePath>
                <Uid>upperarm-1</Uid>
            </InstancePath>
            <Position>0 -0.001016 0</Position>
            <Axis>0 -1 0</Axis>
        </ConstraintGeometry>
        <ConstraintGeometry geomType="plane">
            <InstancePath>
                <Uid>forearm-1</Uid>
            </InstancePath>
            <Position>0 0 0</Position>
            <Axis>0 1 0</Axis>
        </ConstraintGeometry>
    </Coincident>
```

Organization of `Assemblies`

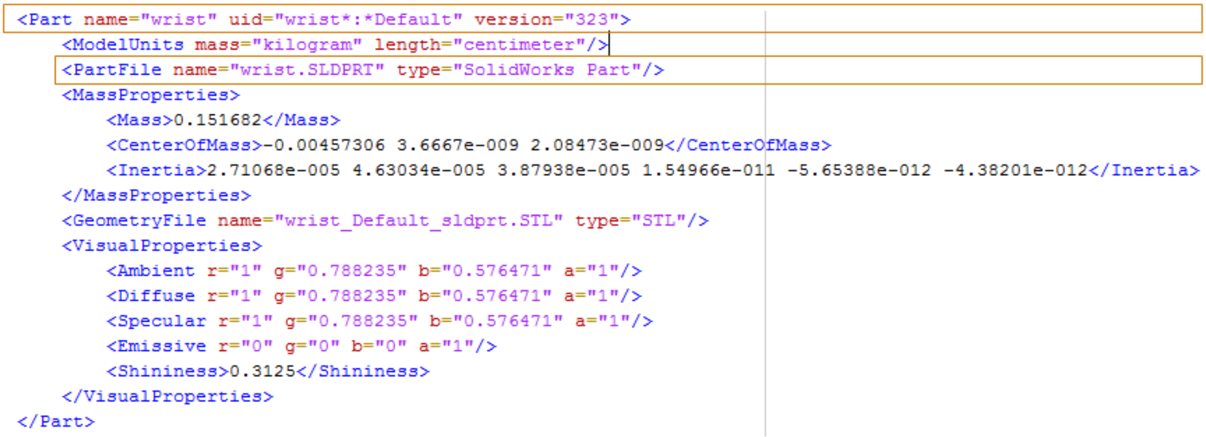
The `Assemblies` section provides the same information present in `RootAssembly`, but with a local non-inertial reference frame acting as the ultimate reference frame. An `InstanceTree` section assigns a local reference frame to each part in an assembly. Each local reference frame appears in a separate `Instance` subsection. In the `Instance` subsection, a rigid transformation rotates and translates a parent frame to obtain the new local reference frame.

A `Constraints` section specifies the kinematic constraints between two parts. The set of constraints between two parts define the kinematic degrees of freedom between them, and are equivalent to SimMechanics joints. During CAD import, SimMechanics interprets each set of CAD constraints, and replaces them with the appropriate set of joints.

Organization of `Parts`

Part Names

Each part receives a unique name. By default, part names originate from the part file names. You can change a part name in SimMechanics, after CAD import, or in the SimMechanics XML import file, before CAD import. The following figure displays the part name section of the SimMechanics XML import file. Colored Boxes highlight part and source file identification information.

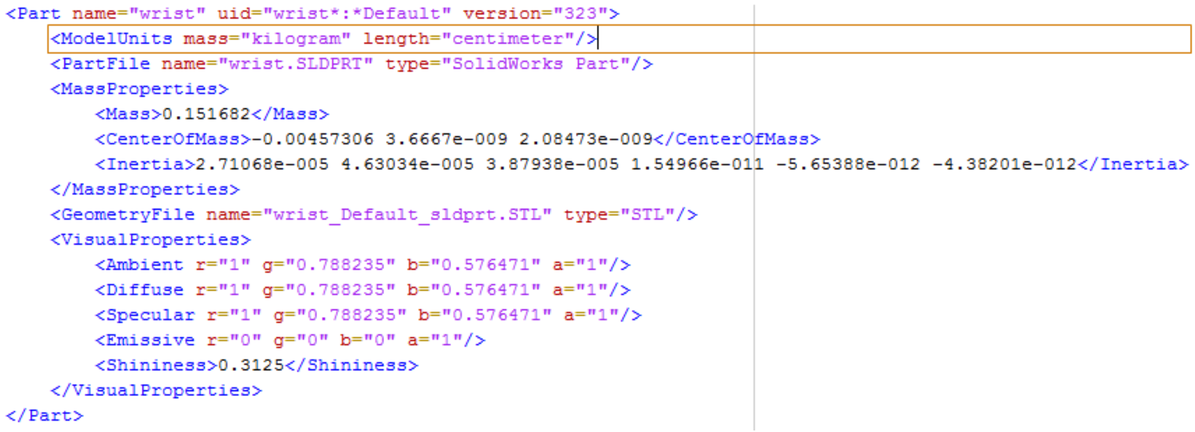


```
<Part name="wrist" uid="wrist*:*Default" version="323">
    <ModelUnits mass="kilogram" length="centimeter"/>
    <PartFile name="wrist.SLDPRT" type="SolidWorks Part"/>
    <MassProperties>
        <Mass>0.151682</Mass>
        <CenterOfMass>-0.00457306 3.6667e-009 2.08473e-009</CenterOfMass>
        <Inertia>2.71068e-005 4.63034e-005 3.87938e-005 1.54966e-011 -5.65388e-012 -4.38201e-012</Inertia>
    </MassProperties>
    <GeometryFile name="wrist_Default_sldprt.STL" type="STL"/>
    <VisualProperties>
        <Ambient r="1" g="0.788235" b="0.576471" a="1"/>
        <Diffuse r="1" g="0.788235" b="0.576471" a="1"/>
        <Specular r="1" g="0.788235" b="0.576471" a="1"/>
        <Emissive r="0" g="0" b="0" a="1"/>
        <Shininess>0.3125</Shininess>
    </VisualProperties>
</Part>
```

Note SimMechanics represents a CAD part as a single rigid body subsystem. A rigid body subsystem inherits its name from the corresponding CAD part.

Physical Units

The SimMechanics XML input file defines the physical units used to resolve the values of inertial parameters. Units originate from the exported CAD assembly file. You can update the physical units in SimMechanics, after CAD import, or in the SimMechanics XML import file, before CAD import. The following figure highlights the physical units used to resolve the inertial properties of CAD part with name `Wrist`.



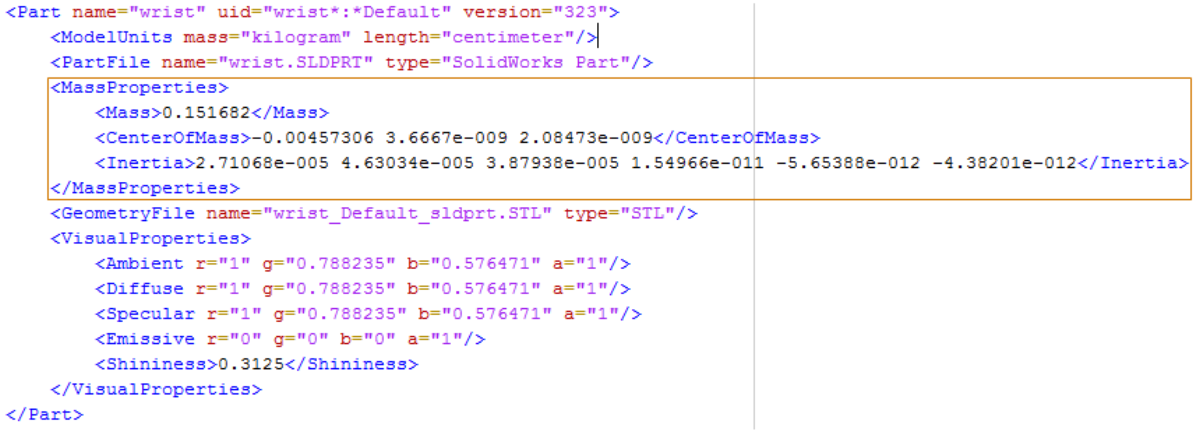
```
<Part name="wrist" uid="wrist*:*Default" version="323">
    <ModelUnits mass="kilogram" length="centimeter"/>
    <PartFile name="wrist.SLDPRT" type="SolidWorks Part"/>
    <MassProperties>
        <Mass>0.151682</Mass>
        <CenterOfMass>-0.00457306 3.6667e-009 2.08473e-009</CenterOfMass>
        <Inertia>2.71068e-005 4.63034e-005 3.87938e-005 1.54966e-011 -5.65388e-012 -4.38201e-012</Inertia>
    </MassProperties>
    <GeometryFile name="wrist_Default_sldprt.STL" type="STL"/>
    <VisualProperties>
        <Ambient r="1" g="0.788235" b="0.576471" a="1"/>
        <Diffuse r="1" g="0.788235" b="0.576471" a="1"/>
        <Specular r="1" g="0.788235" b="0.576471" a="1"/>
        <Emissive r="0" g="0" b="0" a="1"/>
        <Shininess>0.3125</Shininess>
    </VisualProperties>
</Part>
```

Solid Parameters

Solid parameters include inertia and graphic properties. Inertia governs the dynamic response of the solid to an applied force or torque. The SimMechanics XML import file specifies the following inertial parameters:

- Mass
- Center of mass
- Moments and products of inertia

The following figure displays the solid parameters section of the SimMechanics XML import file for a CAD part with name `Wrist`. A box encloses the inertial properties of the CAD part.

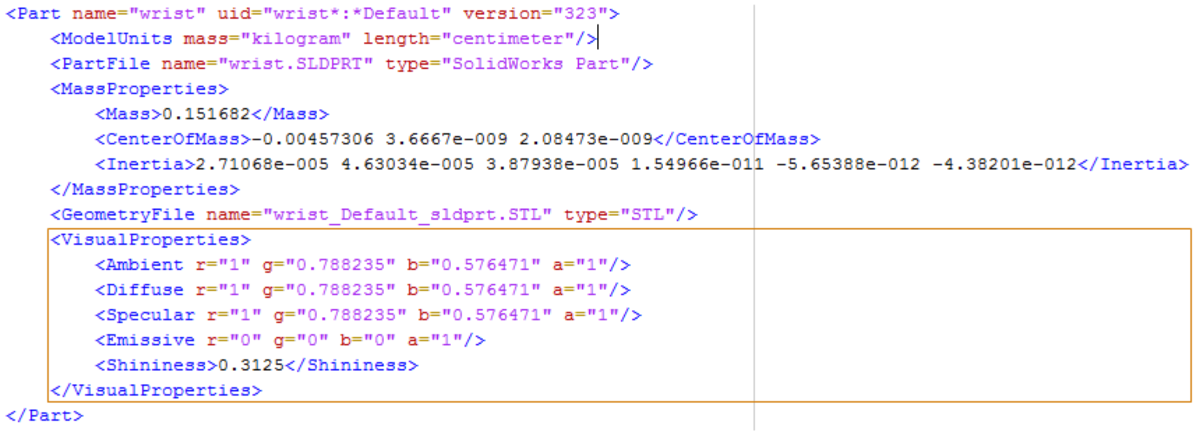


```
<Part name="wrist" uid="wrist*:*Default" version="323">
    <ModelUnits mass="kilogram" length="centimeter"/>
    <PartFile name="wrist.SLDPRT" type="SolidWorks Part"/>
    <MassProperties>
        <Mass>0.151682</Mass>
        <CenterOfMass>-0.00457306 3.6667e-009 2.08473e-009</CenterOfMass>
        <Inertia>2.71068e-005 4.63034e-005 3.87938e-005 1.54966e-011 -5.65388e-012 -4.38201e-012</Inertia>
    </MassProperties>
    <GeometryFile name="wrist_Default_sldprt.STL" type="STL"/>
    <VisualProperties>
        <Ambient r="1" g="0.788235" b="0.576471" a="1"/>
        <Diffuse r="1" g="0.788235" b="0.576471" a="1"/>
        <Specular r="1" g="0.788235" b="0.576471" a="1"/>
        <Emissive r="0" g="0" b="0" a="1"/>
        <Shininess>0.3125</Shininess>
    </VisualProperties>
</Part>
```

Graphic properties govern the visual representation of the solid in Mechanics Explorer. Properties include color and shininess. The following table describes the graphic properties present in the SimMechanics XML import file.

Graphic Property	Type	Description
Ambient Color	RGBA vector	Color of light that hits the solid surface
Diffuse Color	RGBA vector	Color of the solid surface in pure white light
Specular Color	RGBA vector	Color of specular reflection from the solid surface
Emissive Color	RGBA vector	Color of solid self-illumination
Shininess	Scalar	Intensity of specular highlights from the solid surface

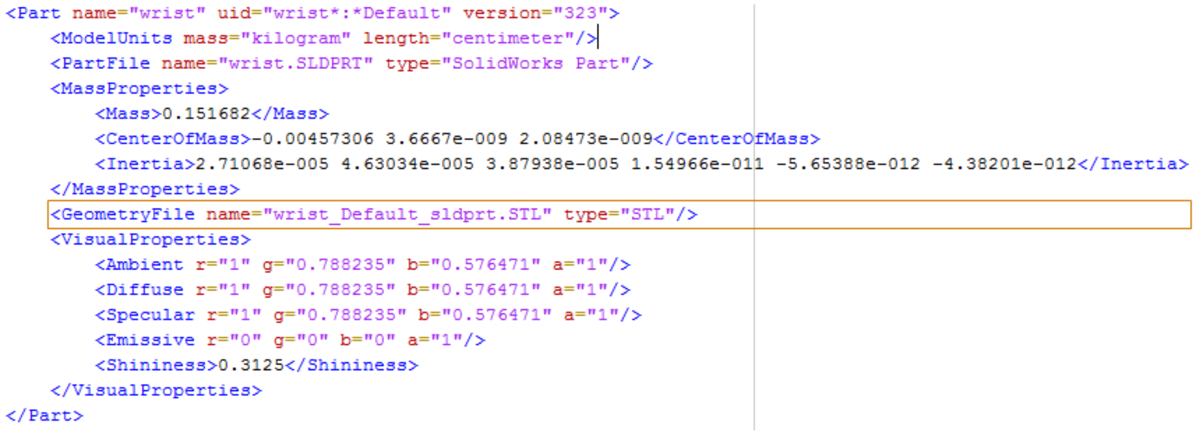
The following figure highlights the graphic properties section of the SimMechanics XML import file.



```
<Part name="wrist" uid="wrist*:*Default" version="323">
    <ModelUnits mass="kilogram" length="centimeter"/>
    <PartFile name="wrist.SLDPRT" type="SolidWorks Part"/>
    <MassProperties>
        <Mass>0.151682</Mass>
        <CenterOfMass>-0.00457306 3.6667e-009 2.08473e-009</CenterOfMass>
        <Inertia>2.71068e-005 4.63034e-005 3.87938e-005 1.54966e-011 -5.65388e-012 -4.38201e-012</Inertia>
    </MassProperties>
    <GeometryFile name="wrist_Default_sldprt.STL" type="STL"/>
    <VisualProperties>
        <Ambient r="1" g="0.788235" b="0.576471" a="1"/>
        <Diffuse r="1" g="0.788235" b="0.576471" a="1"/>
        <Specular r="1" g="0.788235" b="0.576471" a="1"/>
        <Emissive r="0" g="0" b="0" a="1"/>
        <Shininess>0.3125</Shininess>
    </VisualProperties>
</Part>
```

Geometry File References

A set of STL files specifies the 3-D geometry of the solid surface for each CAD part. STL files specify only geometry, without reference to other graphic properties, like color. The SimMechanics XML import file specifies a single STL geometry file for each part in the CAD assembly. The following figure highlights the geometry file reference in the SimMechanics XML input file for a part with name `Wrist`.



```
<Part name="wrist" uid="wrist*:*Default" version="323">
    <ModelUnits mass="kilogram" length="centimeter"/>
    <PartFile name="wrist.SLDPRT" type="SolidWorks Part"/>
    <MassProperties>
        <Mass>0.151682</Mass>
        <CenterOfMass>-0.00457306 3.6667e-009 2.08473e-009</CenterOfMass>
        <Inertia>2.71068e-005 4.63034e-005 3.87938e-005 1.54966e-011 -5.65388e-012 -4.38201e-012</Inertia>
    </MassProperties>
    <GeometryFile name="wrist_Default_sldprt.STL" type="STL"/>
    <VisualProperties>
        <Ambient r="1" g="0.788235" b="0.576471" a="1"/>
        <Diffuse r="1" g="0.788235" b="0.576471" a="1"/>
        <Specular r="1" g="0.788235" b="0.576471" a="1"/>
        <Emissive r="0" g="0" b="0" a="1"/>
        <Shininess>0.3125</Shininess>
    </VisualProperties>
</Part>
```

See Also

`smimport`

Related Examples

- “Install and Register SimMechanics Link Software” on page 7-9
- “Import Robot Arm Model” on page 7-27
- “Import Stewart Platform Model” on page 7-33
- “Find and Fix CAD Import Issues” on page 7-40

Concepts

- “CAD Translation” on page 7-2
- “CAD Import” on page 7-5

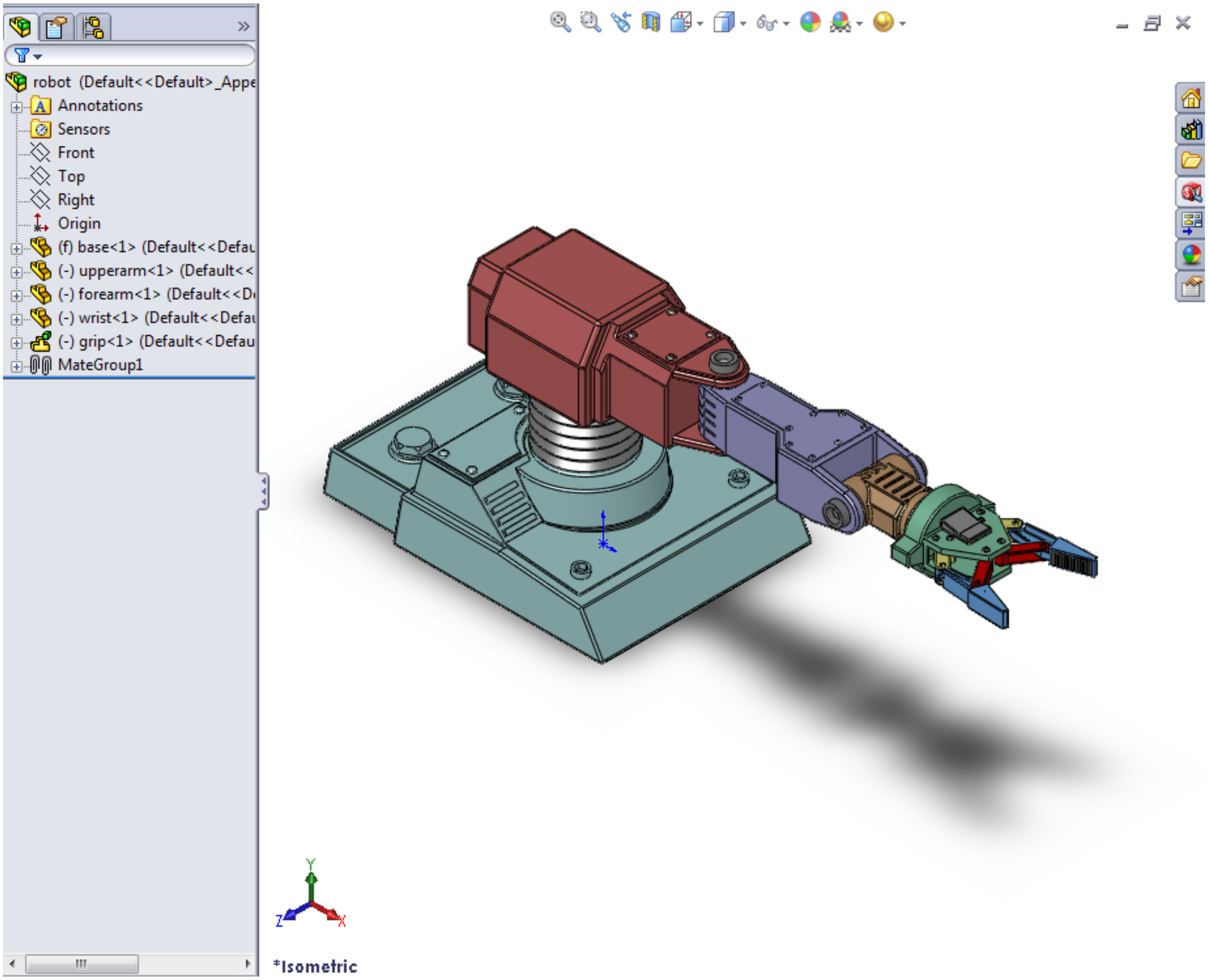
Import Robot Arm Model

In this section...

In this example, you import a CAD assembly with name robot into SimMechanics. SimMechanics provides the `smimport` command so that you can import a CAD assembly. The command is the only SimMechanics tool you need to import a CAD assembly. The CAD import procedure is the same for all CAD platforms.

Note This example uses an XML file and a set of STL files that are present in your SimMechanics installation. You can export the XML and STL files directly from a supported CAD platform, but the names of the files may differ from the example.

The following figure shows the original CAD assembly inside the SolidWorks CAD platform.


robot (Default<<Default>_Appe
Annotations
Sensors
Front
Top
Right
Origin
(f) base<1> (Default<<Defau
(-) upperarm<1> (Default<<
(-) forearm<1> (Default<<D
(-) wrist<1> (Default<<Defau
(-) grip<1> (Default<<Defau
MateGroup1
Y
Z
X
*Isometric

Check Import Files

Before you import the sm_robot CAD assembly, check that the import files exist. The import files include one SimMechanics Import XML file and a set of STL files that specify the geometry of all CAD parts.

1 At the MATLAB command line, enter the following command to change the current working directory to the subdirectory that contains the robot example files:

```
cd(fullfile(matlabroot,'toolbox','physmod','sm','smdemos',
'import','robot'))
```

2 At the MATLAB command line, enter `ls` or `dir` to list all files in the `\robot` directory.

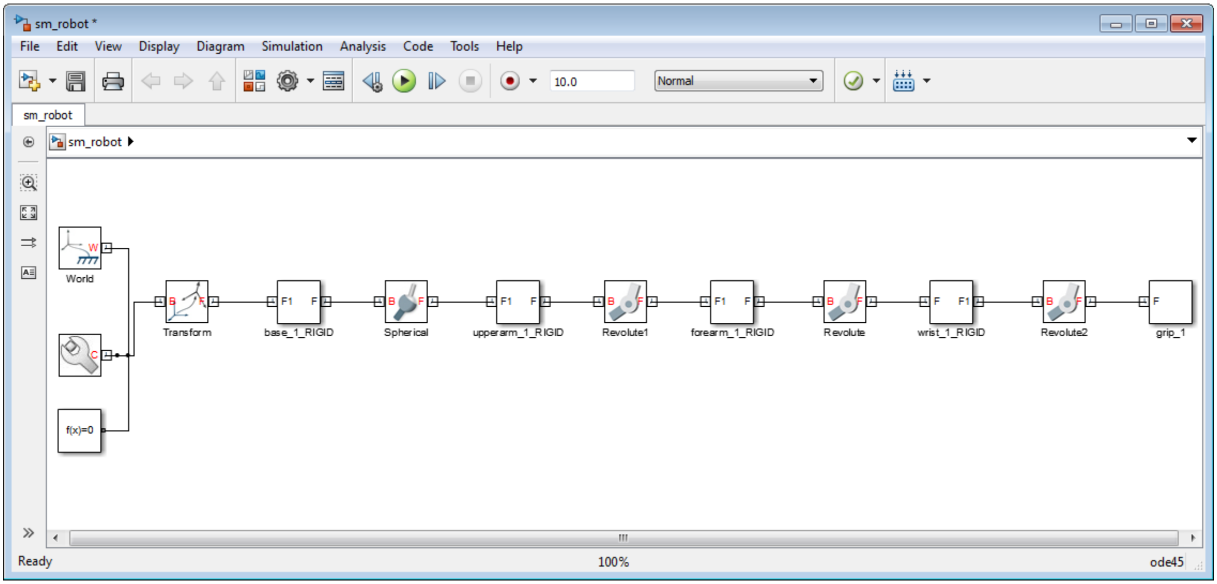
3 Check that the directory contains XML file sm_robot.xml and a set of STL files.

Import Robot Assembly

Once you have verified that all required files exist, proceed to import the assembly.

1 At the MATLAB command line, enter `smimport('sm_robot.xml')`.

2 Confirm that SimMechanics opens a new model with name `sm_robot`.



Note SimMechanics automatically generates the new model without extra input on your part. Review the model and check for errors and inconsistencies in the block diagram.

3 In the Simulink Editor window that contains the model, select **File > Save As**.

4 In the **Save As** dialog box, enter the desired file name and select a convenient directory in which store the model file.

Visualize and Simulate Robot Assembly

1 In the Simulink Editor window that contains the robot model, select **Simulation > Update Diagram** or press **Ctrl+D**.

Note When you update the diagram, SimMechanics automatically updates the model display in Mechanics Explorer. SimMechanics relies on the set of STL files to represent the 3-D geometry of each CAD part. If the files are not available, SimMechanics still generates the model, but Mechanics Explorer cannot display the assembly.

2 In the Mechanics Explorer toolbar, set **View Convention** to `Y up (XY Front)`.

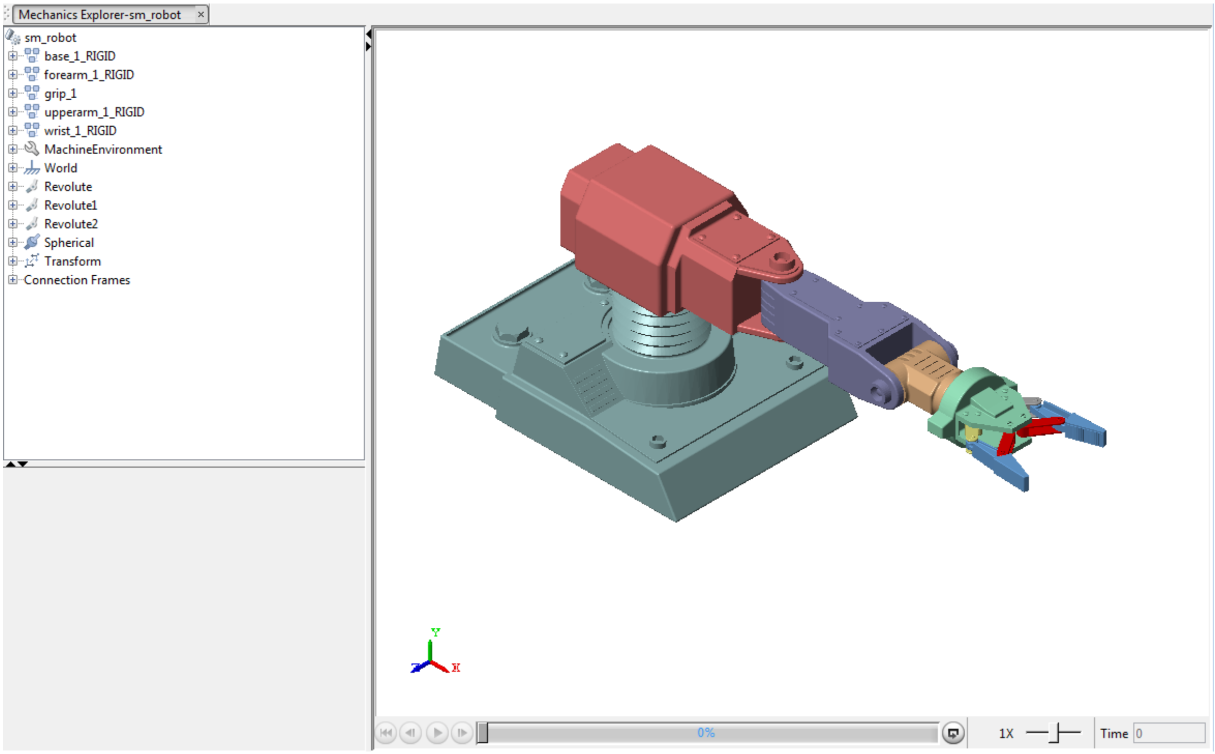
Note Most CAD systems use a `Y up` default view convention. The convention differs from the Mechanics Explorer default setting, `Z up`. Selecting the `Y up` view convention causes Mechanics Explorer to display the assembly with the same orientation used in the CAD platforms.

3 In the toolbar, click the icon for the desired viewpoint.

Note Selecting the `Y up` view convention does not affect the Mechanics Explorer display until you click a view point. You have the choice between seven standard viewpoints: front, back, top, down, left, right, and isometric. Once you select a view point, you can rotate, pan, and zoom to adjust the display of your model. For more information, see:

- "Configure Visualization Settings" on page 6-6
- "Rotate, Pan, and Zoom View" on page 6-18

4 Confirm that a Mechanics Explorer window opens with a static display of the robot assembly.



5. In the Simulink Editor window for the model, select **Simulation > Run** or press **Ctrl+T** to simulate the model.

Tip The model lacks actuation inputs. When you simulate the model, the robot arm moves strictly due to gravity effects. You can change the gravity specification in the Mechanism Configuration block.

You can add actuation inputs to the model. Add a block from the Forces & Torques library to actuate a rigid body. Select an actuation mode in the model joint blocks to actuate a joint.

See Also

`smimport`

Related Examples

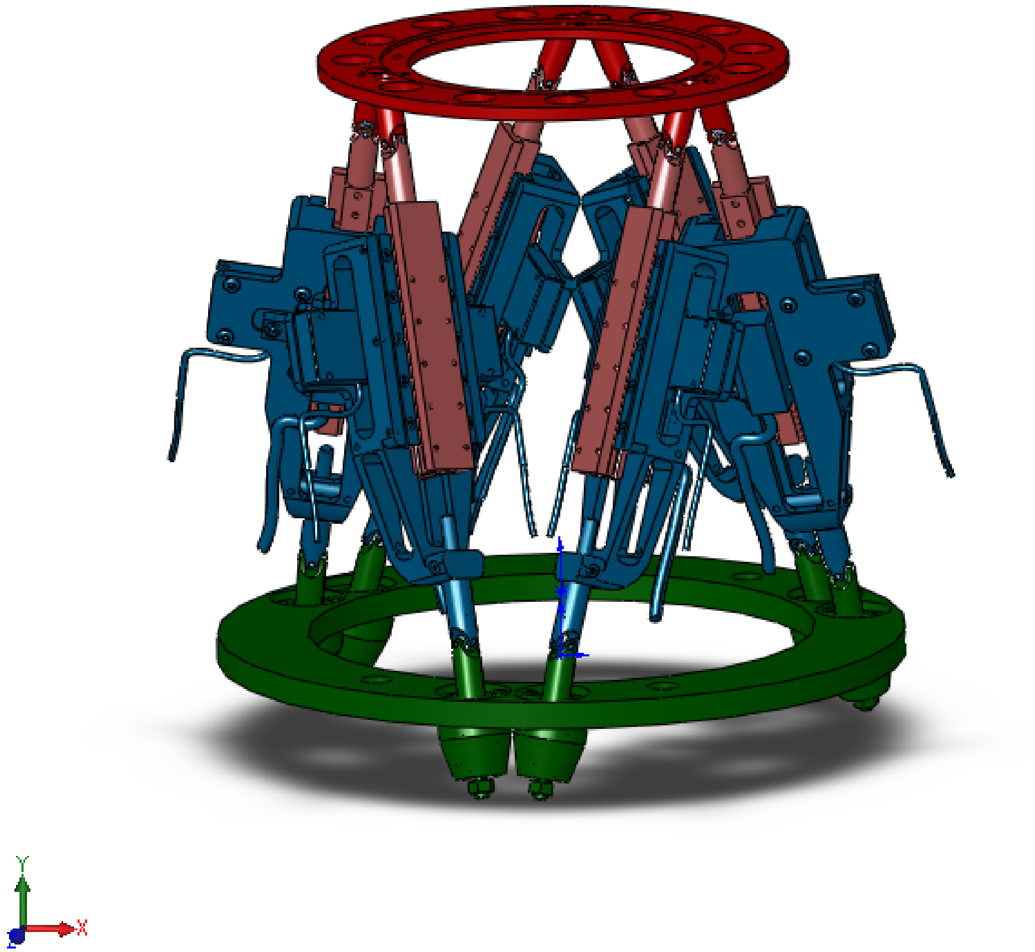
- "Install and Register SimMechanics Link Software" on page 7-9
- "Import Stewart Platform Model" on page 7-33
- "Find and Fix CAD Import Issues" on page 7-40

Concepts

- "CAD Translation" on page 7-2
- "CAD Import" on page 7-5

Import Stewart Platform Model

You can import a CAD assembly into a SimMechanics model. To do this, you use the SimMechanics command `smimport`. In this example, you import the CAD assembly for a Stewart platform. All required files are provided with your SimMechanics installation.



Check Import Files

To import the CAD assembly, you must have access to the SimMechanics Import XML and STL files for this assembly. Check that you have these files before proceeding.

1 Navigate to directory

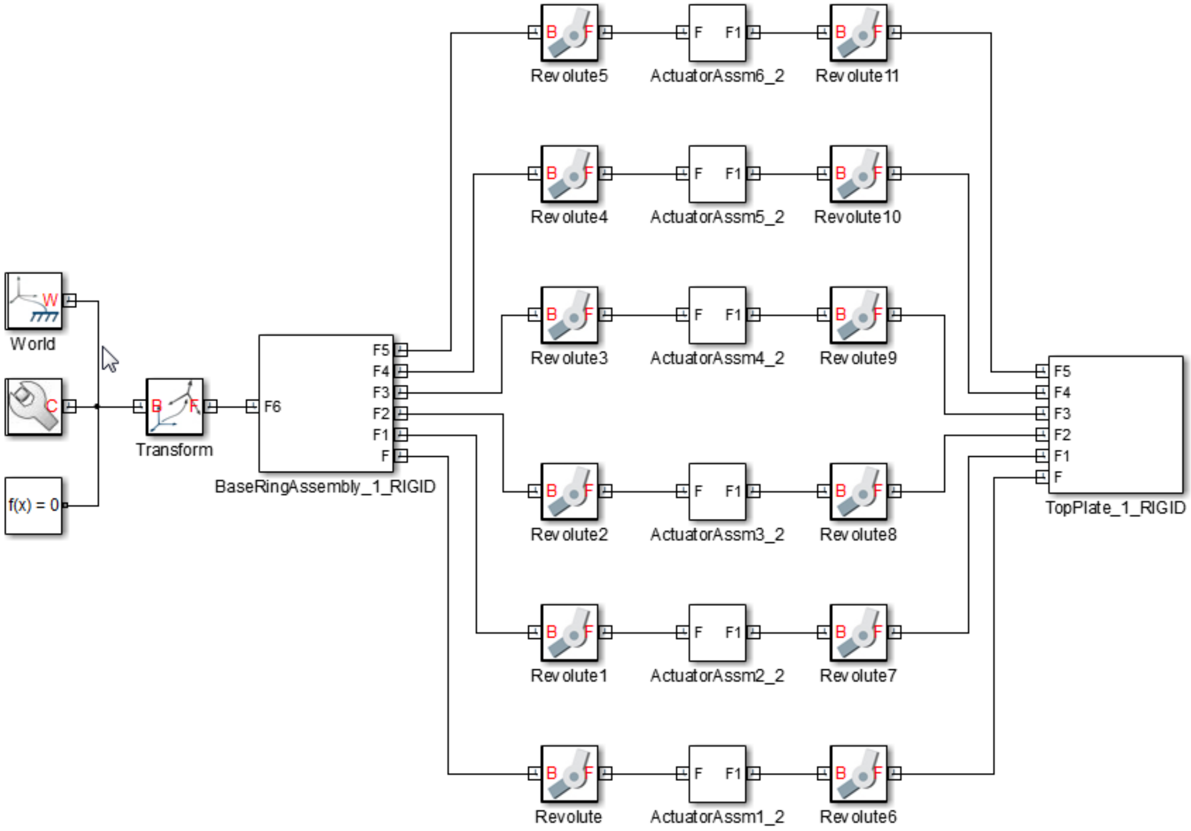
```
<matlabroot>/toolbox/physmod/sm/smdemos/...
...import/stewart_platform
```

2 Check that the following files exist.

File	Quantity	Description
SimMechanics Import XML	One	Provides model structure and parameters
STL	Multiple	Provides part geometry

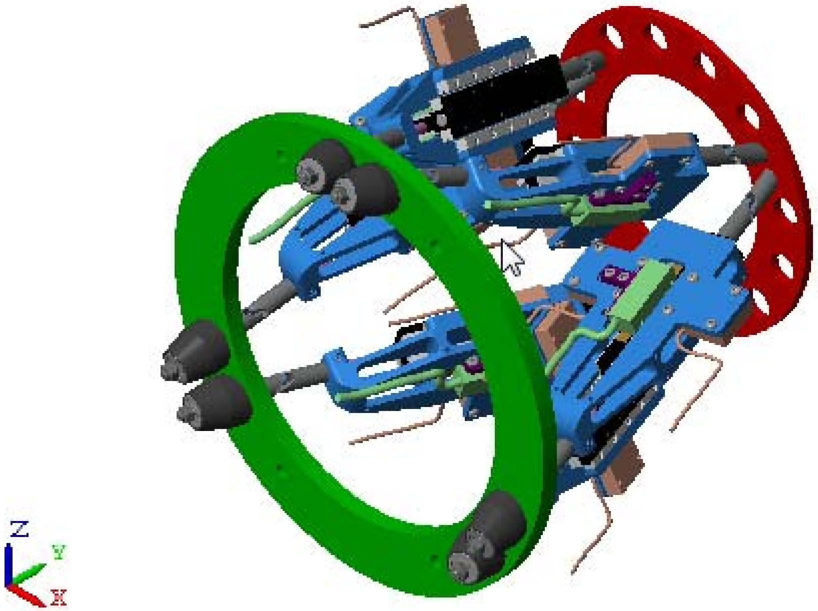
Import Model

If you have access to the import files, you can import the model. To do this, at the MATLAB command line enter `smimport('stewart_platform.xml')`. SimMechanics automatically generates a Stewart platform model. This model replicates the original CAD assembly.



Visualize and Simulate Robot Assembly

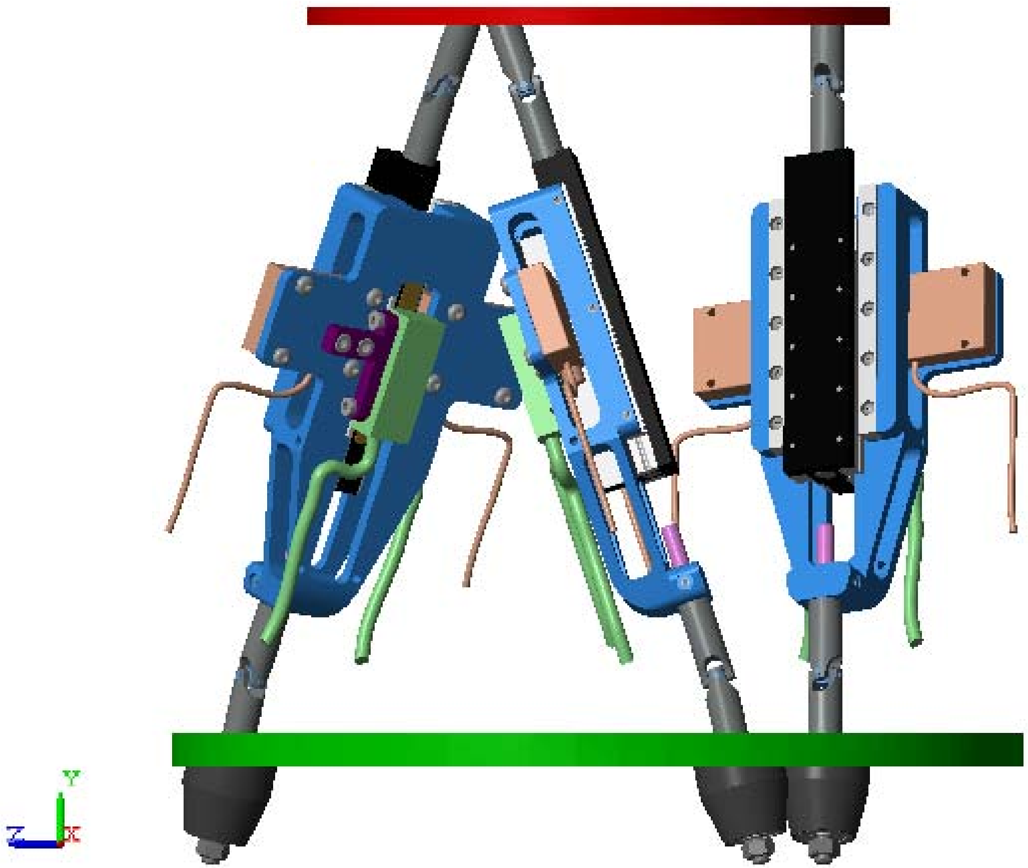
You can now simulate the model that you imported. On the Simulink tool bar, click the **Run** button. Alternatively, press **Ctrl+T**. Mechanics Explorer opens with a dynamic display of your model.



By default, Mechanics Explorer uses a Z axis up view convention. This convention differs from that which most CAD platforms use—Y axis up. The different view conventions cause the Stewart platform to appear sideways in the visualization pane. To fix this issue, change the Mechanics Explorer view convention to Y axis up:

- On the Mechanics Explorer tool bar, in the **View Convention** drop-down list, select **Y Up (XY Front)**.

To refresh the visualization pane using the new view convention, on the Mechanics Explorer tool bar, click any standard view button, e.g., Isometric View.



Tip Actuate the stewart_platform model with blocks from the **Forces and Torques** library. Then, simulate the model and analyze its dynamic behavior in Mechanics Explorer.

See Also

`smimport`

Related Examples

- "Install and Register SimMechanics Link Software" on page 7-9
- "Import Robot Arm Model" on page 7-27
- "Find and Fix CAD Import Issues" on page 7-40

Concepts

- "CAD Translation" on page 7-2
- "CAD Import" on page 7-5

Find and Fix CAD Import Issues

In this section...

Under certain conditions, a model that you import can behave in unexpected ways. Some issues that you can encounter while importing a model include:

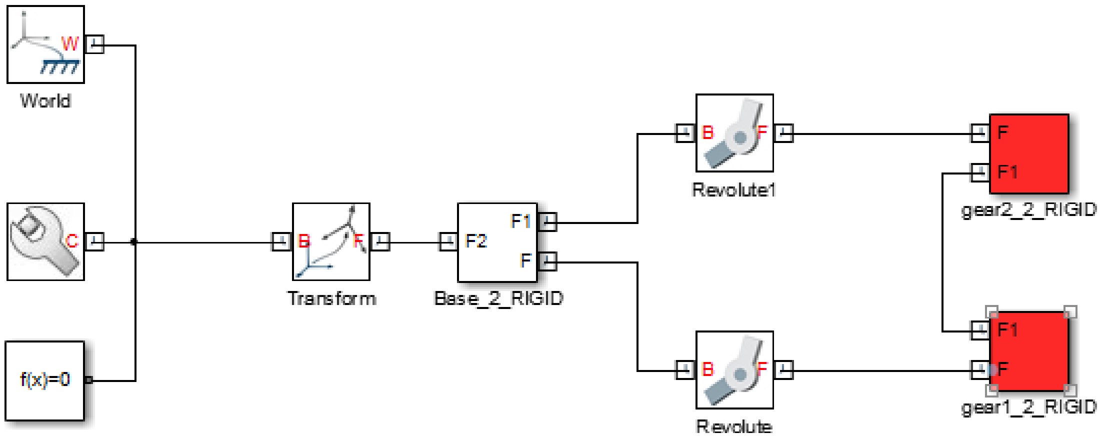
- Model replaces certain CAD constraints with rigid connections
- Model appears with different orientation in Mechanics Explorer
- Part appears invisible in Mechanics Explorer

In this section, learn what causes these issues and, if possible, what approaches you can take to correct them.

Model replaces certain CAD constraints with rigid connections

SimMechanics supports most, but not all, CAD constraints. If you import a CAD assembly with a CAD constraint that SimMechanics does not support, SimMechanics issues a warning message and automatically replaces that constraint with a rigid connection.

The figure shows the imported model of a CAD assembly that contains an unsupported gear constraint. Because SimMechanics does not support that particular gear constraint, it replaces it with a frame line. The frame line represents a rigid connection.

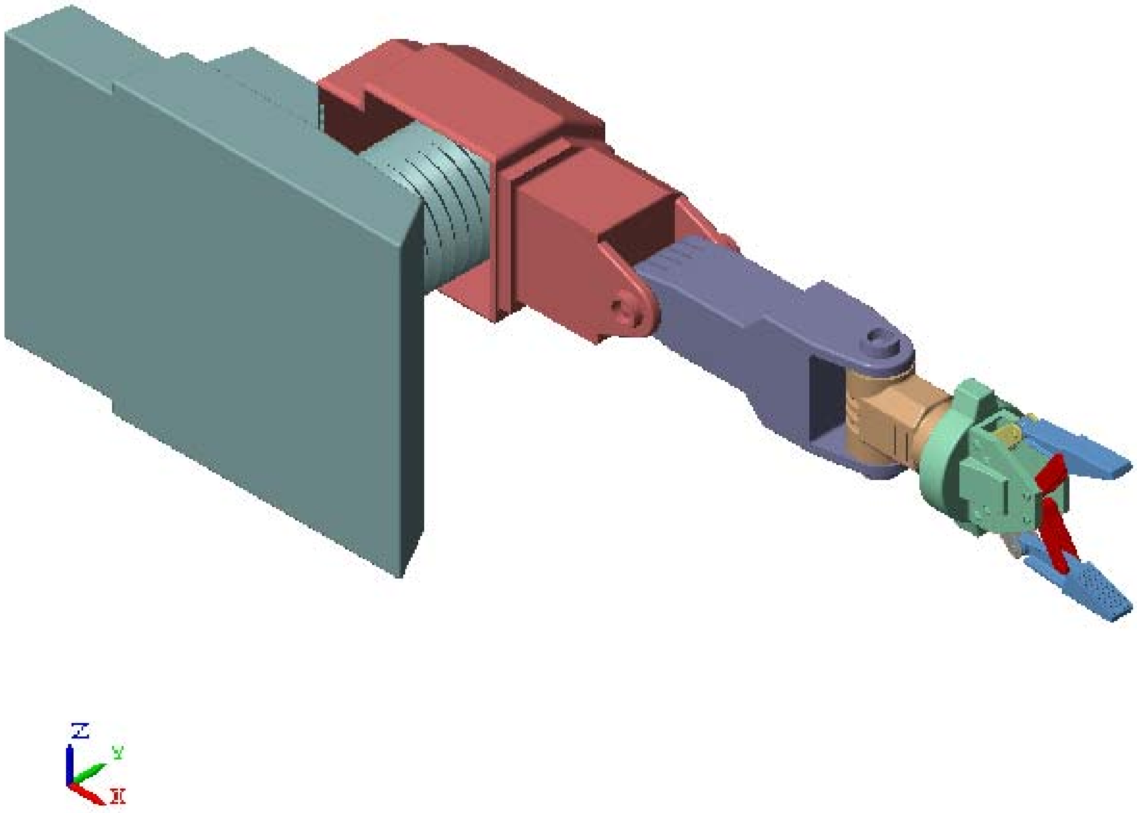


Identify and Change Automatic Rigid Connections

The warning message identifies the blocks and ports that connect to the unsupported constraint. Use this information to identify the new rigid connection in the model. Then, determine if any combination of SimMechanics joint, gear, or constraint blocks adequately replaces the unsupported constraint. If so, replace that rigid connection. Run the simulation to check that the model behaves as you expect.

Model appears with different orientation in Mechanics Explorer

By default, Mechanics Explorer displays a model with the Z axis of the World frame pointing up. Using this convention, the default gravity vector `[0 0 -9.81] m/s^2` points down, a direction that is practical for most applications. However, this convention differs from that which CAD platforms commonly use, Y axis up, causing Mechanics Explorer to display some models sideways. If this happens, you can manually change the view convention to that used in the original CAD assembly. The figure shows the default Mechanics Explorer display of an imported robot arm model.



Change View Convention

To change the view convention of a model:

1. In the Mechanics Explorer toolbar, click the **View Convention** drop-down menu.

2. Select **Y up (ZX Top)**.

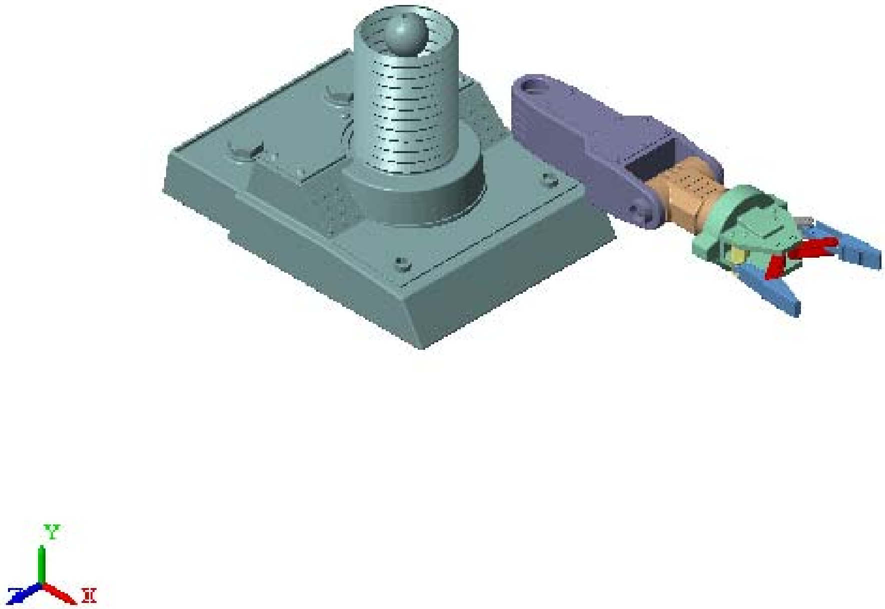
3 Refresh the Mechanics Explorer display by selecting a view point from the Mechanics Explorer tool bar.

Mechanics Explorer displays the model using the new view convention.

Part appears invisible in Mechanics Explorer

During CAD import, SimMechanics uses a set of stereolithographic (STL) files to generate the 3-D surface geometry of each CAD part. If SimMechanics cannot load the STL file for a part, that part appears invisible in Mechanics Explorer. This issue does not affect model update or simulation.

The figure shows the Mechanics Explorer display of an imported model containing an invalid STL file.



Correct Visualization Issue

If a part of an imported model appears invisible in Mechanics Explorer:

1. In Mechanics Explorer, identify the name of each invisible part.
2. In the block diagram, open the dialog boxes of the associated Solid blocks.
3. In the **Geometry** section, check that the name and location of the STL files are correct.

   If either is incorrect, enter the correct information and update the model. Check that Mechanics Explorer displays the invisible part. If not, check if the STL files are valid.

STL File Issues

To visualize a CAD assembly that you import, SimMechanics relies on a set of STL files that specify the 3-D surface geometry of the CAD parts. Each STL file specifies the surface geometry of one CAD part as a set of 2-D triangles. To do this, the STL files contain:

- [X Y Z] coordinates of the triangle vertices
- [X Y Z] components of the normal vectors for the triangles.

If an STL file specifies a normal vector with zero length, SimMechanics issues a warning. The STL file fails to load.

See Also

`smimport`

Related Examples

- "Install and Register SimMechanics Link Software" on page 7-9
- "Import Robot Arm Model" on page 7-27
- "Import Stewart Platform Model" on page 7-33

Concepts

- "CAD Translation" on page 7-2
- "CAD Import" on page 7-5

Deployment

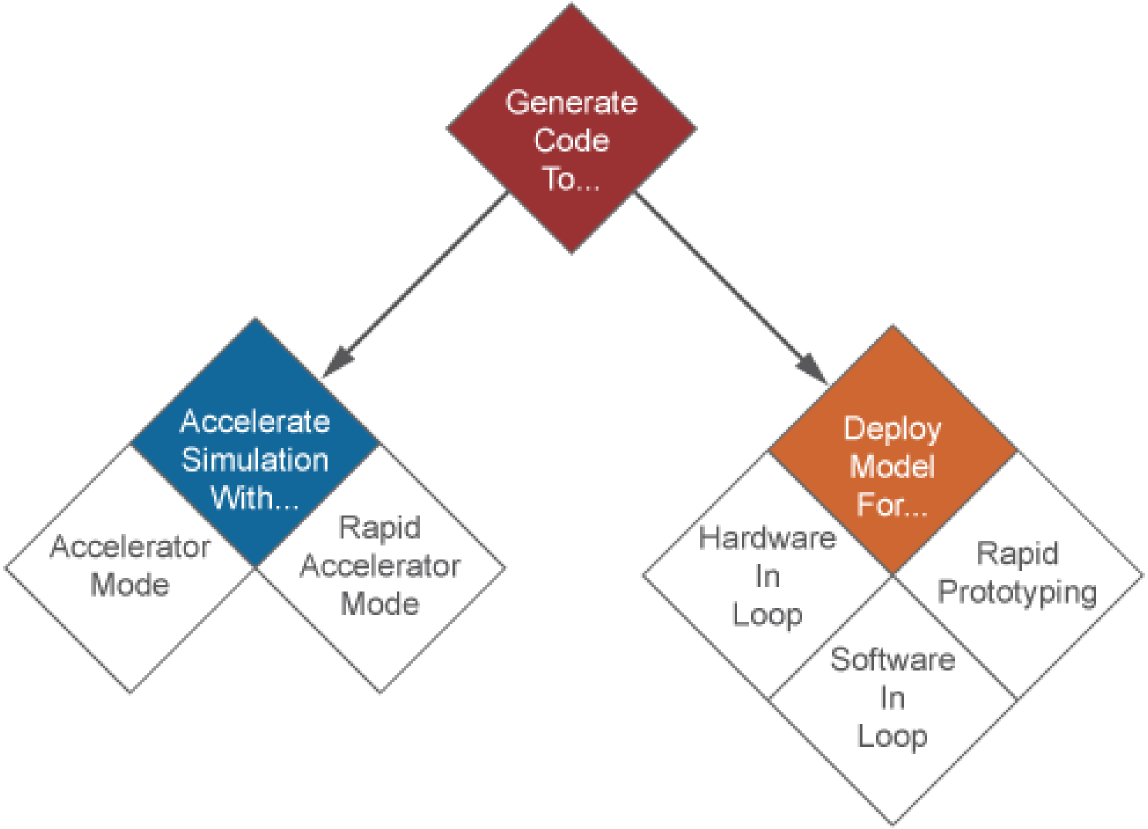
8

Code Generation

About Code Generation

In this section...

SimMechanics supports code generation with Simulink Coder™. You can generate C/C++ code from a SimMechanics model to accelerate simulation or to deploy a model.



Simulation Accelerator Modes

Simulink can generate C/C++ executable code to shorten simulation time. Two simulation modes generate code:

- Accelerator
- Rapid Accelerator

SimMechanics 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 SimMechanics 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 SimMechanics 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 SimMechanics supports, but does not perform, code generation for model deployment. Code generation for model deployment requires the Simulink Coder product.

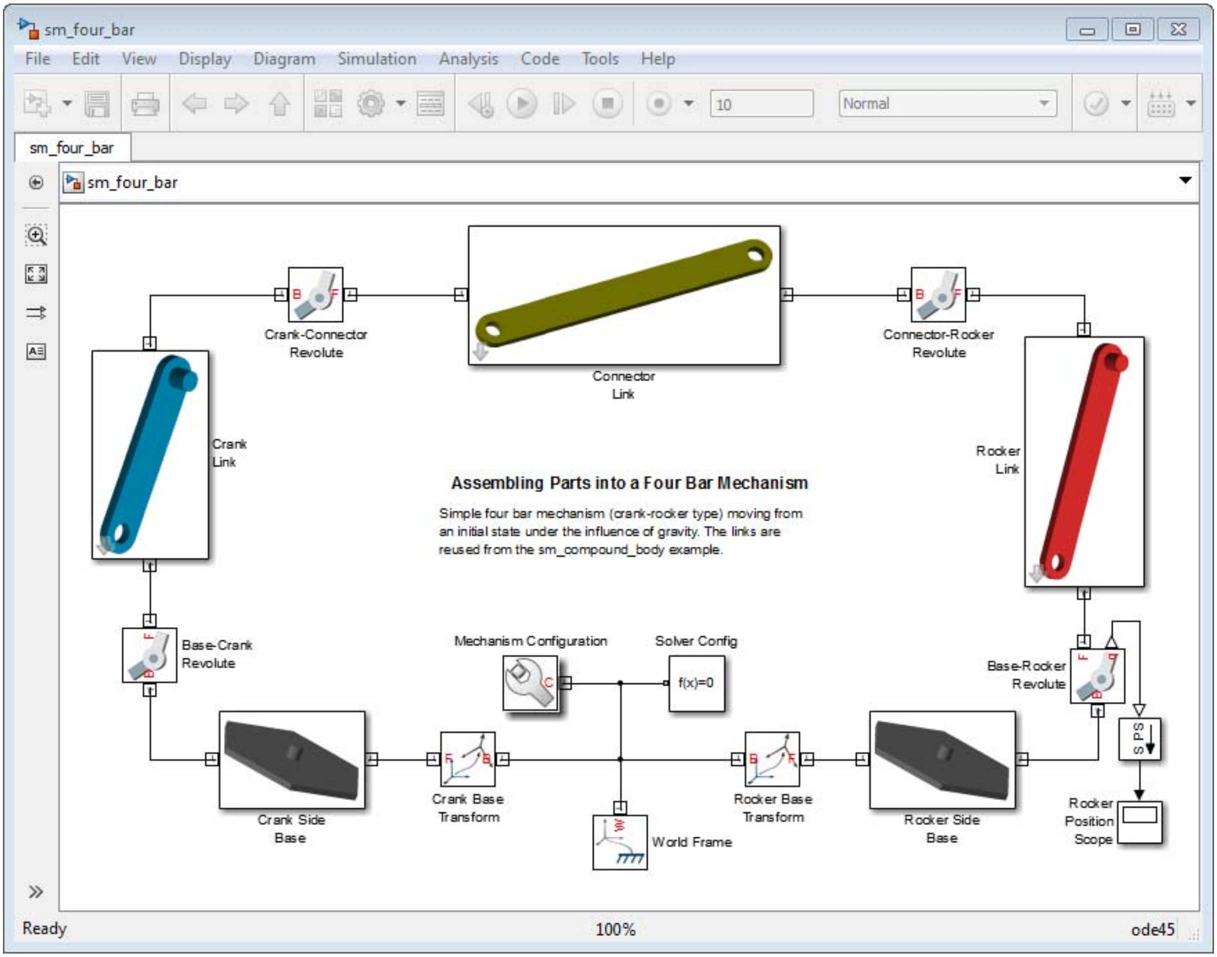
Related Examples

- "Configure Four-Bar Model for Code Generation" on page 8-5
- "Configure Model for Rapid Acceleration Mode" on page 8-8
- "Find and Fix Code Generation Issues" on page 8-12

Configure Four-Bar Model for Code Generation

You can generate code from a SimMechanics model for deployment outside the MATLAB environment. This example shows how to configure a four-bar model for code generation using a variable-step solver with the objective of execution efficiency. The example uses the default Simulink solver `ode45 (Dormand-Prince)`.

The four-bar model is present in your SimMechanics 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.



Assembling Parts into a Four Bar Mechanism

Simple four bar mechanism (crank-rocker type) moving from an initial state under the influence of gravity. The links are reused from the sm_compound_body example.

Configure Model

To configure the model for code generation:

1 In the Simulink Editor window for your model, select **Simulation > Model Configuration Parameters**.

2 In the **Model Configuration Parameters** dialog box, select **Code Generation**.

3 In **Target Selection**, enter `rsim.tlc`.

Note You must use the `rsim.tlc` target each time you use a variable-step solver. You can change the solver type in the **Solver** section of the **Model Configuration Parameters** window.

---

4 In **Code Generation Advisor**, select `Execution Efficiency`.

5 Click **Apply**.

6 To generate C code for your model, click **Build**.

Related Examples

- "Configure Model for Rapid Acceleration Mode" on page 8-8
- "Find and Fix Code Generation Issues" on page 8-12

Concepts

- "About Code Generation" on page 8-2

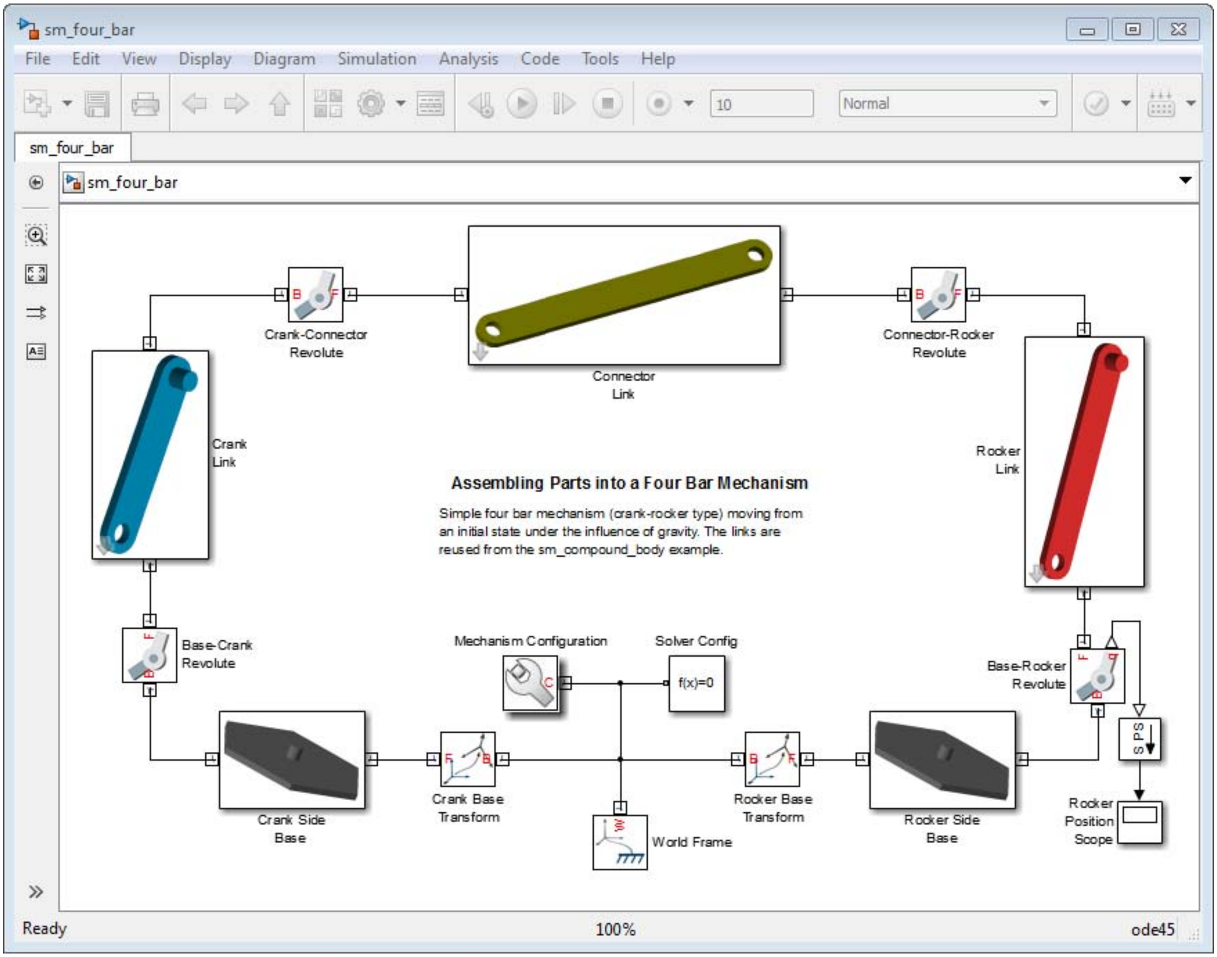
Configure Model for Rapid Acceleration Mode

In this section...

Model Overview

You can run a SimMechanics model in Accelerator and Rapid Accelerator modes. When you select an accelerator mode, SimMechanics 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 SimMechanics 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.



Assembling Parts into a Four Bar Mechanism

Simple four bar mechanism (crank-rocker type) moving from an initial state under the influence of gravity. The links are reused from the sm_compound_body example.

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 **SimMechanics 2G** 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 SimMechanics 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 SimMechanics models with long simulation times.

The Rapid Accelerator mode does not support visualization. Mechanics Explorer does not open, and you can not view a dynamic simulation of the model. All other simulation capabilities remain functional, including graphics and scopes.

Related Examples

- “Configure Four-Bar Model for Code Generation” on page 8-5
- “Find and Fix Code Generation Issues” on page 8-12

Concepts

- “About Code Generation” on page 8-2

Find and Fix Code Generation Issues

In this section...

SimMechanics supports code generation using Simulink Coder. However, certain guidelines and limitations apply. These include:

- Variable step Simulink solver requires `rsim.tlc` target.
- Simulink solver must be continuous.
- SimMechanics does not support visualization in accelerator mode.
- SimMechanics does not support run-time parameters.

Note To generate code for a SimMechanics model, you must have an active Simulink Coder installation.

Variable step Simulink solver requires `rsim.tlc` target

Code generation is compatible with fixed- and variable-step solvers. If you select a variable-step solver, you must specify system target file `rsim.tlc`. To specify the `rsim.tlc` system target file, follow these steps:

1. In the Simulink Editor window for your model, select **Simulation > Model Configuration Parameters**.
2. In the left pane of the **Model Configuration Parameters** dialog box, select **Code Generation**.
3. In **System target file**, enter `rsim.tlc`.

4 Click **Apply**.

5 Click **Build** to generate code for the active model.

Simulink solver must be continuous

Both fixed- and variable-step solvers can be continuous or discrete. Generating code from a SimMechanics model requires a continuous solver. SimMechanics blocks use continuous time samples, and are incompatible with discrete solvers. If you attempt to generate code with a discrete solver, Simulink Coder issues an error.

If you receive an error stating that SimMechanics does not support a discrete solver, select a continuous Simulink solver. To change the Simulink solver, follow these steps:

1 In the Simulink Editor window for your model, select **Simulation > Model Configuration Parameters**.

2 In **Solver**, under **Solver options**, click **Solver**.

3 In the drop-down menu, select any solver with the exception of `discrete (no continuous states)`.

SimMechanics does not support visualization in accelerator mode

SimMechanics supports Accelerator and Rapid Accelerator simulation modes. Selecting an accelerator mode generates executable code that shortens the time required to run a simulation. However, the simulation produces no visualization output. Mechanics Explorer does not open, and you cannot visualize the model simulation. To restore visualization, select the Normal simulation mode.

If you simulate a model in Accelerator or Rapid Accelerator mode, SimMechanics issues a warning indicating that accelerator modes do not support visualization. To remove the warning, disable visualization with Mechanics Explorer:

1 In the Simulink Editor window for your model, select **Simulation > Model Configuration Parameters**.

2 In the **Model Configuration Parameters** window, expand the **SimMechanics 2G** node.

3 Select **Explorer**.

4 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 Mechanics Explorer. When you return to Normal simulation mode, check the box to restore visualization with Mechanics Explorer.

SimMechanics Does Not Support Run-Time Parameters

Model parameters are fixed during code generation. To change model parameters, edit the parameters in SimMechanics and regenerate code for the model. You can only change model parameters in SimMechanics itself.

Related Examples

- "Configure Four-Bar Model for Code Generation" on page 8-5
- "Configure Model for Rapid Acceleration Mode" on page 8-8

Concepts

- "About Code Generation" on page 8-2